ABSTRACT

Title of dissertation: GEOMETRIC METHODS IN MACHINE LEARNING AND DATA MINING

Arvind Agarwal, Doctor of Philosophy, 2012

Dissertation directed by: Professor Hal Daumé III
Department of Computer Science

In machine learning, the standard goal of is to find an appropriate statistical model from a *model space* based on the training data from a *data space*; while in data mining, the goal is to find interesting patterns in the data from a *data space*. In both fields, these spaces carry geometric structures that can be exploited using methods that make use of these geometric structures (we shall call them geometric methods), or the problems themselves can be formulated in a way that naturally appeal to these methods. In such cases, studying these geometric structures and then using appropriate geometric methods not only gives insight into existing algorithms, but also helps build new and better algorithms. In my research, I develop methods that exploit geometric structure of problems for a variety of machine learning and data mining problems, and provide strong theoretical and empirical evidence in favor of using them.

My dissertation is divided into two parts. In the first part, I develop algorithms to solve a well known problem in data mining i.e. distance embedding problem. In particular, I use tools from *computational geometry* to build a unified framework for solving a distance embedding problem known as *multidimensional scaling* (MDS). This geometry-inspired framework results in algorithms that can solve different variants of MDS better than previous state-of-the-art methods. In addition, these algorithms come with many other attractive properties: they are simple, intuitive, easily parallelizable, scalable, and can handle missing data. Furthermore, I extend my unified MDS framework to build scalable algorithms for dimensionality reduction, and also to solve a sensor network localization problem for mobile sensors. Experimental results show the effectiveness of this framework across all problems.

In the second part of my dissertation, I turn to problems in machine learning, in particular, use geometry to reason about conjugate priors, develop a model that hybridizes between discriminative and generative frameworks, and build a new set of generative-process-driven kernels. More specifically, this part of my dissertation is devoted to the study of the geometry of the space of probabilistic models associated with statistical generative processes. This study — based on the theory well grounded in *information geometry* — allows me to reason about the appropriateness of *conjugate* priors from a *geometric* perspective, and hence gain insight into the large number of existing models that rely on these priors. Furthermore, I use this study to build hybrid models more naturally i.e., by combining discriminative and generative methods using the *geometry underlying them*, and also to build a family of kernels called *generative kernels* that can be used as off-the-shelf tool in any kernel learning method such as support vector machines. My experiments of generative kernels demonstrate their effectiveness providing further evidence in favor of using geometric methods.

GEOMETRIC METHODS IN MACHINE LEARNING AND DATA MINING

by

Arvind Agarwal

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Dedication

To my Baba (Grandfather)

Acknowledgments

My thesis work has benefited tremendously from a large number of people both from inside and outside the academic world. First and foremost, I am eternally grateful to my adviser Hal Daumé for his continuous support and guidance throughout my graduate career. He has been extremely helpful, always available, and not to mention very generous – be it money for travel, time to discuss a problem, or just hosting a party at his home. I have learned a lot from him, and it has truly been a privilege to work with him. I would even go as far as to say that he is the best adviser one can have.

I would like to extend my thanks to my committee members Suresh Venkatasubramanian, Amol Deshpande, Jordan Boyd-Graber, and Min wu; all of them have influenced this work in one way or the other. I am indebted to each one of them for their valuable discussions, feedback and input at various steps of this dissertation. My special thanks goes to Suresh for his continuous support and guidance from the very start of my graduate career, and to Min Wu for agreeing to serve on my committee as dean's representative on a short notice.

I have had the good fortune of collaborating with Jeff Phillips, Suresh Venkatasubramanian, and Samuel Gerber. I am grateful to all of them for broadening my horizon in the field of machine learning and beyond. I am also grateful to my managers, mentors, and my colleagues during my two internships at IBM T J Watson Research Center where I had the opportunity to work on question answering system Watson. Thanks to Richard Lawrence, James Fan, Eric Brown, Prem Melville, Hema Raghavan, David Gondek, Karthik Subbian for giving me this opportunity, and for their valuable discussions, help and guidance.

I am thankful to Prof Ellen Riloff at the University of Utah who gave me the first opportunity to work in Natural Language Processing, and is to be blamed for my interest in machine learning. She has been extremely helpful at various stages of my PhD, and I am indebted to her for her support and guidance.

My PhD life would have been dull and boring if it were not for those people around me at the University of Utah and University of Maryland. I would like to thank my colleagues and friends at the University of Utah, Avishek Saha, Adam Teichert, Nathan Gilbert, Piyush Rai, Ruihong Huang, and Siddharth Patwardhan; and at the University of Maryland, Abhishek Kumar, Amit Goyal, Greg Sanders, He He, Jagadeesh Jagarlamudi, Jiarong Jiang, Raul David Guerra, Snigdha Chaturvedi, and Taesun Moon. My special thanks goes to Raul, Greg and Taesun for giving his comments on my dissertation.

My thanks to my friends outside-my-lab goes without saying. I am thankful to my many friends (impossible to name everyone here) — who once-in-a-while reminded me that there is a life outside my lab as well — Ashish Shrivastava, Neha Gupta, Prathyusha Kanala, Preeti Bharghava, Rajesh Chitnis, Sharmistha Chowdhury, Udayan Khurana, and Varun Kumar Nagaraja. My special thanks goes to my college friends Peeyush Raghav, Rahul Tiwari, Vijendra Purohit, and Anita Tamboli who have always made life fun, and kept me away from my PhD life whenever I needed to.

These acknowledgments would be incomplete without mentioning the immense support of the staff at School of Computing at University of Utah and at the department of Computer Science at University of Maryland. My special thanks goes to Karen Feinauer (Utah), Jennifer Story and Fatima Fatima Bangura (Maryland). All three of them are extremely nice, helpful and supportive. They all made sure that I was on track meeting all university deadlines and requirements, and had least of the problems regarding logistics and administration..

I am saving the best for last. None of this PhD work would have been possible if it were not for my family. I am eternally thankful to my grandparents Ramjilal and Badami, parents Geeta and Suresh, my brother Shyam, and my sisters Preeti, Prerana and Poonam. My grandparents and parents raised me, made me capable of making use of opportunities that I have been given, and most of all, always been supportive of me in everything that I have chosen to do. My brother and sisters fought with me, made life fun, and above all, provided much needed emotional support to complete this dissertation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning and geometry have mostly been two separate fields with little interdisciplinary research between them. In this dissertation, my goal is to demonstrate that the methods based on geometry can be used to gain insight into existing learning algorithms, and to design new and better algorithms.

In machine learning, a learning problem takes a set of training points $\{x_i\}_{i=1}^n$ as an input which are assumed to come from some space $\mathcal{X}$, the *data space*. Given a such set of training points, the objective of a learning problem is to find a model from a space Θ, the model space, that explains the data well and generalizes to the future data. The space Θ can either be a constrained set of parameters i.e. parametric space or an unconstrained set of parameters i.e. space of all possible models. When finding an appropriate model for the given training data, a learning problem often makes certain assumptions about the geometries of the data space and the model space, either explicitly or implicitly, which have significant impact on both the solution, and the algorithm to find that solution.

To understand geometric structure in data space, consider an example of a *spherical* distance embedding problem. In this problem, we are given a set of $n \times n$ pairwise distances and a target space $\mathbb{S}^k$ that is a spherical space of a given dimension k, i.e., $\mathbb{S}^k := \{x \in \mathbb{R}^{k+1} : \|x\| = 1\}$. The objective of the problem is to find a set of n

points in the spherical space such that the distances are best preserved. Now note that this spherical distance embedding problem carries a geometric structure inherent to the target space (target space is spherical) which can be exploited to build better algorithms. This spherical distance embedding problem and such other problems that have geometric structures inherent to them can be better solved using methods that make use of underlying geometry. This is an example of the problem where data space carries a geometric structure.

Now consider another problem where geometric structure is found in model space. Consider the problem of text classification i.e., classify given text documents into predefined categories. In the learning problem of text classification, it is common to assume that the text documents are bag-of-words and are generated by some multinomial distribution; and the goal of the learning problem is to find an appropriate model from the space of multinomial distributions that classify future text documents into their correct categories. It is well known [Amari, 1990; Amari and Nagaoka, 2001; Kass and Vos, 1997] that the space of multinomial distributions forms a statistical manifold which is nothing but a special case of Riemannian manifold. Riemannian manifolds (and therefore statistical manifolds) are known to have a *natural* geometric structure associated with them. So when finding an appropriate model for a text classification problem, it is important that one makes use of the geometric structure induced by the statistical manifold. Such consideration of underlying geometric structure can change the algorithms (and their outputs) significantly, and has been shown to be useful in building algorithms that outperform state-of-the-art algorithms that do not consider the underlying geometry [Lebanon and Lafferty, 2004]. This geometric structure underlying the statistical manifold associated with multinomial distributions is not specific to multinomial distributions, but in fact, all statistical manifolds associated with distributions belonging to exponential family are Riemannian manifold and carry geometric structure.

In order to show how consideration of underlying geometry can find a better quality solution, consider the example of the text classification problem again. Information geometry is a field that studies the geometry associated with statistical manifolds. In information geometry, it is widely believed [Amari, 1990; Amari and Nagaoka, 2001] that a *natural* notion of distance for statistical manifolds is the information distance[1], mainly because, it is computed using Fisher information metric which is invariant under re parametrization of the statistical manifold. For the text documents that are bag of words and are generated by a multinomial distribution, this information distance has been shown to be equivalent to the geodesic distance on the sphere [Kass and Vos, 1997; Lebanon and Lafferty, 2004], assuming that the text documents are represented by normalized frequency counts of the words. Specifically, if x_i and x_j are two vectors representing normalized frequency counts of words in respective documents, then the information distance between two documents based on the multinomial generative model is given by:

$$d_g(x_i, x_j) = \cos^{-1}(\langle x_i, x_j \rangle)$$

where $\langle x_i, x_j \rangle$ is the inner product between vectors x_i and x_j. Note that the considera-

[1]Information distance is the geodesic distance on the statistical manifold represented by the underlying probability distribution

tion of underlying geometry changes the distance function along with the structure (now all data points i.e. text documents lie on sphere) and hence the solution.

Although relatively easy for multinomial generative model, the analysis and the consideration of inherent geometry is not so easy for other complex generative models, and becomes increasingly difficult as the complexity of models grow. For example, in complex models such as unsupervised or semi-supervised hidden Markov models, this analysis is not so apparent as in simple generative models such as multinomial generative model. For complex generative models, difficulty arises because one cannot get a closed form expression for the distance function. Fortunately, in machine learning, most of the popular models that we deal with directly or indirectly belong to a special class of distributions known as exponential family distributions [Nielsen, 1978; Brown, 1986] which has been extensively studied by several authors (see [Amari, 1990] and references therein). For such models, it is possible to identify the underlying distance function and hence the geometry. However when a model does not belong to exponential family, identifying the underlying distance function becomes challenging. In my dissertation I deal with probabilistic models that belong to exponential family, and solve a number of problems using methods that are based on the underlying distance function and hence based on the geometry defined by that distance function. These methods driven by geometry not only result in simple and effective algorithms, but also provide insight into existing algorithms.

In my dissertation, I use *geometric methods* to solve a variety of problems, both in machine learning and data mining. Here “geometric methods" refers to methods that make use of geometric structure underlying the data space and the model space to solve a problem. Geometric methods make use of these structures in many ways – they can use these geometric structures *explicitly* by using the distances induced by these geometries, or use them implicitly by developing algorithms that rely on the operations that are based on underlying geometries. In the former, geometric methods involve deriving the appropriate distance function that is associated with the underlying geometry and then use that distance function in the algorithm explicitly, while in the latter, geometric methods rely on geometric operations – since these operations take place in a space, they must respect the geometry of that space. In this latter case, geometric methods treat data points as vectors, and perform *geometric operations* on them – respecting the geometry of the space. Examples of such operations include transformations, moving points, the construction of geometric objects such as lines, triangles, or circles, and performing geometric operations on these objects such as intersection computation. It is important to understand here that the geometric methods (or algorithms) rely on geometric operations in the *operating* space. For example, if a space has a spherical geometry, all operations (e.g. moving points) must be carried out in the spherical space. The strength of geometric methods lie in the fact that they help us visualize the problem, thus providing additional insight into problems and existing algorithms, and often leading to the development of simpler and improved algorithms.

In order to solve a problem using geometric methods, I first study the geometric structure associated with the space of that problem, and then apply appropriate geometric tools to solve it. These geometric methods are based on the concepts of *information geometry* [Spivak, 1999; Amari and Nagaoka, 2001] when dealing with the problems with geometric structure underlying the model space Θ; and the concepts based on *computational geometry* Preparata and Shamos [1985] when dealing with the problems with geometric structure underlying the data space $\mathcal{X}$. In addition to providing intuitive and stronger algorithms to solve different problems, these methods bridge two separate fields "geometry" and "machine learning", and show the advantages of studying these fields together.

In my dissertation, I develop geometric methods for three problems: distance embedding problem (and related problems), building hybrid models by combining generative and discriminative models, and building generative kernels. In the next section, I provide a brief introduction of each of these problems and the summary of my approaches to solving them. The detailed discussion on the motivation, related work and the approaches to solve them will be presented in their respective chapters.

1.1 Geometric Methods in Data Mining

Many problems in data mining rely on a distance function and on the geometry induced by that distance function. For example, the clustering algorithm k-means depends on the underlying distance function to find clusters, k-nearest neighbors (k-NN) classification or search problem depends on finding nearest neighbors where “nearness" is determined by the underlying distance function. The k-means algorithm with Euclidean distance is widely popular, however there are also instances when the distance and therefore the underlying geometry is not Euclidean e.g. spherical k-means [Dhillon and Modha, 2001; Dhillon et al., 2001; Zhong, 2005]. Similarly, k-NN problem has also been studied when the underlying space is not Euclidean but a spherical space [Lunga and Ersoy, 2011] or a general metric space [Yianilos, 1993]. Both k-means and k-NN are examples of the problems that have geometric structure associated with the data space, and therefore, can be better solved by methods that make use of these geometric structures. In this dissertation, I solve a distance embedding problem, in which, one is given a set of pairwise distances, and the objective is to find a set of points such that these distances are best preserved. Similar to k-means and k-NN, this distance embedding problem takes the distances as input and hence naturally appeal to the methods based on distance geometry. In my dissertation, I use geometric methods to solve the distance embedding problem and its variants. This distance embedding problem comes in many flavors and has wide applications in dimensionality reduction, data visualization, sensor network localization, graph embedding, molecular conformation etc.

1.1.1 Distance Embedding

Distance embedding is the problem of finding a set of points that respect the set of given distances [Blumenthal, 1953; Gower, 1982; Liberti et al., 2012]. Formally, given a set of pairwise distances d_{ij} for $(i, j) \in E$, the objective of the distance embedding problem is to find a set of points $\{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$ in some target space $\mathcal{T} = (\mathcal{X}, f)$ such that d_{ij} best approximates the distances in the target space i.e. $f(x_i, x_j)$. Here f is the distance function defined on the set of points in $\mathcal{X}$. This distance embedding problem is a generic problem and can be instantiated to take various forms depending on specific inputs. For example, one can get a different variant of the problem by changing the underlying target space i.e., by varying the distance function f, or the set of points $\mathcal{X}$, or both. Other variations can be obtained by varying the set of given distances E (if E is set of all pairs or only few pairs), or how the approximation between the given distance d_{ij} and the target distances $f(x_i, x_j)$ is measured, etc. A taxonomy of different distance embedding problems can be found in [Liberti et al., 2012].

In the first part of my dissertation, I solve the distance embedding problem using tools from computation geometry. I assume that the data space is a geometric space i.e. the space is endowed with some geometric structure, and points in this space are entities that are allowed to move, construct objects, and participate in geometric operations. In machine learning, it is a standard practice to treat data points as vectors, however it is only in very few cases that these points actually participate in geometric operations leading to the development of new algorithms. Often in machine learning, a learning problem is cast as an optimization problem, and the solution is sought by directly solving the optimization problem using standard optimization methods. Although this approach to solving a learning problem is often successful, it is not the best solution for all problems. Many times, a better quality solution can be found by attacking the problem indirectly – by studying the geometry associated with the problem, and then using the appropriate geometric tools. For example, in my work, I solve a well known distance embedding problem called *multidimensional scaling* (MDS) by giving it a geometric treatment. Earlier methods solve this problem by minimizing the cost function directly using the optimization methods, while I, on the other hand, use geometric operations to solve it. By using these geometric operations, I am able to transform the original problem into a subproblem that is well studied in computational geometry. This geometric treatment of the problem results in a *generalized* framework that not only solves the original MDS problem in *Euclidean* space more effectively than state-of-the-art methods, but also generalizes to other spaces i.e., *spherical* space; and has many desirable properties. In addition to the formal guarantees of convergence, the algorithms derived from this framework are accurate; in most cases, they converge to better quality solutions than existing methods in comparable time. Moreover, they have a small memory footprint and scale effectively for data sets larger than what has previously been reported.

1.1.2 Distance Embedding on Sphere

Distance embedding on a sphere (spherical embedding) is the problem of embedding the given distances under the assumption that the target space is a sphere. Mathematically, in target space $\mathcal{T} = (\mathcal{X}, f)$, $\mathcal{X}$ is the set of all points on a sphere of given dimension k i.e. $\mathcal{X} := \{x \in \mathbb{R}^{k+1} : \|x\| = 1\}$, while function f is the distance function that computes geodesic distance between two points measured along the surface of the sphere. This problem of spherical embedding was first introduced by Cox and Cox [2000], and later followed by [Bronstein et al., 2006, 2008]. Spherical embedding has various applications into texture mapping and image analysis [Bronstein et al., 2008], and is a generalization of the *spherical dimensionality reduction* problem, where the goal is to map points from a high dimensional sphere onto a low-dimensional sphere, with the latter problem being closely related to dimensionality reduction for finite dimensional distributions. A well-known isometric embedding takes a distribution represented as a point on the d-dimensional simplex to the d-dimensional sphere while preserving the Hellinger distance between distributions [Kass and Vos, 1997]. A spherical dimensionality reduction result is an important step to representing high dimensional distributions in a lower-dimensional space of distributions, and will have considerable impact in domains that represent data natively as histograms or distributions, such as in document processing [Pereira et al., 1993; Joachims, 2002; Blei et al., 2003; Lebanon and Lafferty, 2004], image analysis [Lowe, 2004; Dalal and Triggs, 2005] and speech recognition [Gray et al., 1980].

In this dissertation, I use my generalized MDS framework to solve the spherical embedding problem. In hindsight, one strength of using geometric methods to solve the distance embedding problem is that it naturally extends to the spherical case, with the only difference being is that all geometric operations i.e. move, intersection, geometric mean are now performed on the surface of the sphere (spherical space) instead of in the Euclidean space as in Euclidean MDS. This approach to solving the spherical embedding problem can be extended to include other *generalized* distance embedding problems as well, for example, one can generalize this embedding to an arbitrary space where the target space is an arbitrary manifold and the distances are geodesic distances measured along the surface of the manifold.

1.1.3 Large Scale Streaming Distance Embedding/DR

In the distance embedding problem, one is given the pairwise distances among all pairs which makes the time and space complexity of any algorithm at least $O(n^2)$ making it infeasible for large data. In my dissertation, I extend the generalized MDS framework for large scale data using *adaptive sampling*. In addition to providing a linear time, and a constant memory algorithm for distance embedding, a byproduct of my approach is that it gives user the freedom to choose the number of passes, making it suitable for streaming data (when the number of passes is chosen to be one).

1.1.4 Applications of Distance Embedding: SNL

The standard distance embedding problem takes as input pairwise distances among *all* pairs, and can be thought of as a specialized case of a more general problem i.e. sensor network localization (SNL) problem. In SNL problem, one is given a network of sensors that can communicate with each other using communication methods such as radio-signals, infrared-signals etc; and since it is possible (in fact a standard practice) to transform these signals into distances, the SNL problem becomes a distance embedding problem. In the SNL problem, given a set of pairwise distances (or rather obtained by transforming the corresponding communication signals to distances), the objective is to find the locations of the sensors. Since a sensor's communication is only limited to its nearby sensors because of the limited communication radius of sensors, the SNL problem only has access to the distances among nearby pairs making it a more general problem than the standard distance embedding problem. In the SNL problem, in addition to the nearby pairwise distances, one is also given the locations of few sensors, known as “anchors". Although only slightly different from the standard *Euclidean* distance embedding problem, SNL is a significantly difficult problem primarily because of two reasons. First, we are only given distances among few pairs, and second, these distances are computed by transforming signals and therefore are noisy. In this dissertation, I extend my generalized framework to solve the SNL problem, especially for the case when senors are mobile. Although one can use the proposed framework for static SNL problem as well, our experimental study shows that it is more effective for mobile senors.

1.2 Geometric Methods in Machine Learning

In machine learning, a learning problem selects an appropriate model from a model space Θ based on some training data from a data space $\mathcal{X}$. In this dissertation, I focus on solving two machine learning problems using geometry underlying the model space. First, where geometric structure underlying exponential family distributions is used to reason about the appropriateness of conjugate priors, and consequently used to build a semi-supervised hybrid model – a combination of discriminative and generative models, and second, where geometric structure is used to build a new set of generative-process-driven kernels called “generative kernels”

1.2.1 Explaining Conjugate Priors

I study probabilistic models from a geometric perspective by treating the space of parameters Θ as a geometric space and a parameter vector $\theta \in \Theta$ as a point in that space. In machine learning, it is common to assume that data is generated from an *exponential family* distribution parametrized by $\theta \in \Theta$, which has a geometric structure associated with it, given by a natural metric i.e., Fisher information metric and an affine connection. In this work, I study this geometric structure, and use it to reason about the appropriateness of conjugate priors. Specifically, I formulate conjugate prior in the form of Bregman divergence and show that it is the inherent geometry of conjugate priors that makes them appropriate and intuitive. This geometric interpretation allows one to view the hyperparameters of conjugate priors as the *effective* sample points, and to reason about the use of conjugate prior as a coupling prior in hybrid models (see next section).

1.2.2 Combining Discriminative and Generative Models

In machine learning, models of classification have been divided into two categories: discriminative models and generative models; both of which have complimentary properties. In the past few years, there have been many attempts to combine these two types of models and build a hybrid model that performs better than any individual model. One such attempt is to combine these two models using a *coupling* prior. In my work, first, I empirically show that combining these two models using *conjugate* prior as a *coupling* prior is better than combining them using a non-conjugate prior. And then I use geometric understanding of conjugate priors to show that the coupling of these two models using a conjugate prior is nothing but interpolating them using the distance function that is natural to the models under consideration. This geometric interpretation of the hybrid model not only gives us insight into the model but also corroborates our empirical results.

1.2.3 Building Generative Kernels

The above geometric understanding of exponential family models is extended further to build a new set of kernels called *generative kernels*. These kernels are called *generative kernels* because they take into account the generative process of data. My proposed method considers geometry of the space of the statistical model associated with the distribution that generated the data, to build a set of efficient *closed-form* kernels best suited for that distribution. Since these kernels are built considering the geometry associated with the generative model and are used in discriminative models as in SVM, and therefore, they naturally combine the power of both discriminative and generative models and exhibit hybrid behavior. My experiments demonstrate this hybrid behavior, that is, the performance of an SVM using generative kernels is better that an SVN using other kernels, and also better than a pure generative model.

1.3 Contributions

In this dissertation, I claim that the geometric methods are useful not only to gain insight into existing algorithms, but also to develop new algorithms. In order to support my claim, I take a variety of problems and show that the geometric methods are indeed useful, by either getting a better understanding of a problem or by developing better algorithms to solve it. In this work, I specifically take three problems which cover a wide range of problems in both in machine learning and data mining on many fronts. For example, these problems (or the proposed solutions) involve using geometry from both spaces, data space and model space; these problems cover different learning categories e.g., supervised, semi-supervised, and unsupervised; they cover both kind of problems - analysis problems where geometric methods are used only to get a better understanding of a problem without resulting into any new algorithm, and the problems where geometric methods are used not only to get a better understanding of a problem, but also to develop new and better algorithms. The contribution of this dissertation can be summarized as follows:

1. I build a *generalized* distance embedding framework that is suitable for a *variety* of distance embedding problems with different spaces and cost functions. This framework is extended to handle large scale data, and applied to a more general distance embedding problem i.e. sensor network localization problem. Experimental results demonstrate improved performance of the proposed framework compared to state-of-the-art algorithms across all problems: generalized multidimensional scaling problem, large scale dimensionality reduction problem and the sensor network localization problem for mobile sensors.

2. I study the geometry underlying exponential family distributions and use it to reason about the appropriateness of conjugate priors. This geometric under-

standing of conjugate priors allows us not only to explain the effectiveness of these priors in semi-supervised hybrid model, but also to build this hybrid model *more naturally* using the geometry underlying discriminative/generative models under consideration.

3. Finally, I extend the geometric understanding of exponential family distributions to build a new set of kernels called generative kernels, which are based on the geometry of the generative process of data (training and test data of a problem). Experimental results show the effectiveness of these kernels i.e., when used in discriminative methods such as in SVM, these kernels bring in the generative power into discriminative methods and outperform a purely generative or discriminative method.

1.4 Dissertation Outline

In order to show the utility of geometric methods, I take a variety of problems, which as discussed above, cover a wide range of learning problems – from *unsupervised* to *semi-supervised* to *supervised*. Under the unsupervised category, I use geometric methods to develop new algorithms for solving a generalized distance embedding problem (Chapters 2, 3, and 4), and apply it to sensor network localization problem 5. Under the semi-supervised category, I first build a hybrid model suitable for semi-supervised learning using coupling prior (Chapter 6), and then show, how a geometric understanding of the underlying distributions can be used not only to gain insight into this hybrid model but also to build it naturally (Chapter7). In Chapter 8, I use geometric methods to build a family of kernels called *generative kernels* that are then later used in supervised learning. This dissertation is organized as follows:

Part I: Data Mining

Chapters 2 shows the utility of geometric methods by building a unified algorithmic framework for solving many known variants of MDS. In addition to formal convergence proof, in this chapter I experiment on a variety of problems with different cost functions, and show the effectiveness of the algorithms derived from this framework compared to state-of-the-art algorithms.

Chapter 3 shows the application of the unified MDS framework to the spherical embedding problem. Experimental results indicate that the proposed framework solves the spherical embedding problem better than existing algorithms.

Chapter 4 extends the unified MDS framework for large scale data. In this chapter. I propose an approximate version of the unified MDS framework which takes linear time and constant memory, as opposed to the exact version that takes square time and linear memory. An empirical study of the proposed approximate algorithm across different data sets shows improved performance over state-of-the-art algorithms.

Chapter 5 modifies the unified MDS framework to solve the sensor network localization (SNL) problem. Experimental results show that my algorithm outperforms existing algorithms across a variety of networks, especially when sensors are moving (mobile sensors).

Part II: Machine Learning

Chapter 7 describes building of a hybrid semi-supervised learning model by combining discriminative and generative models. This hybrid model is based on the principle of coupling prior, more specifically, it uses conjugate prior to couple these two types of models, unlike the previous work where a Gaussian prior is used to couple them. Empirical results show the effectiveness of using conjugate prior as coupling prior over non-conjugate prior.

Chapter 7 is devoted to understanding how geometric methods can be used to gain insight into existing algorithms by studying conjugate priors from a geometric perspective, and justifying their use in hybrid generative/discriminative model.

Chapter 8 uses geometric methods to build *generative kernels*, a family of kernels that can be used as off-the-shelf tool in any kernel method such as SVM. Experimental results show that these generative kernels when used in SVM not only outperform other kernels based on same principle, but also outperforms a purely generative or a purely discriminative model.

Part III: Conclusion and Future Work

Chapter 9 summarizes the findings of this dissertation and presents directions for future work.

Part I

Data Mining

The first part of this dissertation is devoted to solving a generalized distance embedding problem for different spaces and cost functions, developing scalable algorithms for it, and applying it to sensor network localization task. In particular:

In Chapter 2, I use tools from computational geometry to build an algorithmic framework to solve a distance embedding problem or more specifically a multidimensional scaling problem. I call this algorithmic framework PLACECENTER which is based on simple geometric operations such as moving points in space and finding centroid. The proposed framework is simple, scalable, and can handle a variety of spaces and cost functions. Experimental results show the efficacy of the algorithms derived from this framework over the existing state-of-the-art methods.

In Chapter 3, I apply PLACECENTER to perform distance embedding on spheres. Experimental results indicate that PLACECENTER not only outperforms state-of-the-art methods but is also free of any hyperparameters, unlike other methods to solve spherical embedding problem.

In Chapter 4, I extend PLACECENTER for large scale datasets. Although PLACECENTER can already handle larger amount of data than existing methods because of its linear memory requirement, I extend it to handle even larger dataset using landmarks (sampling). This landmark based PLACECENTER—called PLACE-RANDOM— uses *adap-*

tive sampling, and is shown to outperform other similar sampling-based distance embedding methods.

In Chapter 5, I show an application of PlaceCenter on a sensor network localization (SNL) problem for mobile sensors. SNL problem is similar to a distance embedding problem with two main differences. First, unlike the standard distance embedding problem, SNL only have access to *partial* but *noisy* distance matrix; and second, in addition to the distance matrix, it also has access to the locations of some of the sensors. This chapter describes how PlaceCenter can be extended to solve SNL problem efficiently and effectively especially when sensors are mobile.

Chapter 2

Universal Multidimensional Scaling

This chapter demonstrates the power of geometric methods by building a *unified* algorithmic framework to solve the *generalized* multidimensional scaling (MDS) problem. By using simple tools from computational geometry, I am able to build a framework that performs comparable to or better than the state-of-the-art methods for many known variants of MDS, and, at the same time, provides algorithms for the variants that have not yet been studied. In additional to superior performance, the algorithms derived from this geometry-driven framework have many attractive properties i.e. they are simple, intuitive, easily parallelizable, and scale well to large data. This framework easily generalizes to spaces other than Euclidean space because it is based on simple geometric operations such as moving points in the space, intersection of line with surfaces etc., and a well known *min-sum* or centroid problem [Karcher, 1977; Buss and Fillmore, 2001], which is closely related to the facility location problem [Hochbaum, 1982; Bajaj, 1986] — a problem well studied problem in theoretical computer science and computational geometry.

2.1 Motivation

Multidimensional scaling (MDS) [Kruskal and Wish, 1978; Cox and Cox, 2000; Borg and Groenen, 2005] is a widely used method for embedding a general distance matrix into a low dimensional Euclidean space, used both as a preprocessing step for many problems, as well as a visualization tool in its own right. MDS has been studied and used in psychology since the 1930s [Young and Householder, 1938; Torgerson, 1952; Kruskal, 1964] to help visualize and analyze data sets where the only input is a distance matrix. More recently MDS has become a standard dimensionality reduction and embedding technique to deal with the challenges associated with high dimensional data sets [Cayton and Dasgupta, 2006; Chen and Buja, 2009; Pless and Simon, 2002; Bronstein et al., 2008].

In general, the problem of embedding an arbitrary distance matrix into a fixed dimensional Euclidean space with minimum error is nonconvex. Thus, in addition to the standard formulation [de Leeuw, 1977], many variants of MDS have been proposed, based on changing the underlying error function [Young and Householder, 1938; Cayton and Dasgupta, 2006] – function that measures the error between given distances and the embedded distances. There are also applications where the target space, rather than being a Euclidean space, is a manifold (e.g. a low dimensional sphere), and various methods for MDS in this setting have also been proposed [de Leeuw and Mair, 2009; Bronstein et al., 2008].

Each such variant is typically addressed by a different method, including majorization [de Leeuw and Mair, 2009], singular value decomposition [Torgerson, 1952], semidefinite programming [Cayton and Dasgupta, 2006], subgradient methods [Cayton and Dasgupta, 2006], and standard Lagrange-multiplier-based methods (in both primal and dual settings) [de Leeuw and Mair, 2009]. Some of these methods are efficient (in terms of time), and others are not; in general, every new variant of MDS seems to require different ideas.

In this chapter, I build a unified algorithmic framework called UMDS for solving many variants of MDS. My approach is based on an iterative local improvement method, and can be summarized as follows: "Pick a point (among the points from the previous iteration) and move it so that the cost function is locally optimal. Repeat this process until convergence." The improvement step reduces to a well-studied and efficient family of iterative minimization techniques, where the specific algorithm depends on the variant of MDS. This proposed method does not solve the optimization problem associated with MDS *directly*, rather it breaks the problem into smaller subproblems where each subproblem can be solved by using well known geometric tools. The resulting algorithm is generic, efficient, and simple. The high level framework can be written in 10-12 lines of MATLAB code, with individual function-specific subroutines needing only a few more lines each. A useful feature of this framework is that it is parameter-free, requiring no tuning parameters or Lagrange multipliers in order to perform at its best. Further, this approach compares well with the best methods for all the variants of MDS, and scales to large data sets (because of small memory footprint).

Summary of contributions. The main contributions of this chapter can be summarized as follows:

- In Section 4.2 I present an iterative framework, illustrate how it is applied to specific MDS variants and prove a convergence result.
- In Section 4.3 I present a comprehensive experimental study that compares my approach to the prior best known methods for different MDS variants. In

addition, I also show how the proposed framework can handle the data of a scale bigger than the existing methods.

2.2 Definitions

Let D be a $n \times n$ distance matrix representing the pairwise distances among a set of n points $P = \{p_1, \dots p_n\}$ [1], and $\mathscr{T} = (\mathcal{X}, f)$ be a target space where $\mathcal{X}$ is the set of all points in the target space, and $f : \mathcal{X} \times \mathcal{X} \to R^+$ is the distance function that given two points in the target space, computes the distance between them. I assume that $f(x,x) = 0$ for any $x \in \mathcal{X}$. One example of such a target space is when $\mathcal{X} = \mathbb{R}^k$, and f is the Euclidean distance $f(x_i, x_j) = \|x_i - x_j\|_2$. In general, it is assumed that D is symmetric (i.e $d_{ij} = d_{ji}$), although my method does not formally require this. Also when D is not given but one has access to the point set P and a distance function h, D can be computed using h that is $h(p_i, p_j) = d_{ij}$.

The multidimensional scaling problem takes as input D, target space $\mathscr{T}$, and k, and asks for a mapping $\pi : D \to X = \{x_1, x_2, \dots, x_n\} \subset \mathcal{X}$, from D to a set of points X in a k-dimensional target space $\mathscr{T}$ such that the difference between the original and resulting distances is minimized. There are many different ways to measure the difference between the sets of distances, and one popular way to do to use the following function, known as generalized *stress* function:

$$C(X, D) = \sum_i \sum_j \mathrm{Err}(f(x_i, x_j) - d_{ij}), \tag{2.1}$$

[1]The point set P is mentioned only for the illustrative purpose. The MDS problem only needs a set of n^2 pairwise distances as input.

where Err measures the discrepancy between the source and target distances, and f denotes the function that measures distance in the target space. Below are some instances of the above generalized MDS formulation for specific choice of target space $\mathcal{T}$, distance function f, and discrepancy measure Err.

- $\mathcal{T} = \mathbb{R}^k, \mathrm{Err}(\delta) = \delta^2, f(x, x') = \|x - x'\|_2$: This is a general form of the MDS problem, which I refer to as fMDS.
- $\mathcal{T} = \mathbb{R}^k, \mathrm{Err}(\delta) = |\delta|, f(x, x') = \|x - x'\|_2$: This is a *robust* variant of MDS called rMDS, first suggested by Cayton and Dasgupta [Cayton and Dasgupta, 2006].
- $\mathcal{T} = \mathbb{S}^k$, a k-dimensional sphere; $\mathrm{Err}(\delta) = |\delta|$ or δ^2, $f(x, x')$ is either chordal(c) or geodesic distance (g)[2] on $\mathbb{S}^k$. I refer to this family of problems as {c,g}-{1,2}-sMDS (see Chapter 3).

It will be convenient to split the expression into individual component terms. I define

$$C_i(X, D, x) = \sum_j \mathrm{Err}(f(x, x_j) - d_{ij}) \tag{2.2}$$

which allows us to write

$$C(X, D) = \sum_i C_i(X, D, x_i).$$

[2]For two points x_i and x_j, chordal distance is nothing but the Euclidean distance between them, while geodesic distance is the length of the shortest path from x_i and x_j measured along the great circle passing through x_i and x_j.

Remarks:

- The actual measure studied by Cayton and Dasgupta [2006] is not rMDS. It is a variant which takes the absolute difference of the *squared* distance matrices. I call this measure r^2MDS.

- One can also have a weighted version of the above MDS formulation (2.1), defined as:

$$C(X, D, W) = \sum_i \sum_j w_{ij} \, \mathrm{Err}(f(x_i, x_j) - d_{ij}), \tag{2.3}$$

  where weights $W = [w_{ij}]$ are the importance given to the individual distances. This weighted variant can also handle missing distances (i.e., some of the distances in the distance matrix are not known) by setting the corresponding weights to be 0. Note that not all existing methods (e.g. classical MDS) to solving MDS can handle the weighted variant. We will see that my method naturally handle the weighted version without requiring any additional complexity.

- I assume that the target space comes with a distance function such that $f(x_i, x_j) = 0$. I also assume $d_{ij} = 0$ for the ease of notations, however it is not required.

2.3 Background and Existing Methods

Multidimensional scaling is a *family* of problems for embedding a distance matrix into a fixed dimensional space (Euclidean or an arbitrary but given manifold e.g. sphere). There is a general taxonomy of MDS problems [Cox and Cox, 2000], however, in this dissertation, I will focus on the *generalized* MDS problem with the error function being the *stress function* (see (2.1)).

The traditional formulation of MDS [Kruskal and Wish, 1978] assumes that a given distance matrix D comes from points in some k-dimensional Euclidean space. Under this assumption, a simple transformation takes the distance matrix $D = [d_{ij}]$[3] with the n^2 distances to preserve to a matrix of *similarities* $S = [s_{ij}]$, where $s_{ij} = \langle x_i, x_j \rangle$. More specifically, under this transformation, the Euclidean distance matrix D is *centered* to $\bar{D}$ so the mean of all points $X = \{x_1, x_2 \ldots x_n\}$ is 0. Denote the ith column vector of $\bar{D}$ as the point x_i. Then $\bar{D}$ is converted to a similarities matrix S where $S_{i,j} = \langle x_i, x_j \rangle$ or $S = XX^T$. The MDS problem (2.1) then reduces to finding a set of points $X = \{x_1, \ldots x_n\}$ in k-dimensional space such that XX^T approximates S in terms of squared error ($\text{Err}(\delta) = \delta^2$). This can be done *optimally* using the top-k singular values and vectors from the singular value decomposition of S. While the prime objective of MDS is to preserve the distances, the most popular method to MDS [Torgerson, 1952; Cox and Cox, 2000] takes the above described alternative route i.e., preserve the similarities, and is often referred to as *classical MDS*.

A more direct approach called SMACOF that drops the k-dimensional Euclidean space assumption uses a technique known as stress majorization [Marshall and Olkin, 1979; de Leeuw, 1977; de Leeuw and Mair, 2009]. SMACOF has been adapted to many other MDS variants e.g., variants where distances are assumed to come from points that lie on quadratic surfaces and spheres [de Leeuw and Mair, 2009].

[3]When one is not given the distance matrix D, but has access to a set of points $P = \{p_1, p_2, \ldots p_n\}$ in some higher d-dimensional space ($d > k$), and the distance function h, d_{ij} can be computed as $d_{ij} = h(p_i, p_j)$.

The above algorithms preserve distances by minimizing the squared error between the original distances and the distances computed from embedded points, and since the squared error metric is sensitive to outliers, Cayton and Dasgupta [2006] proposed a robust variant based on an ℓ_1 error metric. They relax the optimization problem by removing the rank constraint, and solve it using either semidefinite programming or a subgradient heuristic, followed by a singular value decomposition to enforce the rank constraints.

Embeddings that guarantee bounded error. A complementary line of work in dimensionality reduction fixes an error bound for *every* pair of distances (rather than computing an average error), and asks for the minimum dimension a data set can be embedded in while maintaining this error. The Johnson-Lindenstrauss Lemma [Johnson and Lindenstrauss, 1984] states that any collection of distances that come from n points in a *d-dimensional Euclidean space* can be embedded in a $k = O((1/\varepsilon^2)\log n)$ dimensional Euclidean space that preserves all distances within a relative error of ε. If the points instead lie in an abstract metric space, then the best possible result is an embedding into $O(\log n)$-dimensional Euclidean space that preserves distances up to a factor of $O(\log n)$. An exhaustive survey of the different methods for dimensionality reduction is beyond the scope of this thesis - the reader is directed to the survey by Indyk and Matousek for more information [Indyk and Matousek, 2004].

2.4 My Algorithm

I now present the proposed algorithmic framework PlaceCenter (Algorithm 1) that finds a mapping $D \to X$ minimizing $C(X,D)$. For now it is assumed that we are given an initial embedding $X_0 \in \mathbb{R}^k$ to seed the algorithm. The experimental results (see Section 4.3)indicate the SVD-based approach [Young and Householder, 1938] is almost always the best way to seed the algorithm, and I will use it unless specifically indicated otherwise.

Algorithm 1 PlaceCenter $(D, \epsilon_d, \epsilon_a)$

```
Choose an initial embedding $X_0$.
$X \leftarrow X_0$
**repeat**
  $err \leftarrow C(X,D)$.
  **for** $i = 1$ **to** $n$ **do**
    $x_i \leftarrow$ Place$_i(X, D, x_i, \epsilon_a)$. {this updates $x_i \in X$}
  **end for**
**until** $(err - C(X,D) < \epsilon_d)$ {for a fixed threshold $\epsilon_d$}
Return $X$.
```

The algorithm operates by employing a technique from the block-relaxation class of heuristics [De Leeuw and Michailidis, 1994, and references therein]. The cost function (2.1) can be expressed as a sum of costs for each point x_i (2.2), and so in each step of the inner loop it finds the best placement for x_i while keeping all other points fixed, using the algorithm Place$_i(X, D, x_i, \epsilon_a)$. A key insight driving this approach is that Place$_i(X, D, x_i, \epsilon_a)$ can be implemented either iteratively or exactly for a wide class of distance functions with provable convergence guarantees. The process terminates when over all i, invoking Place$_i(X, D, x_i, \epsilon_a)$ does not reduce the cost $C(X,D)$ by more than a threshold ϵ_d. The algorithm takes $O(n^2)$ for each iteration,

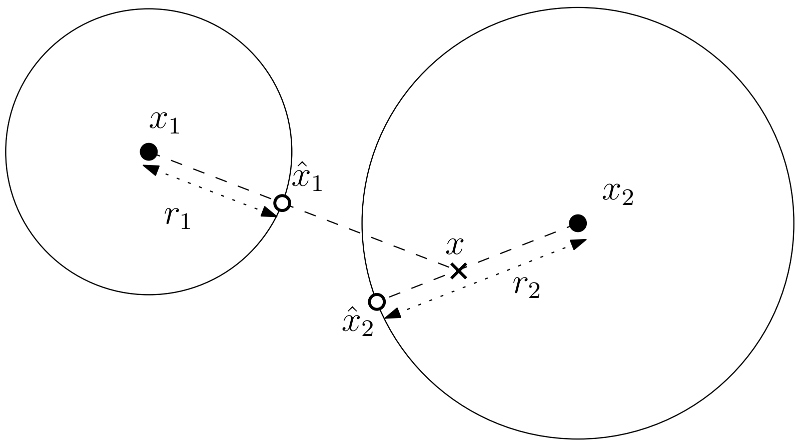


Figure 2.1: A geometric interpretation of the error term.

since $\textsc{Place}_i(X, D, x_i, \epsilon_a)$ will take $O(n)$ time and computing $C(X, D)$ takes $O(n^2)$ time.

A Geometric perspective on Place$_i(X, D, x_i, \epsilon_a)$. The routine $\textsc{Place}_i(X, D, x_i, \epsilon_a)$ is the heart of the proposed algorithm. This routine finds the optimal placement of a fixed point x_i with respect to the cost function $C_i(X, D, x_i) = \sum_j \mathrm{Err}(f(x_i, x_j) - d_{ij})$. Set $r_j = d_{ij}$. Then the optimal placement of x_i is given by the point x^* minimizing the function

$$g_i(x) = \sum_j \mathrm{Err}(f(x, x_j) - r_j).$$

Note that the terms $f(x, x_i)$ and $r_i = d_{ii}$ are zero, so we can ignore their presence in the summation for ease of notation. As mentioned earlier, d_{ii} does not necessarily have to be zero,but even when $d_{ii} \neq 0$, it can be ignored simply because it only adds the constant (term independent of x) to $g_i(x)$ and adding or keeping it will not change the solution x^*. Also note that $C_i(X, D, x)$ and $g_i(x)$ are the same functions, with latter used only for the ease of notations.

Now I give a geometric interpretation of $g_i(x)$ which is illustrated in Figure 3.1.

Consider a sphere around the point x_j of radius r_j. Let $\hat{x}_j$ be the point on this sphere that intersects the ray from x_j towards x. Then the distance $f(x, \hat{x}_j) = |f(x, x_j) - r_j|$. Thus, $g_i(x)$ can be rewritten as

$$g_i(x) = \sum_j \text{Err}(f(x, \hat{x}_j)).$$

Here computation of $\hat{x}_j$ depends on the underlying target space – for the Euclidean space, it can be computed as follows(for the spherical case, see Chapter 3):

$$\hat{x}_j = x_j - \frac{r_j}{\|x_i - x_j\|_2}(x_j - x_i).$$

This function $g_i(x)$ is well-known in combinatorial optimization as the *min-sum* problem. For $\text{Err}(\delta) = \delta^2$, $g_i(x)$ finds the point minimizing the sum-of-squared distances from a collection of fixed points (the 1-mean), which is the centroid $x^* = \frac{1}{n}\sum_j \hat{x}_j$. For $\text{Err}(\delta) = |\delta|$, $g_i(x)$ finds the 1-median: the point minimizing the sum of distances from a collection of fixed points. Although there is no closed form expression for the 1-median, there are numerous algorithms for solving this problem both exactly [Weiszfeld, 1937] and approximately [Bose et al., 2003]. Methods that converge to the global optimum exist for any $\text{Err}(\delta) = |\delta|^p, p \leq 2$; it is known that if p is sufficiently larger than 2, then convergent methods may not exist [Brimberg and Love, 1993].

While $g_i(x)$ can be minimized optimally for error functions Err of interest, the location of the points $\hat{x}_j$ depends on the location of the solution x^*, which is itself unknown! This motivates an alternating optimization procedure, where the current iterate x is used to compute $\hat{x}_j$, and then these $\hat{x}_j$ are used as input to the min-sum problem to solve for the next value of x. The alternating optimization procedure is presented in Algorithm 2.

Algorithm 2 $\textsc{Place}_i(X, D, x_i, \epsilon_a)$

```
repeat
    $\epsilon \leftarrow g_i(x_i)$
    for $j = 1$ to $n$ do
        $\hat{x}_j \leftarrow$ intersection of sphere of radius $r_j$ around $x_j$ with ray from $x_j$ towards $x_i$.
    end for
    $x_i \leftarrow \textsc{Recenter}(\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\})$.
until $(\epsilon - g_i(x_i) < \epsilon_a)$ {for a fixed threshold $\epsilon_a$}
Return $x_i$.
```

2.4.1 Implementing Recenter

Up to this point, the description of PlaceCenter and Place has been generic, requiring no specification of Err and f. In fact, all the domain-specificity of the method appears in Recenter, which solves the min-sum problem. We now demonstrate how different implementations of Recenter allow us to solve the different variants of MDS discussed above.

2.4.1.1 The original MDS: fMDS

Recall from Section 2.2 that the fMDS problem is defined by $\mathrm{Err}(\delta) = \delta^2$ and $f(x, x') = \|x - x'\|_2$. Thus, $g_i(x) = \sum_j \|x - \hat{x}_j\|^2$. As mentioned earlier, the minimum of this function is attained at $x^* = (1/n)\sum_j \hat{x}_j$. Thus, $\textsc{Recenter}(\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\})$ merely outputs $(1/n)\sum_j \hat{x}_j$, and takes $O(n)$ time per invocation.

2.4.1.2 Robust MDS: rMDS

The robust MDS problem rMDS is defined by $\mathrm{Err}(\delta) = |\delta|$ and $f(x, x') = \|x - x'\|_2$. Minimizing the resulting function $g_i(x)$ yields the famous Fermat-Weber problem, or the 1-median problem as it is commonly known. An exact iterative algorithm for solving this problem was given by Weiszfeld [1937], and works as follows. In each iteration of the algorithm, the value x_i is updated by

$$x_i \leftarrow \sum_j \frac{\hat{x}_j}{\|x_i - \hat{x}_j\|} \Bigg/ \sum_j \frac{1}{\|x_i - \hat{x}_j\|}.$$

This algorithm is guaranteed to converge to the optimal solution [Kuhn, 1973; Ostresh, 1978], and in most settings converges quadratically [Katz, 1974].

Other norms and distances. If $\mathrm{Err}(\delta) = |\delta|^p, 1 < p < 2$, then an iterative algorithm along the same lines as the Weiszfeld algorithm can be used to minimize $g_i(x)$ optimally [Brimberg and Love, 1993]. In practice, this is the most interesting range of values for p. It is also known that for p sufficiently larger than 2, this iterative scheme may not converge.

We also can tune PLACECENTER to the r^2MDS problem (using squared distances) by setting $r_j = d_{ij}^2$. Although the convergence proofs (below) do not hold in this case, the algorithm does converge in practice (see results in Section 2.5.3).

2.4.2 Convergence Proofs

Here we prove that each step of PlaceCenter converges as long as the recursively called procedures reduce the relevant cost functions. Convergence is defined with respect to a cost function κ, so that an algorithm converges if at each step κ decreases until the algorithm terminates.

Theorem 1. *If each call to $\tilde{x}_i \leftarrow \text{Place}_i(X, D, x_i, \epsilon_a)$ decreases the cost $C_i(X, D, x_i)$, then $\text{PlaceCenter}(D, \epsilon_d, \epsilon_a)$ converges with respect to $C(\cdot, D)$.*

Proof. Let $\tilde{X} \leftarrow \text{Place}_i(X, D, x_i, \epsilon_a)$ result from running an iteration of $\text{Place}_i(X, D, x_i, \epsilon_a)$. Let $\tilde{X} = \{x_1, \ldots, x_{i-1}, \tilde{x}_i, x_{i+1}, \ldots, x_n\}$. Then we can argue

$$
\begin{aligned}
&C(X, D) - C(\tilde{X}, D) \\
&\quad = \; 2\sum_{j=1} \text{Err}(f(x_i, x_j) - d_{i,j}) - 2\sum_{j=1} \text{Err}(f(\tilde{x}_i, x_j) - d_{i,j}) \\
&\quad = \; 2C_i(X, D, x_i) - 2C_i(\tilde{X}, D, \tilde{x}_i) > 0.
\end{aligned}
$$

The last line follows because X and $\tilde{X}$ only differ at x_i versus $\tilde{x}_i$, and by assumption on $\text{Place}_i(X, D, x_i, \epsilon_a)$, this sub-cost function must otherwise decrease. □

Theorem 2. *If each call $x_i \leftarrow \text{Recenter}(\hat{X})$ reduces $\sum_{j=1}^{n} f(x_i, \hat{x}_j)^p$, then $\text{Place}_i(X, D, x_i, \epsilon_a)$ converges with respect to $C_i(X, D, \cdot)$.*

Proof. First we can rewrite

$$C_i(X, D, x_i) = \sum_{j=1}^{n} \text{Err}(f(x_i, x_j) - d_{i,j})$$

$$= \sum_{j=1}^{n} \mathrm{Err}((f(x_i, \hat{x}_j) + d_{i,j}) - d_{i,j})$$
$$= \sum_{j=1}^{n} \mathrm{Err}(f(x_i, \hat{x}_j)).$$

Since $\mathrm{Err}(f(x_i, \hat{x}_j))$ measures the distance to the sphere $\circ_j$ ($\hat{x}_j$ is the closest point to x_i on $\circ_j$). Then choosing x'_i to minimize (or decrease) $\sum_{j=1}^{n} \mathrm{Err}(f(x'_i, \hat{x}_j))$, must decrease the sum of distances to each point $\hat{x}_j$ on each sphere $\circ_j$. Now let $\hat{x}'_j$ be the closest point to x'_i on $\circ_j$. Hence $\mathrm{Err}(f(x'_i, \hat{x}'_j)) \leq \mathrm{Err}(f(x'_i, \hat{x}_j))$ and thus

$$C_i(X, D, x'_i) = \sum_{j=1}^{n} \mathrm{Err}(f(x'_i, \hat{x}'_j)) \leq \sum_{j=1}^{n} \mathrm{Err}(f(x'_i, \hat{x}_j))$$
$$\leq \sum_{j=1}^{n} \mathrm{Err}(f(x_i, \hat{x}_j)) = C_i(X, D, x_i)$$

where equality only holds if $x_i = x'_i$, in which case the algorithm terminates. □

2.4.3 Relationship with Gradient Descent

I now show that Place$_i$ with the Euclidean space as the target space (fMDS) is nothing but a gradient descent algorithm with learning rate $\lambda = 1/2n$. Recall that the cost function is for the point x_i is:

$$C_i(X, D, x_i) = \sum_{j=1}^{n} \left(\|x_i - x_j\| - d_{i,j} \right)^2.$$

Taking the gradient with respect to x_i

$$\begin{aligned}
\frac{\partial C_i(X,D,x_i)}{\partial x_i} &= \sum_{j=1}^{n} 2\left(\|x_i - x_j\| - d_{i,j}\right)\frac{\partial \|x_i - x_j\|}{\partial x_i} \\
&= \sum_{j=1}^{n} 2\left(\|x_i - x_j\| - d_{i,j}\right)\frac{(x_i - x_j)}{\|x_i - x_j\|} \\
&= \sum_{j=1}^{n} 2\left(1 - \frac{d_{i,j}}{\|x_i - x_j\|}\right)(x_i - x_j).
\end{aligned}$$

Now the new point can be computed iteratively using the gradient update procedure (using learning rate $\lambda = \frac{1}{2n}$):

$$\begin{aligned}
x_i^* &= x_i - \lambda\frac{\partial C_i(X,D,x_i)}{\partial x_i} \\
&= x_i - \frac{1}{2n}\sum_{j=1}^{n} 2\left(1 - \frac{d_{i,j}}{\|x_i - x_j\|}\right)(x_i - x_j) \\
&= x_i - \frac{1}{n}\sum_{j=1}^{n}(x_i - x_j) + \frac{1}{n}\sum_{j}\left(\frac{d_{i,j}}{\|x_i - x_j\|}\right)(x_i - x_j) \\
&= \frac{1}{n}\sum_{j=1}^{n}\left(x_j + (x_i - x_j)\frac{d_{i,j}}{\|x_i - x_j\|}\right) \\
&= \frac{1}{n}\sum_{j=1}^{n}\hat{x}_j.
\end{aligned}$$

These $\hat{x}_j = x_j + (x_i - x_j)\frac{d_{i,j}}{\|x_i - x_j\|}$ are interpolated points between x_j and x_i at the fractional distance $q = \frac{d_{i,j}}{\|x_i - x_j\|}$. Although this algorithm is simply the gradient descent algorithm with learning rate $1/2n$, it is to be noted that algorithm always monotonically reduces the cost, which is not true for general gradient descent algorithms

2.4.4 Working Space Usage

PlaceCenter$(D, \epsilon_d, \epsilon_a)$ takes an $n \times n$ distance matrix as input, but each invocation of Place$_i(X, D, x_i, \epsilon_a)$ only operates on a single point. This means that although the input complexity is $O(n^2)$, the working memory footprint of the algorithm is only $O(n)$. This is a significant advantage of PlaceCenter$(D, \epsilon_d, \epsilon_a)$ over many existing MDS methods that require the entire matrix D to be stored in memory. In Section 4.3 we will see that this small memory footprint enables us to run PlaceCenter$(D, \epsilon_d, \epsilon_a)$ for values of n well beyond the point where other methods start to fail.

2.5 Experiments

In this section I evaluate the performance of PlaceCenter (PC). Since PlaceCenter generalizes to many different cost functions, I compare it with the best known algorithm for each cost function, if one exists. For the fMDS problem the leading algorithm is SMACOF [de Leeuw and Mair, 2009]; for the r^2MDS problem the leading algorithm is by Cayton and Dasgupta (CD) [Cayton and Dasgupta, 2006]. I know of no previous scalable algorithm designed for rMDS. I note that the Cayton-Dasgupta algorithm REE does not exactly solve the r^2MDS problem. Instead, it takes a non-Euclidean distance matrix and finds a Euclidean distance matrix that minimizes the error without any rank restrictions. Thus, as suggested by the authors [Cayton and Dasgupta, 2006], to properly compare the algorithms, I let CD refer to running REE and then projecting the result to a k-dimensional subspace using the SVD technique [Young and Householder, 1938] (our plots show this projection after each step). With regards to each of these Euclidean measures I compare our algorithm with SMACOF and CD. I also compare with the popular SVD-based method [Young and Householder, 1938], which solves the related cMDS problem based on similarities, by seeding all three iterative techniques with the results of the closed-form SVD-based solution.

The subsections that follow focus on individual cost measures. I then discuss the overall behavior of our algorithm in Section 2.5.5.

Data sets, code, and setup. Test inputs for the algorithms are generated as follows. I start with input consisting of a random point set with $n = 300$ points in $\mathbb{R}^d$ for $d = 200$, with the target space $\mathscr{T} = \mathbb{R}^k$ with $k = 10$. Many data sets in practice have much larger parameters n and d, but I limit myself to this range for most of the experiments because for larger values CD becomes prohibitively slow, and both SMACOF and CD run into memory problems. In Section 2.5.4 I explore the performance of our algorithm on larger data sets (up to $50,000$ points). The data is generated to first lie on a k-dimensional subspace, and then (full-dimensional) Gaussian noise is applied to all points up to a magnitude of 30% of the variation in all dimensions. Finally, I construct the Euclidean distance matrix D which is provided as input to the algorithms.

These data sets are Euclidean, but "close" to k-dimensional. To examine the behavior of the algorithms on distance matrices that are non-Euclidean, I generate data as before in a k-dimensional subspace and generate the resulting distance matrix D. Then I perturb a fraction of the elements of D (rather than perturbing the points) with Gaussian noise. The fraction perturbed varies in the set $(2\%, 10\%, 30\%, 90\%)$.

All algorithms were implemented in MATLAB. For SMACOF, I used the implementation provided by Bronstein[4], while for all other algorithms, I used our own implementation[5]. In all cases, I compare performance in terms of the error function $C(X,D)$ as a function of clock time.

2.5.1 The rMDS Problem

Figure 2.2 shows the cost function $C(X,D)$ associated with rMDS plotted with respect to runtime. PlaceCenter always reaches the best local minimum, partially because only PlaceCenter can be adjusted for the rMDS problem. It also observed that the runtime is comparable to SMACOF and much faster than CD in order to get to the same $C(X,D)$ value. Although SMACOF initially reaches a smaller cost that PlaceCenter, it later converges to a larger cost because it optimizes a different cost function (fMDS).

I repeat this experiment in Figure 2.3 for different values of k (equal to $\{2, 20, 50, 150\}$) to analyze the performance as a function of k. Note that PlaceCenter performs even better for lower k in relation to CD. This is likely as a result of CD's reliance on the SVD technique to reduce the dimension. At smaller k, the SVD technique has a tougher job to do, and optimizes the wrong metric. Also for $k = 150$ note that CD oscillates in its cost; this is again because the REE part finds a nearby Euclidean distance matrix which may be inherently very high dimensional and the SVD projection is very susceptible to

[4]Accelerated MDS, `http://tosca.cs.technion.ac.il/book/resources_sw.html`

[5]All of the code may be found at `http://www.cs.utah.edu/~suresh/papers/smds/smds.html`.

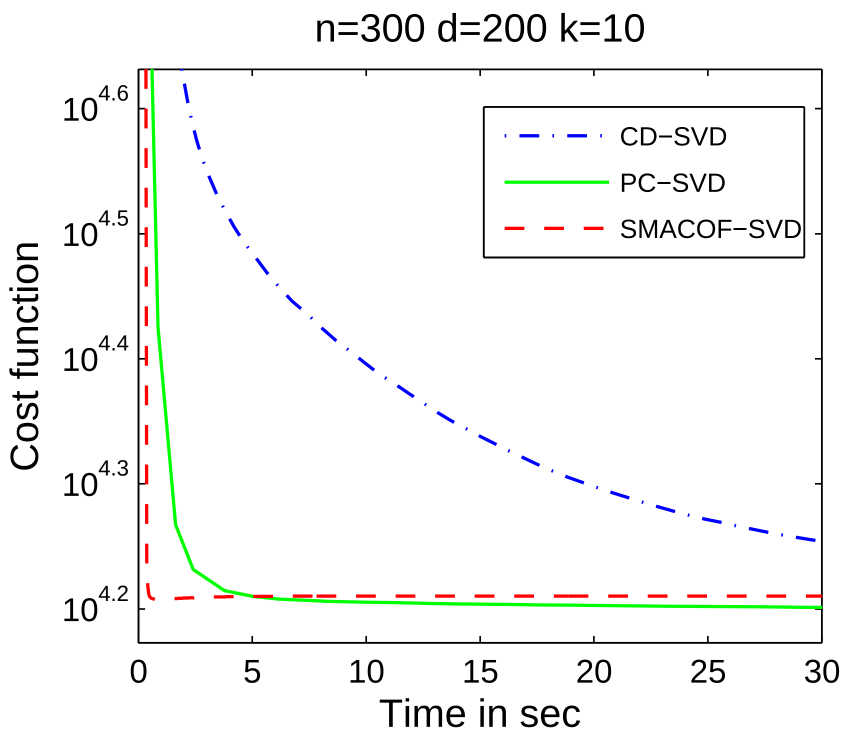


Figure 2.2: rMDS: A typical behavior of the PC, CD and SMACOF for rMDS problem.

changes in this matrix for such large k. I observe that SMACOF is the fastest method to reach a low cost, but does not converge to the lowest cost value. The reason it achieves a cost close to that of PLACECENTER is that for this type of data the rMDS and fMDS cost functions are fairly similar.

In Figure 2.4 I evaluate the effect of changing the amount of noise added to the input distance matrix D, as described above. I consider two variants of the CD algorithm, one where it is seeded with an SVD-based seed (marked CD+SVD) and one where it is seeded with a random projection to a k-dimensional subspace (marked CD+rand). In both cases the plots show the results of the REE algorithm after SVD-type projections back to a k-dimensional space.

The CD+SVD technique consistently behaves poorly and does not improve with

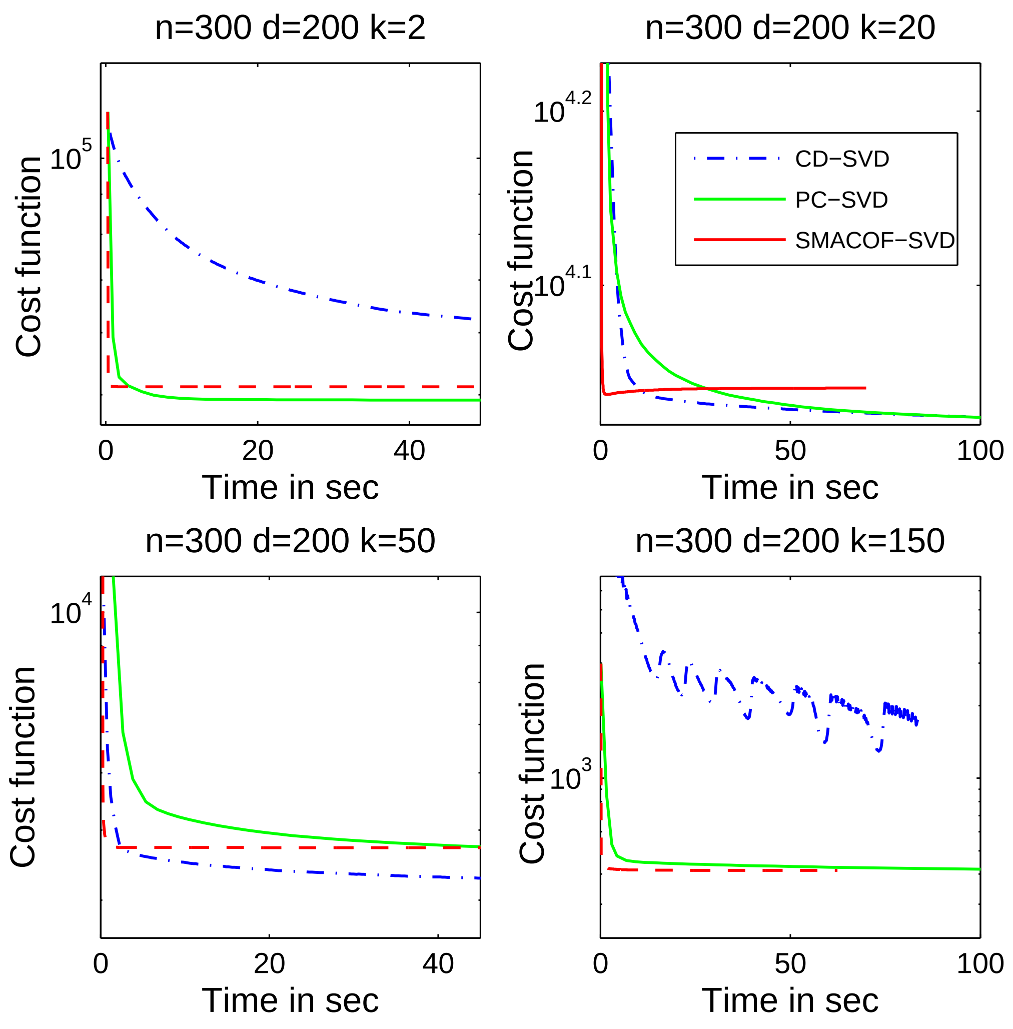


Figure 2.3: rMDS: Variation with $k = 2, 20, 50, 150$.

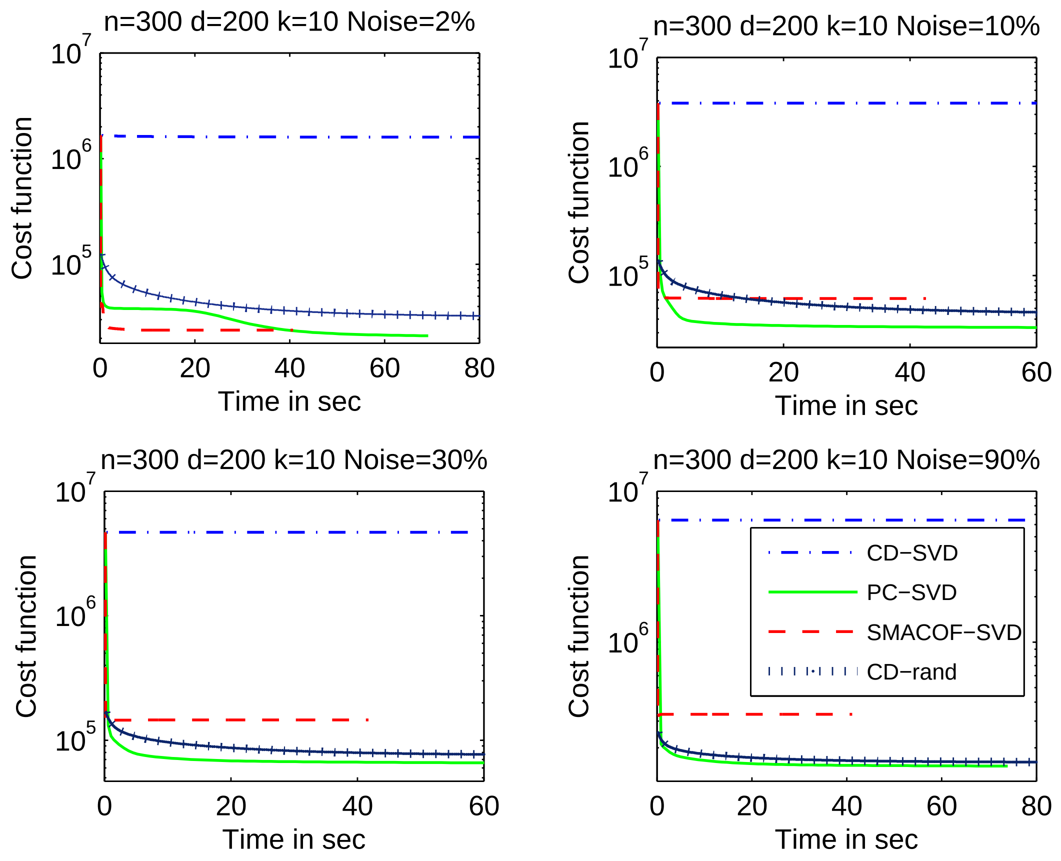


Figure 2.4: rMDS: Variation with noise= $2, 10, 30, 90$.

further iterations. This probably is because the REE component finds the closest Euclidean distance matrix which may correspond to points in a much high dimensional space, after which it is difficult for the SVD to help. The CD+rand approach does much better, likely because the random projection initializes the procedure in a reasonably low dimensional space so REE can find a relatively low dimension Euclidean distance matrix that is nearby. SMACOF is again the fastest algorithm, but with more noise, the difference between fMDS and rMDS is larger, and thus SMACOF converges to a configuration with much higher cost than PLACECENTER. I reiterate that PLACECENTER consistently converges to the lowest cost solution among the different methods, and consistently is either the fastest or is comparable to the fastest algorithm. I will see this trend repeated with other cost measures as well.

2.5.2 The fMDS Problem

I next evaluate the algorithms PlaceCenter, SMACOF, and CD under the fMDS distance measure. The results are very similar to the rMDS case except now both SMACOF and PlaceCenter optimizing the correct distance measure and converge to the same local minimum. SMACOF is still slightly faster that PlaceCenter, but since they both run very fast, the difference is of the order of less than a second even in the very worst part of the cost/time tradeoff curve shown in Figure 2.5. Note that CD performs poorly under this cost function here except when $k = 50$. For smaller values of k, the SVD step does not optimize the correct distance and for larger k the REE part is likely finding an inherently very high dimensional Euclidean distance matrix, making the SVD projection very noisy.

For the fMDS measure, SMACOF and PlaceCenter perform very similarly under different levels of noise, both converging to similar cost functions with SMACOF running a bit faster, as seen in Figure 2.6. CD consistently runs slower and converges to a higher cost solution.

2.5.3 The r^2MDS Problem

In this setting I would expect CD to perform consistently as well as PlaceCenter because both minimize the same cost function. However, this is not always the case

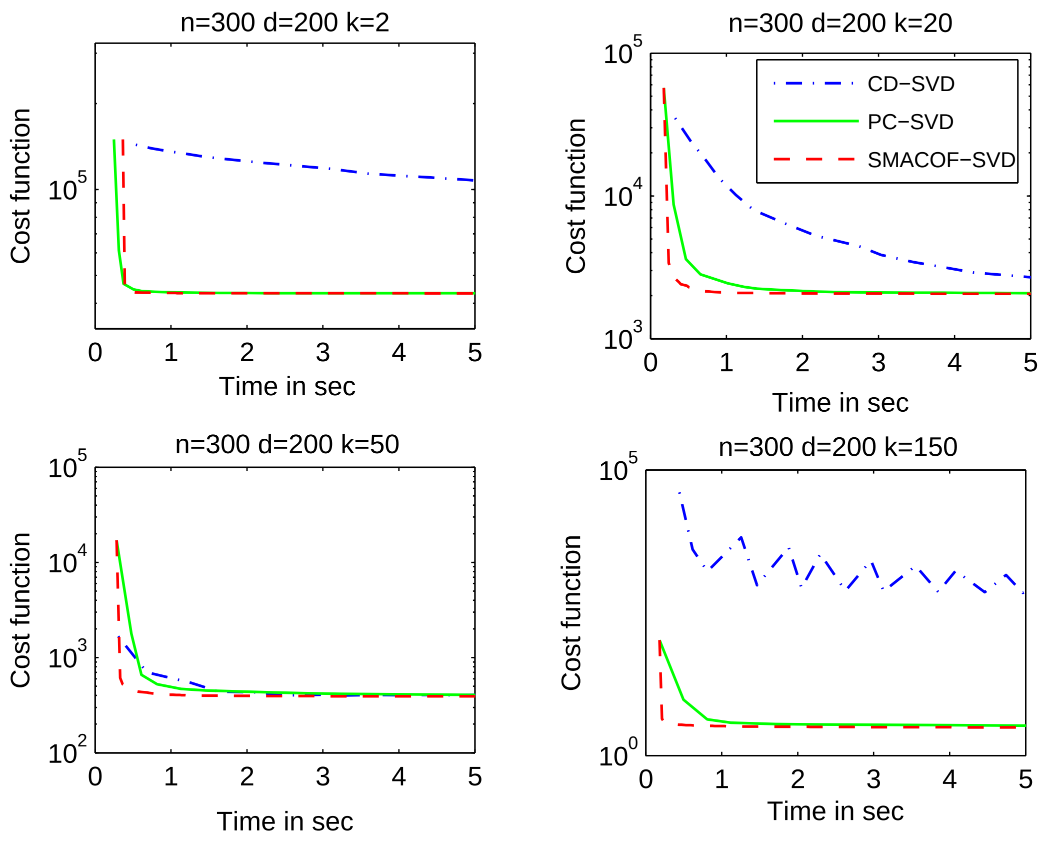


Figure 2.5: fMDS: Variation with $k = 2, 20, 50, 150$

because CD requires the SVD step to generate a point set in $\mathbb{R}^k$. As seen in Figure 2.7 this becomes a problem when k is small ($k = 2, 10$). For medium values of k, CD converges slightly faster than PlaceCenter and sometimes to a slightly lower cost solution, but again for large k ($= 150$), the REE part has trouble handling the amount of error and the solution cost oscillates. SMACOF is again consistently the fastest to converge, but unless k is very large (i.e. $k = 150$) then it converges to a significantly worse solution because the fMDS and r^2MDS error functions are different.

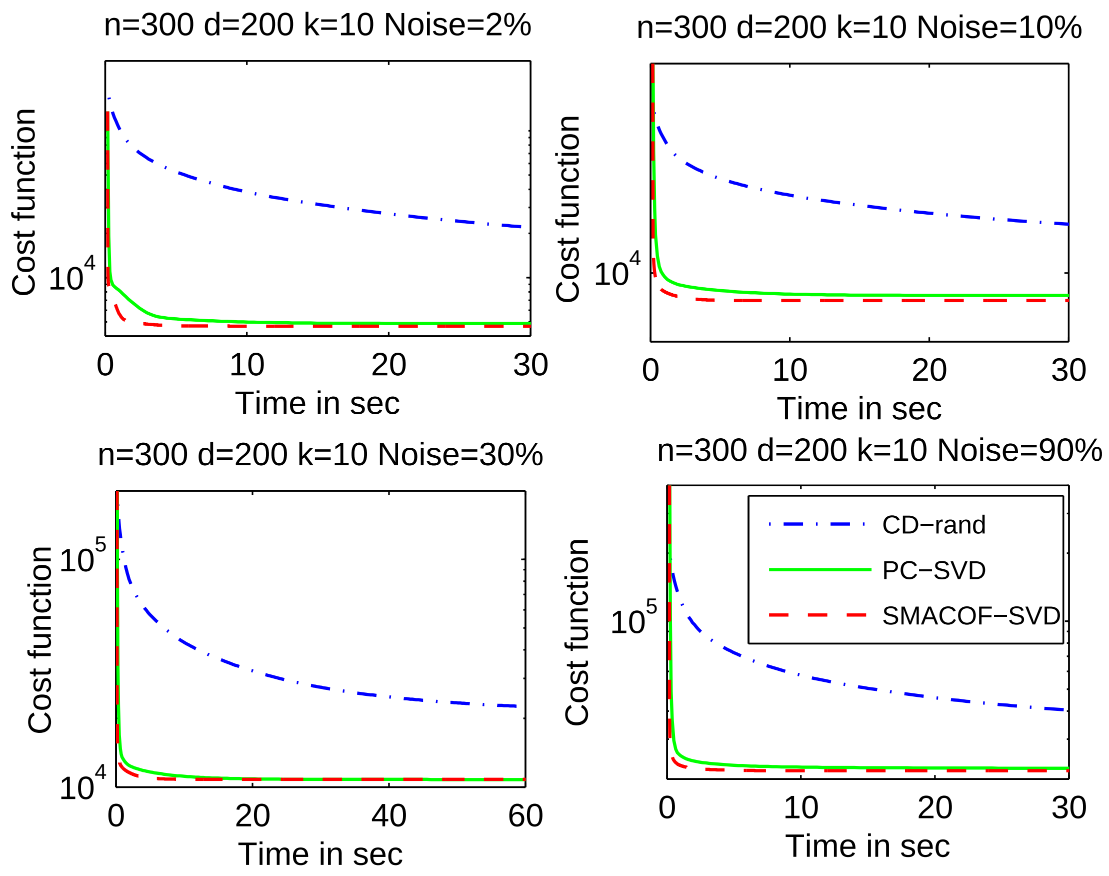


Figure 2.6: fMDS: Variation with noise= $2, 10, 30, 90$.

2.5.4 Processing Large Data Sets

As mentioned in Section 4.2, the memory footprint of PlaceCenter is linear in the number of points. I ran PlaceCenter for fMDS and compared it to SMACOF and CD (Figure 2.8). Both SMACOF and CD fail to run after $n = 5000$ because they run out of memory, while the performance of PlaceCenter scales fairly smoothly even up to $50,000$ points. Before $n = 5000$, SMACOF performs quite well, but the performance of CD starts deteriorating rapidly.

I avoid storing the full $O(n^2)$-sized distance matrix D, by recomputing each

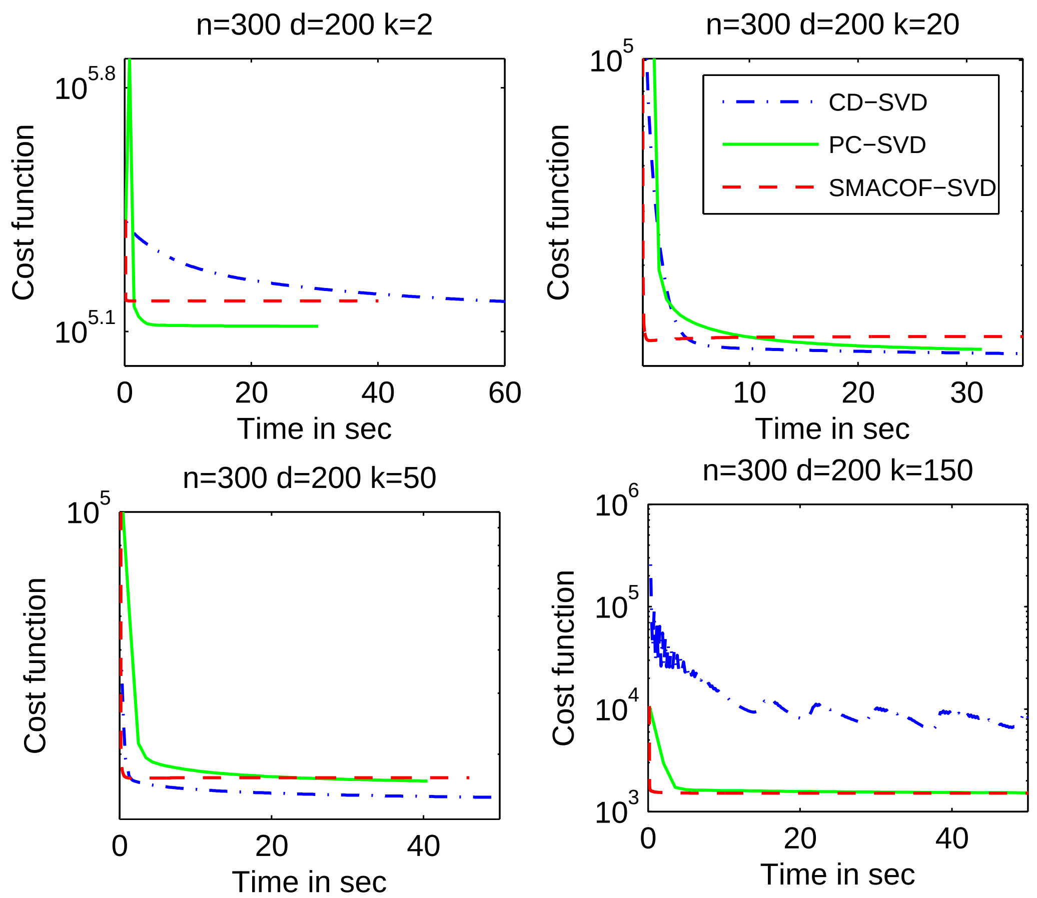


Figure 2.7: r^2MDS: Variation with $k = 2, 20, 50, 150$

distance $d_{i,j}$ as needed. Thus I only store the original point set, which has size $O(nd)$. This approach works for any data set where distances $d_{i,j}$ can be quickly recomputed, not just Euclidean data in $\mathbb{R}^d$ for moderate d. Alternatively, we could have read distances from disk for the point currently being processed instead of recomputing them on the fly. Note, I also seed all algorithms with a random projection instead of cMDS since cMDS also has a memory bottleneck of around $n = 5000$.

These results indicate that my method can be effective on larger data sets. In Chapter 4, I will extend the PLACECENTER for large scale data by using sampling, and compare it other scalable MDS methods like FastMap[Faloutsos and Lin, 1995], Metric

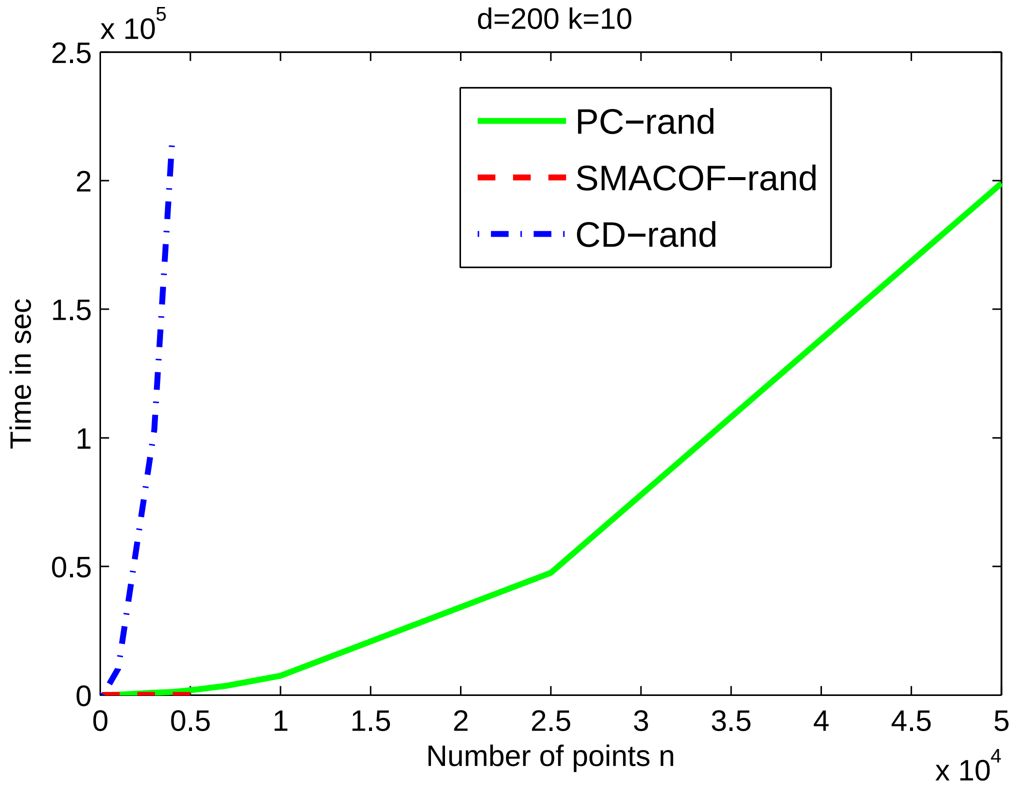


Figure 2.8: The behavior of PC, SMACOF and CD for large values of n. The curves for CD and SMACOF terminate around $n = 5000$ because of memory limitations.

Map[Wang et al., 1999], Landmark MDS[Silva and Tenenbaum, 2003] and Pivot-MDS[Brandes and Pich, 2007] that sacrifice quality (by reducing the number of "pivot" or "landmark" points supplied to the MDS routine) for speed.

2.5.5 Summary of Results

In summary, here are the main conclusions that can be drawn from this experimental study. Firstly, PLACECENTER is consistently among the top performing methods, regardless of the choice of cost function, the nature of the input, or the level of noise in the problem. Occasionally, other methods will converge faster, but will not in general return a better quality answer, and different methods have much more variable behavior with changing inputs and noise levels.

Chapter 3

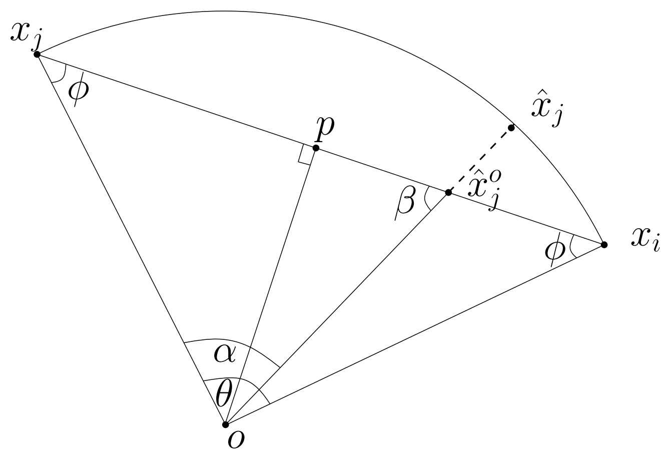
Spherical Multidimensional Scaling

In this chapter, I solve an important variant of MDS called *spherical* MDS, using the generalized MDS framework PLACECENTER presented in the previous chapter. Spherical MDS is the problem of embedding a matrix of distances onto a k-dimensional sphere for a given k. Spherical MDS has applications in texture mapping and image analysis [Bronstein et al., 2008], and is a generalization of the *spherical dimensionality reduction* problem, where the goal is to map points from a high dimensional sphere onto a low-dimensional sphere.

Many techniques have been proposed for performing spherical MDS. Among them are majorization methods [Pietersz and Groenen, 2004] and SMACOF-Q [de Leeuw and Mair, 2009]), a multiresolution approach due to Elad, Keller and Kimmel [Elad et al., 2005] and an approach based on computing the classical MDS and re-normalizing [Pless and Simon, 2002]. My generalized MDS framework presented in the previous chapter applies directly to this setting, where for the local improvement step I adapt a technique first developed by Karcher for finding geodesic means on a manifold [Karcher, 1977].

The spherical MDS problem is closely related to dimensionality reduction for finite dimensional distributions. A well-known isometric embedding takes a distribution represented as a point on the d-dimensional simplex to the d-dimensional sphere while preserving the Hellinger distance between distributions [Kass and Vos, 1997]. A spherical dimensionality reduction result is an important step to representing high dimensional distributions in a lower-dimensional space of distributions, and will have considerable impact in domains that represent data natively as histograms or distributions, such as in document processing [Pereira et al., 1993; Joachims, 2002; Blei et al., 2003; Lebanon and Lafferty, 2004], image analysis [Lowe, 2004; Dalal and Triggs, 2005] and speech recognition [Gray et al., 1980].

Spherical embeddings that guarantee bounded error. The Johnson-Lindenstrauss lemma (discussed in the previous chapter (Chapter 2)) can be extended to data lying on manifolds. Any manifold M with “linearization dimension” k (a measure of its complexity) can be embedded into a $O((1/\varepsilon^2)k\log(kn))$ dimensional space so that all pairwise *Euclidean* distances between points on M are distorted by at most a relative $(1+\varepsilon)$-factor [Agarwal et al., 2007; Sarlós, 2006; Magen, 2002]. A k-dimensional sphere has linearization dimension $O(k)$, so this bound applies directly for preserving the chordal (i.e Euclidean) distance between points on a sphere, however it does not directly apply to geodesic distances. The geodesic distance between points on a sphere can be interpreted as the angle between the points in radians, and a result by Magen [2002] show that $O((1/\varepsilon^2)\log n)$ dimensions preserve angles to within a relative factor of $1+\varepsilon$. In this work, we prove a Johnson-Lindenstrauss-type result for the sphere; namely, that n points lying on a d-dimensional sphere can be embedded on a $O((1/\varepsilon^2)\log n)$-dimensional sphere while approximately preserving the geodesic distances between pairs of points, that is, no distance changes by more than a relative



$$\theta = \cos^{-1}\frac{\langle x_i, x_j\rangle}{\zeta^2}$$

$$\alpha = \frac{r_j}{\zeta}$$

$$\phi = \pi/2 - \theta/2$$

$$\begin{aligned}\beta &= \pi - \phi - \alpha\\ &= \pi/2 - (\alpha - \theta/2))\end{aligned}$$

Figure 3.1: A pictorial representation of how to compute the point intersecting with the geodesic ray on the sphere.

$(1+\varepsilon)$-factor. This latter result is stronger than the results of Magen [2002], and can be seen as complementary to the local improvement scheme; the formal embedding result guarantees the error while being forced to use $\log n$ dimensions, while the local improvement strategy generates a mapping into any k dimensional hypersphere but provides no formal guarantees on the error.

3.1 Algorithm

I use the generalized MDS framework from Chapter 2 to solve the spherical MDS problem with the specific implementations of Place and Recenter. Recall that in generalized MDS framework, Place determines how to move points, while Recenter computes the mean of points obtained from the Place step which is equivalent to solving the min-sum problem. For spherical MDS, implementation of Place is easy. Rather than drawing spheres around each x_j, we draw *geodesic spheres*, which are the set of points at a fixed *geodesic* distance from x_j. On the sphere, this set of points can be easily described as the intersection of an appropriately chosen halfplane with the sphere. Next, instead of computing the intersection of this geodesic sphere with the ray from x_j towards the current estimate of x_i, we compute the intersection with a *geodesic ray* from x_j towards x_i. This intersection can be found using simple trigonometric operations. Let $\hat{x}_j$ be the intersecting point of *geodesic ray* connecting x_j and x_i with the sphere of radius of r_j around x_j then $\hat{x}_j$ is given by (refer Figure 3.1):

$$\hat{x}_j = \frac{\zeta \hat{x}_j^o}{\|\hat{x}_j^o\|}$$

where,

$$\hat{x}_j^o = x_j - \eta(x_j - x_i).$$

Here is ζ is the radius of the sphere[1] . Note that η is just the interpolation parameter interpolating between two extreme points x_j and x_i such as for $\eta = 0$, $\hat{x}_j = x_j$ while for $\eta = 1$, $\hat{x}_j = x_i$.

In Figure 3.1, from triangle $\triangle x_j o \hat{x}_j^o$, we know:

$$\frac{\|x_j - \hat{x}_j^o\|_2}{\sin\alpha} = \frac{\zeta}{\sin\beta},$$

and

$$\eta = \frac{\|x_j - \hat{x}_j^o\|_2}{\|x_i - x_j\|_2}$$

[1]This is the *target* sphere where all the points are being embedded, and not to be confused with the sphere around x_j.

$$= \frac{\zeta}{\|x_i - x_j\|_2} \quad \frac{\sin \alpha}{\sin \beta}$$

$$= \frac{\zeta}{\|x_i - x_j\|_2} \quad \frac{\sin \alpha}{\cos\left(\alpha - \frac{\theta}{2}\right)}$$

where α and θ are defined in the Figure 3.1.

The above closed form expression of $\hat{x}_j$ gives us an easy implementation of PLACE, leaving us to implement RECENTER which is not as easy for the spherical space as it is for the Euclidean space and poses some challenges. Consider the case where $\text{Err}(\delta) = \delta^2$, and $f(x, x')$ is given by geodesic distance on the sphere. Now unlike in the case of $\mathbb{R}^k$, simply computing the centroid of the given points will not work here because this centroid will not in general lie on the sphere, however one can force it to lie on the sphere by projecting it. In my experiments, I use this approach – centroid followed by the projection – for the simplicity. Note that this approach no longer guarantees the optimality, and for the optimality, one will have to resort to more principled way of computing this mean e.g., [Karcher, 1977].

For the robust case ($\text{Err}(\delta) = |\delta|$), the Karcher scheme no longer works. For this case, I make similar adaptation but to the Weiszfeld approach i.e., Weiszfeld algorithm followed by the projection on the sphere. Similar to $\text{Err}(\delta) = \delta^2$, such adaptation no longer guarantees convergence but it seems to work well in practice (see experimental results in next section). A more principled approach in this case is [Fletcher et al., 2009].

3.2 Experiments

In this section, I take the spherical MDS problems {c,g}-{1,2}-sMDS, and compare the proposed algorithms for both of these problems against SMACOF-Q, an adaptation of SMACOF to restrict data points to a low-dimensional sphere, and a technique of Elad, Keller and Kimmel [Elad et al., 2005]. It turns out that the Elad *et.al.* approach consistently performs very poorly compared to both other techniques, and so I do not display it in the graphs. SMACOF-Q has a tunable parameter κ, a larger κ implying a stronger emphasis on satisfying spherical constraint, while small value emphasizing on optimizing the cost. Since the solution produced via this procedure may not lie on the sphere, I normalize all points to the sphere after each step for a fair comparison.

Figure 3.2 and Figure 3.3 compares the performance of PLACECENTER (PC) against SMACOF-Q for the g-1-SMDS and the c-2-SMDS problems respectively. For g-1-SMDS, PLACECENTER does not converge as quickly as SMACOF-Q with small κ, but it reaches a better cost value. However, when SMACOF-Q is run with a larger κ, then PLACECENTER runs faster and reaches nearly the same cost value. For my input data, the solution has similar g-1-MDS and c-1-MDS cost. When I compare SMACOF-Q with PLACECENTER under c-2-MDS (Figure 3.3) then for an optimal choice of κ in SMACOF-Q, both PLACECENTER and SMACOF-Q perform very similarly, converging to the same cost function and in about the same time. But for larger choices of κ SMACOF-Q does much worse than PLACECENTER.

In both cases, it is possible to find a value of κ that allows SMACOF-Q to match PLACECENTER. However, this value is different for different settings, and varies from

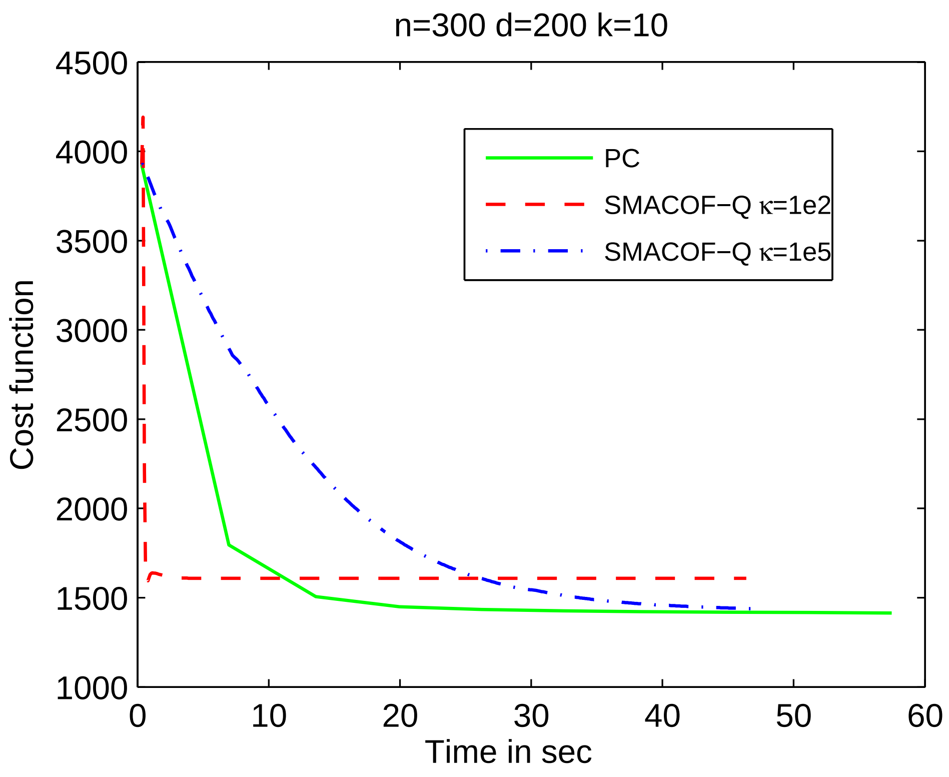


Figure 3.2: Comparing PLACECENTER (PC) with SMACOF-Q for different values of penalty parameter κ for g-1-SMDS

input to input. The key observation here is that since PLACECENTER is *parameter-free* (no hyperparameter except for the convergence criterion ϵ which is needed for both algorithms), it can be run regardless of the choice of input or cost function, and consistently performs well.

3.3 A JL Lemma for Spherical Data

In this section I present a Johnson-Lindenstrauss-style bound for mapping data from a high dimensional sphere to a low-dimensional sphere while preserving the distances to within a multiplicative error of $(1 + \epsilon)$.

Consider a set $Y \subset \mathbb{S}^d \subset \mathbb{R}^{d+1}$ of n points, defining a distance matrix D where

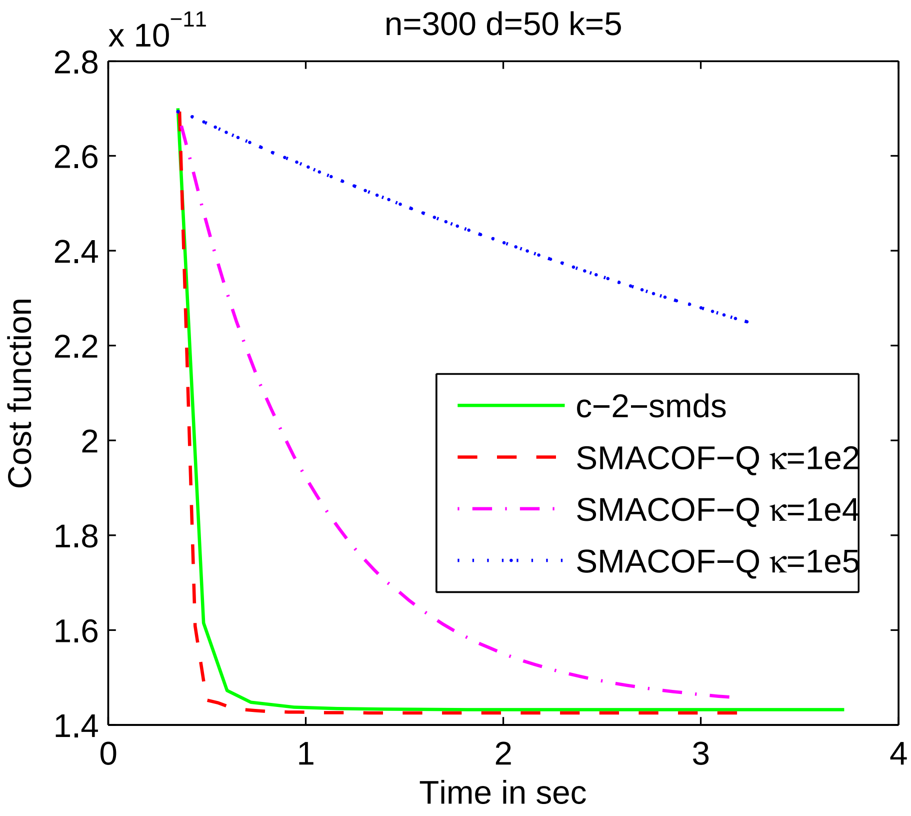


Figure 3.3: Comparing PLACECENTER (PC) with SMACOF-Q for different values of penalty parameter κ for for c-2-SMDS

the element $d_{i,j}$ represents the geodesic distance between y_i and y_j on $\mathbb{S}^k$. I seek an embedding of Y into $\mathbb{S}^d$ that preserves pairwise distances as much as possible. For a set $Y \in \mathbb{S}^d$ and a projection $\pi(Y) = X \subset \mathbb{S}^k$ I say the X has γ-*distortion* from Y if these exists a constant c such that for all $x_i, x_j \in X$

$$(1-\gamma)f(y_i, y_j) \leq cf(x_i, x_j) \leq (1+\gamma)f(y_i, y_j).$$

For a subspace $H = \mathbb{R}^k$, let $\pi_H(Y)$ be the projection of $Y \in \mathbb{R}^d$ onto H and then scaled by d/k. For $X \in \mathbb{R}^k$, let $S(X)$ be the projection to $\mathbb{S}^{k-1}$, that is for all $x \in X$, the corresponding point in $S(X)$ is $x/||x||$.

When $f(y_i, y_j) = ||y_i - y_j||$, and $Y \in \mathbb{R}^d$, then the Johnson-Lindenstrauss (JL) Lemma [Johnson and Lindenstrauss, 1984] says that if $H \subset \mathbb{R}^d$ is a random k-dimensional linear subspace with $k = O((1/\varepsilon^2)\log(n/\delta))$, then $X = \pi_H(Y)$ has ε-distortion from Y with probability at least $1 - \delta$.

I now present the main result of this section. Note that recent results [Agarwal et al., 2007] have shown similar results for point on a variety of manifolds (including spheres) where projections preserve Euclidean distances. I reiterate that my results extend this to geodesic distances on spheres which can be seen as angle $\angle_{x,y}$ between the vectors to points $x, y \in \mathbb{S}^k$. Another recent result [Magen, 2002] shows that $k = O((1/\varepsilon^2)\log(n/\delta))$ dimensions preserves $\sqrt{\varepsilon}$-distortion in angles, which is weaker than the following result.

Theorem 3. *Let $Y \subset \mathbb{S}^d \subset \mathbb{R}^{d+1}$, and let $H = \mathbb{R}^{k+1}$ be a random subspace of $\mathbb{R}^d$ with $k = O((1/\varepsilon^2)log(n/\delta))$ with $\varepsilon \in (0, 1/4]$. Let $f(y_i, y_j)$ measure the geodesic distance on $\mathbb{S}^d$ (or $\mathbb{S}^k$ as appropriate). Then $S(\pi_H(Y))$ has ε-distortion from Y with probability at least* $1 - \delta$.

This implies that if we project n data points that lie on any high-dimensional sphere to a low-dimensional sphere $\mathbb{S}^k$ with $k \sim \log n$, then the pairwise distances are each individually preserved. The proof of the above theorem is given in Appendix B.

3.4 Conclusion

In this chapter, I have shown how PlaceCenter— algorithm developed in the previous chapter — can be used to do the embedding on a sphere. Experimental results show that the PLACECENTER is efficient and effective than the existing algorithms. In addition, I have also proved a formal dimensionality reduction result that embeds a set of n points on a high-dimensional sphere into a sphere of dimension $O((1/\epsilon^2)\log n)$ while preserving all distances to within relative error of $(1+\epsilon)$ for any $\epsilon > 0$.

Chapter 4

Large Scale Distance Embedding

In this chapter, we[1] extend MDS framework proposed in previous chapter for large scale data sets. Recall that MDS is a dimensionality reduction formulation, particularly relevant when the target dimension is small and fixed. In its most general form, the input is given as a set of n^2 distances, and the goal is to embed n associated points into a fixed space (usually $\mathbb{R}^k$) where the average distortion on those distances is small. This problem has been studied [Young and Householder, 1938; Torgerson, 1952; Kruskal, 1964; de Leeuw, 1977; Cox and Cox, 2000; Bronstein et al., 2008; Kruskal and Wish, 1978]; and efficient solutions have been developed [Borg and Groenen, 2005; de Leeuw and Mair, 2009; Agarwal et al., 2010a] for data sets where all n^2 distances fit in memory.

However, as datasets have grown, this assumption that n^2 distances fit in memory is no longer valid; and in fact, in many cases the assumption that n points fit in memory has been violated as well, making many existing MDS algorithms infeasible. This second observation often co-occurs with these n^2 distances not being available directly on input, but rather being represented implicitly by some function on a set of n points P. The most common setting is where each point $p \in P \subset \mathbb{R}^b$ is a d-dimensional feature vector (e.g. SIFT features of images, profile data on customers, attributes of products), and the distance between $p, q \in P$ is defined as $\|p - q\|_2$; thus the input

[1]Part of this work was done in collaboration with another student Chad Brubaker.

can be stored in $O(nd)$ space and each distance can be calculated in $O(d)$ time. A host of other feature distances can be used, and other similar “implicit" problems arise on sparse graphs [Davis and Hu, pear; Buja et al., 2008; Chen and Buja, 2009] using the shortest-path graph distance.

To deal with these large-data[2] versions of the MDS problem two general schemes have arisen – first, landmark-based methods which include Nyström methods such as LMDS [de Silva and Tenenbaum, 2004], MetricMap [Wang et al., 1999], FastMap [Faloutsos and Lin, 1995] and PMDS [Brandes and Pich, 2007] and second greedy methods such as Iterative PCA [Artač et al., 2002]. The Nyström methods select a subset of $\ell \ll n$ landmarks thus reducing the number of rows and columns in a sub-matrix which is used to determine the projection. Second, greedy methods such as Iterative PCA [Artač et al., 2002] (iPCA), inside classical MDS, maintain the best estimate of the top k components of the decomposition, but may suffer from early pruning decisions. By restricting ℓ or k to be constant, these approaches are scalable because they require space that is independent of n, and time that is linear with respect to the data size n; however, as we will reaffirm in experiments of our own, for small values of k and ℓ, these methods are unstable and allow too much error. This inaccuracy comes from two restrictions. First, all of these methods are based on the principal of ”classical MDS“ (spectral methods), that is, they preserve similarities not the distances (hence they solve a different problem). It is well known that classical MDS gives a perfect embedding when distances actually come from a k-dimensional

[2]Here by large data we mean the size of the distance matrix for large n. If this distance matrix is assumed to come from points in some high d-dimensional space, we assume that d is fixed and is relatively small, hence solution is often constrained by n not by d.

Euclidean space, but when they do not, classical MDS or any method based on the same principal performs poorly. This performance degradation increases as one deviates from the k-dimensional Euclidean space i.e. as the difference between the dimension of the source space and the target space increases. In addition to preserving similarities, like classical MDS, these methods perform *linear projections* and thus cannot harness the improved accuracy by allowing *non-linear projections*. Second limitation of the landmark-based methods is that they suffer from potentially poor early decisions (a fixed sample for LMDS, and a initial guess of principal components for iPCA).

Our proposed method (called PLACE-RANDOM) handles large-data MDS while overcoming these restrictions by extending PLACECENTER to work with an adaptive sample, unlike a fixed sample as in LMDS. Our proposed method PLACE-RANDOM not only allows for non-linear projections and inherit all generalizations of PLACECENTER (e.g. for robust MDS [Cayton and Dasgupta, 2006]), but it can refine a fixed size sample over multiple passes of the data, iteratively improving the embedding. Furthermore, the proposed iterative coordinate descent approach of PLACECENTER scales better than iPCA (in terms of d) and FastMap (in terms of k), while all algorithms scale linearly in n. We also demonstrate that PLACE-RANDOM empirically outperforms iPCA and FastMap which will be our two main baselines, in terms of runtime and accuracy on a thorough suite of experiments. These experiments also investigate the effect of noise in the data and the choice of ℓ on the error; notably, the effect of ℓ on the error is independent of n.

4.1 Background And Related Work

Now we give formal definition of the large scale MDS problem, and explore the existing methods used to solve it.

4.1.1 Problem Formulation

Recall that the core MDS problem takes as input an $n \times n$ distance matrix D representing the pairwise distances among a set of n points, and a target space $\mathcal{T} = (\mathcal{X}, f)$; and finds a set of n points $X = \{x_1 \ldots x_n\} \in \mathcal{X}$ such that distances are best preserved. Specifically, the goal is to find X that minimizes the cost:

$$C(X,D) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathsf{Err}\left(f(x_i, x_j - d_{ij})\right). \tag{4.1}$$

d_{ij} is the (i,j) element of the distance matrix D, and $\mathsf{Err}(\delta)$ is a loss function, which typically is $(\delta)^2$ or $|\delta|$ (robust case).

For large-scale data the input is generally presented as a point set $P \subset \mathbb{R}^d$ of size n, a distance function $h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, and a target space $\mathcal{T} = (\mathcal{X}, f)$. In most of the cases, f and h are same, but they do not have to be. The goal is to find a mapping $\mu : P \to \mathcal{X} \subset \mathbb{R}^k$ to preserve the distances $h(p_i, p_j)\ \forall i, i = 1 \ldots n$ as best possible, where typically $k \ll d$. Specifically, the goal is to find a mapping $\mu : P \to \mathcal{X}$ that minimizes the cost

$$C(P, \mu(P)) = \sum_{p_i \in P}\sum_{p_j \in P} \mathrm{Err}\left(f(\mu(p_i), \mu(p_j)) - h(p_i, p_j)\right). \tag{4.2}$$

Here the notation C has been overloaded for the brevity, specific meaning of which should be clear from the context. Similar to the Equation (2.2), we split the cost in (4.2) for individual point:

$$C_i(P, \mu(P), \mu(p_i), p_i) = \sum_{p_j \in P} \mathrm{Err}\left(f(\mu(p_i), \mu(p_j)) - h(p_i, p_j)\right). \tag{4.3}$$

These formulations captures several common instances.

- In Euclidean MDS, f and h are just the Euclidean distance functions, with $\mathcal{X} = \mathbb{R}^k$, and $P \subset \mathbb{R}^d$.
- In Spherical MDS [Elad et al., 2005], $\mathcal{X} = \mathbb{S}^k$, $P \subset \mathbb{S}^d$ where $\mathbb{S}^k$ and $\mathbb{S}^d$ denote the d-dimensional and k-dimensional spheres, respectively. f and h are the (geodesic) distance functions on the sphere, giving the distance along the great circle between two points on the sphere. This generalizes to other explicitly defined non-Euclidean spaces.
- In the robust setting [Cayton and Dasgupta, 2006], the spaces and distances are the same, that is Euclidean space and Euclidean distance, but error is measured as $\mathsf{Err}(\delta) = |\delta|$.
- The input can also be a sparse graph [Davis and Hu, pear; Buja et al., 2008; Chen and Buja, 2009] and the goal is to use one of the above target spaces and errors to embed the shortest-path graph distance between vertices. This distance can be very expensive to compute (often done with Djikstra's algorithm) and should be avoided when possible.

4.1.2 Related Work

In Section 2.3, I described various methods to solve MDS. These methods are efficient and give accurate embedding when n is small i.e. when all n^2 distances can fit in memory, but for large n these methods are not suitable. Among these methods, spectral methods such as classical MDS are admissible and give accurate answer when distances come from a k-dimensional Euclidean space, but when distances do not come from a k-dimensional Euclidean space, SMACOF [Marshall and Olkin, 1979; de Leeuw, 1977; de Leeuw and Mair, 2009] and PLACECENTER—the method proposed in Chapter 4— give fast and accurate embedding. While SMACOF does not assume that the distances come from a k-dimensional Euclidean space but requires n^2 memory to store the distances; PLACECENTER, not only does not make the k-dimensional Euclidean space assumption but takes *linear* memory as well.

When the size of the dataset is so large that one cannot even afford to have a memory requirement that is linear in the size of the data, other methods have been proposed. All known approaches to scaling MDS to large data sets fall into the category of spectral methods — methods based on eigenvalue decomposition, hence they focus on the computational bottleneck of the matrix decomposition step (both in space and time). Incremental PCA (or iPCA) [Artač et al., 2002] maintains the top k principal components incrementally as points are processed one at a time. This procedure has the short-coming that if a key component of the data is not captured early in the process, that component may not be recognized as data is updated.

A more general approach is based on the use of the Nyström approximation technique [Platt, 2005; Zhang et al., 2008]. Here, the idea is to identify (via sampling) rows and columns of the given distance matrix[3] that can be used to define a sub-matrix. This sub-matrix can be decomposed efficiently, and the resulting projection can be extended to apply to the entire data set. Platt [2005] shows that a number of methods for scaling MDS such as LMDS [de Silva and Tenenbaum, 2004], MetricMap [Wang et al., 1999], FastMap [Faloutsos and Lin, 1995] fall into this general framework.

In LMDS, a small set of ℓ *landmark* points L is chosen from the data set. Given this subset $L \subset P$, the full data set can be embedded with respect to a mapping μ_L that is optimized with respect to L. de Silva and Tanenbaum [2004] propose two ways to choose the landmarks: (1) letting L be a fixed random subset of P, and (2) first running an online furthest-point heuristic [Gonzalez, 1995] to choose the set of points L which maximize the minimum pair-wise distance. Zhang *et al.* [2008] recently proposed a third heuristic: (3) run Lloyd's algorithm minimizing the ℓ-mean objective function for 10 rounds and use the ℓ centers as landmarks.

Another Nyström approximation method MetricMap is a modified version of LMDS with the difference that it only uses k (the dimension of the target space) landmarks (instead of ℓ), and these k landmarks are selected by embedding a $2k \times 2k$ distance sub-matrix of the original distance matrix D. The resulting projection from these k landmarks is then applied to embed the rest of points as in LMDS.

FastMap is based on the principal of LMDS but unlike LMDS which finds all of the eigenvectors (or the projection space) by embedding $\ell \times \ell$ submatrix, FastMap

[3]In case when the input is a set of points, this is equivalent to subsampling a set of points and computing the distance among those subset of points.

works in a sequential manner and finds one eigenvector at a time. Specifically, it uses two landmarks (as opposed to ℓ) to project all n points in one dimension $k = 1$, which is equivalent to the projecting along the first eigenvector. In the second round, it computes the residual distances, and finds the second eigenvector by embedding the residual distances using two new landmarks. This process is repeated for k times.

There is another method to deal with large-scale MDS PMDS [Brandes and Pich, 2007] which is also based on landmarks; but unlike the above landmarks based methods, eigen-decomposition step in this method uses the distances from landmark points to all other n points, instead of the pairwise distances just among the landmark points as in LMDS.

Note that all of the existing methods to scaling MDS to large data are spectral methods which means that they perform spectral decomposition which is equivalent to linear projection of the similarity matrix to do the embedding. This projection is contractive (all pair-wise distances can only become shorter, not longer), but is often quite useful as an initial state of μ for the optimization techniques for solving (4.2), at least on small scale problems. This method is especially useful when there is no embedding error i.e. given data can be perfectly embedded into k-dimensional space, because in the absence of noise preserving similarities also preserves distances, as it recovers exact embedding. As we will show in experiments later that these methods based on preserving similarities become less desirable when there is noise in the data, since approximately preserving similarities does not guarantee any approximation on the preservation of distances; the embedding error can be arbitrarily bad.

In our experiments, we will compare the proposed method PLACE-RANDOM with the above methods however FastMap and iPCA will be the two main baselines for the reasons explained later. In our experiments, we do not include MetricMap because both LMDS and FastMap have been shown to outperform MetricMap [Platt, 2005], which we include in our baselines.

Manifold learning. The *nonlinear* nature of the embedding we describe brings to mind the class of methods known as *manifold learning* that try to identify a k-dimensional *manifold* containing (or nearly containing) a set of points in $\mathbb{R}^d$ and then map this data to $\mathbb{R}^k$ to preserve the manifold-based geodesic distance. IsoMap [Tenenbaum et al., 2000] attempts to preserve the global distances by approximating them using the graph distance on the t nearest neighbor graph. Both LLE [Roweis and Saul, 2000] and Laplacian Eigenmaps [Belkin and Niyogi, 2002] attempt to preserve the local manifold structure, in the former case assuming the manifold is locally linear. In all of these approaches, landmark-based methods inspired by the Nyström approach have been used to enable scaling [Silva and Tenenbaum, 2003; Bengio et al., 2004].

A key difference between the problem we solve and these methods is that they attempt to find a mapping to capture the non-linearity *in the data*, often using local distance information only. This local non-linear behavior in the data is captured either explicitly as in LLE and Laplacian Eigenmaps, or implicitly as in ISOMAP by embedding geodesics (geodesics are based on the nearest neighbor graphs). In contrast, MDS in general attempts to preserve global distance structure without assumptions on the underlying structure of the data[4].

[4]Another view: MDS assumes that "straight-line" distance correctly captures the relationship between points, while manifold learning methods preserve a non-Euclidean distance that is approximated using locally-linear assumptions.

Further, a general method for solving MDS actually acts as a *subroutines* for ISOMAP (c.f. [de Silva and Tenenbaum, 2004]). Thus, the proposed method could be used as a subroutine as part of a larger manifold dimensionality reduction method.

4.2 The Algorithm

We start by introducing some notation. Let $X = \mu(P)$ denote the current set of embedded points, and let $x_i = \mu(p_i)$. We maintain a set $Y \subset X$ of ℓ *landmark* points that has preimage L (L is the subset of P such that $\mu(L) = Y$). At every step, the algorithm attempts to minimize $C(L, Y)$, or more specifically $C_i(L, Y, x, p_i)$ for some $x \in \mathcal{X}$, as defined in (4.2) and (4.3), respectively.

Algorithm 3 place-random(P, T, ϵ_a)

```
Randomly project $P$ to $X \subset \mathbb{R}^k$
Set $Y$ to be the first $\ell$ points of $X$
for $t = 1$ to $T$ do
  for $i = 1$ to $|X|$ do
    $x_i \leftarrow$ PLACE2$(Y, x_i, L, p_i, \epsilon_a)$ as argmin$_x \tilde{g}_i(x)$
    Possibly replace some $y \in Y$ with $x_i$ (see Section 4.2.2)
  end for
end for
Return $X$
```

Algorithm 4 PLACE2$(Y, x_i, L, p_i, \epsilon_a)$

```
repeat
  $\epsilon \leftarrow \tilde{g}_i(x_i)$
  for $x_j \in Y$ and $p_j \in L$ do
    $r_j = h(p_i, p_j)$
    $\hat{x}_j \leftarrow$ intersection of sphere of radius $r_j$ around $x_j$ with ray from $x_j$ towards $x_i$.
  end for
  $x_i \leftarrow$ RECENTER$(\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\})$.
until $(\epsilon - \tilde{g}_i(x_i) < \epsilon_a)$ {for a fixed threshold $\epsilon_a$}
Return $x_i$.
```

4.2.1 The Basic Algorithm

The algorithm is based on MDS framework PLACECENTER (from Chapter 2) and is presented in Algorithm 3. It starts with an initial embedding of $P \subset \mathbb{R}^d$ to $X \subset \mathbb{R}^k$ and an initial subsets of ℓ landmarks $L \subset P$ usually chosen arbitrarily as the first ℓ points in P, along with their corresponding images in X i.e. $Y = \mu(L)$. The body of the algorithm consists of two nested loops. The first loop repeats T times (in our experiments, we find a small constant, such as $T = 2$, works well), indicating the number of passes made over the full dataset. Each run of the second loop is a pass on the full dataset P that processes each $x_i \in X$. Each step of this second loop, assigns a new position to $x_i \in X$ using just the embedded landmark points Y, and then decides whether or not to keep x_i as a landmark (replacing one of the current landmark points if so; see Section 4.2.2). The assignment of x_i to a new position is done using PLACE2 function (Algorithm 4) which converges towards the optimal position of x_i with respect to just the landmarks L and Y and takes $O(\ell d)$ time. Note that unlike in PLACE where $g_i(x)$ is defined with respect to all n points, $\tilde{g}_i(x)$ in PLACE2 is defined with respect to only landmarks:

$$\tilde{g}_i(x) = \sum_{\substack{x_j \in Y \\ p_j \in L}} \mathrm{Err}\left(f(x, x_j) - h(p_i, p_j) \right)$$

Remarks:

- The algorithm is time and IO efficient. It makes only a constant T passes over the entire dataset avoiding random disk access, and each point only needs to interact with ℓ other points which fit in memory (space is $O(\ell)$). Section 4.3.3 demonstrates that ℓ can be chosen independent of n.

- It is quite general. It can optimize any cost function for which a PLACE2 subroutine exists to place a single point; this includes those in Chapters 2 and 3, or any other routine which optimizes the location of a single point with respect to a landmark set. Furthermore, it can incorporate several strategies to maintain a set of landmark points L (options discussed below in Section 4.2.2).

4.2.2 Constructing Landmark Points

We consider three ways of choosing and maintaining landmark points L. For all variations, we seed the landmark set based on an arbitrary ordering of the points $P = p_1, p_2, \ldots, p_n$; we always choose the first ℓ points $p_1, p_2, \ldots, p_\ell$ as the initial set L. Two variations presented here are based directly on the original Landmark MDS paper [de Silva and Tenenbaum, 2004] with the key difference being that the landmark set is allowed to change. For completeness, we also consider the k-means variant of Zhang et al. [2008].

Random: We use reservoir sampling [Vitter, 1985] to maintain a random sample of points. Each new point x_i is kept with probability $1/i$ and replaces a point from Y chosen uniformly at random. This guarantees that the landmark set Y is a random sample over the points seen so far.

MaxMin: We use a variation of Gonzalez's algorithm [Gonzalez, 1995] to maintain a set of points spread as far apart as possible. We maintain the $\binom{\ell}{2}$ interpoint distances in Y in a priority-queue. After embedding x_i we calculate $\delta = \min_{y \in Y} \|x_i - y\|$, and if δ is greater that the minimum distance $\|y - y'\|$ in the priority queue, we replace one of the points (y or y') in Y with x_i; this process takes $O(\ell \log \ell)$ time per point processed.

k-Means: Without projecting any points, we run 10 passes of Lloyd's algorithm [Lloyd, 1982] to solve an ℓ-means problem. This can be done using only $O(\ell d)$ space on each pass and with $O(\ell d)$ time for each point processed. We use these ℓ points as a fixed set of landmarks (as in [Zhang et al., 2008]). This method requires an extra 10 passes on the data initially, and then we find it behaves similarly to MinMax but without the extra priority queue.

4.2.3 Multiple passes

The main challenge in scaling MDS is dealing with the internal memory space needed to store the data; our algorithm deals with this by enforcing a *streaming* paradigm [Muthukrishnan, 2005]. Each pass IO-efficiently reads each block of data once, and the only other internal memory space used is for the landmarks, whose size is independent of the full data set's size. However, this streaming restriction can increase error. Consider the situation when we make only one pass over the data. It is possible that the initial choice of landmarks is not representative of the entire data set. In this case, trying to embed points based solely on these landmarks will incur large error. While the proposed adaptive approach mitigates this problem by placing landmarks with respect to the current embedding based on all data seen, it cannot change the (bad) embedding for points seen early on in the stream.

Thus, we allow the algorithm to make multiple passes over the data, *while letting the landmark set persist across passes*. This is important, because then the passes are not independent of each other, and the landmark set acts as a succinct representation of all the processing in the previous passes. Note that in the Random framework for generating landmarks, the t^{th} pass retains new points with probability $1/((t-1)n+i)$ to correspond with the point being the $((t-1)n+i)$th point processed.

We shall see in Section 4.3 that two passes ($T=2$) suffices to generate most of the improvement obtained from a multi-pass scheme and so the overhead for multiple passes is small.

We shall also see that our approach does considerably better than the baselines even when one restricts to two passes. If one cares about the minimal embedding error, then it can further be reduced by increasing the number of passes, something that is not possible with other scalable approaches e.g. iPCA and FastMap. Note that the algorithm is only linear in the number of passes, and more passes does not require any more space, so it provides an easy control for the accuracy/time tradeoff.

It is worth mentioning here that one of our main baselines FastMap require the k number of the passes over the data since. This method can therefore be undesirable for large k or when there is constraint on IO.

[‡]The memory requirement heavily depends on the implementation. We assume that one at least needs to store the 2 landmark points to compute the distances from all other n points.

Table 4.1: Time and space (internal memory) complexities on n points in $\mathbb{R}^d$ to $\mathbb{R}^k$ using ℓ landmarks. In all algorithms, we consider the time to compute distances. As implemented, the number of passes T is constant.

Algorithm	**Space**	**Time**	**Preserved Property**
Classical MDS [Torgerson, 1952]	$O(n^2)$	$O(n^3 + n^2 d)$	Similarities
SMACOF [de Leeuw and Mair, 2009]	$O(n^2)$	$O(n^2 d)$	*Distances*
PLACECENTER [Agarwal et al., 2010a]	$O(nd)$	$O(n^2 d)$	*Distances*
iPCA [Artač et al., 2002]	$O(kd)$	$O(nkd^2)$	Similarities
LMDS [de Silva and Tenenbaum, 2004]	$O(\ell^2 + \ell k)$	$O(n\ell k + nd\ell)$	Similarities
PMDS [Brandes and Pich, 2007]	$O(\ell^2 + \ell k)$	$O(n\ell k + nd\ell)$	Similarities
FastMap [Faloutsos and Lin, 1995]	$O(d)$ ‡	$O(ndk)$	Similarities
MetricMap [Wang et al., 1999]	$O(k^2)$	$O(nk^2 + ndk)$	Similarities
PLACE-RANDOM with Random or k-Means landmarks	$O(d\ell)$	$O(nd\ell)$	*Distances*
PLACE-RANDOM with MaxMin landmarks	$O(d\ell + \ell^2)$	$O(n(d\ell + \ell \log \ell))$	*Distances*

4.2.4 Complexity Analysis

We now provide a runtime analysis for a problem with n points in a d-dimensional space being projected to a k dimensional space. We run Algorithm 3 for T passes (a user specified constant) using a landmark set of size ℓ. Each of the schemes will spend time Tn on the outer two loops, with the only difference being in the complexity of the inner loop. Each call of PLACE2 runs for a constant number of steps in time $O(d\ell)$, so the inner loop for the Random and k-Means schemes is $O(d\ell)$. The MinMax approach takes $\ell \log \ell$ time in addition because of the queue manipulations needed to decide whether to retain the current point as a landmark.

We can analyze internal memory space complexity similarly. Random and k-

Means use $O(d\ell)$ space, while MinMax needs $O(\ell^2)$ extra units of space to maintain the priority queue on $\binom{\ell}{2}$ distances. Thus, the space is limited solely by the number of landmarks, and not by the input size n. We demonstrate in Section 4.3.3 that the choice of ℓ should depend only on the error one wishes to achieve, regardless of the size of the data set n. The time and space complexities of other algorithms including that of PLACE-RANDOM is presented in Table 4.1.

4.3 Experiments

In this section, we perform experiments on both artificial and real data sets, and compare our algorithm to various state-of-the-art algorithms.

4.3.1 Experimental Setup

Data sources. We consider both real and artificial data sets to evaluate the performance of the algorithm. Artificial data sets are generated in the following manner. Choose k random basis vectors in d dimensions in order to generate a k-dimensional subspace. Points are then generated in this subspace by selecting k values in the range $[-1, 1]$ and multiplying those by the basis. We then add spherical zero-mean full-dimensional Gaussian noise with standard deviation σ. The algorithm is then run on the generated data with a target dimension of k. Unless otherwise specified, we will use the following values of different data generation parameters $n = 5,000, d = 500, k = 30, \sigma = 0.1$

We also consider two real datasets, both of which have been used by Platt [2005] for the distance embedding problem for a relative study of FastMap, MetricMap and LMDS. These two datasets are:

Text dataset (REUTERS). This data set [Frank and Asuncion, 2010] is collection of Reuters new articles, consisting of a total of $9,117$ documents, each document is a $19,955$ dimensional feature vector. The feature vector is obtained by preprocessing the document - treating a document as a bag of words, removing stop words, stemming and turning the remaining words into their lower case. Each document is represented by tf-idf vector representation. For this data set, we use the following distance function

$$d_{ij} = h(p_i, p_j) = \cos^{-1}(\langle p_i, p_j \rangle)$$

Corel dataset. This data set [Porkaew et al., 1999; Platt, 2005; Frank and Asuncion, 2010] is a collection of images, each image consists of four types of feature: Color Histogram, Color Layout, Color Moments, and Co-occurrence Matrix Texture. This dataset has been used in the image retrieval and is suitable for the distance embedding (or dimensionality reduction task). The dataset contains $68,040$ images, each image has a total of 89 features: 32 color histogram, 32 color layout histogram, 9 color moments and 16 texture features. Each of these features is continuous. We exclude the images with missing features which leaves us a total of $66,615$ images. Following [Platt, 2005], we computed the distance between two images by first computing the distance between each set of features. For the color histogram and color layout histogram, we used the chi-squared distance defined as:

$$d_{ij}^{chi} = \sum_m \frac{2(p_{im} - p_{jm})^2}{p_{im} + p_{jm}}$$

where p_{im} is the m_{th} histogram bin of the i_{th} image.

For the other two set of features, color moments and texture features, we used the standard euclidean distance:

$$d_{ij}^{e} = \sqrt{\sum_m \left(p_{im} - p_{jm}\right)^2}$$

Now the total distances between two images can be computed using a weighted average of the above four distance where the weights are computed so that average of each of the four distances is unity across the database.

$$d_{ij} = \frac{d_{ij}^{colorHist}}{2.457} + \frac{d_{ij}^{layout}}{2.006} + \frac{d_{ij}^{moments}}{4.295} + \frac{d_{ij}^{texture}}{7.212}$$

Baseline algorithms. In our experiments, we considered the following algorithms for the comparison, however, only two of them were considered throughout all experiments while others were discarded for various reasons explained below. Following are the reason for including (or not) the baselines.

LMDS [de Silva and Tenenbaum, 2004] This algorithm was discarded because of its poor embedding error. Although LMDS finds an embedding mostly faster than PLACE-RANDOM, such an embedding is meaningless because this embedding turns out to be worse than the random — an random set of points in $\mathbb{R}^k$.

PMDS [Brandes and Pich, 2007] This algorithm was also discarded for the same reason as LMDS.

iPCA [Artač et al., 2002] This algorithm is included. Although we include this in our experiments, time complexity of this algorithm is square in the number of dimensions $O(d^2)$ which makes it infeasible for the datasets with high d, as will be see later in the experiments that because of higher d, this algorithm could not be run on the *text* data.

Fast Map [Faloutsos and Lin, 1995] we include this in our experiments and is main baseline across all experiments.

Metric Map [Wang et al., 1999] we do not include this in our experiments because this has been shown to perform worse than FastMap and LMDS [Platt, 2005]

Implementation details. While MATLAB is a standard implementation platform for many of the methods we examine, it does not handle larger data sets (or larger memory requirement) very well. Therefore, we used the `NumPy`[5] Python platform for scientific computing. NumPy has highly optimized inline C/C++ and Fortran routines for the matrix manipulations that the methods we compare against use. For iPCA, we used Micha Kaflon's Python implementation [Kalfon, 2010]. We implemented PLACE2 using inline C, and used a Python wrapper around it to drive the sampling strategies and multi-pass method. The Python code interfaces with the inline C code using the `weave`[6] sub-package of `SciPy`[7] (the generic scientific computing framework

[5]`http://numpy.scipy.org/`
[6]`http://scipy.org/Weave`
[7]`http://www.scipy.org/SciPy`

for Python). While it would have been possible to implement our algorithm in “pure” Python (and we tried it), the initial experiments showed that this was at least two orders of magnitude slower than using inline C. Similarly, for PMDS and LMDS as well, we used our own python implementation with inline C using `weave` wherever possible. For FastMap, we used Gunnar Aastrand Grimnes’s python implementation [8].

Evaluation criteria and parameter choices. We measure error as the average scaled error in distance over all pairs of points:

$$error = \frac{1}{n}\sqrt{\left(d_{ij} - sf(x_i, x_j)\right)^2}$$

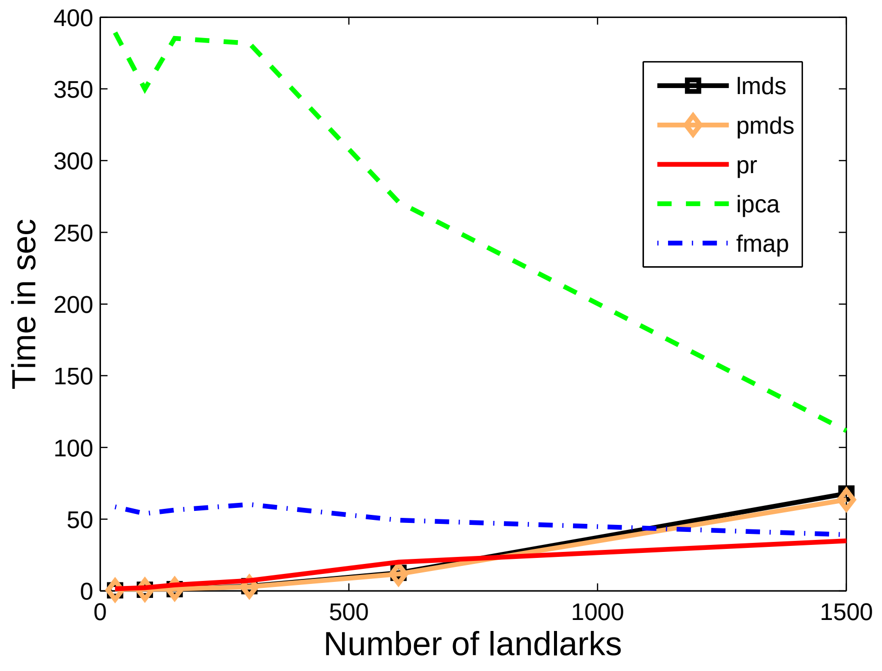
where s is the scaling factor that minimizes the above distortion measure, and can be computed as:

$$s = \frac{\sum_{i,j=1}^{n} f(x_i, x_j)/d_{ij}}{\sum_{i,j=1}^{n} f(x_i, x_j)^2/d_{ij}^2}$$

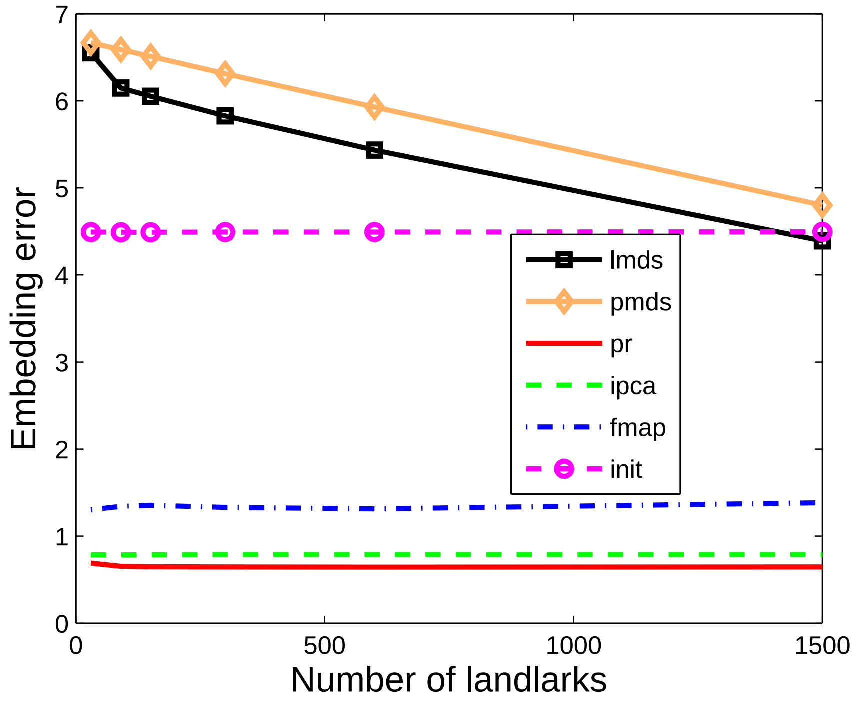
Note that computing $C(P, \mu(P))$ often takes significantly longer than the actual run of our MDS algorithm, therefore we approximate it by computing it over a small sample of n data points (usually $1,000$). We compute this error 5 times and report the average. Time is measured in wall clock time and does not include the time to compute the error. Note that all methods are implemented using the same Python framework, so the timing comparisons are meaningful.

The algorithm (referred to as PLACE-RANDOM) behavior is controlled by a few

[8] `http://gromgull.net/2009/08/fastmap.py`



(a) Time



(b) Error

Figure 4.1: A typical behavior of PLACE-RANDOM and other large scale dimensionality reduction baselines

key parameters: the number of landmarks (default $\ell = 500$), the number of passes (default $T = 2$), and the strategy for landmarks (default Random).

A typical behavior of different algorithms is shown in Figure 4.1(b) (Error) and Figure 4.1(a) (Time). From these plots, we notice the poor performance of LMDS and PMDS in terms of embedding error hence both of these are excluded from any further consideration. Note that these results are even worse than the random – when one chooses a set of points randomly in a k-dimensional space. Although both PMDS and LMDS are faster than PLACE-RANDOM, but such timing results are meaningless because of their bad embedding. During our experiments, we found that both of these methods perform well when number of landmarks is close to n, and there is no embedding error (data is assumed to come from a Euclidean space), an expected behavior of spectral methods.

4.3.2 Overall Performance

We start our experimental suite with a head-to-head comparison of our algorithm with prior work i.e. with FastMap and iPCA.

Efficiency. If space and time are not a constraint, the best approaches for performing MDS are UMDS[9] and SMACOF. However, SMACOF has severe memory limitations and cannot run beyond $n = 5000$. UMDS runs beyond $n = 50,000$ (see Figure 2.8), but its running time is much larger than the methods seen here.

As discussed in Section 4.3.1 and seen in Figure 4.1(b), the best in class large-

[9]We use UMDS and PLACECENTER interchangeable; both refer to the algorithmic framework presented in Chapter 2.

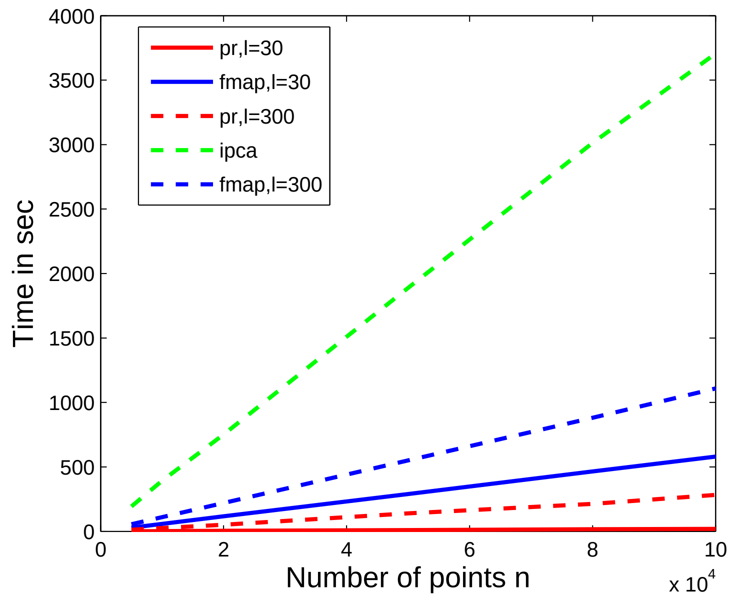


Figure 4.2: Runtimes for iPCA, FastMap and Place-Random from $\mathbb{R}^d$ (for various values of n) to $\mathbb{R}^k$ using $\ell = k = 30$ and $\ell = 10k = 300$.

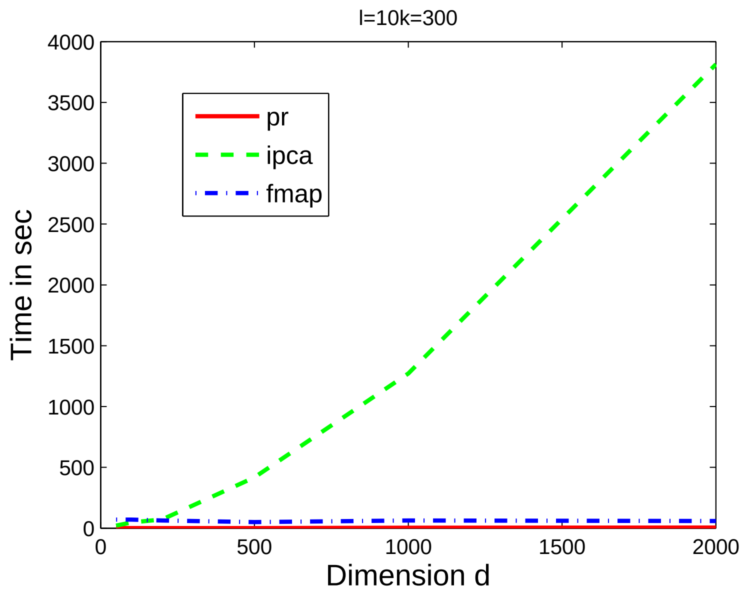


Figure 4.3: Runtimes for iPCA, FastMap and Place-Random from $\mathbb{R}^d$ (for various values of d) to $\mathbb{R}^k$ using $\ell = 10k = 300$. $\ell = k = 30$ gave the similar results and hence excluded from the figure for clarity.

scale methods are represented by iPCA [Artač et al., 2002] and FastMap [Faloutsos and Lin, 1995]. However a direct head-to-head comparison is complicated by the fact that Place-Random takes a parameter ℓ (the number of landmarks) that controls the space usage (and therefore the error) whereas iPCA only stores k eigenvectors when embedding into k dimensions. FastMap does not store any data point, it rather makes k passes over the data. In order to present a clear *time* comparison between the three methods, Figure 4.2 presents results for data generated with up to $n = 100,000$ points, with $k = 30$, and using $\ell = \{30, 300\}$ while iPCA uses 30 eigenvalues. As will be seen in our experiments later, $\ell = 10k$ is usually a good value for ℓ. From this figure, we observe that all three algorithms scale linearly with n, but Place-Random runs faster than both FastMap and iPCA.

For large d, Place-Random clearly outperforms both iPCA because of its squared complexity in d, and also FastMap, as shown in Figure 4.3.

Accuracy. Figure 4.4 shows the average error as sample size varies for Place-Random, FastMap and iPCA (note that both iPCA and FastMap do not have a sample size parameter and is depicted by an almost straight line [10]). The figure shows that the error profile for Place-Random is quite superior to the other methods for small $k = 10, 30$ while for large k iPCA outperforms Place-Random and FastMap. This change in the behavior with respect to the target dimension is expected.

In our data sets, the data is intrinsically k dimensional and the added noise is d dimensional. As we decrease the target dimension, the influence of the noise components can potentially increase, causing the dimensionality reduction to deteriorate in quality. In fact, we see this effect in all three algorithms, where as k decreases, the av-

[10]The line deviates from the straight line because of randomness. Every time we choose a different set of random n points, and error is also measured on a random subset of n points.

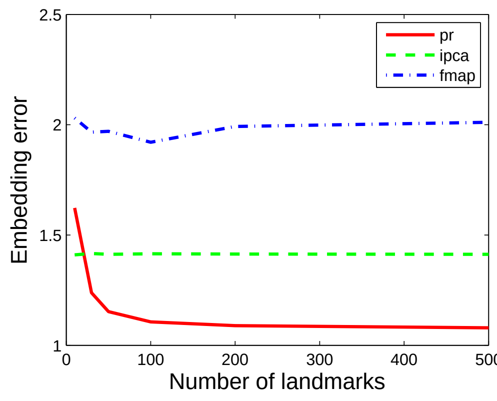
erage error increases. Cayton and Dasgupta [2006] made the point that since spectral methods like iPCA preserve *similarities* and not distances, the two being the same only for a noiseless embedding. When there is no noise in the data i.e. $d = k$), $\widehat{\text{f}}$ull spectral methods such as iPCA, "classical MDS" (the ones not based on landmarks) will give a zero embedding error while it cannot be expected from any landmark based method to give zero error because of the approximation. This figure bears this observation that as the amount of noise reduces or as k goes closer to d, full spectral methods such as iPCA start to outperform the distance based method especially the ones based on landmarks such as PLACE-RANDOM. However this does not discourage one from using PLACE-RANDOM because although for large k, iPCA outperforms PLACE-RANDOM, it would be infeasible to run iPCA if d is large (as will be seen in one of the real data set experiments).

These results indicate that PLACE-RANDOM dominates both FastMap and iPCA from a quality/time stand-point especially for small k.

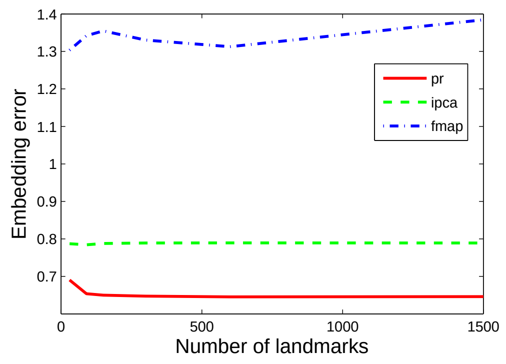
4.3.3 Exploring the Space of Parameters

Error vs sample size. We first consider how the error changes as we adjust the number of landmarks ℓ. Figure 4.5 shows that as ℓ increases, the error decreases with an inverse relationship; following the approximate pattern $0.01 + 1/\ell$. Remarkably, this rate seems to have no dependence on the size of the dataset n, as the curves for $n = \{10K, 30K, 50K, 90K\}$ are indistinguishable.

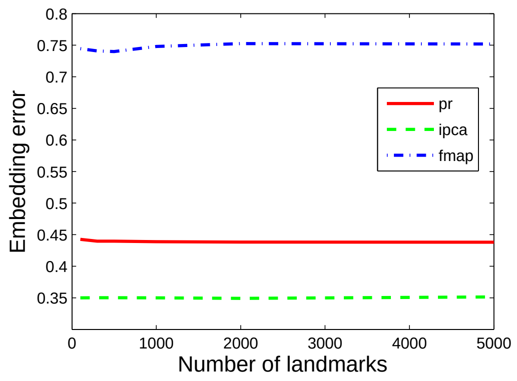
We notice that even for $\ell = 1000$, the error never really vanishes. This can be



(a) $k = 10$



(b) $k = 30$



(c) $k = 100$

Figure 4.4: Error comparison of PLACE-RANDOM with FastMap and iPCA for various values of landmarks ℓ and for different k.

explained by the $\sigma = 0.1$ full dimensional noise added to the synthetic data. Figure 4.6 shows that as σ is decreased, this error approaches 0. This baseline error seems to follow a rate of roughly σ^2 for PLACE-RANDOM (of course with more error with smaller values of ℓ) and $\frac{1}{3}\sigma^2$ for PLACE2. As such, we use $\ell = 500$ for the remainder of the experiments unless otherwise noted since we seem to get diminishing returns at that point. At this size, the error from ℓ (about $1/500 = 0.002$) is significantly less than the expected error due to σ (about $(.1)^2 = 0.01$). As we will see next, the specific constants are not important, but the rates at which these parameters affects the error

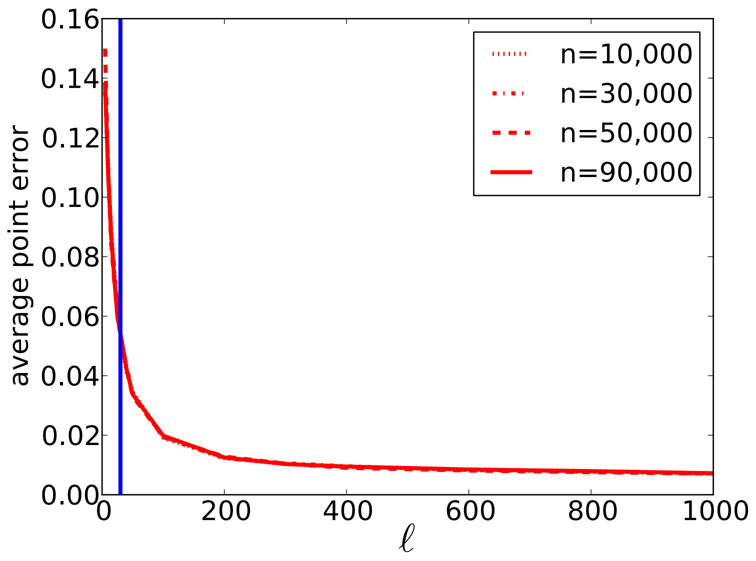


Figure 4.5: Error as ℓ varies from $d = 100$ to $k = 30$. Relationship is unchanged for various n. Vertical blue line shows $\ell = k$.

is informative.

The above experiments always projected from $d = 100$ dimensions to $k = 30$ dimensions. So, finally, in Figure 4.7, we consider varying k and d and plot as a function of ℓ for a fixed $n = 5000$ and $\sigma = .1$. Two observations are apparent: As d shrinks, so does the error; this intuitively is due to the full-dimensional noise added has less room relative to the "true" k-dimensional subspace to distort the distances. Also, as k increases, the error also decreases for the same reason; full-dimensional noise affects the projected distance error more as the difference between d and k increases.

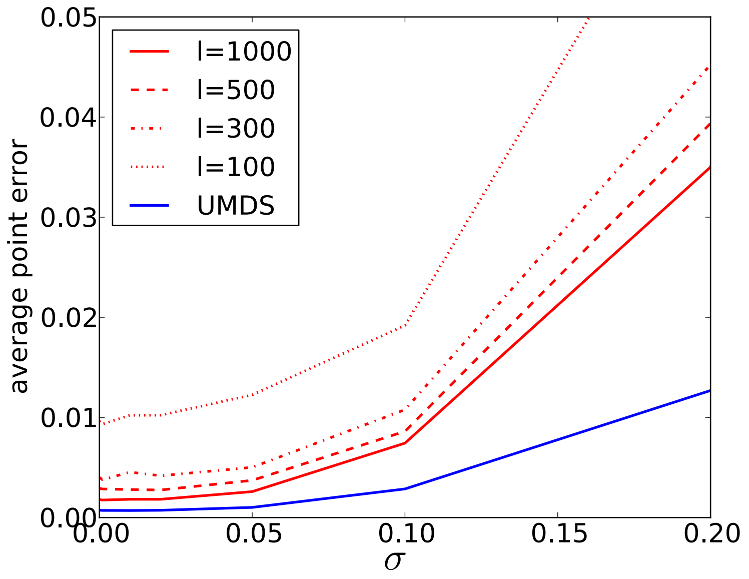


Figure 4.6: Error as σ varies on $n = 10000$ points from $d = 100$ to $k = 30$.

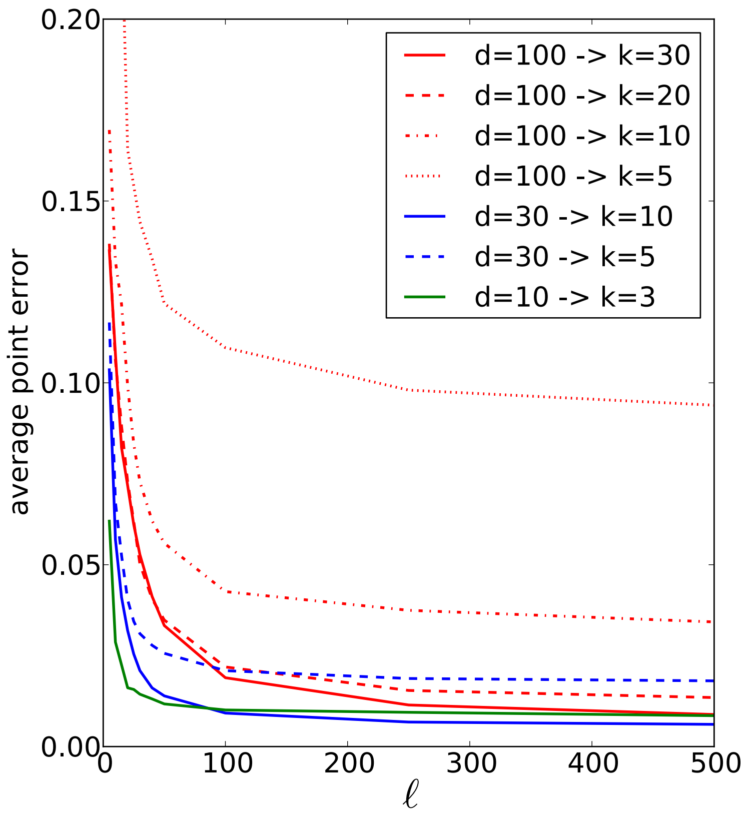


Figure 4.7: Error versus sample size on $n = 5000$ points from $\mathbb{R}^d$ to $\mathbb{R}^k$, for various d and k.

Penalty for sampling. To calibrate the error incurred as a result of sampling, we compare our approach to the original UMDS (PlaceCenter) method that effectively uses the entire point set as landmarks, and runs for many more passes (typically, it converges in 50 "passes" over the input). As Figure 4.6 indicates our algorithm is worse by about a factor 3 in error. But we note (see Table 4.1) that in both space and time, Place-Random is better than UMDS by a factor of n/ℓ, a factor that scales linearly as data increase (in addition to a 25 factor more passes). Thus this error penalty for sampling, which is about on the order of the noise, is not too severe.

Number of passes. Running the algorithm over multiple passes helps "improve" bad initial choices of landmarks. Figure 4.8 illustrates this behavior. We note that while there is steady improvement in error as the number of passes increases, at least half of the improvement is obtained just by running a second pass. This effect is more pronounced as the target dimension increases; notice that at $k = 5$, the improvement is more consistent across passes in comparison to $k = 30$, which shows a steep drop on the second pass. The choice of only $T = 2$ passes in Place-Random seems to account for most of the 3-factor difference in error between Place-Random and UMDS in Figure 4.6. Hence, if one desires a higher accuracy/speed trade-off, running a few more passes is a useful option.

Seeding strategy. We use different seeding strategies to build and maintain the set of landmarks, and a key difference between our strategy and the one adopted by LMDS is the *adaptivity* in our approach. We change the set of landmarks on the fly as new data comes in, depending on either random choice or distance calculations. In

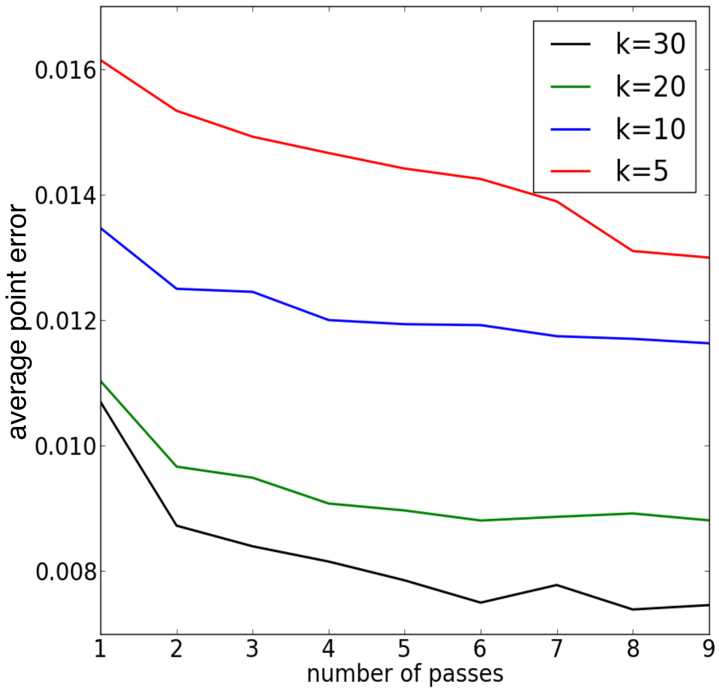
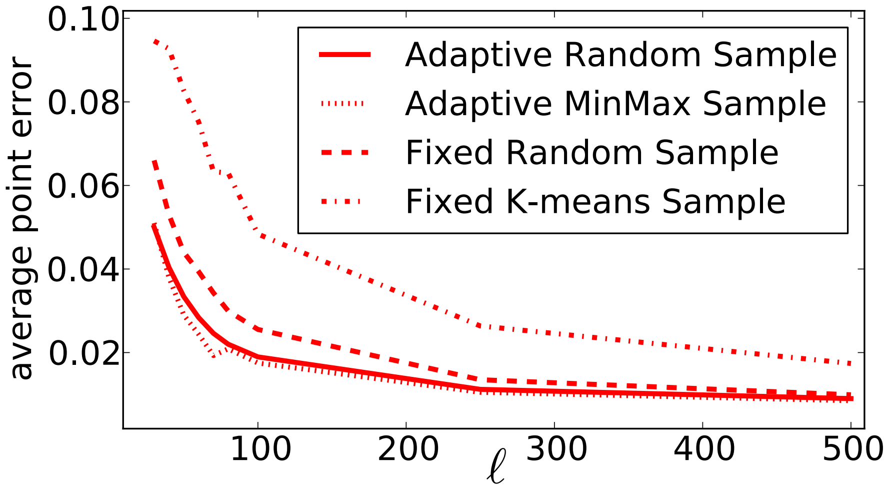


Figure 4.8: Average error versus number of passes for $n = 5000$ points in $d = 100$ dimensions. k is the target dimension, and the sample size $\ell = 500$.

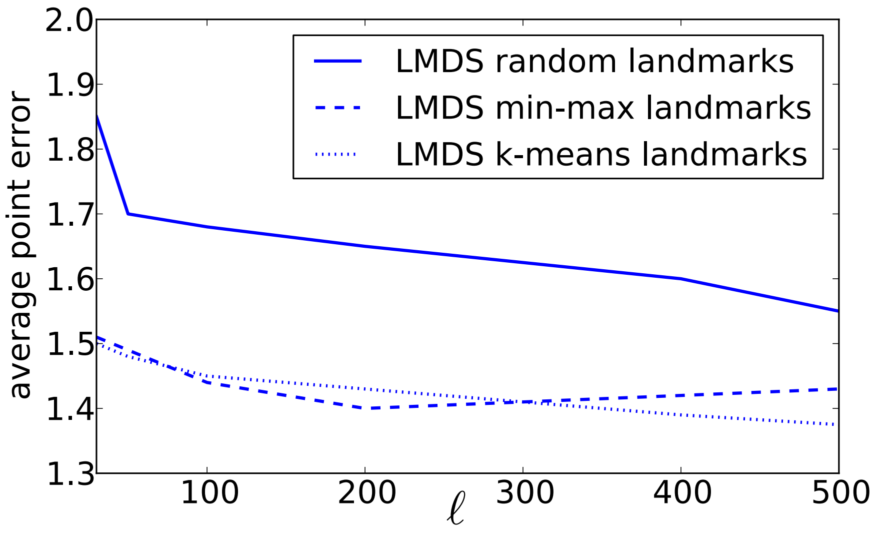
Figure 4.9(a) we examine the error profile generated by different strategies.

Several observations can be gleaned from the figure. First, the difference between the two adaptive seeding strategies (Random and MinMax) is not significant, although the MinMax strategy yields slightly less error. Second, the "adaptivity bonus" is marked; letting the landmarks set change in response to the data improves the error compared to maintaining a fixed landmark set, as LMDS must. Notice on just random samples that we get about the same error (0.02) with $\ell = 100$ for an adaptive sample as we do for $\ell = 200$ with a fixed sample. Furthermore, the k-Means approach (which is not adaptive) is significantly worse than the fixed random sample in this setting.

For another comparison, Figure 4.9(b) shows the effects of the three fixed



(a) Error vs sample/landmarks size for different seeding strategies with $n = 5000$.



(b) Error vs sample/landmarks size for different seeding strategies on LMDS with $n = 5000$.

Figure 4.9: Seeding strategies

sampling strategies on the same set-up but running LMDS. Note that the error is 2 orders of magnitude larger than for PLACE-RANDOM and here the error caused by a non-adaptive is more pronounced compared to the clustering based strategies: MinMax and k-Means.

4.3.4 Real Datasets

Text data. Results for the `text` data are shown in Figure 4.10 (time) and Figure 4.11 (error). From Figure 4.10, we see that time for PLACE-RANDOM increases linearly in both k and ℓ. Although FastMap runs faster than the PLACE-RANDOM for higher ℓ one does not need that many landmarks for PLACE-RANDOM to perform better. In fact, it is seen in Figure 4.11 that PLACE-RANDOM does not benefit from increasing the number of landmarks beyond $\ell = 100$, and for $\ell = 100$, PLACE-RANDOM runs faster than FastMap especially for $k = 10$. For other values of k as well, time taken by FastMap is significantly higher than PLACE-RANDOM. Note that we exclude iPCA results from these plots because iPCA takes $O(d^2)$ times to run which was not feasible for *text* data because of its high dimensionality ($d = 19,955$).

Corel data. In `corel` data, we obtain similar results. From the error plot Figure 4.13, we see that PLACE-RANDOM is better than both iPCA and FastMap for all values of k, while for time (Figure 4.12), PLACE-RANDOM runs almost as fast as FastMap (faster for the smaller ℓ which is sufficient to give a good embedding), while significantly faster than iPCA.

4.4 Discussion

In this chapter, we have presented a new approach for scalable MDS that exhibits a small memory footprint and quickly returns an accurate embedding of the data. Our approach combines adaptive sampling and multi-pass algorithms, built on the

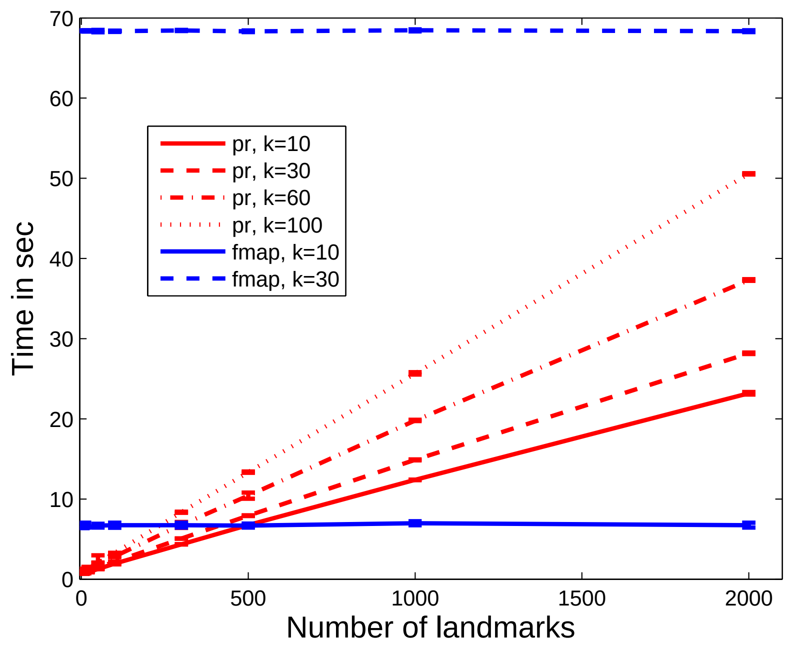


Figure 4.10: Runtimes versus ℓ for PLACE-RANDOM and FastMap for different values of k on the `text` data

PLACECENTER algorithm presented in Chapter 2 which avoids expensive eigenvalue computations and uses a non-linear projection. The approach generalizes to other distance measures and cost metrics, and can run MDS efficiently at data sizes well beyond what has previously been demonstrated for the same accuracy.

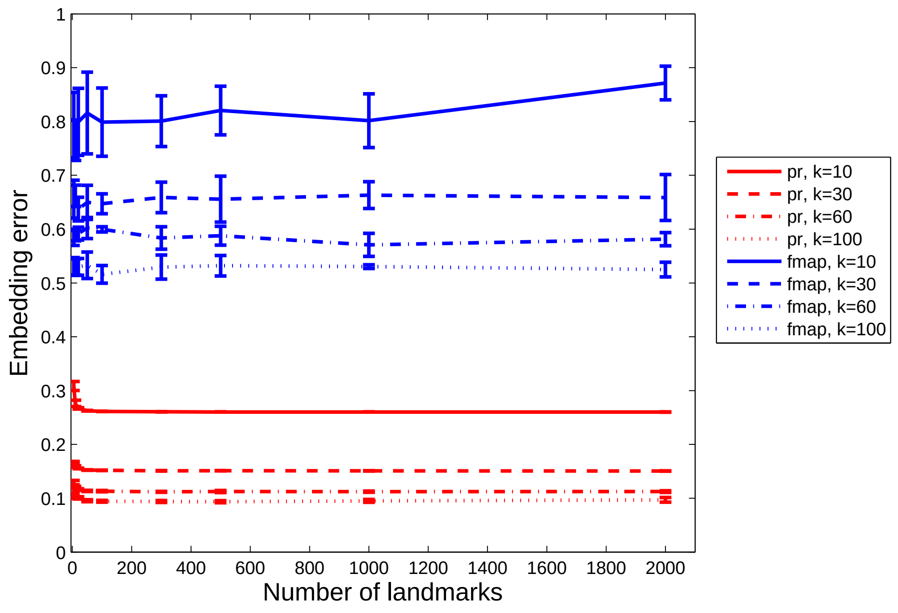


Figure 4.11: Error versus ℓ for PLACE-RANDOM and FastMap for different values of k on the `text` data

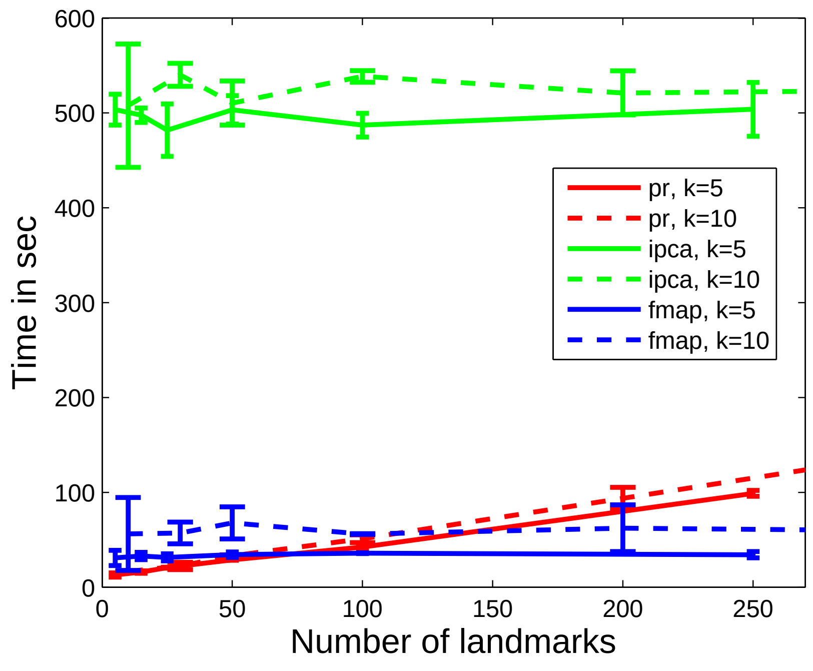


Figure 4.12: Time versus ℓ for PLACE-RANDOM, iPCA and FastMap for different values of k on the `corel` data

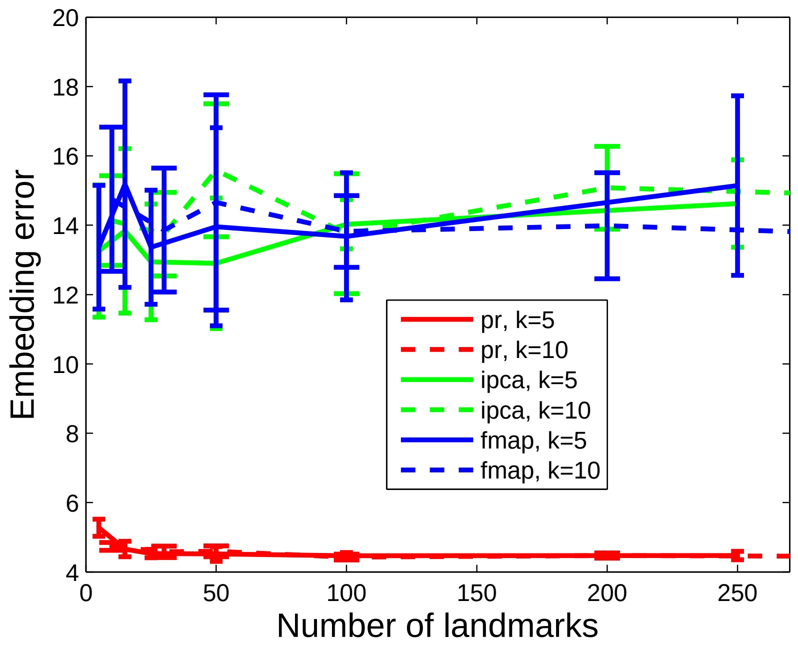


Figure 4.13: Error versus ℓ for PLACE-RANDOM, iPCA and FastMap for different values of k on the `corel` data

Chapter 5

Sensor Network Localization for Mobile Sensors

In this chapter, I modify the proposed MDS framework PLACECENTER to solve a sensor networks localization problem.

With the advancement of computer and wireless technology, wireless sensors are becoming smaller, more computationally capable, and more inexpensive causing wireless sensor networks to be commonplace. Wireless sensors can be effectively used to measure temperature, vibrations, sound, pressure, etc, and has been applied in applications in many areas such as military applications, environment or industrial control and monitoring, wildlife monitoring, and security monitoring; however, in most applications, if the networks are to achieve their purpose, locations of the sensors must be known. Yet, in many networks, locations are given for a few of the sensors (called anchors), and the locations of the remaining sensors must be determined. Although Global Positioning System (GPS) can be used to determine the locations, GPS systems are bulky, consume too much power, are expensive compared to the sensors, and can only be used outdoors. In such scenarios when GPS cannot be used to find the locations, finding locations by other means become important. One way to find the locations of these sensors is to use the intercommunication distances (sensor-sensor and sensor-anchor), and treat the localization problem as a distance embedding problem. Distances between sensors are usually measured by communicating with only nearby sensors, since further communication is limited by sensor power. Such distance measurements provide a sparse distance matrix. Sensor network localization (SNL) [Biswas and Ye, 2004; Biswas et al., 2006; Carter et al., 2006; Tseng, 2007; Pong and Tseng, 2009; Kim et al., 2009; Krislock, 2010] is the problem of determining sensor locations using known anchor locations and (usually noisy) distance measurements (i.e. sparse noisy distance matrix). This is a large, non-convex, constrained optimization problem. Large networks contain many thousands of sensors, whose locations need to be determined accurately and quickly.

In this chapter, I also consider a special case of the SNL problem, when sensors are moving. This moving sensors scenario occurs in many applications e.g., wildlife tracking (animal networks), vehicle tracking etc. In animal networks, sensors are installed on animals to help them track. Although any algorithm that can be used for stationary SNL problem can also be used for moving case, one can do better if it takes into account the movement of the sensors, and is able to exploit the information available along the movement trail. In this work, I present an algorithm that is especially suitable for moving sensors. Although the focus is on moving sensors, there is no reason why one cannot apply the proposed algorithm for stationary case, however in my experiments, I found this algorithm to be more effective for the moving case.

My proposed algorithm is based on the MDS framework PLACECENTER proposed in Chapter 2. Recall that PLACECENTER was proposed to solve a distance embedding problem where all pairwise (sensor-sensor and sensor-anchor pairs) distances are available and there are no anchors. In this chapter, I modify PLACECENTER to exploit the available anchors to position the remaining sensors more accurately, and to handle the sparse and noisy distance measurements. I experiment with the moving scenario for a variety of networks, and show that the proposed algorithm outperforms the existing state-of-art SNL algorithms across almost all experiments. My algorithm comes with many attractive properties:

1. This is a scalable algorithm: it can handle large number of sensors. This is the first ever algorithm that has been tested with number of sensors more than $100,000$ with *noisy* distance measurements. My algorithm can localize $120,000$ sensors in less than 20 minutes.

2. This is a decentralized algorithm, which means that the computation can be carried out in the sensor itself needing only the locations of its neighbors.

3. The proposed algorithm consists of two stages, in which the second is guaranteed to converge, that is, in each iteration, the objective function associated with the corresponding distance embedding problem (see (5.1)) is guaranteed to monotonically decrease. The first stage is a heuristic stage developed to get faster estimation of locations for dense networks.

4. Since this (excluding the heuristic stage) is an iterative algorithm, and always decreases the cost function, it can be used as a post-processing (refinement) step for any solution output by other algorithm.

5. The proposed algorithm is found to be very effective for real-time tracking since it takes only a fraction of second for a tracking problem with 1000 sensors (See results in Section 5.4.5).

6. This algorithm preserves all the properties of the original algorithm PLACECENTER, that is, it can work for different spaces and different cost functions.

5.1 Related Work

In the last decade, there has been a considerable amount of work in stationary SNL, however not much has done for the moving sensors. In this section I discuss some of the recent work for SNL problem, especially for the *large scale noisy* problems. A more comprehensive discussion on the related work can be found in [Dalal and Triggs, 2005; Krislock, 2010].

The literature on solving SNL problems can fit in two broad categories: ones that performs convex relaxation on the cost function (see equation (5.1) below) using semi-definite programming (SDP); and ones that directly minimizes the non-convex cost function using multi-dimensional scaling (MDS)-style algorithms.

In the SDP-type algorithms, one of the most notable works is of Biswas Ye [2004]. They propose a semi-definite relaxation of the SNL problem, now called Biswas-Ye relaxation. Although effective, this relaxation can only be used for small problems with exact distance measurements. Authors later extended this work for larger scale [Biswas and Ye, 2006; Biswas et al., 2008] and noisy problems [Biswas et al., 2006]. This was further extended by Carter et al. [2006] to solve the even larger problems by breaking large problems into a series of small sub-problems, each solved using an SDP.

To handle even larger problem using SDP relaxation, Wang et al. [2008] proposed further relaxation of SNL problem, where unlike the Biswas-Ye relaxation, one does not require the entire distance matrix to be positive semi-definite. Instead, the problem is broken into smaller subproblems, and the semideifinite constraint is enforced only on the subproblems. They proposed two such relaxations edge-based and node-based; the edge-based (ESDP) appears more successful. Although considerably faster than the original Biswas-Ye relaxation, it still takes significant time compared to the other more recent approaches. For $n = 4,000$ sensors (and with noise $\sigma = 0.01$, neighborhood radius $R = 0.035$ in a unit cube) it took more than 10 minutes, which is slower than the some of the recent approaches [Pong and Tseng, 2009; Kim et al., 2009] but better than the SOCP approach of [Tseng, 2007] and Carter et al. [2006]. A noise-aware version of ESDP called ρ-ESDP was proposed by Pong and Tseng [2009] in which they use the Log-barrier Penalty Coordinate Gradient Descent (LPCGD) method to solve the SDP relaxation problem. In my experiments, I have found this to be the best baseline among all competing algorithms both in terms of speed and accuracy. Since the SNL problem typically has sparse input, a sparse version of the Biswas-Ye relaxation is proposed by [Kim et al., 2009] called SFSDP. This allows one to handle the sparse noisy SNL problem efficiently. In their work Kim et al. could handle the problem up to 18,000 sensors and 2000 anchors in less than 10 minutes. Some of the more recent work in SNL is done by Krislock [2010] where smaller SDPs are solved to embed the sensors in a successive manner. This algorithms can handle enormous numbers of sensors (experiments were performed on sensor sets of size up to 100,000), and is remarkably fast and accurate for the exact distance measurements, but when there is a noise in the measurement, algorithms performs very poorly, both in terms of time and accuracy (see results in Section 5.4.3).

In the second category (MDS-style), the work that is most related to this work is by Costa et al. [2006]. They present a distributed algorithm dwMDS which uses iterative refinement, and finds a minimum of the cost function using a majorization algorithm. Although similar to my algorithm, its biggest limitation is that it does not make smart use of available anchors, and thus is not suitable when there are lots of anchors and distance measurements are noisy. Another related work in the MDS-style approaches is done by Moore et al. [2004] which is based on trilateralization. This requires each node to have a degree or 10 or more between any pair of neighbors, or more if the distance measurements are noisy.

5.2 My Approach

Problem 1 (Sensor Network Localization (SNL)). *Consider a graph $G = (V, E, D)$ with vertex set of size $|V| = n$, where D_{ij} is the estimated distance for each edge $(i, j) \in E$, and where there are m anchor vertices $(v_{n-m+1}, \ldots, v_n)$ with known and fixed locations. The SNL problem seeks a mapping $\mu : V \to \mathbb{R}^d$ to minimize*

$$\sum_{(i,j)\in E} \Big(\|\mu(v_i) - \mu(v_j)\|_2 - D_{ij} \Big)^2 \tag{5.1}$$

over all choices of μ that restricts anchor vertices to their known locations $(x_{n-m+1}, \ldots, x_n)$. For notation, we label $X = \{x_i = \mu(v_i)\}$ for all vertices.

I solve SNL problem *directly* without converting it into a related problem and without relaxing any constraints. I propose an algorithm based on PlaceCenter, and call it

Algorithm 5 PLACEMENT($D, X_0, \epsilon_x, \epsilon_d, \sigma, N_a, N_r$)

```
$X \leftarrow$ FindMoreAnchors($D, X_0, \epsilon_x, \epsilon_d, \sigma, N_a$)
for $t = 1$ to $N_r$ do
  for $i = 1$ to $|X|$ do
    $Y = \{x_j : (i,j) \in E\}$
    $S = \{d_{ij} : x_j \in Y\}$
    $x_i \leftarrow$ Place3($Y, S, x_i, t, \epsilon_x$)
  end for
end for
Return $X$
```

PLAce CEnter with Missing ENTries (PLACEMENT). Let $X = \{x_1, x_2 \ldots x_{m-n+1}, \ldots x_n\}$ be a $k \times n$ matrix such that the last m columns of this matrix are anchors. Let X_0 be another $k \times n$ matrix that will be used to initialize X. Let ϵ_x and ϵ_d be terminating parameters of the algorithm, and N_{in}, N_{out} be the number of iterations. Let

$$C_i(x_i, Y, S) = \sum_{(i,j)\in E} \left(\|x - \mu(v_j)\|_2 - D_{ij} \right)^2$$

be the embedding cost of one point v_i at location x where $Y \subset V$ is the set of neighboring of x and $S \subset D$ are the corresponding distances of x to its neighbors U. $C_i(Y, S, x_i)$ is nothing but the cost defined in (2.2) with Err function replaced with the squared cost, and the distances function f replaced with the Euclidean distance, which can be written as:

$$C_i(Y, S, x_i) = \sum_{\substack{x_j \in Y \\ d_{ij} \in S}} \left(\|x_i - x_j\|_2 - d_{ij} \right)^2. \tag{5.2}$$

The main algorithm is presented in Algorithm 5. This algorithm consists of

Algorithm 6 FINDMOREANCHORS$(D, X_0, \epsilon_x, \epsilon_d, \sigma, N_a)$

```
$X \leftarrow X_0$
$isAnchor$ := $|X_0| \times 1$ vector denoting if $x_i$ is an anchor or not.
$OrderIdx$ := ordering of points based on the number of anchors
**for** $t = 1$ **to** $N_a$ **do**
    **for** $u = 1$ **to** $n$ **do**
        $i = orderIdx[u]$
        **if** $isAnchor[i] = false$ **then**
            $Y = \{x_j : (i,j) \in E,$ and $isAnchor[j] = true\}$
            **if** $|Y| \geq k + 2$ **then**
                $S = \{d_{ij} : x_j \in Y\}$
                $ninit = 0$
                **while** $isAnchor[i] \neq true$ & $ninit < 30$ **do**
                    $ninit = ninit + 1$
                    $x_i \leftarrow$ PLACE3$(Y, S, x_i, t, \epsilon_x)$
                    **if** $C_i(Y, S, x_i) < \max(\sigma, \epsilon_d)$ **then**
                        $isAnchor[i] \leftarrow true$
                    **else**
                        Reinitialize $x_i$
                    **end if**
                **end while**
            **end if**
        **end if**
    **end for**
**end for**
Return $X$
```

Algorithm 7 PLACE3$(Y, S, x_i, t, \epsilon_x)$

```
**for** $iter = 1$ **to** $10\sqrt{t}$ **do**
    $x_{old} \leftarrow x_i$
    **for** $j = 1$ **to** $|Y|$ **do**
        $x'_j \leftarrow$ move $x_j \in Y$ towards $x_i$ by $S_j$
    **end for**
    $x_i \leftarrow$ RECENTER$(\{x'_1, x'_2 \ldots x'_{|Y|}\})$
    **if** $\|x_i - x_{old}\|_2 < \epsilon_x$ **then**
        Return $x_i$
    **end if**
**end for**
Return $x_i$
```

two stages. In the first stage, it uses the existing anchors to find more anchors using Algorithm 6. When it can no longer add anchors, it invokes the second stage to embed the remaining points and to refine their location. The significant difference between this algorithm and that of PlaceCenter is the FindMoreAnchors step. In my experiment I have found that for sparse networks, it is not advisable to use the PlaceCenter directly since is very susceptible to local minima. This problem of local minima is alleviated using the FindMoreAnchors subroutine and a good initialization. This FindMoreAnchors subroutine is also critical to give a faster embedding when the number of anchors is large. Consider a case when 90% of the total sensors are anchors, in such a case, algorithm would most likely not enter the second step as all remaining sensors would be found in the FindMoreAnchors step. The part after FindMoreAnchors is same as the one presented in Chapter 2 (Algorithm 2) with the difference that I now minimize the distances with respect to the point in Y and the corresponding distances in S.

I now describe the FindMoreAnchors algorithm (Algorithm 6). It first considers points connected to at least $k+2$ anchors. Since we know the location of the anchors, localizing a point with respect to anchors is a relatively easy problem, and only minimizing the distance with respect to the anchors should lead to a better solution. In FindMoreAnchors, all such points that are connected to at least $k+2$ anchors are localized using Place3 by only considering the distances from anchors; and when these points are localized up to a user-defined tolerance ($\max(\sigma, \varepsilon_d)$), they are added to the set of the anchors. The process is repeated for N_{in} iterations.

5.3 Connection with Other Problems

The SNL problem is closely related to many other problems (See section 2.6 of Krislock's [Krislock, 2010] thesis and the survey by Liberti et al. [2012]). My algorithm can be easily extended to solve these related problems. Two related problems are:

Problem 2 (Euclidean Distance Matrix Completion). *Find a complete EDM (Euclidean Distance Matrix) $\bar{D}$ such that $\bar{d}_{ij} = d_{ij}, \forall (i, j) \in E$.*

EDM is a distance matrix such that each entry $\bar{d}_{ij}$ of the matrix $\bar{D}$ is an Euclidean distance from a set of points in some but not necessarily known Euclidean space.

Problem 3 (EDM with rank constraint(EDM-k)). *It is the same problem as Problem 2, with the difference that the distances $\bar{d}_{ij}$ need to be from a fixed k-dimensional space. Formally, find a set of points $x_1, x_2 \ldots x_n \in \mathbb{R}^k$ such that $\|x_i - x_j\|_2 = d_{ij}, \forall (i, j) \in E$.*

Note that EDM with rank constrain problem is nothing but SNL problem with number of anchors $m = 0$, while EDM problem without rank constraint can easily be reduced to the one with rank constraint by setting $k = n$, since n is the maximum possible rank of n points in any arbitrarily high dimensional space. Now note that my approach can easily handle both related problems: EDM with rank constraint and without rank constraint naturally, since it does not require any anchors [1]. It only requires the dimension of the embedding space, which for EDM without rank constraint can just be n

[1]Having more number of anchors makes the problem easy, but it is not a requirement

Table 5.1: Comparing PLACEMENT with Pong and Tseng (2009) and Krislock et al. (2010) for different values of σ and η for $n = 1000, m = 20, R = 0.06$

σ	η	**PLACEMENT**			**Ting(2010)**			**Krislock et al. (2010)**		
		RMSE	MaxErr	Time	RMSE	MaxErr	Time	RMSE	MaxErr	Time
0	0.01	2.29e-3	1.80e-2	0.54	7.67e-2	3.62e-1	12.75	8.44e-14	67e-13	0.87
0	0.1	2.75e-2	1.39e-1	1.40	8.25e-2	3.73e-1	12.14	2.52e-13	4.20e-12	0.83
0	0.2	4.10e-2	2.07e-1	1.49	7.18e-2	3.67e-1	10.44	1.66e-13	3.23e-12	0.87
0.1	0.01	5.65e-3	3.60e-2	0.73	7.27e-2	3.49e-1	5.62	5.54e+1	1.08e+3	0.92
0.1	0.1	3.09e-2	1.36e-1	1.93	8.63e-2	4.13e-1	6.07	7.95e+2	2.0e+4	1.10
0.1	0.2	3.82e-2	1.59e-1	3.47	7.24e-2	3.42e-1	5.30	7.64e+3	9.5e+4	0.84
0.3	0.01	1.85e-2	9.35e-2	1.80	7.86e-2	3.24e-1	4.40	7.29e+1	1.72e+3	1.44
0.3	0.1	3.38e-2	1.39e-1	1.11	8.56e-2	3.35e-1	4.49	1.16e+2	1.66e+3	1.78
0.3	0.2	4.58e-2	1.92e-1	2.54	7.20e-2	3.17e-1	4.24	9.05	1.18e+2	1.52

Table 5.2: Comparing PLACEMENT with Pong and Tseng (2009) and Krislock et al. (2010) for different values of σ and η for $n = 2000, m = 20, R = 0.05$

σ	η	**PLACEMENT**			**Ting(2010)**			**Krislock et al. (2010))**		
		RMSE	MaxErr	Time	RMSE	MaxErr	Time	RMSE	MaxErr	Time
0	0.01	3.78e-3	2.49e-2	1.59	6.21e-2	3.07e-1	37.69	5.53e-14	4.31e-13	1.30
0	0.1	2.05e-2	1.28e-1	4.35	7.84e-2	3.96e-1	43.14	1.33e-13	8.39e-13	1.14
0	0.2	4.04e-2	2.21e-1	4.24	5.98e-2	3.08e-1	41.67	6.65e-14	5.98e-13	1.13
0.1	0.01	6.28e-3	3.98e-2	2.09	5.79e-2	3.48e-1	20.90	1.08e+2	1.28e+3	1.28
0.1	0.1	2.12e-2	1.19e-1	6.73	6.40e-2	3.20e-1	18.88	6.78e+1	1.15e+3	1.21
0.1	0.2	3.02e-2	1.44e-1	10.08	8.96e-2	3.90e-1	22.27	6.17e+1	1.05e+3	1.20
0.3	0.01	7.49e-3	4.00e-2	0.39	7.42e-2	3.20e-1	15.87	4.12e+3	9.68e+4	1.86
0.3	0.1	2.52e-2	1.28e-1	7.01	6.89e-2	3.13e-1	16.09	9.69e+1	2.87e+3	3.21
0.3	0.2	3.33e-2	1.60e-1	9.40	7.83e-2	3.52e-1	15.93	1.53e+2	3.42e+3	1.27

5.4 Experiments

I now conduct an experimental study comparing PLACEMENT with the prior work. I experiment with a variety of sensor networks and show the behavior of PLACEMENT compared to the other algorithms. Since the behavior of an algorithm

Table 5.3: Comparing PLACEMENT with Pong and Tseng (2009) and Krislock et al. (2010) for different values of σ and η for $n = 4000, m = 20, R = 0.035$

σ	η	**PLACEMENT**			**Ting(2010)**			**Krislock et al. (2010))**		
		RMSE	MaxErr	Time	RMSE	MaxErr	Time	RMSE	MaxErr	Time
0	0.01	1.90e-3	3.10e-2	2.31	5.5e-2	3.05e-2	133.09	1.98e-13	1.92e-12	3.27
0	0.1	1.77e-2	9.63e-2	8.36	5.94e-2	3.17e-2	138.59	1.45e-13	9.7e-13	3.26
0	0.2	2.96e-2	1.48e-1	8.19	7.51e-1	3.73e-1	153.85	1.47e-13	9.91e-13	3.16
0.1	0.01	2.69e-3	2.24e-2	1.78	7.11e-2	3.68e-1	96.96	2.64e+1	8.54e+2	8.84
0.1	0.1	1.75e-2	8.99e-2	15.89	5.9e-2	3.18e-2	85.31	5.88e+3	6.57e+4	6.50
0.1	0.2	2.44e-2	1.24e-1	21.11	7.26e-2	3.45e-1	88.08	1.76e+1	3.12e+2	10.70
0.3	0.01	5.50e-3	3.07e-2	0.82	7.23e-2	3.45e-1	60.62	2.90e+1	4.37e+2	10.79
0.3	0.1	1.69e-2	1.05e-2	12.51	7.64e-2	3.46e-2	62.90	1.02e+2	2.25e+3	19.14
0.3	0.2	2.62e-2	1.36e-1	21.12	9.28e-2	3.96e-1	64.24	1.28e+2	2.72e+3	33.18

depends on the underlying network, I take different networks to cover a wide span of problems.

I initially chose to compare PLACEMENT with following 6 algorithms, but later, I discarded 4 of them for the following reasons. First three of these six algorithms are based on solving SDP, and were found to be relatively very slow even for $n = 1000$; while the fourth algorithm was only suitable for non-anchor case. This leaves us with just two other algorithms i.e., Krislock [2010] and [Pong and Tseng, 2009].

Below is a brief description of all baselines, and reasons why I decided (or not) to choose them.

- **Biswas et al. (2006)** [Biswas et al., 2006]- This algorithm was found to be very slow.

- **Kim et. al (2009)** [Kim et al., 2009]- Although this algorithm is shown to perform better than the ESDP algorithm of Wang et al. (2008) and the original

Biswas-Ye relaxation, it is considerably slower than Pong and Tseng (2009) which I include in the baselines.

- **Wang et al. (2008)** [Wang et al., 2008] - This was discarded because it has been shown to be inferior than Kim et al. [Kim et al., 2009].

- **Fang and Oleary** [Fang and O'Leary, 2011] - Algorithm is only suitable for the non-anchor case.

- **Pong and Tseng (2009)** [Pong and Tseng, 2009] - This is the best baseline among all baselines, especially for noisy measurements, and therefore, it would be my main baseline across different experiments.

- **Krislock (2010)** [Krislock, 2010] - This algorithm is scalable, and have been shown to embed $100,000$ points in few minutes, but only when there is no noise in distance measurement. But when there is a noise in the measurements, algorithm performs very poorly, as will be seen later in the experiments.

For all of the baselines (reported or not reported), I asked authors to provide their code or downloaded publicly available code. I implemented my own algorithm in C. All runtimes reported in experiments are wall clock times and are in seconds. For baselines, all the parameters were taken to be either at the default values or whatever suggested by authors. For my algorithm, I chose the following values of the parameters. $\epsilon_x = 1e^{-7}, \epsilon_d = 1e^{-5}$; $N_r = 20, N_a = 10$ for $\sigma = 0$, otherwise $N_r = 40, N_a = 5$.

5.4.1 Data Generation

Following the previous work [Wang et al., 2008; Krislock, 2010; Pong and Tseng, 2009; Kim et al., 2009], artificial data is generated by first generating n points uniformly at random in $[0,1]^2$ box, and then selecting m points uniformly at random as anchors. Call these set of points as X_0. Here 0 denotes the time. To model moving points, I have them move in a random walk described by a Gaussian distribution on each step. Specifically, in each step I generate the set X_t from X_{t-1} by $X_t = X_{t-1} + \eta\mathcal{N}(0,1))$, where η is the perturbation factor and $\mathcal{N}(0,1)$ is the Gaussian random variable with 0 mean and unit variance. Only sensors move, anchors locations remain fixed. For each set of points except X_0 which is used for initialization; edges are generated based on the radio-distance R. Two nodes are connected if their distance is less than radio-distance

$$D'_{ij} = \begin{cases} \|x_i - x_j\|_2,\ (i,j) \in E & \text{if } \|x_i - x_j\|_2 \leq R \\ \text{unspecified} & \text{otherwise.} \end{cases}$$

Then we add Gaussian noise to each specified distance. For all $(i,j) \in E$ we set $D_{ij} = D'_{i,j}(1 + \sigma\mathcal{N}(0,1))$, where σ is the noise factor. In the moving case, this σ-error is added to the measurement after each time step.

5.4.2 Evaluation

I evaluate the algorithms based on their ability to position sensors at their actual locations, (not based on the cost function). We measure two quantities, RMSE (Root

Mean Square Error) and MaxErr (Maximum Error) defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n-m}\sum_{i=1}^{n-m} \|x_i - x_i^{true}\|^2}$$

$$\text{MaxErr} = \max_i \|x_i - x_i^{true}\|$$

I perform two types of experiments. First, I consider only one time step, where data is initialized at X_0, and after one step, I solve the SNL on X_1 (each sensor moves with $\eta\mathcal{N}(0,1)$). Second, I consider the problem over multiple time steps, e.g. at $t = 1, 2, \ldots, 20$, and analyze how error propagates.

5.4.3 Results

In reporting results, there are five free parameters to generate a problem i.e., n, m, R, σ and η. I vary all five parameters, and for each set of parameters we generate six random problems, and present the mean. All of the above parameters control different properties of the network. n and m and R control the sparseness/density of the network. A higher value of R means more edges and hence a dense graph. σ controls the amount of noise in the distances while η controls the movement of sensors (only speed). All times are reported in seconds unless otherwise noted.

First set of results is reported in Tables 5.1, 5.2, and 5.3. In this set of results, I fix the network structure i.e., value of m and R, but change the amount of noise σ and the amount of perturbation η. I repeat these experiments for different values of $n = 1000, 2000$ and 4000. From these tables, first observation I make is that Krislock et

Table 5.4: Comparison of PLACEMENT with Pong and Tseng (2009) for different networks e.g. for different m and R but for fixed $\sigma = \eta = 0.1$ and $n = 1000$

n	m	R	**PLACEMENT**			**Ting(2010)**		
			RMSE	MaxErr	Time	RMSE	MaxErr	Time
1000	5	0.02	1.18e-1	3.51e-1	0.03	1.12e-1	3.39e-1	0.50
1000	5	0.06	2.77e-2	1.55e-1	1.55	1.96e-1	5.45e-1	17.69
1000	5	0.1	1.40e-2	1.41e-1	4.46	2.22e-1	7.36e-1	9.09
1000	5	0.15	6.04e-3	1.55-2	18.59	1.26e-1	5.86e-1	9.43
1000	20	0.02	1.17e-1	3.56e-1	0.03	1.11e-1	3.67e-1	0.64
1000	20	0.06	3.39e-2	1.36e-1	2.50	8.20e-2	3.49e-1	5.79
1000	20	0.1	7.69e-2	2.18e-1	10.68	3.33e-1	3.23e-1	3.41
1000	20	0.15	1.74e-2	1.32e-1	12.74	5.60e-1	3.00e-1	1.93
1000	50	0.02	1.15e-2	3.47e-1	0.04	1.09e-2	3.00e-1	1.93
1000	50	0.06	2.94e-2	1.44e-1	3.03	3.26e-1	2.39e-1	1.9
1000	50	0.1	2.06e-2	1.23e-1	5.98	2.55e-2	1.97e-1	1.24
1000	50	0.15	3.48e-3	1.18e-2	7.80	1.49e-2	1.18e-1	1.21
1000	200	0.02	1.09e-1	3.45e-1	0.05	1.03e-1	3.45e-1	0.35
1000	200	0.06	1.45e-2	9.24e-2	1.32	1.43e-2	1.26e-1	0.56
1000	200	0.1	9.79e-3	1.37e-1	0.66	2.96e-3	9.99e-3	0.73
1000	200	0.15	2.91e-3	1.03e-2	0.52	2.89e-3	1.00e-2	1.05

Table 5.5: Comparison of PLACEMENT with Pong and Tseng (2009) for different networks e.g. for different m and R but for fixed $\sigma = \eta = 0.1$ and $n = 4000$

n	m	R	**PLACEMENT**			**Ting(2010)**		
			RMSE	MaxErr	Time	RMSE	MaxErr	Time
4000	5	0.02	4.28e-2	2.74e-1	3.38	2.61e-1	6.23e-1	286.11
4000	5	0.1	4.94e-4	1.31e-2	64.55	1.23e-1	5.03e-1	267.35
4000	20	0.02	4.19e-2	2.85e-1	3.85	1.12e-1	3.52e-1	78.08
4000	20	0.1	7.20e-2	2.45e-1	175.18	2.36e-2	2.64e-1	48.99
4000	50	0.02	4.09e-2	2.58e-1	4.26	8.60e-2	3.39e-1	52.04
4000	50	0.1	7.59e-3	1.04e-1	100.69	1.51e-3	5.55e-3	16.11
4000	200	0.02	3.70e-2	2.89e-1	4.03	4.12e-2	2.89e-1	12.27
4000	200	0.1	1.66e-3	7.41e-3	12.43	1.41e-3	5.98e-3	9.69

al. (2010) performs poorly when there is noise in the data. This algorithm is extremely good for non-noisy data, and can give almost perfect position for sensors, however if one only cares for the near-perfect position (of the order of $1e^{-3}$), performance of PLACEMENT is very close to that of Krislock et al. Since Krislock et al. (2010) does not perform so well for the noisy data which is the main focus of this work, I will though include it in further results, Pong and Tseng (2009) would be my main baseline, and the discussion will revolve around that.

On comparing Pong and Tseng (2009) with PLACEMENT, I observe that PLACEMENT outperforms Pong and Tseng (2009) in all cases both in terms speed and accuracy. In some cases (e.g., non-noisy cases) the difference in time is significant. Note that time taken by Pong and Tseng (2009) for $n = 4000$ even with $\eta = 0.2$ perturbation is 153 seconds while PLACEMENT can find a *better* embedding in only 8 seconds. This time difference between Pong and Tseng (2009) and PLACEMENT is consistent for other cases as well. I emphasize here that both algorithms PLACEMENT and Pong and Tseng (2009) require an initial estimation of the locations but Pong and Tseng (2009) is not able to exploit the good estimate of the initial locations as much as PLACEMENT. Notice that when there is small perturbation $\eta = 0.01$, PLACEMENT is significantly faster than Pong and Tseng (2009)—for $n = 4000$, $\sigma = 0$, $\eta = 0.01$ Pong and Tseng (2009) takes 133 seconds while PLACEMENT only takes 2.31 seconds, while providing the better embedding both in terms of RMSE and MaxErr.

In the second set of results, I study the behavior of different algorithms as the network properties are varied, specifically the number of anchors m and the radio distance R. I fix the noise level σ at 0.1 and perturbation η at 0.1, so as not to bias any algorithm. Note from the previous tables that my algorithm will perform very well for small perturbation $\eta = 0.01$ and for small noise, hence I do not keep σ and η to be too small to favor PLACEMENT. The results of this set of experiments are reported in Tables 5.4 and 5.5. Due to the poor performance of Krislock et al. (2010) on noisy data, I exclude it from the table. From this set of experiments, I observe that PLACEMENT has a better performance than Pong and Tseng (2009) when the network is sparse, i.e., when R and m both are small. As I increase R, the performance of PLACEMENT still remains better as long as m is small but when I increase both, Pong and Tseng (2009) starts to catch up. This is mainly because of the relatively poor performance of Pong and Tseng (2009) for small m. Note that all the experiments reported in Pong and Tseng [2009] considered m at 10% of n. Here I keep m to be much smaller, we believe this to be a more practical scenario. When both m and R are large (e.g. $m = 200$, $R = 0.15$), there is not much difference in the performance of Pong and

Table 5.6: Comparing PLACEMENT with Pong and Tseng (2009) and Krislock et al. (2010) on large scale noisy data for $n = 10000, m = 20, R = 0.02$. "-" indicates that job did not complete within 24 hours.

σ	η	**PLACEMENT**			**Ting(2010)**			**Krislock et al. (2010))**		
		RMSE	MaxErr	Time	RMSE	MaxErr	Time	RMSE	MaxErr	Time
0.02	0.01	2.6e-3	2.9e-2	5	6.0e-2	2.6e-1	598	1.1e+2	1.9e+3	39
0.02	0.1	1.5e-2	8.1e-2	31	6.7e-2	3.0e-1	582	2.5e+3	1.8e+5	33
0.1	0.01	2.9e-3	3.0e-2	5	7.1e-2	3.0e-1	592	3.6e+2	2.2e+4	126
0.1	0.1	1.5e-2	8.4e-2	28	6.4e-2	2.6e-1	858	-	-	-

Tseng (2009) and PLACEMENT because, for large m, subroutine FINDMOREANCHORS finds more anchors making the problem simpler, and thus PLACEMENT runs faster.

In certain instances, it is not clear which algorithm performs best, mainly because my algorithm is an iterative algorithm, and one can increase/decrease the time by running it for more/less iterations. For example, $n = 1000, m = 50, R = 0.15$ or $n = 1000, m = 5, R = 0.15$, although PLACEMENT takes more time, but it also gives better accuracy. One can reduce the time by compromising on the accuracy.

5.4.4 Large Scale Experiments

I also perform experiments on much larger problems in the noisy setting. To the best of our knowledge, there has not been any work on this type of problem at this scale for noisy data. So for the large scale, there is no baseline algorithm. But still to see the behavior of both baselines for noisy sparse problem, I run them for $n = 10,000$ and results are reported in Table 5.6. From this table, it is clear that PLACEMENT not only runs significantly faster than both Pong and Tseng (2009) and Krislock et al. (2010) but also gives better accuracy for both RMSE and MaxErr. Results for even

Table 5.7: Performance of PLACEMENT on large scale noisy data

n	m	R	σ	η	**PLACEMENT**		
					RMSE	MaxErr	Time
20000	20	0.018	0.02	0.01	1.5e-3	2.1e-2	14
20000	20	0.018	0.02	0.1	1.1e-2	6.9e-2	79
20000	20	0.018	0.1	0.01	1.8e-3	2.4e-2	13
20000	20	0.018	0.1	0.1	1.1e-2	6.4e-2	87
40000	20	0.015	0.02	0.01	1.2e-3	1.7e-2	46
40000	20	0.015	0.02	0.1	8.6e-3	5.6e-2	316
40000	20	0.015	0.1	0.01	1.3e-3	1.8e-3	50
40000	20	0.015	0.1	0.1	8.4e-3	5.1e-2	305
80000	20	0.010	0.02	0.01	1.4e-3	1.6e-2	123
80000	20	0.010	0.02	0.1	8.2e-3	4.7e-2	699
80000	20	0.010	0.1	0.01	1.4e-3	1.6e-2	125
80000	20	0.010	0.1	0.1	8.3e-3	4.4e-2	729
120000	20	0.008	0.02	0.01	1.4e-3	1.5e-2	197
120000	20	0.008	0.02	0.1	8.0e-3	4.8e-2	1043
120000	20	0.008	0.1	0.01	1.4e-3	1.3e-3	202
120000	20	0.008	0.1	0.1	8.0e-3	4.3e-2	1085

larger problems, $n = 20,000$ to $n = 120,000$ are presented in Table 5.7. Note that my algorithm can handle the sensors up to $n = 120,000$ in less than 20 minutes, with a good accuracy.

5.4.5 Moving Problem

We now consider experiments on moving data over several (up to $t = 20$) time steps. For the moving problem, the dataset is generated as previously. At every time step, I perturb the points from the previous time step by a small amount η and measure the distances (by adding σ noise). Note that both algorithms only take the *initial position* X_0 as input, and in the successive steps, the initial position is the position returned from the previous step (estimated locations from $((t-1)$ step).

So far I have only considered the speed of the movement η not the pattern. When sensors are moving, there can be many moving patterns e.g., moving in a herd, moving in only one direction, disperse moving etc., and an algorithm's behavior might change according to the pattern. In this work, I experiment with following two moving patterns:

Moving-within-boundary in this pattern, sensors move from their initial location by a small amount η, but at any given time all sensors remain inside a fixed boundary. This pattern would imitate if animals are left in a big fenced in ranch (note that animals are randomly distributed in the ranch), and then they start moving. They are not allowed to go out of the ranch.

Moving-without-boundary   this pattern is same as above except, now that there is no notion of boundary and animals are allowed to move freely. Since the size of the area sensors are located in is increasing with the time (and so are the distances), error would also increase (as seen in Figures 5.1 and 5.2) .

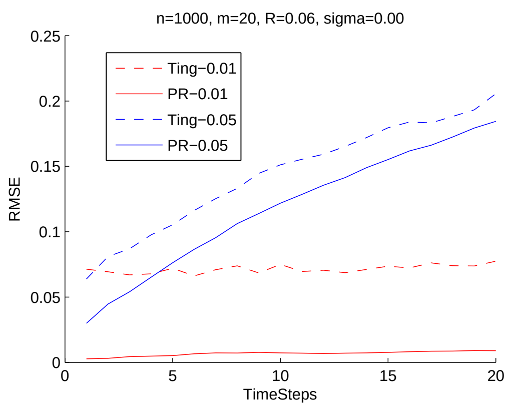
Results for the pattern moving-without-boundary is shown in Figures 5.1 and 5.2. For these experiments, I fix $n = 1000$, but change other parameters m, σ and η. Each subfigure in these figures is captioned with the average time taken in all 20 time steps, in the order of Pong and Tseng (2009) ($\eta = 0.01$), PLACEMENT ($\eta = 0.01$), Pong and Tseng (2009) ($\eta = 0.05$), PLACEMENT ($\eta = 0.05$). Each subfigure has four RMSE plots, two for $\eta = 0.01$ (red lines) and two for $\eta = 0.05$ (blue lines). From both figures, it is clear that PLACEMENT outperforms Pong and Tseng (2009) in terms of both speed and accuracy in all cases. This improvement is even more significant if one considers the time taken by both algorithms. As mentioned above, since we are dealing with the ground with no boundary, as we advance in time, inter-sensor distances increase and so the embedding error, but the relative improvement of PLACEMENT over Pong and Tseng (2009) remains consistent. Note that when $R = 0.06, \sigma = 0.01, \eta = 0.05, m = 5$ (Figure 5.2(c)), Pong and Tseng (2009) takes 19.1 seconds while PLACEMENT only takes 0.2 seconds. I remind readers that such faster response time is important if one were to do a real-time tracking.

Results for the pattern moving-within-boundary is shown in Figures 5.3 and 5.4. In these cases as well, PLACEMENT consistently outperforms Pong and Tseng (2009) both in terms of speed and accuracy. Once again, compare the time for the case $m = 5, R = 0.06, \eta = 0.05, \sigma = 0.01$ (Figure 5.4(c)), Pong and Tseng (2009) takes

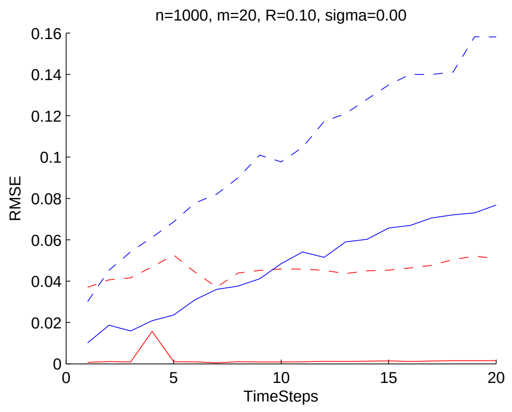
25.6 seconds while PLACEMENT only takes 0.9 seconds.

5.5 Conclusion

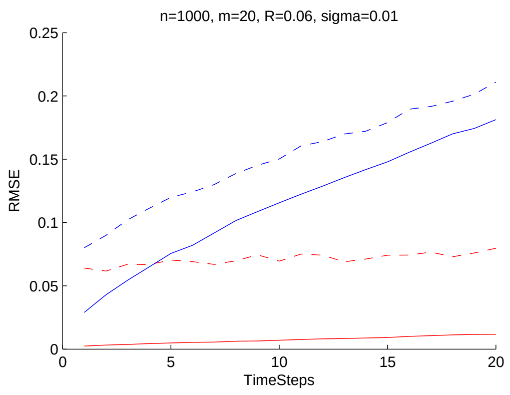
I have presented an iterative scalable MDS-based algorithm for sensor network localization that can handle the noisy data more effectively than the previous scalable algorithms, and is guaranteed to reduce the cost function in every iteration. It can embed sensors up to $120,000$ in less than 20 minutes. The proposed algorithm requires an initial estimate of the sensor locations, and is able to exploit them i.e., a good initial estimate gives a very good performance both in terms of speed and accuracy compared to other algorithms, making this algorithm effective for mobile sensor network problems. Furthermore, my algorithm is naturally distributed (i.e. decentralized computation possible) thus avoiding any centralized computation and hence the communication overhead.



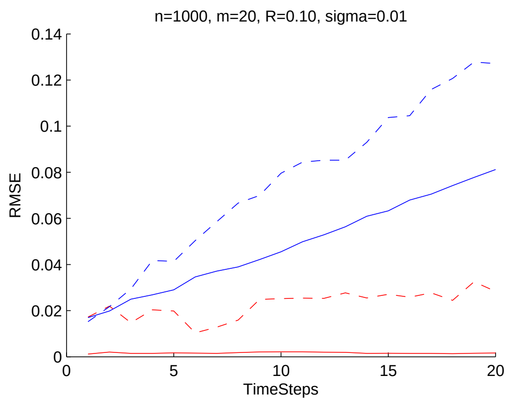
(a) 9.5, 0.3, 7.1, 0.4



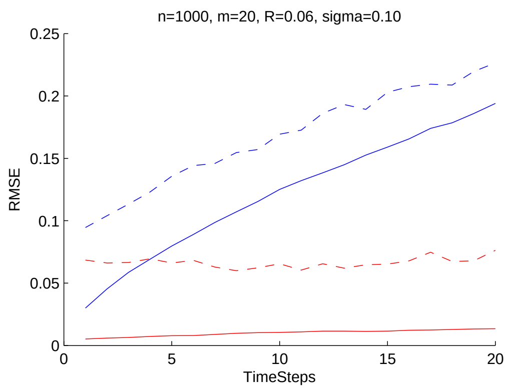
(b) 7.8, 0.7, 8.6, 1.3



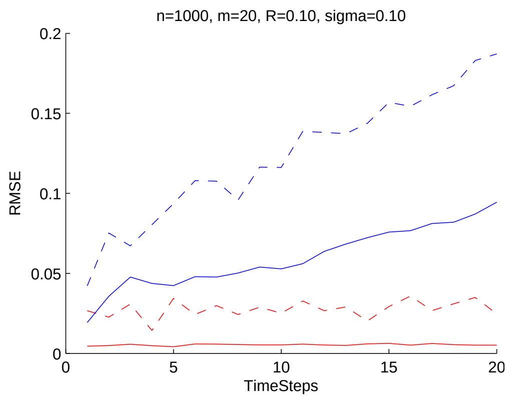
(c) 8.3, 0.3, 5.6, 0.4



(d) 4.0, 0.8, 4.0, 2.2

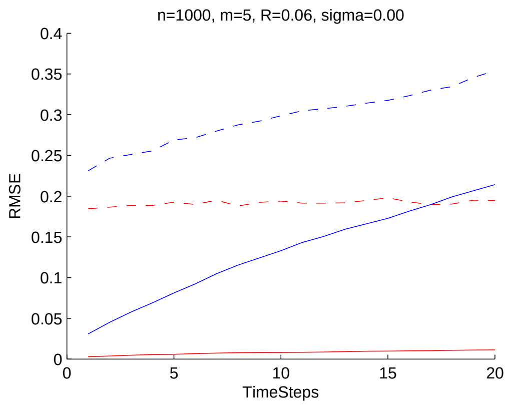


(e) 4.9, 0.2, 3.8, 0.3

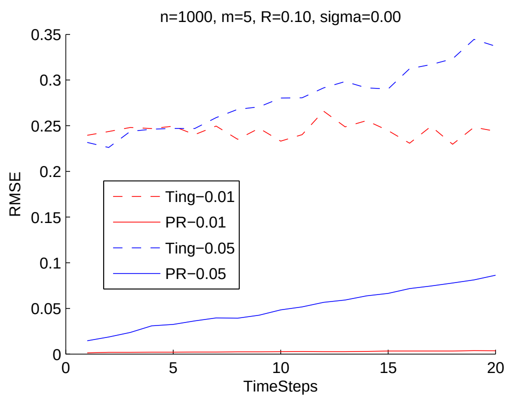


(f) 3.0, 2.7, 3.0, 4.3

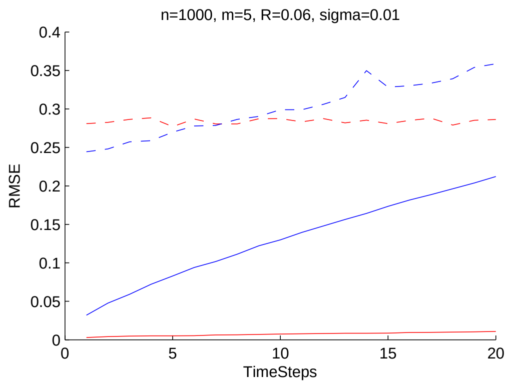
Figure 5.1: RMSE versus time steps for $m = 20$ for moving pattern moving-without-boundary. Here Ting corresponds to Pong and Tseng (2009) and PR corresponds to my algorithm.



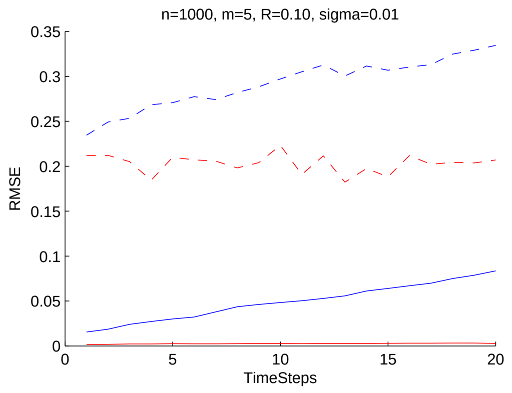
(a) 30.8, 0.3, 16.6, 0.2



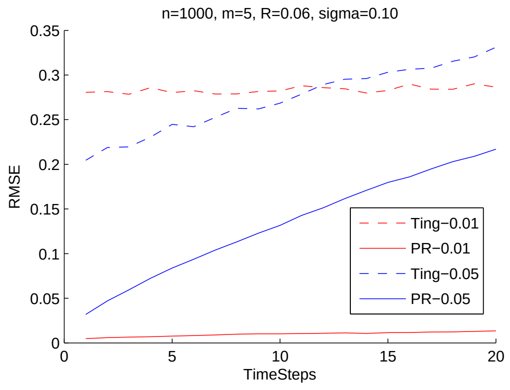
(b) 19.84, 0.71, 19.95, 2.32



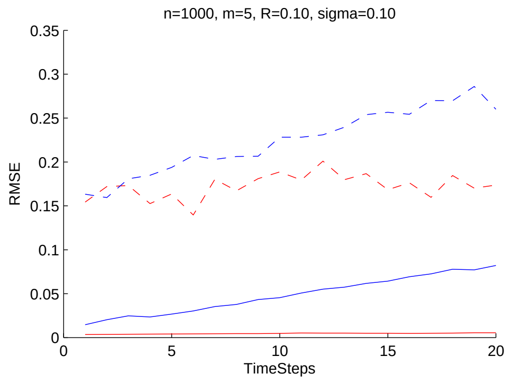
(c) 26.1, 0.3, 19.1, 0.2



(d) 13.6, 0.7, 18.7, 2.7

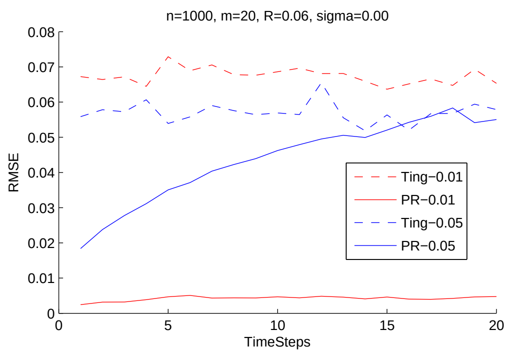


(e) 18.0, 0.2, 10.6, 0.2

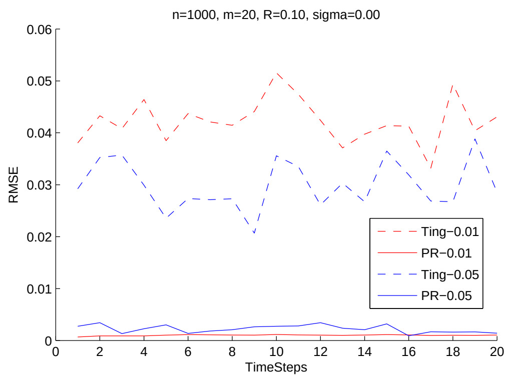


(f) 10.6, 0.4, 9.1, 2.9

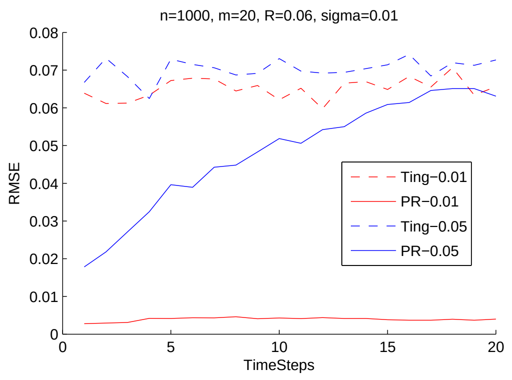
Figure 5.2: RMSE versus time steps for $m = 5$ for moving pattern moving-without-boundary. Here Ting corresponds to Pong and Tseng (2009) and PR corresponds to my algorithm.



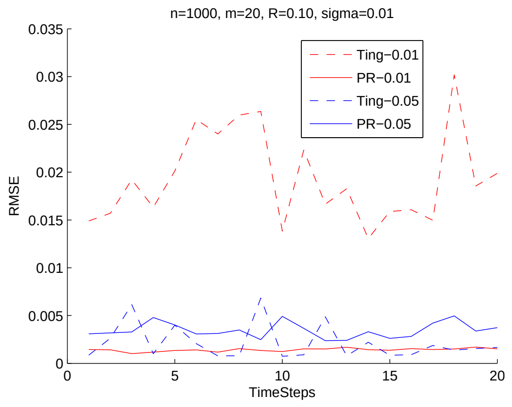
(a) 10.3, 0.4, 10.0, 1.2



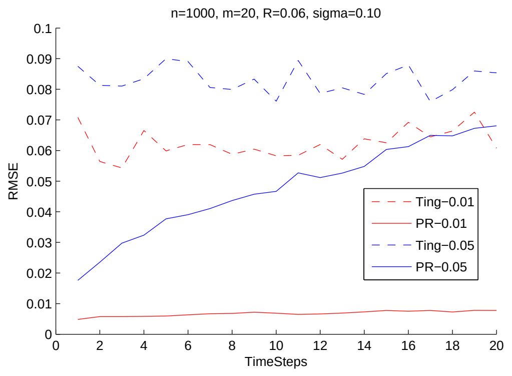
(b) 7.5, 0.6, 8.6, 1.5



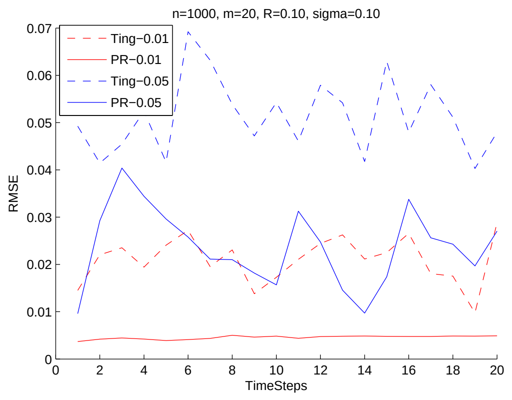
(c) 8.8, 0.4, 8.1, 1.6



(d) 4.3, 1.0, 3.7, 6.4

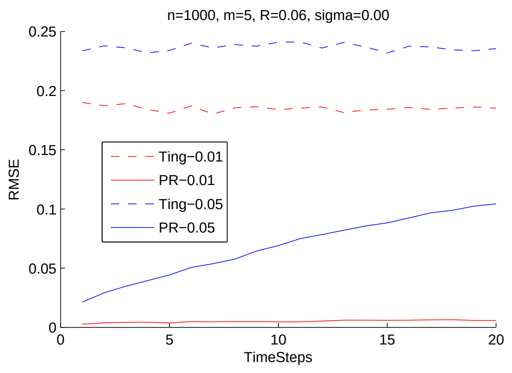


(e) 5.1, 0.2, 5.2, 1.4

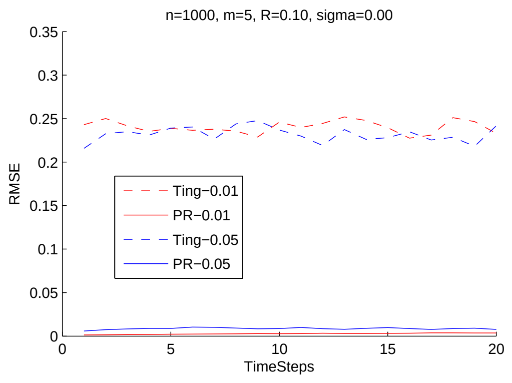


(f) 2.8, 2.6, 2.8, 7.4

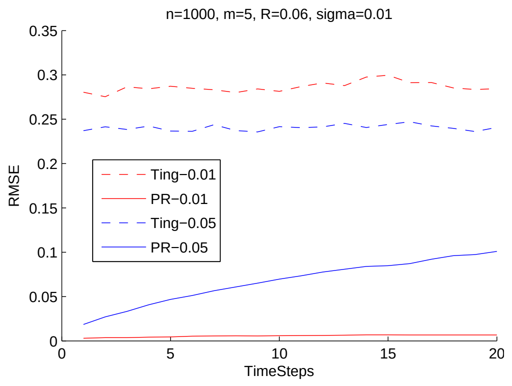
Figure 5.3: RMSE versus time steps for $m = 20$ for moving pattern moving-within-boundary. Here Ting corresponds to Pong and Tseng (2009) and PR corresponds to my algorithm.



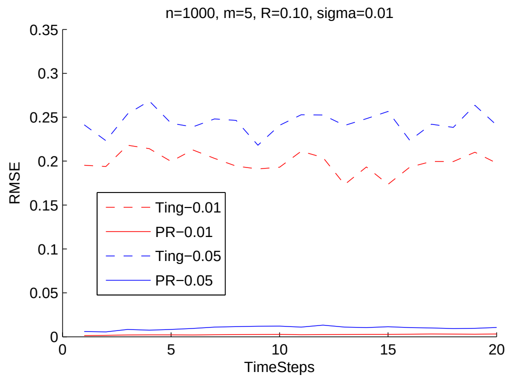
(a) 31.9, 0.4, 23.1, 0.7



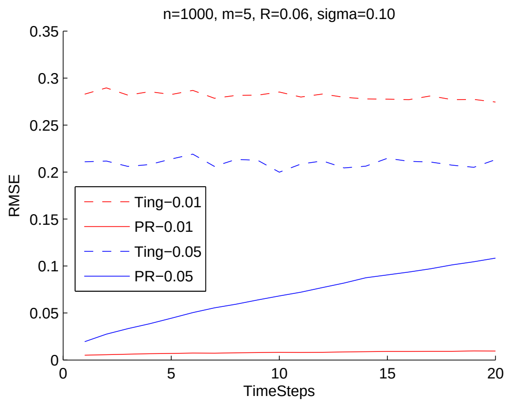
(b) 19.3, 0.7, 20.9, 3.6



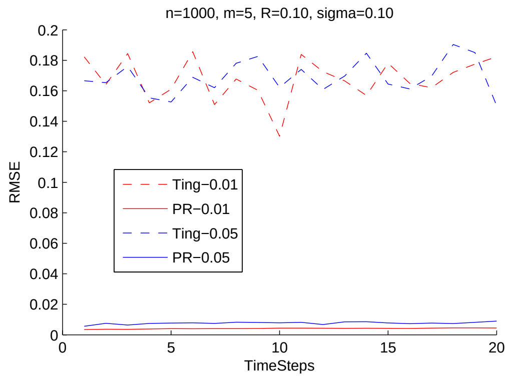
(c) 26.5, 0.4, 25.6, 0.9



(d) 14.3, 0.7, 19.4, 5.8



(e) 18.8, 0.2, 13.8, 0.7



(f) 11.1, 0.4, 9.9, 6.0

Figure 5.4: RMSE versus time steps for $m = 5$ for moving pattern moving-within-boundary. Here Ting corresponds to Pong and Tseng (2009) and PR corresponds to my algorithm.

Part II

Machine Learning

In the first part of my dissertation we have seen how geometric methods can be used to build powerful algorithms by solving a distance embedding problem and its variants. In this part of my dissertation, I consider problems in machine learning and develop geometric methods for them that exploit the geometry underlying probabilistic models. In particular,

In Chapter 6, I build a hybrid model — a combination of discriminative and generative models — using a prior called coupling prior. Unlike previous work, in this work, I combine these two types of models using a conjugate prior, and show the effectiveness of using conjugate prior over non-conjugate prior.

In Chapter 7, I use the geometry underlying probabilistic models to give reasoning about the appropriateness of conjugate priors. In addition, I show how this understanding of conjugate prior can be used to build a hybrid model more naturally. More specifically, I show that there is a natural way to combine discriminative and generative models that is based on the geometry underlying these models.

In Chapter 8, I extend the geometric understanding of probabilistic models to build a family of kernels called *generative kernels*. These kernels when used in a discriminative method like support vector machine, bring in the generative power, and result in a model that performs better than a pure generative and discriminative models. One can think of generative kernels as another way of building hybrid models.

In order to understand the material presented in next few chapters, readers would require a background in Legendre duality, Bregman divergence, and information geometry. A brief introduction to these topics is provided in Appendix A

Chapter 6

A Hybrid Semi-Supervised Learning Model

In this chapter, I present an approach to semi-supervised learning that is based on building a hybrid model — a combination of discriminative and generative models — using a conjugate prior as a coupling prior. This approach generalizes the previous work on coupled priors for hybrid generative/discriminative models. This chapter will also act as a background to the next chapter (Chapter 7) where I will use the geometry underlying the exponential family models to build this hybrid model *geometrically*.

6.1 Motivation

Labeled data on which to train machine learning algorithms is often scarce or expensive. This has led to significant interest in semi-supervised learning methods that can take advantage of unlabeled data [Cozman et al., 2003; Zhu, 2005]. While it is straightforward to integrate unlabeled data in a generative learning framework [Nigam et al., 2000], it is not so in a discriminative framework. Unfortunately, it is well-known both empirically and theoretically [Ng and Jordan, 2002] that discriminative approaches tend to outperform generative approaches when there is enough labeled data. This has led to many recent developments in *hybrid* generative/discriminative models that are able to leverage the power of both frameworks (see Section 6.4). One particular such example is the work of Lasserre et al. [2006], who describe a hybrid framework ("PCP") in which a generative model and discriminative model are jointly estimated, using a *prior* that encourages them to have similar parameters.

In this chapter, I generalize the *parameter coupling prior*[1] (PCP) method [Lasserre et al., 2006] to arbitrary distributions belonging to the *exponential family*. Unlike the PCP method, I do not restrict to the *Gaussian prior*, but instead choose a prior that is natural to the model. Other authors [Bouchard, 2007] have also noted the inappropriateness of the Gaussian prior to couple the generative and discriminative models. My resulting approach for hybridizing discriminative and generative models has the following properties.

1. It is not restricted to a particular class of the models.
2. It is more flexible in choosing the way these two models can be combined.
3. It enables us to achieve a closed form solution for the generative parameters, unlike PCP method, where one has to resort to the numerical optimization

I demonstrate the proposed approach on using a Beta/Binomial conjugate pair on the text categorization problems addressed by Druck et al. [2007].

6.2 Background

In general, machine learning approaches to classification can be divided into two categories: *generative approaches* and *discriminative approaches*. Generative approaches assume that the data is generated though an underlying process. One simple example

[1]Terms *parameter coupling prior* and *coupled prior* were introduced in [Druck et al., 2007] and do not appear in [Lasserre et al., 2006] though they refer to the framework introduced in [Lasserre et al., 2006].

is document categorization: for each example $(\mathbf{x}, y)$, generative process is as follows: first choose a category y, and then produce a document $\mathbf{x}$ conditioned on the category y. The goal in generative modeling is to approximate the joint distribution $p(\mathbf{x}, y)$ that represents this process. On the other hand, discriminative approaches do not assume any underlying process and directly model the probability of category given the document $p(y|\mathbf{x})$. Ng and Jordan [2002] compare these two approaches and show that while discriminative models are asymptotically better than generative models, generative models need less data to train.

In semi-supervised settings, one has access to lots of unlabeled data but only a small amount of labeled data. It is easy to see that unlabeled data is not directly useful in a discriminative setting but can be easily used in generative setting. However, since discriminative methods asymptotically tend to outperform generative methods [Ng and Jordan, 2002], this naturally leads to combining these two approaches and building a hybrid model that does better than the individual models. Earlier work [Bouchard and Triggs, 2004; Lasserre et al., 2006; Druck et al., 2007] has shown the efficacy of the hybrid approach.

6.2.1 Hybrid Model with Coupled Prior

I first define the problem and some of the notations that I will use through-out the paper. My task is to learn a model that predicts a label y given an example $\mathbf{x}$. I am given the data $D = D_L \cup D_U$ where D_L represents the labeled data and D_U represents the unlabeled data. Each instance of the labeled data consists of a pair $(\mathbf{x}, y)$ where

$\mathbf{x}$ is feature vector and y is the corresponding label. Each instance of unlabeled data consists of only feature vector $\mathbf{x}$. The $\mathbf{x}$s are M-dimensional feature vectors, and $\mathbf{x}_d$ denotes the d^{th} feature.

I now give a brief overview of the hybrid model presented by Lasserre et al. [2006]. The hybrid model is a mixture of discriminative and generative components, both of which have separate sets of parameters. These two sets of parameters (hence two models) are combined using a prior called *coupled prior*. Considering only one data point (the extension to multiple data points is straightforward and presented later), the model is defined as follows:

$$\begin{aligned} p(\mathbf{x}, y, \theta_d, \theta_g) &= p(\theta_g, \theta_d) p(y|\mathbf{x}, \theta_d) p(\mathbf{x}|\theta_g) \\ &= p(\theta_g, \theta_d) p(y|\mathbf{x}, \theta_d) \sum_{y'} p(\mathbf{x}, y'|\theta_g) \end{aligned}$$

Here θ_d is a set of discriminative parameters, θ_g a set of generative parameters, and $p(\theta_g, \theta_d)$ provides the natural coupling between these two sets of parameters. $p(y|\mathbf{x}, \theta_d)$ is the discriminative component; $p(\mathbf{x}|\theta_g) = \sum_{y'} p(\mathbf{x}, y'|\theta_g)$ is the generative component.

The most important aspect of this model is the *coupled prior* $p(\theta_g, \theta_d)$, which interpolates the hybrid model between two extremes; generative model when $\theta_d = \theta_g$ and discriminative when θ_d is independent of θ_g. In other cases, the goal of the coupled prior is to encourage the generative model and the discriminative model to have similar parameters. In earlier work [Lasserre et al., 2006; Druck et al., 2007], a

Gaussian prior was used as the coupled prior:

$$p(\theta_g, \theta_d) \propto \exp\left[-\lambda \left|\left|\theta_g - \theta_d\right|\right|^2\right]$$

Unfortunately, the Gaussian prior is not always appropriate [Bouchard, 2007].

6.3 Exponential Family Hybrid Model

In this section, I provide a more general prior for the hybrid model that is not only mathematically convenient but also allows choosing a problem specific prior.

6.3.1 Exponential Family Generalization

First, I generalize the hybrid model defined in Section 6.2.1 for the distributions that come from the exponential family. In other words, all of the distributions (generative, discriminative and coupled prior) of the generalized hybrid model belong to the exponential family. Below are the definitions of discriminative and generative models in terms of exponential family.

Generative model:

$$p(\mathbf{x}, y|\theta_g) = h(\mathbf{x}, y)\exp(\langle\theta_g, T(\mathbf{x}, y)\rangle - G(\theta_g)) \tag{6.1}$$

Discriminative model:

$$p(y|\mathbf{x}, \theta_d) = g(y)\exp(\langle \theta_d, T(\mathbf{x}, y)\rangle - M(\theta_d, \mathbf{x})) \tag{6.2}$$

Next, I break the coupled prior $p(\theta_g, \theta_d)$ into two parts; an independent prior on the discriminative parameters $p(\theta_d)$ and a prior on the generative parameters given discriminative parameters $p(\theta_g|\theta_d)$. This formulation lets us model the dependency of the generative component over the discriminative component. The new hybrid model can now be written as:

$$p(\mathbf{x}, y, \theta_d, \theta_g) = \Big[p(\theta_d)p(y|\mathbf{x}, \theta_d)\Big]\; p(\theta_g|\theta_d)\, \Big[\sum_{y'} p(\mathbf{x}, y'|\theta_g)\Big] \tag{6.3}$$

For convenience and interpretability (later I will show that it also improves the performance), I choose the coupled prior $p(\theta_g|\theta_d)$ to be conjugate with the generative model.

Conjugate prior:

$$p(\theta_g|\theta_d) = m(\theta_d)\,\exp(\langle \theta_g, \alpha(\theta_d)\rangle - \beta(\theta_d)G(\theta_g)) \tag{6.4}$$

Here, $\alpha(\cdot)$ and $\beta(\cdot)$ are user-defined functions that map the discriminative parameters θ_d into hyperparameters for the conjugate prior. I discuss suitable choices of these functions in Section 6.3.4.

Substituting the exponential definitions of generative model Eq (6.1), discriminative model Eq (6.2), and coupled prior Eq (6.4) in Eq (6.3), and taking a log, I obtain a log joint probability of data and parameters:

$$
\begin{aligned}
L = \log p(\mathbf{x}, y, \theta_d, \theta_g) = & \\
& \log p(\theta_d) + \\
& \log m(\theta_d) + \langle \theta_g \alpha(\theta_d) \rangle - \beta(\theta_d) G(\theta_g) + \\
& \log g(y) + \sum_{(\mathbf{x},y)\in D_L} \Big[\langle \theta, T(\mathbf{x}, y) \rangle - M(\theta_d, \mathbf{x}) \Big] + \\
& \quad \sum_{\mathbf{x}\in D} \log \sum_{y'} \Big[h(\mathbf{x}, y') \exp(\langle \theta_g, T(\mathbf{x}, y') \rangle - G(\theta_g)) \Big]
\end{aligned} \tag{6.5}
$$

Note that here discriminative part is defined only for labeled data while generative part is defined for both labeled and unlabeled data.

6.3.2 Parameter Optimization

The parameter optimization of the above expression (6.5) is performed by a coordinate descent method, alternating between optimizing the discriminative parameters θ_d and optimizing the generative parameters θ_g.

For the generative parameters, I take the partial derivative of the log probability in Eq (6.5) with respect to θ_g:

$$
\begin{aligned}
\frac{\partial L}{\partial \theta_g} &= \alpha(\theta_d) - \beta(\theta_d) G'(\theta_g) + \\
& \quad \sum_{\mathbf{x}\in D} \sum_{y'} p(y' | \mathbf{x}, \theta_g)(T(\mathbf{x}, y') - G'(\theta_g)).
\end{aligned}
$$

Here, $p(y'|\mathbf{x}, \theta_g)$ is the probability based on the parameters estimated in the last iteration $p(y'|\mathbf{x}, \theta_{g_{old}})$. Substituting this in the above equation and setting it equal to zero, gives:

$$\begin{aligned} G'(\theta_g) &= \frac{\sum_{\mathbf{x} \in D} \sum_{y'} p(y'|\mathbf{x}, \theta_{g_{old}}) T(\mathbf{x}, y') + \alpha(\theta_d)}{N + \beta(\theta_d)} \\ &= \frac{\hat{\mathbb{E}}_{\mathbf{x} \sim D} \, \mathbb{E}_{y \sim \theta_{g_{old}}}(T(\mathbf{x}, y')) + \alpha(\theta_d)}{N + \beta(\theta_d)}. \end{aligned} \tag{6.6}$$

Here $G'(\theta_g)$ denotes the partial derivative of $G(\theta_g)$ with respect to θ_g. As discussed in Section 6.1, choosing a conjugate prior gives us a closed form solution for $G'(\theta_g)$. From Eq (A.7), we know that $G'(\theta_g)$ is equivalent to the expected sufficient statistics of the generative model.

Having solved for the generative parameters θ_g, I now solve the hybrid model for discriminative parameters θ_d.

$$\begin{aligned} \frac{\partial L}{\partial \theta_d} &= \frac{\partial \log p(\theta_d)}{\partial \theta_d} + \frac{\partial \log m(\theta_d)}{\partial \theta_d} + \theta_g \alpha'(\theta_d) - \\ &\quad \beta'(\theta_d) G(\theta_g) + \sum_{(\mathbf{x}, y) \in D_L} (T(y, \mathbf{x}) - M'(\theta_d, \mathbf{x})) \end{aligned} \tag{6.7}$$

There is no closed form solution to the above expression therefore I solve it using numerical methods. In my implementation, I use stochastic gradient descent.

6.3.3 Hybrid Model: Naive Bayes and Logistic Regression

In this section, I demonstrate how this hybrid model can be applied in practice. I first choose a generative model that is suitable to my application, and then choose the coupled prior conjugate to the generative model. Since later on, I intend to use the hybrid model for the document classification task, I use a *naive Bayes*[2] (NB) model for the generative part and *logistic regression* for the discriminative part, akin to the study of Ng and Jordan [2002]. The generative part of my model (naive Bayes) is given by:

$$\begin{aligned} p(y, \mathbf{x}|\pi, v) &= p(y|\pi)p(\mathbf{x}|y, v) \\ &= \prod_k \pi_k^{1_{\{y=k\}}} \prod_d v_{yd}^{\mathbf{x}_d}(1 - v_{yd})^{1-\mathbf{x}_d}, \end{aligned} \tag{6.8}$$

where $\sum_{y'} \pi_{y'} = 1$ and $0 \leq v_{yd} \leq 1$. $1_{\{y=k\}}$ is an indicator function that takes value 1 if $y = k$ and 0 otherwise. This generative model can be written in the canonical form of the exponential family (see Appendix A).The discriminative part is:

$$p(y|\mathbf{x}, w, b) = \frac{1}{Z_\mathbf{x}} \exp\Big(b_y + \sum_d \mathbf{x}_d w_{yd}\Big), \tag{6.9}$$

where $Z_\mathbf{x} = \sum_{y'} \exp\big(b_{y'} + \sum_d \mathbf{x}_d w_{y'd}\big)$ is a normalization constant. Note here that since these models form generative/discriminative pair, number of parameters is same in both models. It is easy to see that there is one-to-one relationship between these two sets of parameters. b_y in the discriminative model behaves similar to π_y in the generative model, and w_{yd} behaves similar to v_{yd}. Since w_{yd} and v_{yd} are the parameters that capture most of the information, I use coupled prior to couple these sets of parameters and do not couple b_y and π_y. It is important to note the difference between the canonical parameters of the exponential family representation of the model and the mean parameters. In the generative(or discriminative) model, $\theta_{g_{yd}}$ (or

[2]It should be noted that "naive Bayes" classifiers come into (at least) two different versions: the "multivariate Bernoulli version" and the "multinomial version" [Mccallum and Nigam, 1998]. Because of its generality, in my implementation, I use multivariate Bernoulli.

$\theta_{d_{yd}}$) denote the canonical parameters while v_{yd} (or w_{yd}) denote the mean parameters.

Having defined the appropriate discriminative and generative models, now I can get equivalent exponential family forms of these models. First I show the exponential form of the generative model. The generative model in Eq (6.8) can be broken into two parts: one is class probability $p(y|\pi)$ and other class conditional probability $p(\mathbf{x}|y, v)$. Since the parameters of these distributions are independent, one can get their exponential representations separately. Considering the class conditional probability for one feature, Eq (6.8) can be written in the following form:

$$p(\mathbf{x}_d|y, v_{yd}) = \exp\left(\mathbf{x}_d \log\frac{v_{yd}}{1-v_{yd}} + \log(1-v_{yd})\right)$$

Comparing this with Eq (6.1) gives $\theta_{g_{yd}} = \log\frac{v_{yd}}{1-v_{yd}}$; $G(\theta_{g_{yd}}) = \log(1+e^{\theta_{g_{yd}}})$ and $T(y, \mathbf{x}) = \mathbf{x}_d$. Substituting these along with the appropriate conjugate prior in Eq (6.6) gives us a closed form solution for $G'(\theta_{g_{yd}})$, which, in the naive Bayes model is equal to v_{yd}.

$$G'(\theta_{g_{yd}}) = v_{yd} = \frac{\sum_{\mathbf{x}\in D} p(y|\mathbf{x}, \theta_{g_{old}})\mathbf{x}_d + \alpha(\theta_d)}{N + \beta(\theta_d)} \tag{6.10}$$

In other words, v_{yd} is the normalized expected count of the d_{th} feature in class y, with smoothing parameters that are controlled by the coupled prior hyperparameters $\alpha(\theta_d)$ and $\beta(\theta_d)$.

Next I solve for π_y by directly optimizing the objective function Eq (6.8) with respect to π_y with the given constraints. This gives us $\pi_y = \frac{\sum_{\mathbf{x}\in D} p(y|\mathbf{x}, \theta_{g_{old}})}{N}$ which is the normalized expected number of examples in class y.

Having solved for generative parameters, I now solve for the discriminative parameters. Ideally, we would like to first get an equivalent exponential form of Eq (6.9) and then solve it using Eq (6.7). Since Eq (6.7) is only defined for discriminative parameters that are coupled (w), we can not use Eq (6.7) unless we break Eq (6.9) into two exponential forms separate for w and b and, it is not clear how to do so. Therefore, I solve for discriminative parameters directly, without converting Eq (6.9) into exponential form. It is important to note here that mean parameters w in Eq (6.9) is equal to the canonical parameters $\theta_{d_{yd}}$. I place Gaussian prior $p(\theta_d) = N(\theta_d|0,\sigma^2)$ on $w = \theta_d$ and an improper uniform prior on b. Taking derivatives, I obtain:

$$\begin{aligned}\frac{\partial L}{\partial w_{yd}} &= -\frac{w}{\sigma^2} + \frac{\partial \log m(w_{yd})}{\partial w_{yd}} + \langle \theta_{g_{yd}} \alpha'(w_{yd}) \rangle - \beta'(w_{yd}) G(\theta_{g_{yd}}) \\ &\quad + \sum_{(\mathbf{x},y')\in D_L} \left\{ 1_{\{\mathbf{x}_d=1\}} - \frac{1}{Z_{\mathbf{x}}} \exp(b_{y'} + \sum_d \mathbf{x}_d w_{y'd}) 1_{\{\mathbf{x}_d=1\}} \right\} \\ \frac{\partial L}{\partial b_y} &= \sum_{(\mathbf{x},y')\in D_L} \left\{ 1_{\{y=y'\}} - \frac{1}{Z_{\mathbf{x}}} \exp(b_y + \sum_d \mathbf{x}_d w_{yd}) \right\}\end{aligned}$$

6.3.4 Conjugate Beta Prior

Recall that the conjugate prior crucially depends on two functions: $\alpha(\theta_d)$ and $\beta(\theta_d)$ that "convert" the discriminative parameters θ_d into a prior on the generative parameters $p(\theta_g|\theta_d)$. In the case of the binomial likelihood, the conjugate prior is Beta.

Exponential form of Beta prior is defined as:

$$p(\theta_{g_{yd}}|\theta_{d_{yd}}) = m(\theta_{d_{yd}})\exp(\theta_{g_{yd}}\alpha(\theta_{d_{yd}}) - \beta(\theta_{d_{yd}})G(\theta_{g_{yd}})),$$

where $m(\theta_{d_{yd}}) = \frac{\Gamma(\beta(\theta_{g_{yd}})+2)}{\Gamma(\alpha(\theta_{g_{yd}})+1)\Gamma(\beta(\theta_{d_{yd}})-\alpha(\theta_{d_{yd}})+1)}$ and $G(\theta_{g_{yd}} = \log(1+e^{\theta_{g_{yd}}})$.

I select the function $\alpha(\theta_{d_{yd}})$ and $\beta(\theta_{d_{yd}})$ to be such that: (1) the *mode* of the conjugate prior is $\theta_{d_{yd}}$ and (2) the *variance* of the conjugate prior is controllable by the hyperparameter γ. As noted from Figure 6.1, as γ goes to ∞, variance goes to 0 and prior forces generative parameters to be equal to the discriminative parameters (pure generative model) and as γ goes to 0, variance goes to ∞ which implies the independence between generative and discriminative parameters (pure discriminative model). Other values of γ interpolate between these two extremes. Thus, I choose $\alpha(\theta_{d_{yd}}) = \gamma/(1+e^{-\theta_{d_{yd}}})$ and $\beta(\theta_{d_{yd}}) = \gamma$. This gives mode of $p(\theta_{g_{yd}}|\theta_{d_{yd}})$ at $\theta_{d_{yd}}$ with the variance that decreases in γ, as desired.

It is important to note that my choice of hyperparameters for the conjugate prior is not specific to this example, but holds true in general. In the general case, let G be the log-partition function associated with the generative model, then, the conjugate prior hyperparameters should be $\alpha(\theta_d) = \gamma G'(\theta_d)$ and $\beta(\theta_d) = \gamma$. This gives us the mode of conjugate prior at θ_d with the variance that decreases in γ. In the beta/binomial hybrid model, $G'(\theta_d) = G'(w) = 1/(1+e^{-w})$. Also note that in the beta/binomial example, $G'(\theta_d)$ is also the transformation function T that transforms the discriminative mean parameters w to the generative mean parameters v.

In Figure 6.1, I also compare the Beta prior (solid blue curves) to an "equivalent"

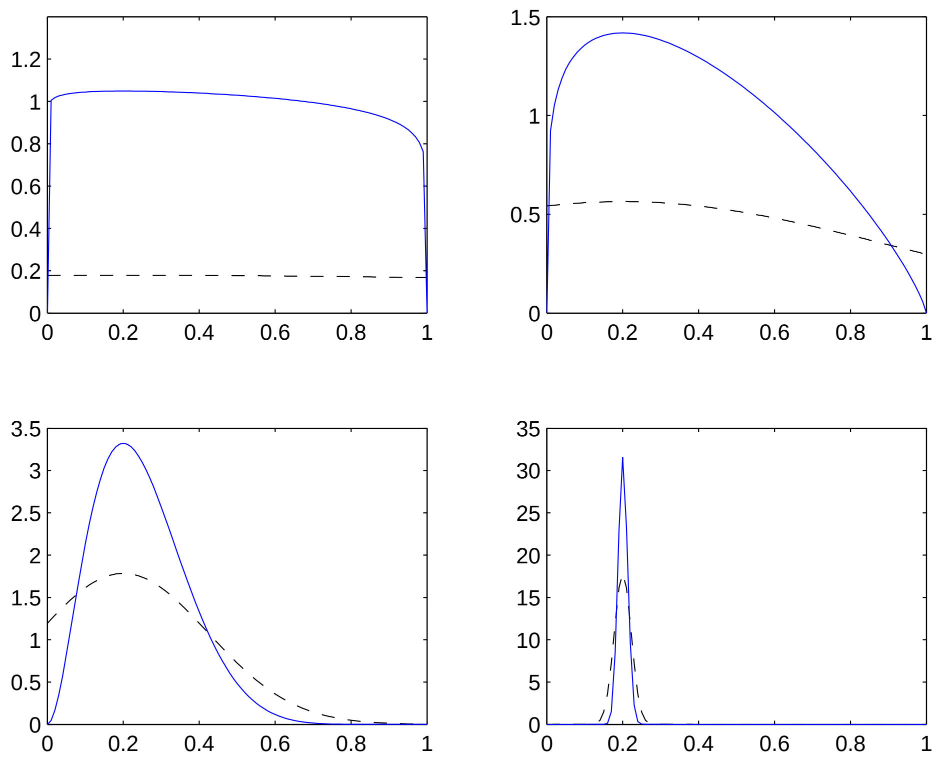


Figure 6.1: Effect of gamma on the Beta prior (solid curve) and Normal prior (dashed curve) for gamma=0.1, 1, 10, 100 (top-left, top-right, bottom-left, bottom-right) and for the transformed discriminative parameter $T(w) = 0.2$

Normal prior (dashed black curves) for four settings of γ. The Normal is parametrized to have the same mode and variance as the Beta prior. As we can see, for high values of γ (wherein the model is essentially generative), the two behave quite similarly. However, for more moderate settings of γ, the priors are qualitatively quite different.

6.4 Related Work

There have been a number of efforts to combine generative and discriminative models to obtain a hybrid model that performs better than either individually. Some of the earlier works [Raina et al., 2003; Bouchard and Triggs, 2004] use completely different approaches to hybridize these models; Raina et al. [2003] present a model for the document classification task where a document is split into multiple regions and complementary properties of generative/discriminative models are exploited by training a large set of the parameters generatively and only a small set of parameters discriminatively. Bouchard and Triggs [2004] build a hybrid model by taking a linear combination of generative and discriminative model. This model is similar to the multi-conditional learning model presented by McCallum et al. [2006]. Jaakkola and Haussler [1999] describe a scheme in which the kernel of a discriminative classifier is extracted from a generative model. Though these models have shown to perform better than just the discriminative or generative model, none of them combine the hybrid model in natural way.

My work builds on the work of Lasserre et al. [2006] and Druck et al. [2007], which are discussed in Section 6.2.1. Along these lines, Fujino et al. [2007] present another hybrid approach where a generative model is trained using a small number of labeled examples. Since the generative model has high bias, a generative "bias-correction" model is trained in a discriminative manner to discriminatively combine the bias-correction model with the generative model. Most of these work focus on the application and little on the theory of the hybrid model. There has been a recent work by Bouchard [2007] that presents a unified framework for the "PCP" model and the "convex-combination" model [Bouchard and Triggs, 2004], and proves performance properties.

Table 6.1: Description of the datasets used in the experiments

Dataset	No. of Features	Dataset description
movie	24,841	classifies the sentiments of the review of the movies from IMDB as *positive* or *negative*
webkb	22,824	classifies webpages from university as *student, course, faculty* or *project*
sraa	77,494	classifies messages by the newsgroup to which they were posted: *simulated-aviation*, *real-aviation*, *simulated-autoracing*, *real-autoracing*

6.5 Experiments

I now show empirical results of my approach and compare them with the existing (and the most related to my method) state-of-the-art semi-supervised methods [Druck et al., 2007].

6.5.1 Experimental Setup

In order to have a fair comparison, I use experimental setup of Druck et al. [2007] and perform experiments only for the datasets where PCP model have shown to perform best, There are three such datasets: `movie`, `sraa` and `webkb`. Description of these datasets is given in Table 6.1.

Although all of the examples in these datasets are labeled, I perform experiments by taking a subset of dataset as labeled and treating the rest of the examples as unlabeled. I use either 10 or 25 labeled examples from each class and vary unlabeled examples from 0 to a maximum of 1000. Number of unlabeled examples are same in each class. I show my results for two sets of experiments: (1) I show how performance varies as the number of unlabeled examples is varied; (2) I show how performance varies with respect to λ. Here λ normalizes the $\gamma \in [\infty, 0]$ in the range of $[0, 1]$ using $\gamma = ((1-\lambda)/\lambda)^2$. Now $\lambda = 0$ corresponds to the pure generative case while $\lambda = 1$ corresponds to the pure discriminative case. As in the work of Druck et al. [2007], the success of the semi-supervised learning depends on the quality of the labeled examples, therefore I choose five random labeled sets and report the average on them. In my results, I report the percentage classification accuracy which is the ratio of number of examples correctly classified to the total number of test examples.

6.5.2 Results and Discussion

Results on the above mentioned three datasets are presented in Table 6.2. Table shows the results for the PCP model with the Gaussian prior (PCP-Gauss) and with the Beta prior (PCP-Beta). Since PCP-Beta uses the binomial version of NB, I reimplemented the PCP-Gauss for the binomial version of NB and compare the results with it. Though I also show the results for PCP-Gauss multinomial [Druck et al., 2007], a fair comparison would be to compare only binomial models. %change is the change in PCP-Beta with respect to the PCP-Gauss binomial version. As we see, PCP-Beta performs better than PCP-Gauss binomial in all experiments and better than PCP-Gauss multinomial in all experiments except `sraa(10)`. Compared to PCP-Gauss binomial, PCP-Beta performs significantly better on `sraa` and `movie` datasets.

Comparing multinomial and binomial versions of PCP-Gauss, we see that for `movie` and `webkbb` datasets, binomial version performs better (or almost equal) than

Table 6.2: Comparative results for pcp with Gaussian prior and pcp with Beta prior. Parenthesized values denote the number of labeled examples per class and the standard deviation.

Dataset	pcp-Gauss Mult	pcp-Gauss Bin	pcp-Beta Bin	% change
`movie` (10)	64.6	63.4 (3.2)	**68.3** (5.5)	+7.7%
`movie` (25)	68.6	69.0 (1.5)	**76.7** (1.2)	+11.1%
`webkb` (10)	72.5	73.7 (3.7)	**75.3** (2.9)	+2.2%
`webkb` (25)	76.7	83.8 (1.3)	**83.9** (1.6)	+1.1%
`sraa` (10)	**81.6**	67.7 (6.8)	79.1 (4.0)	+16.8%
`sraa` (25)	84.1	76.6 (3.5)	**86.1** (1.0)	+12.4%

the multinomial while for `sraa` dataset, multinomial performs better. I conjecture that reason for this behavior could be because `sraa` has a large number of features and feature independence assumption is less violated in multinomial NB than in binomial NB. When datasets do not have too many features, binomial version tends to perform better because binomial NB accounts for both presence and absence of the features, in contrast to multinomial NB which only accounts for the presence of the features.

Figure 6.2 and Figure 6.3 show the results for accuracy vs. λ for different number of unlabeled examples for `sraa` and `movie` datasets respectively. Remember that $\lambda = 0$ is the purely generative model and $\lambda = 1$ is the purely discriminative model. In both of these figures, It is clear that as one increases the number of unlabeled examples, performance improves. In `sraa`, it is observed that increasing the number of unlabeled examples results in the shifting of optimal λ (λ^*) towards rights. It gives an optimal $\lambda^* = 0.2$ for a fully supervised model while for 1000 unlabeled examples, $\lambda^* = 0.5$. All the curves in this experiment are uni-modal which means that there is a unique value of λ where hybrid model performs best.

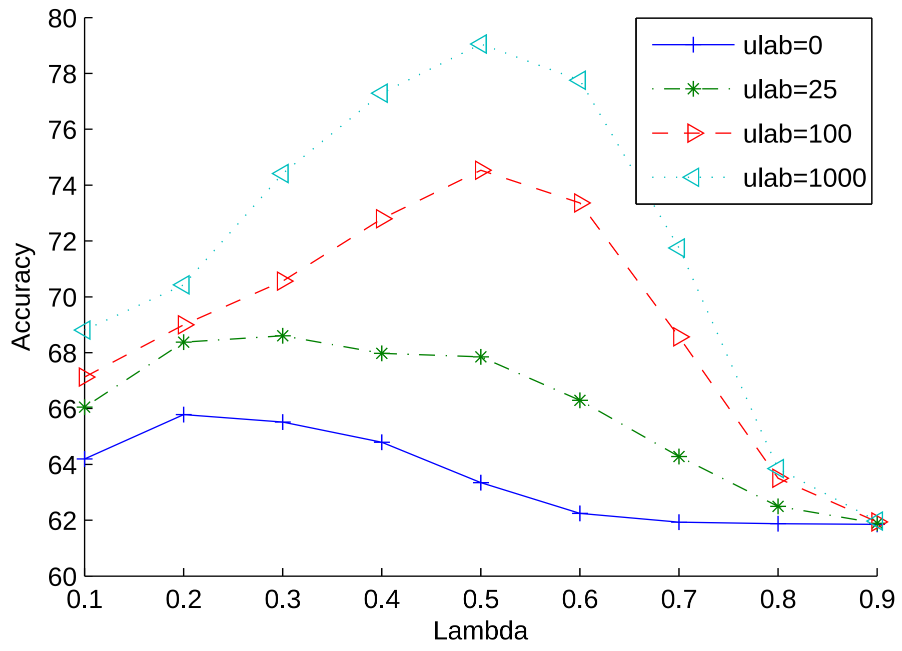


Figure 6.2: Results for `sraa` dataset for different number of unlabeled examples. Number of labeled examples=10.

Unfortunately, these nearly-perfectly shaped curves are not common to all settings, e.g., it is not observed in the other (`movie`) dataset (Figure 6.3). There are values of λ where a fully supervised model performs better than the best semi-supervised model. This experiment emphasizes the need for choosing the right value of λ and also shows the importance of the hybrid model. If one does not choose the right value of λ, he/she might end up hurting the model by using the unlabeled data. Note that `movie` dataset gives us a bi-modal curve in contrast to the uni-modal curve obtained in the `sraa`. From this figure, it is also noted that the curve is a uni-modal in the supervised setting but as one introduces unlabeled examples, the curves not only become bi-modal but also shift towards the left-hand side (best accuracy is achieved close to the generative end). This naturally suggests that generative model is actually

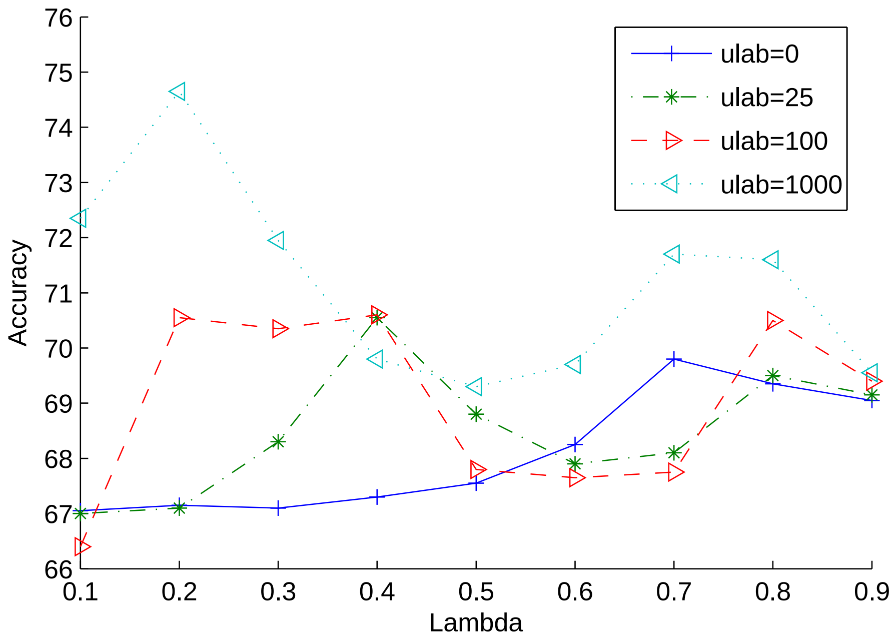


Figure 6.3: Results for `movie` dataset for different number of unlabeled examples, Number of labeled examples=25.

affecting the hybrid model in a positive manner and exploiting the strength of the unlabeled examples.

6.6 Conclusion and Future Work

I have presented a generalized "PCP" hybrid model for the exponential family distributions and have experimentally shown that the prior conjugate to the generative model is more appropriate than the Gaussian prior. In addition to the performance advantage, the conjugate prior also gives us a closed form solution for the generative parameters.

Chapter 7

A Geometric View of Conjugate Priors

In this chapter I show how geometric methods can be used to gain insight into existing algorithms. More specifically, I study the geometry underlying exponential family models and use it to reason about the appropriateness of conjugate priors. I use this study to gain insight into hybrid models Lasserre et al. [2006] – a family of models built by combining discriminative and generative models – and justify the use of conjugate prior to build one [Agarwal and Daumé III, 2009] as done in the previous chapter (Chapter 6).

In probabilistic modeling, a practitioner typically chooses a likelihood function (model) based on her knowledge of the problem domain. With limited training data, a simple maximum likelihood estimation (MLE) of the parameters of this model will lead to overfitting and poor generalization. One can regularize the model by adding a prior, but the fundamental question is: which prior? In this work, I give a turn-key answer to this problem by analyzing the underlying *geometry* of the likelihood model and suggest choosing the unique prior with the same geometry as the likelihood. This unique prior turns out to be the *conjugate* prior, in the case of the exponential family. This provides justification beyond "computational convenience" for using the conjugate prior in machine learning and data mining applications.

In this work, first I formulate the MLE problem into a completely geometric problem with no explicit mention of probability distributions (Section 7.2.1), and then give geometric argument in the favor of using conjugate priors (Sections 7.2.3 and 7.2.4). Empirical evidence showing the effectiveness of the conjugate priors can be found in our earlier work [Agarwal and Daumé III, 2009]. One important outcome of this analysis is that it allows us to treat the hyperparameters of the conjugate prior as the effective sample points drawn from the distribution under consideration. Finally, this geometric interpretation of conjugate priors is extended to analyze the hybrid model given by Lasserre et al. [2006] in a purely geometric setting and justify the argument presented in the Chapter 6 (i.e. a *coupling prior* should be conjugate) using a much simpler analysis (Section 7.3).

7.1 Motivation

The motivation for this analysis is the desire to understand the geometry of the conjugate priors for the exponential families. I motivate this analysis by asking myself the following question: Given a parametric model $p(x;\theta)$ for the data likelihood, and a prior on its parameters θ, $p(\theta;\alpha,\beta)$; what should the hyperparameters α and β of the prior encode? It is known that θ in the likelihood model is the estimation of the parameter using the given data points. In other words, the estimated parameter fits the model according to the given data while the prior on the parameter provides the generalization. This generalization is enforced by some prior belief encoded in the hyperparameters. Unfortunately, one does not know what is the likely value of the parameters; rather one might have some belief in what *data points* are likely to be sampled from the model. Now the question is: Do the hyperparameters encode this belief in the parameters in terms of the sampling points? My analysis shows that the hyperparameters of the conjugate prior is nothing but the effective sampling points. In case of non-conjugate priors, the interpretation of hyperparameters is not clear.

A second motivation is the following geometric analysis. Before we go into the problem, consider two points in the *Euclidean* space which one would like to interpolate using a parameter $\gamma \in [0, 1]$. A natural way to do so is to interpolate them linearly i.e., connect two points using a straight line, and then find the interpolating point at the desired γ, as shown in Figure 7.1(a). This interpolation scheme does not change if we move to a *non-Euclidean* space. In other words, if we were to interpolate two points in the non-Euclidean space, we would find the interpolating point by connecting the two points by a geodesic (an equivalent to the straight line in the non-Euclidean space) and then finding the point at the desired γ, shown in Figure 7.1(b).

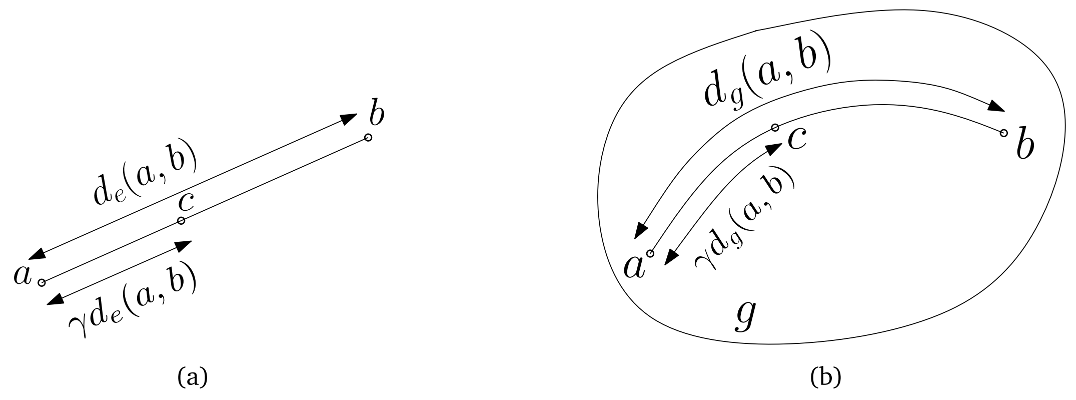


Figure 7.1: Interpolation of two points a and b using (a) Euclidean geometry, and (b) non-Euclidean geometry. Here geometry is defined by the respective distance/divergence functions d_e and d_g. It is important to notice that the divergence is a generalized notion of the distance in the non-Euclidean spaces, in particular, in the spaces of the exponential family statistical manifolds. In these spaces, it is the divergence function that define the geometry.

This situation arises when one has two models and wants to build a better model by interpolating them. This exact situation is encountered in Lasserre et al. [2006] where the objective is to build a hybrid model by interpolating (or coupling) discriminative and generative models. In our earlier work [Agarwal and Daumé III, 2009], we used conjugate prior to couple these two models, and empirically showed that using a conjugate prior for the coupling outperforms the original choice [Lasserre et al., 2006] of a Gaussian prior. In this work, the hybrid model is instead built by interpolating the two models using the *inherent geometry*[1] of the space (interpolate along the geodesic in the space defined by the inherent geometry) which automatically results in the expression of the conjugate prior along with its hyperparameters.

7.2 Likelihood, Prior and Geometry

In this section, I first formulate the MLE problem into a Bregman median problem (Section 7.2.1) and then show that the corresponding MAP problem can also be converted into a Bregman median problem (Section 7.2.3). The MAP Bregman median problem consists of two parts: a likelihood model and a prior. I argue (Sections 7.2.3 and 7.2.4) that a Bregman median problem makes sense only when both of these parts have the same geometry. Having the same geometry amounts to having the same log-partition function leading to the property of conjugate priors.

[1]In exponential family statistical manifold, inherent geometry is defined by the divergence function because it is the divergence function that induces the metric structure and connection of the manifold. Refer [Amari and Nagaoka, 2001] for more details.

7.2.1 Likelihood in the form of Bregman Divergence

Let $\mathcal{X}$ be the input data space, and Θ be the parameter space such that for each $\theta \in \Theta$, $p(x;\theta)$ is an exponential family distribution as defined in (A.5) (the likelihood of the point $x \in \mathcal{X}$ under distribution given by θ). Following [Collins et al., 2001], we can write the log likelihood of exponential family distributions in terms of Bregman divergence

$$\log p(x;\theta) = \log p_o(x) + F(\phi(x)) - B_F(\phi(x) \| \nabla G(\theta)). \tag{7.1}$$

This relationship depends on two observations: $F(\nabla G(\theta)) + G(\theta) = \langle \nabla G(\theta), \theta \rangle$ and $(\nabla F)^{-1}(\theta) = \nabla G(\theta) \Rightarrow (\nabla F)(\nabla G(\theta)) = \theta$. These two observations can be used with (A.3) to see that (7.1) is equivalent to the probability distribution defined in (A.5). This representation of likelihood in the form of Bregman divergence gives insight in the geometry of the likelihood function.

In learning problems, one is interested in estimating the parameters θ of the model which results in low generalization error, and perhaps, the most standard estimation method is *maximum likelihood*, which solves the following problem:

Definition 1. MLE. *Given a set of data points X_n and a family of distribution $p(x;\theta)$ parametrized by $\theta \in \mathbb{R}^d$, MLE finds $\hat{\theta}_{ML} \in \Theta$ such that $\hat{\theta}_{ML} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log p(x_i;\theta)$.*

It turns out that for exponential family distributions, the MLE problem can be transformed into a geometric problem, in particular into a Bregman median problem.

Lemma 1. *Let X_n be a set of n i.i.d. training data points drawn from the exponential family distribution with the log partition function G, and F be the dual function of G. Then the dual of the MLE ($\hat{\theta}_{ML}$) of X_n under the assumed exponential family model solves the following Bregman median problem:*

$$\hat{\theta}^*_{ML} = \hat{\mu}_{ML} = \arg\min_{\mu \in \mathcal{M}} \sum_{i=1}^{n} B_F(\phi(x_i) \| \mu) \tag{7.2}$$

Proof. The log-likelihood of X_n under the assumed exponential family distribution is given by $\log p(X_n; \theta) = \sum_{i=1}^{n} \log p(x_i; \theta)$ which along with (7.1) can be used to compute MLE i.e., $\hat{\theta}_{ML}$:

$$\begin{aligned}\hat{\theta}_{ML} &= \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \Big(\log p_o(x_i) + F(\phi(x_i)) - B_F(\phi(x_i) \| \nabla G(\theta))\Big) \\ &= \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} B_F(\phi(x_i) \| \nabla G(\theta)) \end{aligned} \tag{7.3}$$

which using $\nabla G(\theta) = \mu$, takes the entire problem into the $\mathcal{M}$-space and gives the desired result. □

The above theorem converts the problem of maximizing the log likelihood $\log p(X_n; \theta)$ into an equivalent problem of minimizing the corresponding Bregman divergences which is nothing but a *Bregman median* problem, the solution to which is given by $\hat{\mu}_{ML} = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i)$. MLE $\hat{\theta}_{ML}$ can now be computed using the expression $\nabla G(\theta) = \mu$, $\hat{\theta}_{ML} = (\nabla G)^{-1}(\hat{\mu}_{ML})$.

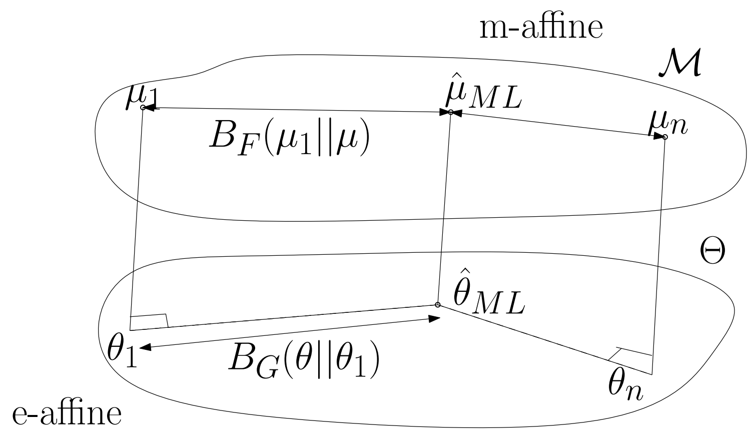


Figure 7.2: MLE as Bregman median problem into Θ and $\mathcal{M}$ spaces for exponential family distributions. In Θ space, it is a problem over θ_is with optimal θ appearing in the first argument of the divergence function while in $\mathcal{M}$-space, it is a problem over mean parameters with optimal μ appearing in the second argument in the divergence function.

Corollary 1 (MLE for Single Point). *Let x_i be the only point observed, $\hat{\mu}_{i,ML}$ under this observed point is $\phi(x_i)$.*

Proof. In Lemma 1, for single point, (7.2) is minimized when $\hat{\mu}_{i,ML} = \phi(x_i)$. □

Unless otherwise stated, I will use μ_i instead of $\hat{\mu}_{i,ML}$ to make notations less cluttered.

Theorem 4 (MLE as Bregman Median). *Let $\mathcal{M}$ and Θ be dual spaces as defined earlier, θ_i be the MLE of data point x_i under the exponential family parametric model $p(x;\theta)$ with log partition function $G(\theta)$. Given such $\{\theta_1,\dots,\theta_n\}$ for all points $\{x_1,\dots,x_n\}$, $\hat{\theta}_{ML}$ is equivalent to:*

$$\hat{\theta}_{ML} = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} B_G(\theta\|\theta_i) \tag{7.4}$$

Proof. From Corollary 1, $\phi(x_i) = \mu_i$, now replacing this in (7.2) gives

$$\hat{\mu}_{ML} = \arg\min_{\mu \in \mathcal{M}} \sum_{i=1}^{n} B_F(\mu_i \| \mu).$$

Now for $\mu_1, \mu_2 \in \mathcal{M}$ and $\theta_1, \theta_2 \in \theta$, using the relationship $B_F(\mu_1 \| \mu_2) = B_G(\theta_2 \| \theta_1)$ takes the entire problem from $\mathcal{M}$-space to Θ-space giving the desired result. □

Figure 7.2 gives a pictorial summarization of these MLE(s) as Bregman median problems in Θ and $\mathcal{M}$ spaces. The above expression requires us to find a θ so that divergence from θ to other θ_i is minimized. Now note that G is what defines this divergence and hence the geometry of the Θ space (as discussed earlier in Section 7.1). Since G is the log partition function of an exponential family, **it is the log-partition function that determines the geometry of the space**. I emphasize that divergence is measured from the parameter being estimated to other parameters θ_i(s), as shown in Figure 7.3.

7.2.2 Conjugate Prior in the form of Bregman Divergence

An expression similar to the likelihood can be written for the conjugate prior:

$$\log p(\theta|\alpha, \beta) = \log m(\alpha, \beta) + \beta(\langle \theta, \frac{\alpha}{\beta} \rangle - G(\theta)) \tag{7.5}$$

(7.5) can be written in the form of Bregman divergence by a direct comparison to (A.5), replacing $\phi(x)$ with α/β.

$$\log p(\theta|\alpha,\beta) = \log m(\alpha,\beta) + \beta\left(F\left(\frac{\alpha}{\beta}\right) - B_F\left(\frac{\alpha}{\beta}\|\nabla G(\theta)\right)\right) \tag{7.6}$$

The expression for the joint probability of data and parameters is given by:

$$\begin{aligned}\log p(x,\theta|\alpha,\beta) = \log p_o(x) + \log m(\alpha,\beta) + F(\phi(x)) + \beta F\left(\frac{\alpha}{\beta}\right) \\ - \left(B_F(\phi(x)\|\nabla G(\theta)) + \beta B_F\left(\frac{\alpha}{\beta}\|\nabla G(\theta)\right)\right)\end{aligned}$$

Combining all terms that do not depend on θ:

$$\log p(x,\theta|\alpha,\beta) = \text{const} - B_F(\phi(x)\|\mu) - \beta B_F\left(\frac{\alpha}{\beta}\|\mu\right) \tag{7.7}$$

7.2.3 Geometric Interpretation of Conjugate Prior

In this section I give a geometric interpretation of the term $B_F(\phi(x)\|\mu) + \beta B_F(\frac{\alpha}{\beta}\|\mu)$ from (7.7).

Theorem 5 (MAP as Bregman median). *Given a set X_n of n i.i.d. examples drawn from the exponential family distribution with the log partition function G and a conjugate prior as in* (7.6)*, MAP estimation of parameters is $\hat{\theta}_{MAP} = \hat{\mu}^*_{MAP}$ where $\hat{\mu}_{MAP}$ solves the following problem:*

$$\hat{\mu}_{MAP} = \arg\min_{\mu\in\mathcal{M}} \sum_{i=1}^{n} B_F(\phi(x_i)\|\mu) + \beta B_F\left(\frac{\alpha}{\beta}\|\mu\right) \tag{7.8}$$

which admits the following solution:

$$\hat{\mu}_{MAP} = \frac{\sum_{i=1}^{n} \phi(x_i) + \alpha}{n + \beta}$$

Proof. MAP estimation by definition maximizes (7.7) for all data points X_n which is equivalent to minimizing $B_F(x_i\|\mu) + \beta B_F(\frac{\alpha}{\beta}\|\mu)$. One can expand this expression using (A.3) and use conditions $F(\nabla G(\theta)) + G(\theta) = \langle \nabla G(\theta), \theta \rangle$ and $(\nabla F)^{-1}(\theta) = \nabla G(\theta)$ to obtain the desired solution. □

The above solution gives a natural interpretation of MAP estimation. One can think of prior as β number of extra points at position α/β. β works as the effective sample size of the prior which is clear from the following expression of the dual of the $\hat{\theta}_{MAP}$:

$$\hat{\mu}_{MAP} = \frac{\sum_{i=1}^{n} \phi(x_i) + \sum_{i=1}^{\beta} \frac{\alpha}{\beta}}{n + \beta}. \tag{7.9}$$

The expression (7.8) is analogous to (7.2) in the sense that both are defined in the dual space $\mathcal{M}$. One can convert (7.8) into an expression similar to (7.4) in the dual space which is again a Bregman median problem in the parameter space.

$$\hat{\theta}_{MAP} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} B_G(\theta\|\theta_i) + \sum_{i=1}^{\beta} B_G\Big(\theta\|(\frac{\alpha}{\beta})^*\Big) \tag{7.10}$$

here $(\frac{\alpha}{\beta})^* \in \Theta$ is dual of $\frac{\alpha}{\beta}$. The above problem is a Bregman median problem consisting of $n + \beta$ points, $\{\theta_1, \theta_2 \ldots \theta_n, \underbrace{(\alpha/\beta)^*, \ldots, (\alpha/\beta)^*}_{\beta \text{ times}}\}$, as shown in Figure 7.3 (left).

A geometric interpretation is also shown in Figure 7.3. When the prior is conjugate to the likelihood, they both have the same log-partition function (Figure 7.3, left). Therefore they induce the same Bregman divergence. Having the same divergence means that distances from θ to θ_i (in likelihood) and the distances from θ to $(\alpha/\beta)^*$ are measured with the same divergence function, yielding the same geometry for both spaces.

It is easier to see using the median formulation of the MAP estimation problem that one must choose a prior that is conjugate. If one chooses a conjugate prior, then the distances among all points are measured using the same function. It is also clear from (7.9) that in the conjugate prior case, the point induced by the conjugate prior behaves as a sample point $(\alpha/\beta)^*$. A median problem over a space that have different geometries is an ill-formed problem, as discussed further in the next section.

7.2.4 Geometric Interpretation of Non-conjugate Prior

The expression (7.10) was considering the prior to be conjugate to the likelihood function. Had the prior been non-conjugate with the log-partition function e.g., Q, one would have instead obtained:

$$\hat{\theta}_{ML} = \min_{\theta \in \Theta} \sum_{i=1}^{n} B_G(\theta \| \theta_i) + \sum_{i=1}^{\beta} B_Q\left(\theta \parallel \left(\frac{\alpha}{\beta}\right)^*\right). \tag{7.11}$$

Here G and Q are different functions defined over Θ. Since these are the functions that define the geometry of the space parameter, having $G \neq Q$ is equivalent to consider them as being defined over different (metric) spaces. Here, it should be noted that

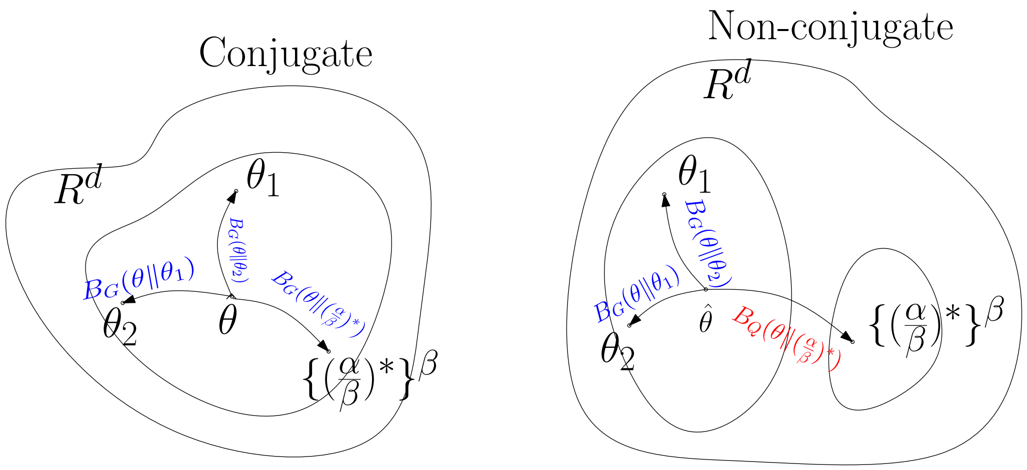


Figure 7.3: Prior in the conjugate case has the same geometry as the likelihood while in the non-conjugate case, they have different geometries.

distance between the sample point (θ_i) and the parameter θ is measured using the Bregman divergence B_G. On the other hand, the distance between the point induced by the prior $(\alpha/\beta)^*$ and θ is measured using the divergence function B_Q. This means that $(\alpha/\beta)^*$ can *not* be treated as one of the sample points. This tells us that, unlike the conjugate case, belief in the non-conjugate prior can not be encoded in the form of the sample points.

Another problem with considering a non-conjugate prior is that the dual space of Θ under different functions would be different. Thus, one will not be able to find the alternate expression for (7.11) equivalent to (7.8), and therefore not be able to find the closed-form expression similar to (7.9). This tells us why non-conjugate does not give us a closed form solution for $\hat{\theta}_{MAP}$. A pictorial representation of this is also shown in Figure 7.3. Note that, unlike the conjugate case, in the non-conjugate case, the data likelihood and the prior both belong to different spaces.

I emphasize that it is possible to find the solution of (7.11) i.e., in practice, there is nothing that prohibits the use of non-conjugate prior, using the conjugate prior is intuitive, and allows one to treat the hyper-parameters as pseudo data points.

7.2.5 Generalization to α-affine manifold

Not all probability distributions belong to the exponential family (although many do). A broader family of distributions is the "α-family" [Amari and Nagaoka, 2001]. Although a full treatment of this family is beyond the scope of the work, we briefly discuss an extension of my results to the α-family. An α-family distribution is defined as:

$$\log p_\alpha(x;\theta) = \begin{cases} \frac{2}{1-\alpha} p(x;\theta)^{(1-\alpha)/2} & \alpha \neq 1 \\ \log p(x;\theta) & \alpha = 1 \end{cases}$$

where $p(x;\theta)$ defined as in (A.5). Note that the exponential family is a special case of α-family for $\alpha = 1$.

MAP estimation of the parameters in the exponential family can be cast as a median problem, where an appropriate Bregman divergence is used to define the geometry. In other words, for exponential family, a Bregman-median problem naturally arose as an estimation method.

By using an appropriately defined, "natural," divergence for the α-family, one can actually obtain a similar result for this broader family of distributions. Using such a natural divergence, one can also define a "conjugate prior" for the α-family. Zhang

[2004] shows that such a natural divergence exist for α-family and is given by:

$$D_G^{\alpha}(\theta_1, \theta_2) = \frac{4}{1-\alpha^2}\left(\frac{1-\alpha}{2}G(\theta_1) + \frac{1+\alpha}{2}G(\theta_2) - G\left(\frac{1-\alpha}{2}\theta_1 + \frac{1+\alpha}{2}\theta_2\right)\right)$$

Like the exponential family, this divergence also induces the Fisher information metric.

7.3 Hybrid model

In this section, I show an application of the above analysis to a common supervised and semi-supervised learning framework, in particular, to a generative/discriminative hybrid model presented in the Chapter 6 [Agarwal and Daumé III, 2009; Druck et al., 2007; Lasserre et al., 2006] that has been shown to be successful in many applications. I will show how the coupling using the conjugate prior can be transformed into a coupling which is based on the distance geometry underlying generative/discriminative models.

Recall the hybrid model from the previous chapter which can be written as:

$$\begin{aligned} p(\mathbf{x}, y, \theta_d, \theta_g) &= p(\theta_g, \theta_d)p(y|\mathbf{x}, \theta_d)p(\mathbf{x}|\theta_g) \qquad (7.12) \\ &= p(\theta_g, \theta_d)p(y|\mathbf{x}, \theta_d)\sum_{y'} p(\mathbf{x}, y'|\theta_g) \end{aligned}$$

here θ_d is a set of discriminative parameters, θ_g a set of generative parameters, and $p(\theta_g, \theta_d)$ provides the natural coupling between these two sets of parameters.

As mentioned in the previous chapter, the most important aspect of this model is the *coupling prior* $p(\theta_g, \theta_d)$, which *interpolates* the hybrid model between two extremes: fully generative when the prior forces $\theta_d = \theta_g$, and fully discriminative when the prior renders θ_d and θ_g independent. In non-extreme cases, the goal of the coupling prior is to encourage the generative model and the discriminative model to have similar parameters. It is easy to see that this effect can be induced by many functions. One obvious way is to *linearly* interpolate them as done by Lasserre et al. [2006] and Druck et al. [2007] using a Gaussian prior (or the *Euclidean* distance) of the following form:

$$p(\theta_g, \theta_d) \propto \exp\left(-\lambda \left|\left|\theta_g - \theta_d\right|\right|^2\right) \tag{7.13}$$

where, when $\lambda = 0$, model is purely discriminative while for $\lambda = \infty$, model is purely generative. Thus λ in the above expression is the interpolating parameter, and is same as the γ in Section 7.1. Note that the log of the prior is nothing but the squared *Euclidean* distance between two sets of parameters.

In the previous chapter we showed that using a prior that is conjugate to the generative model outperforms the non-conjugate priors. It has been noted by other authors [Bouchard, 2007] that a Gaussian prior is not always appropriate, and the prior should instead be chosen according to models being considered. In our previous chapter, our main argument for choosing the conjugate prior came from the fact that this provides a closed form solution for the generative parameters and therefore is mathematically convenient. In this work, I will show that it is more than convenience that makes conjugate prior appropriate by using their geometric interpretation studied in the previous section. In addition to providing a justification behind their use as a coupling prior, this analysis also derives the expression and hyperparameters of the prior automatically (an additional advantage of using geometric methods).

7.3.1 Generalized Hybrid Model

In order to see the effect of geometry, recall the the generalized hybrid model for distributions that belong to the exponential family. We recall the expression for the generalized generative model:

$$p(\mathbf{x}, y|\theta_g) = h(\mathbf{x}, y)\exp(\langle\theta_g, T(\mathbf{x}, y)\rangle - G(\theta_g)) \tag{7.14}$$

where $T(\cdot)$ is the potential function similar to ϕ in (A.5), now only defined on $(\mathbf{x}, y)$.

Let G^* be the dual function of G; the corresponding Bregman divergence (retaining only the terms that depend on the parameter θ) is given by:

$$B_{G^*}\left((\mathbf{x}, y)\|\nabla G(\theta_g)\right). \tag{7.15}$$

Solving the generative model independently reduces to choosing a θ_g from the space of all generative parameters Θ_g which has a geometry defined by the log-partition function G. Similarly to the generative model, the exponential form of the discriminative model is given as:

$$p(y|\mathbf{x}, \theta_d) = g(y)\exp(\langle\theta_d, T(\mathbf{x}, y)\rangle - M(\theta_d, \mathbf{x})) \tag{7.16}$$

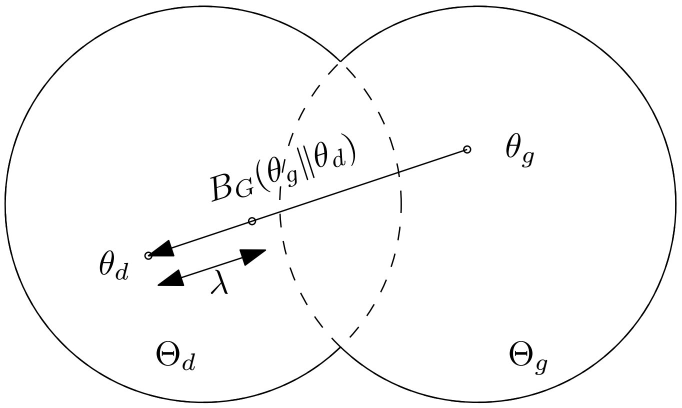


Figure 7.4: Hybrid model by interpolating between θ_d and θ_g using the Bregman divergence

Importantly, the sufficient statistics T are the *same* in the generative and discriminative models; such generative/discriminative pairs occur naturally: logistic regression/naive Bayes and hidden Markov models/conditional random fields are examples. However, observe that in the discriminative case, the log partition function M depends on both $\mathbf{x}$ and θ_d which makes the analysis of the discriminative model harder. Unlike the generative model, one does not have the explicit form of the log-partition function M that is independent of $\mathbf{x}$. This means that the discriminative component (7.16) can not be converted into an expression like (7.15), and the MLE problem can not be reduced to the Bregman median problem like the one given in (7.4).

7.3.2 Geometry of the Hybrid Model

The analysis of the hybrid model is simplified by writing the discriminative model in an alternate form. This alternate form makes obvious the underlying geometry of the discriminative model. Note that the only difference between the two models is that discriminative model models the conditional distribution while the generative model models the joint distribution. This observation can be used to write the discriminative model in the following alternate form using the expression $p(y|x,\theta) = \frac{p(y,x|\theta)}{\sum_{y'} p(y'x|\theta)}$ and (7.14):

$$p(y|x,\theta_d) = \frac{h(\mathbf{x},y)\exp(\langle\theta_d, T(\mathbf{x},y)\rangle - G(\theta_d))}{\sum_{y'} h(\mathbf{x},y')\exp(\langle\theta_d, T(\mathbf{x},y')\rangle - G(\theta_d))} \tag{7.17}$$

Denote the space of parameters of the discriminative model by Θ_d. It is easy to see that geometry of Θ_d is defined by G since function G is defined over θ_d. This is same as the geometry of the parameter space of the generative model Θ_g. Now let us define a new space Θ_H which is the *affine* combination of Θ_d and Θ_g. Now, Θ_H will have the same geometry as Θ_d and Θ_g i.e., geometry defined by G. Now the goal of the hybrid model is to find a $\theta \in \Theta_H$ that maximizes the likelihood of the data under the hybrid model. These two spaces are shown pictorially in Figure 7.4.

7.3.3 Prior Selection

As mentioned earlier, the coupling prior is the most important part of the hybrid model, which controls the amount of coupling between the generative and discriminative models. There are many ways to do this, one of which is given by Lasserre et al. [2006] and Druck et al. [2007]. By their choice of Gaussian prior as coupling prior, they *implicitly* couple the discriminative and generative parameters by the squared Euclidean distance. I suggest coupling these two models by a general prior, of which the Gaussian prior is a special case.

Bregman Divergence and Coupling Prior: Let a general coupling be given by $B_S(\theta_g \| \theta_d)$. Notice the direction of the divergence. This direction was chosen because the prior is induced on the generative parameters, and it is clear from (7.10) that parameters on which prior is induced, are placed in the first argument in the divergence function. The direction of the divergence is also shown in Figure 7.4.

Now recall the relation (7.6) between the Bregman divergence and the prior. Ignoring the function m (this is consumed in the measure defined on the probability space) and replacing $\nabla G(\theta)$ by θ^*, we get the following expression:

$$\log p(\theta_g|\alpha,\beta) = \beta(F(\frac{\alpha}{\beta}) - B_F(\frac{\alpha}{\beta}\|\theta_g^*)) \tag{7.18}$$

Now taking the $\alpha = \lambda\theta_d^*$ and $\beta = \lambda$, we get:

$$\log p(\theta_g|\lambda\theta_d^*, \lambda) = \lambda(F(\theta_d^*) - B_F(\theta_d^*\|\theta_g^*)) \tag{7.19}$$

$$p(\theta_g|\lambda\theta_d^*, \lambda) = \exp(\lambda(F(\theta_d^*)))\exp(-\lambda B_F(\theta_d^*\|\theta_g^*)) \tag{7.20}$$

For the general coupling divergence function $B_S(\theta_g\|\theta_d)$, the corresponding coupling prior is given by:

$$\exp(-\lambda B_{S^*}(\theta_d^*\|\theta_g^*)) = \exp(-\lambda(F(\theta_d^*)))\, p(\theta_g|\lambda\theta_d^*, \lambda) \tag{7.21}$$

The above relationship between the divergence function (left side of the expression) and coupling prior (right side of the expression) allows one to define a Bregman divergence for a given coupling prior and vise versa.

Coupling Prior for the Hybrid Model: It is known that that the geometry of the space underlying the Gaussian prior is just Euclidean, which does not necessarily match the geometry of the likelihood space. The relationship between prior and divergence (7.21) allows one to first define the appropriate geometry for the model, and then define the prior that respects this geometry. In the above hybrid model, this geometry is given by the log partition function G of the generative model. This argument suggests to couple the hybrid model by the divergence of the form $B_G(\theta_g \| \theta_d)$. The coupling prior corresponding to this divergence function can be written using (7.21) as:

$$\exp(-\lambda B_{G^*}({\theta_d}^* \| {\theta_g}^*)) = p(\theta_g | \lambda {\theta_d}^*, \lambda) \exp(-\lambda F(\theta_d^*)) \tag{7.22}$$

where $\lambda = [0, \infty]$ is the interpolation parameter, interpolating between the discriminative and generative extremes. In dual form, the above expression can be written as:

$$\exp(-\lambda B_G(\theta_g \| \theta_d)) = p(\theta_g | \lambda {\theta_d}^*, \lambda) \exp(-\lambda F(\theta_d^*)). \tag{7.23}$$

Here $\exp(-\lambda F({\theta_d}^*)) = \exp(\lambda(G(\theta_d) - \theta_d \nabla G(\theta_d)))$ can be thought of as a prior on the discriminative parameters $p(\theta_d)$. In the above expression, $\exp(-\lambda B_G(\theta_g \| \theta_d)) = p(\theta_g | \theta_g) p(\theta_d)$ behaves as a joint coupling prior $P(\theta_d, \theta_g)$ as originally expected in the model (6.3). Note that hyperparameters of the prior α and β are naturally derived from the geometric view of the conjugate prior. Here $\alpha = \lambda \theta_d^*$ and $\beta = \lambda$.

7.4 Related Work and Conclusion

To the best of my knowledge, there have been no previous attempts to understand Bayesian priors from a geometric perspective. However, one related piece of work [Snoussi and Mohammad-Djafari, 2002] uses the Bayesian framework to find the best prior for a given distribution. It is noted that, in that work, the authors use the δ-geometry for the data space and the α-geometry for the prior space, and then show different cases for different values (δ, α). It is emphasized that even though it is possible to use different geometry for the both spaces, it always makes more sense to use the same geometry. As mentioned in *remark* 1 in Snoussi and Mohammad-Djafari [2002], useful cases are obtained only when we consider the same geometry.

In this chapter, I have shown how geometry and the methods derived from it, are useful to get a better understanding of existing models. More specifically, I have considered the geometry underlying the model space and used it to transform the MLE problem into a geometric problem, *Bregman median* problem. This geometric interpretation of the space and the likelihood function allowed me to reason about the conjugate prior beyond their mathematical convenience. In addition to providing an insight in the hybrid model, this interpretation also justified the use of conjugate prior as a coupling prior; and as an added advantage, derived the expression and the hyperparameters of this coupling prior.

Chapter 8

Generative Kernels

In this chapter, I use geometric methods to build a family of kernels that can be used as off-the-shelf tool for any kernel learning method such as support vector machines. Generative models provide a useful statistical language for describing data; discriminative methods achieve excellent classification performance. In this chapter, I use geometric understanding of generative models to propose a family of kernels i.e., *generative kernels*, defined as a family of kernels built around generative models for use in discriminative classifiers. The key idea of generative kernels is to use the generative model of the data to automatically define a statistical manifold, on which a particular natural divergence (based on the Fisher information metric) can be translated into a positive definite kernel. My approach is applicable to any statistical model belonging to the exponential family, which includes common distributions like the Gaussian and multinomial, as well as more complex models.

Apart from the geometric perspective, there are other reasons to consider the data distribution when constructing the kernels. It is commonly observed that generative models perform well when only a small amount of data is available [Ng and Jordan, 2002], especially when the model is a true model of the data distribution [Liang and Jordan, 2008]. However, as the model becomes less true, or as the amount of data grows, discriminative approaches prevail, both theoretically and empirically [Ng and

Jordan, 2002; Liang and Jordan, 2008]. Ideally, one would like to take advantage of both methods, and build a hybrid method, that performs well for small training data and does not rely too much on the generative assumption. One way to build such a hybrid model is to combine discriminative and generative models using a coupling prior as I described in the previous chapter. Another way to build hybrid model is to encode the generative process information in the kernel and then use this kernel in the discriminative method.

There have been previous efforts to build kernels from generative models: the Fisher kernel [Jaakkola and Haussler, 1999], the heat kernel[Lafferty and Lebanon, 2005], and the probability product (PP) kernel [Jebara et al., 2004]. The first two consider the generative process by deriving the geometry for the statistical manifold associated with the generative distribution family using the fundamental principle of the *information distance*. Unfortunately, these kernels are intractable to compute exactly even for very simple distributions due to the need to compute the Fisher information metric. The approximations required to compute these kernels result in a function that is not guaranteed to be positive definite [Martins et al., 2008]. More discussion on the related work can be found in our published work [Agarwal and Daumé III, 2011]. There is another family of kernels named semi-group kernels [Cuturi et al., 2005] which turns out to have the same functional form as generative kernels though both are derived completely differently. Unlike Cuturi et al. [2005], I consider the geometry of the data distribution and reason why it is appropriate to call these kernels generative kernels. Note that semi-group kernels are not generative kernels for general probability distributions therefore empirical study of these kernels under the discriminative/generative paradigm is not considered in Cuturi et al. [2005]. In this chapter, generative kernels are studied under generative/discriminative paradigm, in particular, experiments are performed to see what happens when the data generation assumption is violated i.e., when the model is *mis-specified*.

These generative kernels have a number of desirable properties:

- They are applicable to *any* exponential family distribution.
- They are built using the natural geometry of the model space associated with data distribution.
- They are closed-form, efficient to compute, and by construction are always positive definite.
- Empirical comparisons to the best published kernels using the same data and experimental setup show improved performance.
- Empirical results with these kernels show that these kernels are able to exploit the generative properties i.e., perform as well as a generative model when there is less training data.

Unlike other distribution-based kernels, a discriminative method based on the proposed generative kernel is able to exploit the properties of the generative methods i.e., perform well when not enough data is given. Figure 8.1 shows typical result from a real world classification task on the text data. The blue curve which represents the generative method (Naive Bayes (NB)) performs better when training size n is small but as n is increased, discriminative methods (other curves) start to take over. Although

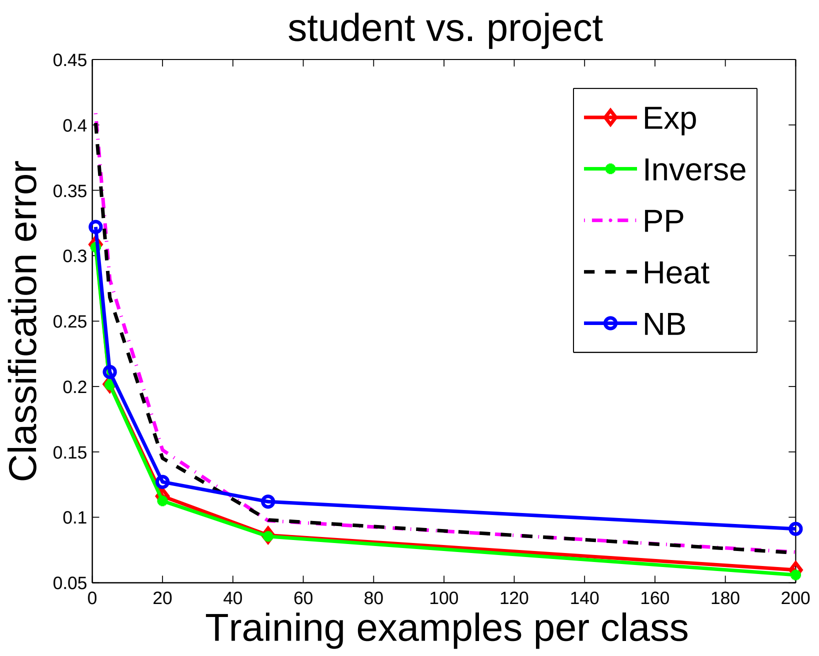


Figure 8.1: A typical example of relative performance on real dataset from multinomial distribution. *Exp* and *Inverse* are the generative kernels.

other discriminative methods perform poorer than the NB for small n, discriminative methods when used with generative kernel, perform better for all n. Generative kernels (red and green curves) perform equal/better to NB when n is small, and outperform all other methods as n gets large. Generative kernel curves are lower envelopes of all other curves, giving us the best of both worlds.

8.1 Background: Statistical Manifolds and Dualistic Structure

In this section, I define statistical manifolds and the dualistic structure associated with them. I in particular give reasons why it is important to choose the KL divergence to define the kernel for the exponential family. A statistical manifold $\mathscr{S}$ is a d-dimensional manifold $\mathscr{S} = \{\theta \in \Theta\}$ such that every $\theta \in \Theta$ induces a probability distribution over some space $\mathcal{X}$.

Following Amari and Nagaoka [2001], it is well known that all arbitrary divergences induce a *dualistic structure* and a *metric*. In particular, for statistical manifolds, the most natural divergence is the one that induces the *Fisher information metric*. The Fisher information metric is a special Riemannian metric that has many attractive properties (e.g. invariance under reparametrization), and is considered to be a natural metric for statistical manifolds. A divergence function that induces the Fisher information metric on the exponential family manifolds is KL divergence. It is also known as D^{-1} divergence (a special case of D^{η} divergence for $\eta = -1$). Since exponential family manifolds have dualistic structure (in fact they are dually flat), there exists a dual space, where one can define the dual divergence i.e. D^{1} divergence. Following Zhang [2004], this duality is called *referential duality*. In referential duality, for two points $p, q \in \Theta$ $D^{-1}(p\|q) = D^{1}(q^{*}\|p^{*})$ or $KL(p\|q) = KL^{*}(q^{*}\|p^{*})$. For exponential family statistical manifold, this duality is closely related to the other form of duality i.e., Legendre duality, and is also known as *representational duality*; and in such duality, $B_F(p\|q) = B_G(q^{*}\|p^{*})$, where F and G are dual of each other.

8.2 Generative Model to Metric

In this section, I develop generative kernels. I first take the *Bregman median* problem corresponding to the exponential family distribution that generated the data, and transform it into a problem that minimizes the KL divergence in Θ-space. As mentioned above, a natural divergence in Θ-space is the KL divergence. I use this KL

divergence to build a metric in Θ-space. Using duality, this metric is projected into $\mathcal{M}$-space, and then finally in the Section 8.3, this projected metric is converted into a positive definite kernel.

Here, I first borrow the *Bregman median* problem from Section 7.2.1 corresponding to the generative model belonging to the exponential family. Recall, the *Bregman median* problem is:

$$\hat{\theta}_{ML} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} B_G(\theta \| \theta_i)$$

where $\mathcal{M}$ and Θ are dual spaces of each other as defined earlier, and $\{\theta_1, \ldots, \theta_n\}$ are the MLE of data points $\{x_1, \ldots, x_n\}$, under the exponential family model $p(x;\theta)$. Now this Bregman median problem in Θ-space can be transformed into a KL minimization problem using the following Lemma.

Lemma 2 (KL and Bregman for Exponential Family). *Let $KL(\theta_1 \| \theta_2)$ be the KL divergence for $\theta_1, \theta_2 \in \Theta$, then $KL(\theta_1 \| \theta_2) = B_G(\theta_2 \| \theta_1)$*

Proof. This directly follows from the definitions of KL divergence and exponential family; and from the relation, $\mathbb{E}_\theta(x) = \nabla G(\theta)$ for exponential family. □

Theorem 6. *MLE $\hat{\theta}_{ML}$ of data points X_n generated from exponential family is now given by:*

$$\hat{\theta}_{ML} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} KL(\theta_i \| \theta) \tag{8.1}$$

Proof. From Lemma 2, $KL(\theta_i \| \theta) = B_G(\theta \| \theta_i)$, substituting this in (7.4) gives the desired result. □

In (8.1), It is worth nothing that the parameter being estimated comes in the second argument of KL divergence. This observation along with the following definition is used to construct a metric in Θ-space.

Definition 2 (JS Divergence). *For $\theta_1, \theta_2 \in \Theta$ and $\tilde{\theta} = \frac{(\theta_1+\theta_2)}{2}$, $JS(\theta_1, \theta_2)$ is defined as:*

$$JS(\theta_1, \theta_2) = \frac{1}{2}\Big(KL(\theta_1 \| \tilde{\theta}) + KL(\theta_2 \| \tilde{\theta})\Big)$$

It is well known that $\sqrt{JS}$ is a metric. $\sqrt{JS}$ have also been shown to be Hilbertian [Fuglede and Topsoe, 2004; Berg et al., 1984]. A metric $d(x, y)$ is said to be Hilbertian metric if and only if $d^2(x, y)$ is a negative definite(n.d.) [Schoenberg, 1938]. Since $\sqrt{JS}$ is a Hilbertian metric, JS is n.d..

It is important to understand the purpose of the above analysis in deriving the metric based on the JS divergence. I will call this metric as JS metric. This analysis builds a bridge between the JS metric and the generative models. The JS metric can therefore be used to build kernels that can exploit the generative properties of the data. Establishing the connection between the JS metric and the generative models in theory, and showing the efficacy of the kernels based on this metric in practice, is the main contribution of this chapter.

The JS metric, which is based on symmetrized KL divergence has been known for a long time [Cuturi et al., 2005]. However, what is not known is the generative behavior of the JS metric. In the existing literature, JS metric is usually derived for any probability distribution, and for a general probability distribution, JS metric does not consider the generative model of the data, and therefore can not be used to build generative kernels. I, in this work, only consider the distributions belonging to the exponential families, for which, the JS metric can be shown to have been derived considering the generative model. This connection between the JS metric and the generative model allows us to build kernels that can be used to build hybrid models.

Definition 3 (**Dual JS Divergence**). *For $\mu_1, \mu_2 \in \mathcal{M}$ and $\tilde{\mu} = \frac{(\mu_1+\mu_2)}{2}$, let $KL^*(.\|.)$ be the dual of KL divergence then the dual JS divergence $DJS(\mu_1, \mu_2)$ is defined as:*

$$DJS(\mu_1, \mu_2) = \frac{1}{2}\Big(KL^*(\tilde{\mu}\|\mu_1) + KL^*(\tilde{\mu}\|\mu_2)\Big)$$

Theorem 7. *DJS is negative definite.*

Proof. Result is direct consequence of the fact that JS divergence is symmetric. Using the relationship between KL and KL^*, one can simply take the dual of JS which is $DJS(\theta_1, \theta_2) = \frac{1}{2}(KL(\theta_1\|\tilde{\theta}) + KL(\theta_2\|\tilde{\theta})) = JS(\theta_1, \theta_2)$ which is n.d.. $\square$

Theorem 8. *Let $\psi(\mu_1, \mu_2) = DJS(\mu_1, \mu_2)$ be a n.d. function on $\mathcal{M}$, then*

$$\psi(\mu_1, \mu_2) = \frac{F(\mu_1) + F(\mu_2)}{2} - F\Big(\frac{\mu_1 + \mu_2}{2}\Big) \tag{8.2}$$

Proof. Using the duality, $KL^*(\mu_1\|\mu_2) = B_F(\mu_2\|\mu_1)$; $DJS(\mu_1, \mu_2) = \frac{1}{2}(B_F(\mu_1\|\tilde{\mu}) + B_F(\mu_2\|\tilde{\mu}))$. Expanding the expression for the Bregman divergence and using some algebra yields the result. $\square$

Though not analyzed, this expression is also observed in Chen et al. [2008]. It is to be noted that this metric (8.2) is defined over the mean parameters, so in order to define the metric over the data points X_n, one can use Corollary 1, according to which, $\psi(\mu_1, \mu_2) = \psi(\phi(x_1), \phi(x_2))$. Recall $\phi(x)$ here is the sufficient statistics.

8.3 Metric to Kernel

In this section, I convert the previously constructed Hilbertian metric (n.d. function) into a family of kernels that will be used later in the experiments. I now state some results that can be used along with (8.2) to build the kernels called "generative kernels". Although there could be many ways [Berg et al., 1984; Schoenberg, 1938] to transform a metric into a kernel, I mention a few here:

Proposition 1 (Centering). *Let function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function, and $x_0 \in \mathcal{X}$. Let $\varphi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be*

$$\varphi(x, y) = \psi(x, x_0) + \psi(y, x_0) - \psi(x, y) - \psi(x_0, x_0),$$

then φ is n.d. if and only if ψ is positive definite (p.d.).

Proposition 2 (Exponentiated(Exp)). *The function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is n.d. if and only if exp$(-t\psi)$ is p.d. for all $t > 0$.*

Proposition 3 (Inverse). *The function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is n.d. if and only if $(t + \psi)^{-1}$ is p.d. for all $t > 0$.*

Note that the computation complexity of generative kernels depends on the computational complexity of the dual of log-partition function.

8.4 Related Work

In this section, we briefly discuss the kernels that are close to generative kernels. One of the earliest distribution-based kernels is Fisher kernels [Jaakkola and Haussler, 1999]. These kernels are constructed by taking the inner product in the tangent space of Θ using the Fisher information metric. Although constructed in a principled manner, these kernels are impractical due to the intractability of the Fisher information metric computation.

Other kernels (heat kernels) based on the principle of the Fisher information metric are proposed by Lafferty and Lebanon [2005]. Similar to generative kernels, these kernels also consider the geometry of the statistical manifold. Like Fisher kernels, these kernels also suffer from the intractability issue, and so for even most simple geometries, there is closed-form solution.

Jebara et al. [2004] propose a family of kernels which is most similar to generative kernels, and at the same time, is fundamentally very different. Similar to my work, they define kernels by considering the data distribution. Their kernel is defined on the parameters space Θ while my kernel is defined on dual of Θ. For exponential family their results reduce to the Bhattacharyya kernels [Bhattacharyya, 1943] $K(\theta_1, \theta_2) = \exp\Big(G(\frac{\theta_1+\theta_2}{2}) - \frac{G(\theta_1)+G(\theta_2)}{2}\Big)$ while my kernel (Exp form) look like $K(\mu_1, \mu_2) = \exp\Big(F(\frac{\mu_1+\mu_2}{2}) - \frac{F(\mu_1)+F(\mu_2)}{2}\Big)$. In my formulation, we use the dual of the log-partition function F while they use the log-partition function G. These two kernels are very different, one because of the space they are defined over, and other because, there is no explanation as such what divergence (or metric) Bhattacharyya kernels induce on the statistical manifold while generative kernels by construction, induce the Fisher information metric.

There has also been some recent work on the kernels on the probability measures [Martins et al., 2008; Hein and Bousquet, 2005] which are very restricted because they are only defined on the probability measures not on the arbitrary spaces. Although we take a completely different approach to derive the generative kernels, it can be shown that for the exponential family distributions, Jensen Tsallis q-kernels [Martins et al., 2008] reduce to generative kernels for $q = 1$.

As mentioned earlier, generative kernels have the same expression as the semi-group kernels [Cuturi et al., 2005] for the exponential family, though both are derived completely differently. I emphasize that semi-group kernels are defined for the general probability distributions by considering the symmetrized KL divergence, and for general probability distributions, these kernels are not the generative kernels because in general, they do not induce the natural divergence (or natural metric) on the statistical manifold. For general probability distributions, KL divergence is not the natural divergence, therefore use of JS divergence to define the kernel is questionable. Moreover, semi-group kernels do not justify why KL divergence (with $\tilde{\theta}$ in the second argument) is the appropriate divergence. In my work, we take an information geometric approach to derive these kernels and show that these kernels are actually the natural kernel because they induce the Fisher information metric. Semi-group kernels only consider (both in theory and their evaluation) *exp* form of the kernel while we consider a different number of kernels e.g. *exp*, *inv*, *centering*, and study them in discriminative/generative framework. Note that *exp* version usually does not perform as well as others (see Tables 1 and 2).

8.5 Experiments

In this section I evaluate generative kernels on several text categorization tasks (multinomial distributions), as a first evidence of their effectiveness. Later in the proposed work, I plan to apply them for more complex models (tasks). In my preliminary experiments, I use multinomial distribution for this evaluation – mainly for two reasons. First, multinomial is one of the most widely used distributions after Gaussian[1]. Second, other principally similar kernels which I would like to compare against, have been shown to work only for the multinomial geometry, for computational reasons. Note that for multinomial distribution, $\phi(x)$ is simply the observed frequency vector.

I have performed experiments with six kernels: generative(centering, exp and inverse), linear, Heat and PP kernels. In these experiments, generative kernels are compared with some of the best published results for multinomial distribution i.e., heat and PP kernel. In order to see the discriminative/generative behavior, I also include the results of a generative model (NB with α-smoothing). In order to make graphs look less cluttered, I exclude generative centering and linear kernels. In most of the cases they underperformed other kernels. For statistical significance, all of the results are averaged over 20 runs. Although in my results, I only report the mean error, variance was found to be very low (~ 0.0001), and hence not reported for the clarity. For evaluation, I use SVM toolbox[Canu et al., 2005].

[1]Gaussian is uninteresting because all kernels, heat, PP and generative reduce to RBF

Here I report results on the synthetic data (generated from multinomial distribution) because these are more intuitive and give insight into hybrid behavior. I have also performed experiments on real datasets mainly text classification tasks (i.e. multinomial distribution) which can be found in our published work [Agarwal and Daumé III, 2011].

For the multinomial distribution, the relative size of each trial w compared to the dimension of the data d is important because that's what makes problems difficult or easier. if multinomial distributions are considered to be the documents, then long documents compared to the vocabulary size(low d/w, dense setting) makes problem easier while short documents (high d/w, sparse settings) makes the problem difficult. I show results for both dense and sparse settings. For each of these settings, I perform two kind of experiments, In one n is varied (no noise), and in other the noise level($n = 50$) is varied. Noise is introduced by copying the result of previous trial with probability $p =$noise level. In sparse setting $d/w = 100$ while in dense $d/w = 5$ with $w = 20$. These results are shown in Figure 8.2 and Figure 8.3 respectively. Since in all of these experiments, generative kernels outperform other kernel methods, an important comparison would be to see how a method based on generative kernels performs compared to a pure generative method (NB) mainly because discriminative models based on generative kernels can be thought of as a mixture of discriminative/generative models.

In all of these experiments, results are found to be interesting and consistent with previous known facts. In the non-noise dense settings, it is observed that generative kernels based method outperforms all other methods except for NB. It is known [Liang

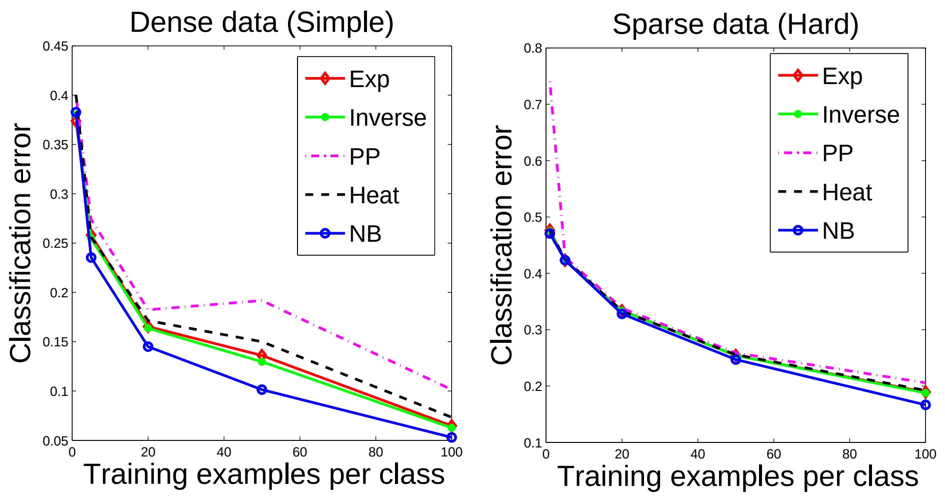


Figure 8.2: Performance variation with n on random multinomial dataset in *sparse* and *dense* settings

and Jordan, 2008] that in case of correct model assumption, generative models outperform discriminative methods hence in this case, one can not hope to beat NB, though it beats all other discriminative models for all n. It is also emphasized that when n is small generative kernels based method performs as well as the NB because for small n, generative properties dominate discriminative properties, but as we increase the data, discriminative properties tend to dominate and model starts to perform poor. Similar results are obtained for the sparse setting except that problem is now harder, and for hard problem, relative difference between generative and discriminative is not very high.

An interesting phenomena occurs when I introduce noise, or when the model is mis-specified. Results are presented in Figure 8.3. For simple problem (dense setting), NB outperforms all discriminative methods for all noise levels. In simple problem, introducing noise does not make much difference, and problem is still simple enough for NB to perform better, therefore, it is the hard problem that is more interesting. In

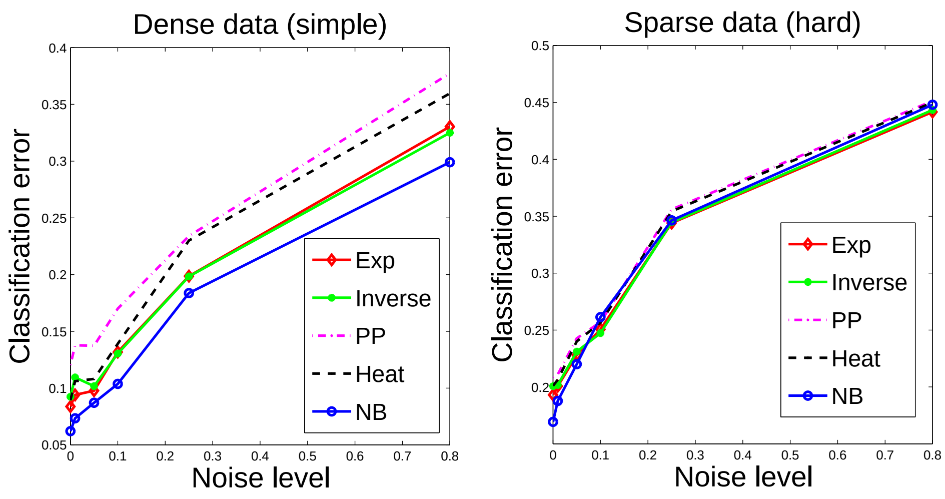


Figure 8.3: Performance variation with different noise levels on random multinomial dataset in *sparse* and *dense* settings

hard problem (sparse setting), NB performs better when there is less noise ($\sim$ 10%), but as we increase the noise, correct model assumption breaks and generative kernels based method starts to outperform.

8.5.1 Real Datasets

For experiments, I consider two standard datasets `WebKB` and `Reuters`[2]. Datasets were preprocessed in a standard manner (short words and stopwords removal, stemmer)[3].

`WebKB` dataset contains web pages classified into four categories, student, faculty, course and student. I take all four classes and construct binary classification tasks by choosing class pairs, giving us a total of six tasks. For two such tasks, performance variation with n is shown in Figure 8.4. These are typical results and were found

[2]Available at `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/` and `http://www.daviddlewis.com/resmyces/testcollections/`

[3]`http://web.ist.utl.pt/~acardoso/datasets/`

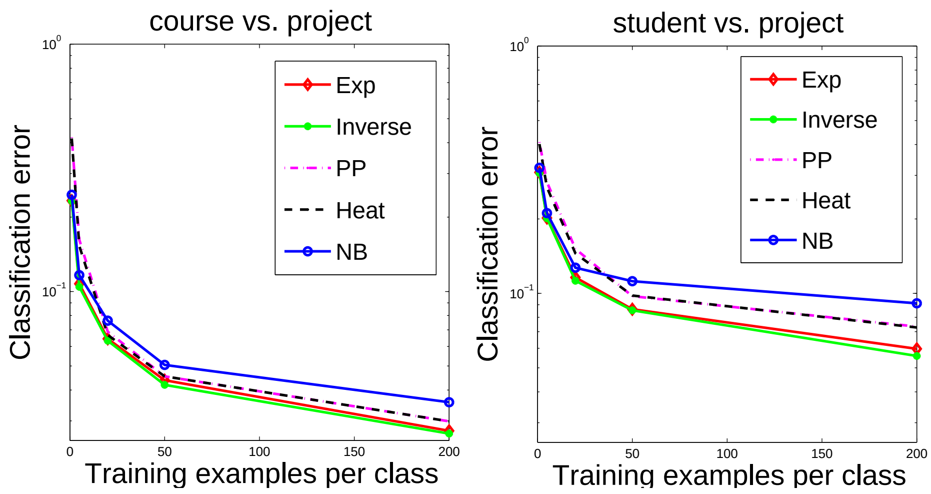


Figure 8.4: Performance variation with n on WebKB dataset. Note that Y-axis is log-scaled hence relative difference is not equal along the X-axis. Although not visible from plot, relative improvement for $n = 1$) is $\sim 10\%$

to be consistent among other tasks. The results are very promising, giving the ideal results for the discriminative/generative hybrid models. Generative kernel curves form the lower envelopes of the discriminative and generative curves. Though, we outperform all models in all cases, we perform significantly better $\sim$ 10%) when n is small. Observe that NB performs better than other discriminative models when n is small, an important property of the generative models [Ng and Jordan, 2002], which we are able to exploit, and perform better/equivalent to generative models for small n, and better than discriminative methods for large n. Table 8.1 shows the results for all six tasks for $n = 20$. For `WebKB` dataset, Generative-centering does not perform very good but other generative kernels outperform all methods. In some case i.e. faculty vs. student difference is as much as 5%.

`Reuters` dataset is a collection of newswire articles classified according to the topics. Although there are more than 140 topics, we take topics with largest number

Table 8.1: Error comparison of generative kernels with other kernels on `WebKB` dataset

Task	**Generative Kernels**			Linear	PP	Heat	Naive
	Centering	Exp	Inverse	Kernel	Kernel	Kernel	Bayes
faculty vs. course	0.1727	0.0443	**0.0413**	0.1268	0.0616	0.0593	0.0719
student vs. project	0.2735	0.1142	**0.1114**	0.2166	0.1386	0.1364	0.1340
course vs. project	0.2120	**0.0602**	**0.0602**	0.1747	0.0699	0.0687	0.0663
faculty vs. project	0.2012	0.1310	**0.1256**	0.2259	0.1539	0.1509	0.2036
faculty vs. student	0.2945	0.1499	**0.1476**	0.2272	0.1926	0.1896	0.2098
student vs. course	0.3165	0.0541	**0.0515**	0.1227	0.0858	0.0819	0.0590

of labeled examples: acq, earn, crude, grain and money-fx. I again consider class pair as a binary classification task which gives us a total of 10 tasks. Performance variation with n for two such tasks is shown in Figure 8.5. Results are similar to the `WebKB` dataset, generative kernel curves being the lower envelopes of all other curves. Table 8.2 shows the results for all 10 tasks for $n = 20$. I see that generative kernels are able to outperform all other methods but here unlike `WebKB` dataset, difference among different version of generative kernels is not as high, in fact all of the generative kernels perform almost equally good.

Table 8.2: Error comparison of generative kernels with other kernels on `Reuters` dataset.

Task	**Generative Kernels**			Linear	PP	Heat	Naive
	Centering	Exp	Inverse	Kernel	Kernel	Kernel	Bayes
acq vs. earn	0.0694	0.0747	**0.0664**	0.0754	0.0673	0.0671	0.0684
acq vs. crude	0.0432	0.0410	**0.0407**	0.0966	0.0472	0.0469	0.0696
acq vs. money-fx	0.0074	0.0078	**0.0071**	0.0596	0.0110	0.0106	0.0191
earn vs. grain	0.0119	0.0142	**0.0116**	0.0776	0.0142	0.0123	0.0164
grain vs. money-fx	0.0138	0.0134	**0.0131**	0.0687	0.0198	0.0194	0.0265
acq vs. grain	**0.0131**	0.0153	0.0134	0.0791	0.0168	0.0168	0.0194
crude vs. money-fx	**0.0082**	0.0096	0.0089	0.0394	0.0121	0.0117	0.0209
earn vs. money-fx	**0.0117**	0.0142	0.0121	0.0567	0.0142	0.0135	0.0199
earn vs. crude	**0.0326**	0.0366	0.0329	0.0814	0.0363	0.0348	0.0450
grain vs. crude	**0.0179**	0.0194	0.0183	0.0705	0.0246	0.0246	0.0448

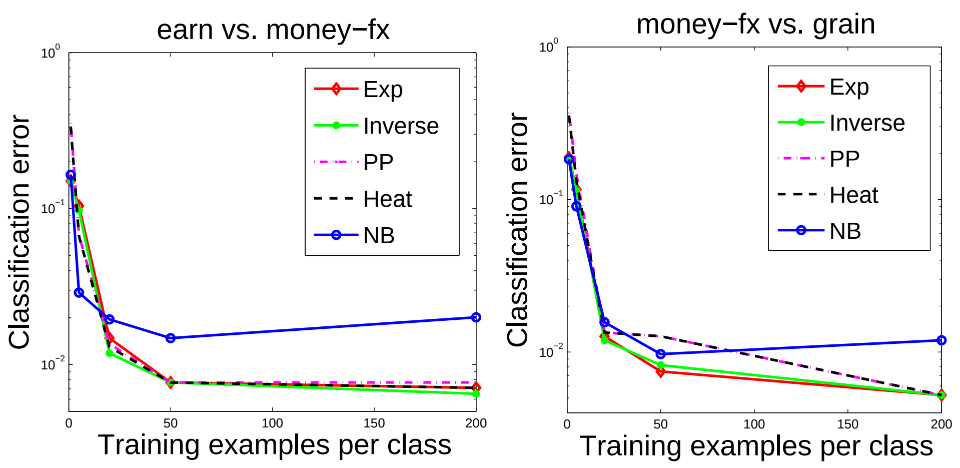


Figure 8.5: Performance variation with n on *Reuters* dataset. Y-axis is log-scaled.

8.6 Conclusion and Future Work

In this chapter, I have used the geometric structure of the space associated with the generative models to build a family of kernels called *generative kernels* that can be used as off-the-shelf tool in any kernel learning method. When used in discriminative models, these kernels exploit the properties of both, generative and discriminative models, giving us the best of both worlds. Empirical results demonstrate this *hybrid* behavior, in addition to the superior performance over the other state-of-the-art kernels.

Part III

Conclusion and Future Work

This part presents the conclusion and findings of my dissertation.

Chapter 9

Summary and Future Work

My dissertation demonstrates the power of geometric methods by applying them to a variety of problems, both in data mining and machine learning. For these problems, I show that the geometric methods are useful not only to gain insight into existing algorithms, but also to build new and effective algorithms.

In data mining, I use geometric methods to solve a well known distance embedding problem known as multidimensional scaling (MDS). These geometric methods result in a powerful unified algorithmic framework that can be used to solve a number of MDS variants. Furthermore, I modify the algorithms derived from this framework to develop a scalable algorithm for dimensionality reduction, and also to solve a sensor network localization problem.

In machine learning, I apply geometric methods to three problems: First, I use them to reason about the appropriateness of conjugate priors, second, to use them to build a hybrid model by combing discriminative and generative models using the geometry underlying these models, and finally, to build a set of kernels called generative kernels using the geometric structure underlying the generative process of data. In all of these three problems, geometric methods exploit geometric structures underlying probabilistic models, giving us powerful tools not only to analyze existing algorithms but also to build new algorithms.

9.1 Summary

Below, I enumerate problems that I solve in my dissertation, and discuss in detail the contributions and findings in each problem. A list of relevant publications that came out of this dissertation can be found in Appendix C.

Generalized Multidimensional Scaling: I have demonstrated the power of geometric methods by building a unified algorithmic framework to solve the *generalized* multidimensional scaling (MDS) problem. This framework not only solves the original MDS problem in *Euclidean* space more effectively than state-of-the-art methods, but also generalizes to other spaces e.g., *spherical* space; and has many desirable properties. In addition to the formal guarantees of convergence, the algorithms derived from this framework are accurate; in most cases, they converge to better quality solutions than existing methods in comparable time. Moreover, they have a small memory footprint and scale effectively for large data sets.

Large Scale Dimensionality Reduction: I have extended my generalized MDS framework to build scalable algorithms for dimensionality reduction. In particular, I have modified the MDS framework such that the resulting framework has a time complexity that is *linear* in data size, and a memory complexity that is *independent* of data size. My approach combines *adaptive* sampling with multipass algorithm, which avoids expensive eigenvalue computations and uses a non-linear projection. The approach generalizes to other distance measures and cost metrics, and can run MDS efficiently at data sizes well beyond what has previously been demonstrated for the same accuracy.

Sensor Network Localization: I have also modified my generalized MDS framework to solve the sensor network localization problem for mobile sensors. The resulting algorithm is accurate, and scales to large data effectively i.e., it can handle large number of sensors more efficiently that what has previously been reported. This algorithm is the first ever algorithm that has been tested with number of sensors more than $100,000$ with *noisy* distance measurements — it can localize $120,000$ sensors in less than 20 minutes. The experimental study across a variety of networks with different moving patterns show that the proposed algorithm outperforms the existing state-of-the-art methods. In our experiments, it is especially found to be effective for real-time tracking for mobile sensors, as it takes only a fraction of second for a tracking problem with 1000 sensors. Furthermore, this algorithm is a decentralized algorithm, which means that the localization computation can be carried out in the sensor itself needing only the locations of its neighbors, and thus, adding less communication overhead which is a significant bottleneck in sensor network localization problems.

Conjugate Priors and Hybrid Models: In my dissertation, I have shown that the geometric structure associated with statistical models can be used to gain insight into existing algorithms by using it to reason about the appropriateness of conjugate priors, and then using this reasoning to justify the use of these conjugate priors in hybrid models. These hybrid models are models that hybridize between discriminative and generative models. In my work, I first demonstrate that using a conjugate prior as a coupling prior is more effective than using a non-conjugate prior (as done previously), and then show that the hybrid model built using conjugate prior as coupling prior can be built more *naturally* by simply interpolating between the participating generative and discriminative models using the *distance geometry* underlying these models, and thus justifying the use of conjugate priors to build these hybrid models in the first place.

Generative Kernels: My dissertation shows that the geometric structure associated with the statistical process of a learning problem can also be used to build better machine learning tools — later to be used in various other algorithms — by building a family of kernels called *generative kernels*. My proposed method considers the geometry of the space of statistical models associated with the distribution that generated data, to build a set of efficient *closed-form* kernels best suited for that distribution. Since these kernels are built considering the geometry associated with the generative model, and are used in discriminative models as in SVM, they naturally combine the power of both discriminative and generative models, thus give us the hybrid behavior. My experiments demonstrate this hybrid behavior, that is, the performance of an SVM using generative kernels is better that an SVN using other kernels, and also better than a pure generative model.

9.2 Future Work

Here I discuss future directions of my research in the context of the problems considered in this dissertation.

9.2.1 Distance Embedding

As mentioned earlier, the distance embedding is a generic class of problems encompassing various other problems that take distance matrix as input and output a set of points in a given space. These problems differ from each other depending on the assumptions made about the input and the output. The multidimensional scaling (MDS) problem is one such problem where the input is set of pairwise distances among *all* pairs, and the output is a set of points in a given space. In my dissertation, I have used geometric methods to build a unified algorithmic framework to solve the MDS problem that performs better than existing methods for a variety of target spaces and cost functions. In my experiments, the performance of MDS is measured by its ability to converge to a solution (distortion cost). One algorithm is said to be better than the other when it converges to a lower distortion cost in a comparable or lesser time.

Although useful in many cases, such a *relative* measurement has limited use in the absence of any *absolute* measurement of the quality of the solution. Ideally one would like to know the quality of the solution with respect to the best solution i.e. global optimal, but unfortunately, the global optimal solution is not available for such problems. In fact, there exists no work in the existing literature on MDS or in general on distance embedding problems, that provides theoretical guarantees on the quality of a solution. Consequently, one natural future direction of this work would be to extend it to do a theoretical analysis so as to determine the quality of the embedding in an absolute sense or relative to the global optimal. Since my geometric framework allows one to handle many variants of MDS with different cost functions and different target spaces, a theoretical analysis in the context of the proposed framework will naturally extend to other spaces and cost functions, leading to powerful MDS algorithms with guarantees on the quality of the solution.

9.2.2 Spherical Embedding

In this dissertation, I have applied my generalized MDS framework to solve a spherical embedding problem. Recall from our earlier discussion that the spherical embedding problem is useful to reduce the dimension of histograms, and therefore, the dimension of the text documents because text documents are nothing but histograms when represented by normalized frequency vectors. One future avenue of my research would be to show the effectiveness of spherical embedding in the context of text analysis.

It is well known [Kass and Vos, 1997; Lebanon and Lafferty, 2004] that text documents, when represented as term-frequency (tf) vectors can be isometrically embedded into positive orthant of a sphere, under the assumption that the text documents are generated from a multinomial distribution, and a natural distance between these documents is *information distances* (computed using Fisher information metric on multinomial distribution). Since documents are represented as term-frequency vectors, they are naturally of high dimension which being as high as the size of the vocabulary. In many machine learning problems, this high dimensional nature of documents poses several challenge i.e., high computational complexity, sparsity. One obvious way to tackle these challenges is to preprocess the documents and reduce their dimension.

There are many methods that can be used to reduce the dimension of documents, one natural method would be the one that preserves their geometric structure. Note that text documents have an inherent geometry to them i.e., they lie on a high-dimensional sphere, so when reducing the dimension of these documents, it would be useful to preserve this inherent spherical property (they must lie on low-dimensional sphere), a problem that is exactly solved by spherical MDS. Once the documents have been processed to reduce their dimension, their geometric structure can be exploited when performing a further analysis task on these documents. One method that exploits the spherical structure of text documents in a classification task is the method proposed by Lebanon [2005]. In his work, Lebanon assumes that text documents lie on a sphere and classifies them using a hyperplane defined on the sphere. Hence one natural extension of spherical embedding framework would be to reduce the dimension of text documents, and then use the classification method presented by Lebanon [2005] which will hopefully lead to a better classification for text documents.

Parallelization. Although I have mentioned that the proposed unified MDS framework is easily parallelizable, a practical evidence showing the effectiveness of such a parallel framework is yet to be shown. In our geometric framework, an update of a point requires the most updated positions of other points, which means that before an update can be made, positions of all other points have to be communicated to the node where the point being updated resides. Such a parallelism would not only add communication overhead, but it will also break convergence guarantees. Note that the convergence guarantees that the cost function monotonically decreases in each update only applies when these updates take place in a sequential manner.

9.2.3 Senor Network Localization

Recall that the SNL problem is a generalized version of the MDS problem because in SNL, we are given a subset of all distances (i.e. distances among few pairs), and the positions of some of the points (i.e. anchors), both of which can be adjusted to give the original MDS problem. More specifically, when the given subset includes all pairwise distances, and the number of anchors is zero, one recovers the standard Euclidean MDS problem. Note that this seemingly simple generalization of MDS to SNL brings in significant difficulties when solving the problem, and have been shown to be closely related to other problems such as Euclidean distance matrix completion, graph rigidity, graph realization and many more [Liberti et al., 2012]. Since in SNL, one is interested in recovering *actual* locations of the sensors, it is important that the underlying optimization problem has one unique solution because if there are multiple solutions to the optimization problem, we might end up finding the solution that has the same cost as the actual solution but is different in terms of the positions of the sensors); and figuring out if there exists a unique solution to the given set of distances in itself is a challenging problem, and is an active area of research [Hendrickson, 1992;

Alfakih, 2001, 2005; Connelly, 2005; BELK and CONNELLY, 2007; Connelly, 2009].

One of the main challenges when dealing with an SNL problem can be attributed to the *partial* (and noisy) distance matrix. In order to understand this, consider an extreme case of the problem, i.e., consider the case when we are not given *any* distances. In such a case, it is easy to see that any solution would be a solution to the underlying optimization problem, however it would not be a solution of the actual SNL problem. Even assuming that such extreme and degenerate cases do not exist, i.e., problem is well behaved in the sense it has a unique solution, finding the one is still a challenge. In the context of the work presented in this dissertation, this challenge comes from the fact that the partial distance makes algorithm *more* susceptible to local miinima, compared to the case when the distance matrix is full or relatively dense.

One way to alleviate the problem of local minimia is to consider non-neighbor points, and hence the distances from them making the distance matrix denser. That is, when updating for one point x_i, one should not only consider points that are neighbors of the point x_i, but also the points that are not neighbors. More specifically, one should try to position the point x_i relative to the neighbors x_j at the given distances d_{ij}, while at the same time, position it so that it is more than R (communication radius) distance away from non-neighbors. Note that in the SNL problem definition, any point that is less than R distance away from x_i is a neighbor of x_i. Mathematically, it should optimize the following cost function:

$$C_i(Y, S, x_i) = \sum_{\substack{x_j \in Y \\ d_{ij} \in S}} \left(\|x_i - x_j\|_2 - d_{ij} \right)^2$$

$$\text{s.t.} \quad \|x_i - x_j\|_2 \geq R \; \forall x_j \notin Y.$$

Note the difference between the above problem from the problem (5.2) on page 100. The above problem has additional constraints that take into account repulsion from non-neighbors. A similar formulation where both neighbors and non-neighbors constraints are enforced to place a point is also considered in the work of Pong and Tseng [2009].

Identifying how to minimize the above cost function would be one direction of future research. One possible way to solve this problem is to transform it into an unconstrained optimization problem using a regularization parameter as done in Chen and Buja [2009]:

$$C_i(Y, S, x_i) = \sum_{\substack{x_j \in Y \\ d_{ij} \in S}} \left(\|x_i - x_j\|_2 - d_{ij}\right)^2 + \lambda \sum_{x_j \notin Y} \left(\|x_i - x_j\|_2 - R\right)^2$$

I believe that my MDS framework can be extended to solve the above unconstrained optimization problem, since this is nothing but a special instance of (2.3) on page 26 with $w_{ij} = \lambda, \; \forall x_j \notin Y$.

9.2.4 Generative Kernels

In my dissertation, I have used generative kernels built using geometry underlying probabilistic models that represent the generative process of a problem, and shown their effectiveness on a document classification task where the generative process consisted of a multinomial distribution. Although in principle, these generative kernels can be derived for any exponential family models, one has yet to establish practical evidences. In machine learning, many exponential family models can be represented as graphical models, and one future direction of this dissertation in the context of generative kernels would be to build these kernels for such graphical models. Recall that the computation of these kernels requires the computation of the *dual* of the log partition function F, an important extension of this work would be to study how to compute F. Once the dual of the log partition function is known, computing the kernel is straightforward.

9.3 Conclusion

I conclude my dissertation with the remark that the geometric methods are indeed powerful tools. The understanding of the geometric structure underlying a problem and then using it appropriately results in algorithms that are helpful in many ways. In addition to being simple and intuitive, these algorithms, in most cases, are more effective than existing algorithms, and often provide insight into problems and existing algorithms.

Appendix A

Exponential Family and their Properties

In chapters 7 and 8, I use the geometry underlying exponential family models to reason about *conjugate priors*, and build *generative kernels*. This chapter gives the required background, in particular, concepts related to Legendre duality, Bregman divergence, and exponential families.

A.1 Legendre Duality

Let $\mathcal{M} \subseteq \mathbb{R}^d$ and $\Theta \subseteq \mathbb{R}^d$ be two spaces and let $F : \mathcal{M} \rightarrow \mathbb{R}^+$ and $G : \Theta \rightarrow \mathbb{R}^+$ be two non-negative real valued convex functions. F and G are *conjugate duals* of each other if:

$$F(\mu) := \sup_{\theta \in \Theta}\{\langle \mu, \theta \rangle - G(\theta)\}, \quad \forall \mu \in \mathcal{M} \tag{A.1}$$

and

$$G(\theta) := \sup_{\mu \in \mathcal{M}}\{\langle \theta, \mu \rangle - F(\mu)\}, \quad \forall \theta \in \Theta \tag{A.2}$$

here $\langle a, b \rangle$ denotes the dot product of vectors a and b, $\mathbb{R}$ denotes the space of real numbers, $\mathbb{R}^d$ denotes the d-dimensional space of real numbers, and $\mathbb{R}^+$ denotes the space of non-negative real numbers. The spaces (Θ and $\mathcal{M}$) associated with

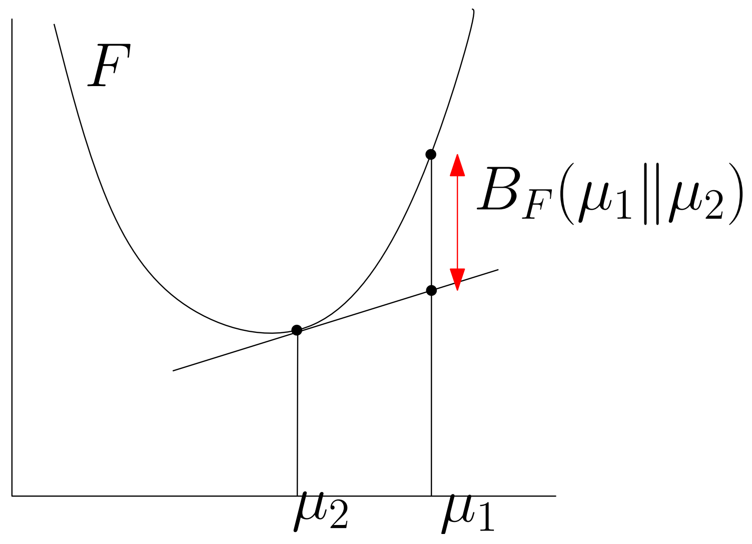


Figure A.1: Geometric representation of Bregman divergence

these dual functions are called *dual spaces*; this relationship between them is know as Legendré duality or just the duality. There are standard notations to refer this duality i.e., $G = F^*$ and $F = G^*$. A particularly important connection between dual spaces is that: for each $\mu \in \mathcal{M}, \nabla F(\mu) = \theta \in \Theta$ (denoted as $\mu^* = \theta$)) and similarly, for each $\theta \in \Theta, \nabla G(\theta) = \mu \in \mathcal{M}$ (or $\theta^* = \mu$)). For more details, refer to [Rockafellar, 1996].

A.2 Bregman Divergence

I now give a brief overview of Bregman divergence (for more details see [Banerjee et al., 2005]). Let $F : \mathcal{M} \rightarrow \mathbb{R}^+$ be a continuously-differentiable real-valued and strictly convex function. The Bregman divergence (see Figure A.1) associated with F for points $\mu_1, \mu_2 \in \mathcal{M}$ is:

$$B_F(\mu_1 \| \mu_2) = F(\mu_1) - F(\mu_2) - \langle \nabla F(\mu_2), (\mu_1 - \mu_2) \rangle \tag{A.3}$$

If G is the *conjugate dual* of F then:

$$B_F(\mu_1 \| \mu_2) = B_G(\mu_2^* \| \mu_1^*) \tag{A.4}$$

here μ_1^* and μ_2^* are the duals of μ_1 and μ_2 respectively. Note that the arguments are flipped in (A.4). It is emphasized that Bregman divergence is not symmetric i.e., in general, $B_F(p\|q) \neq B_F(q\|p)$, therefore it is important what directions these divergences are measured in.

A.3 Exponential Family

The exponential family is a set of distributions defined over a space $\mathcal{X}$, whose probability density function for some $x \in \mathcal{X}$ can be expressed in the following form:

$$p(x;\theta) = p_o(x)\exp(\langle \theta, \phi(x) \rangle - G(\theta)) \tag{A.5}$$

here $\phi(x) : \mathcal{X} \to \mathbb{R}^d$ is a vector *potentials* or *sufficient statistics* [Nielsen, 1978], $p_o(x)$ is a function that depends on the base measure ν i.e., $d\nu = p_0(x)\,dx$, and $G(\theta)$ is a normalization constant or *log-partition function*. With the potential functions $\phi(x)$ fixed, every θ induces a particular member $p(x;\theta)$ of the family. In the proposed research, I deal with exponential families that are *regular* and have the *minimal representation*. (For details on this, refer to [Wainwright and Jordan, 2008]).

The exponential family has a number of convenient properties and subsumes many common distributions. It includes the Gaussian, Binomial, Beta, Multinomial and Dirichlet distributions, Bayes nets, etc.. Below I give two examples of Naive Bayes and Gaussian distributions to show that they are exponential family distributions by representing them in the canonical form (A.5)

Gaussian. The distribution can be written as:

$$
\begin{aligned}
p(x;a) &= \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{1}{2\sigma^2}\|x-a\|^2\right) \\
&= \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \quad \exp\left(\langle x, \frac{a}{\sigma^2}\rangle - \frac{1}{\sigma^2}\|a\|^2 - \frac{1}{\sigma^2}\|x\|^2\right) \\
&= \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{1}{\sigma^2}\|x\|^2\right) \quad \exp\left(\langle x, \frac{a}{\sigma^2}\rangle - \frac{1}{\sigma^2}\|a\|^2\right) \\
&= p_o(x) \quad \exp(\langle x, \theta\rangle - G(\theta)),
\end{aligned}
$$

where $p_o(x)$ is independent of θ, and comparing with (A.5), $\theta = \frac{a}{\sigma^2}$ and $G(\theta) = \frac{\sigma^2}{2}\|\theta\|^2$.

Naive Bayes. The distribution can be written as following considering that each feature x_j is binary and is generated by Bernoulli distribution:

$$
\begin{aligned}
p(x;a) &= \prod_{j=1}^{d} a_j^{x_j}(1-a_j)^{(1-x_j)} \\
&= \exp\left(\log\left(\prod_{j=1}^{d} a_j^{x_j}(1-a_j)^{(1-x_j)}\right)\right) \\
&= \exp\left(\sum_{j=1}^{d} \log\left(a_j^{x_j}(1-a_j)^{(1-x_j)}\right)\right) \\
&= \exp\left(\sum_{j=1}^{d} \left(x_j \log a_j + (1-x_j)\log(1-a_j)\right)\right)
\end{aligned}
$$

$$= \exp\left(\sum_{j=1}^{d}\left(x_j \log\frac{a_j}{1-a_j} + \log(1-a_j)\right)\right)$$

$$= \exp(\langle x, \theta\rangle - G(\theta))$$

where $p_o(x) = 1$, $\theta_j = \log\frac{a_j}{1-a_j}$, and $G(\theta) = \sum_{j=1}^{d} \log(1-a_j)$.

One important property of the exponential family is the existence of conjugate priors. Given any member of the exponential family as in (A.5), the *conjugate prior* is a distribution over its *parameters* with the following form:

$$p(\theta|\alpha,\beta) = m(\alpha,\beta)\exp(\langle\theta,\alpha\rangle - \beta G(\theta)) \tag{A.6}$$

here α and β are hyperparameters of the conjugate prior. Importantly, the function $G(\cdot)$ is same between the exponential family member and its conjugate prior.

For exponential family, there exist a relationship between the log-partition function $G(\theta)$ and the sufficient statistics. In particular, we have:

$$\frac{\partial G}{\partial\theta} = \mathbb{E}_\theta\left[T(\mathbf{x})\right] \tag{A.7}$$

A second important property of exponential family member is that log-partition function G is a continuously differentiable *strictly* convex function and defined over the convex set $\Theta := \{\theta \in \mathbb{R}^d : G(\theta) < \infty\}$. These properties of log-partition function induce a Bregman divergence on the space Θ. Convexity of log-partition function G ensures that there exists a space $\mathcal{M}$, dual to Θ and a dual function F defined over $\mathcal{M}$. Here duality refers to Legendre duality. This convexity property also induces a

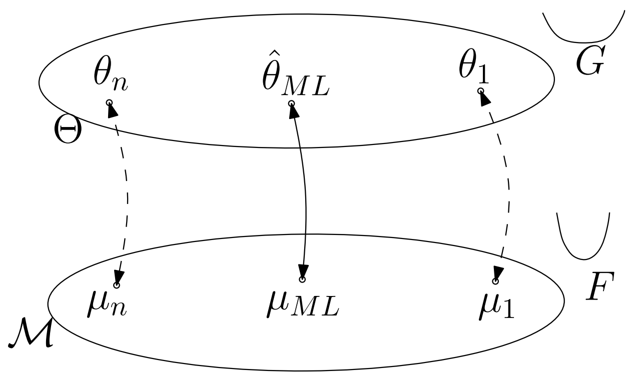


Figure A.2: Duality between mean parameters and canonical parameters. Notice the convex functions defined over both spaces. these functions are dual of each other and so are the spaces.

Bregman divergence on both Θ and $\mathcal{M}$.

Another important property of the exponential family is the *one-to-one* mapping between the *canonical parameters* θ and the so-called "*mean parameters*" which we denote by μ. For each canonical parameter $\theta \in \Theta$, there exists a unique corresponding mean parameter μ, which belongs to the space $\mathcal{M}$ defined as:

$$\mathcal{M} := \left\{ \mu \in \mathbb{R}^d : \mu = \int \phi(x) p(x;\theta)\, dx, \quad \theta \in \Theta \right\} \tag{A.8}$$

My notation in the above expression has been deliberately suggestive. Θ and $\mathcal{M}$ are dual spaces, in the sense of Legendre duality because of the following relationship between the log-partition function $G(\theta)$ and the expected value of the sufficient statistics $\phi(x)$:

$$\nabla G(\theta) = \mathbb{E}(\phi(x)) = \int \phi(x) p(x;\theta)\, dx = \mu.$$

In Legendre duality, we know that two spaces Θ and $\mathcal{M}$ are dual of each other if for each $\theta \in \Theta$, $\nabla G(\theta) = \mu \in \mathcal{M}$. Under this duality, there exists a dual function F

on $\mathcal{M}$ such that $F = G^*$. A pictorial representation of the duality between canonical parameter space Θ and mean parameter space $\mathcal{M}$ is given in Figure A.2.

Appendix B

Proof of JL Lemma for Spherical Data

Before we proceed with the proof, we require a key technical lemma.

Lemma 3. *For $\varepsilon \in [0, 0.5]$ and $x \in [0, 0.7]$,*

(1) $\sin((1-2\varepsilon)x) \leq (1-\varepsilon)\sin(x)$*, and*

(2) $\sin((1+2\varepsilon)x) \geq (1+\varepsilon) sin(x)$*.*

Proof. Let $g_\varepsilon(x) = (1-\varepsilon)\sin x - \sin((1-2\varepsilon)x)$. We will show that for $x \in [0,1]$ and $\varepsilon \in [0, 0.5]$, $g_\varepsilon(x)$ is concave. This implies that it achieves its minimum value at the boundary. Now $g_\varepsilon(0) = 0$ for all ε, and it can be easily shown that $g_\varepsilon(0.7) \geq 0$ for $\varepsilon \in [0, 0.5]$. This will therefore imply that $g_\varepsilon(x) \geq 0$ in the specified range.

It remains to show that $g_\varepsilon(x)$ is concave in $[0, 0.7]$.

$$
\begin{aligned}
g''_\varepsilon(x) &= (1-2\varepsilon)^2 \sin((1-2\varepsilon)x) - (1-\varepsilon)\sin x \\
&\leq (1-\varepsilon)(\sin((1-2\varepsilon)x) - \sin x)
\end{aligned}
$$

which is always negative for $\varepsilon \in [0, 0.5]$ and since $\sin x$ is increasing in the range $[0, 0.7]$.

This proves the first part of the lemma. For the second part, observe that $h_\varepsilon(x) = \sin((1+2\varepsilon)x) - (1+\varepsilon) sin(x)$ can be rewritten as $h_\varepsilon(x) = g_{-\varepsilon}(-x)$. The rest

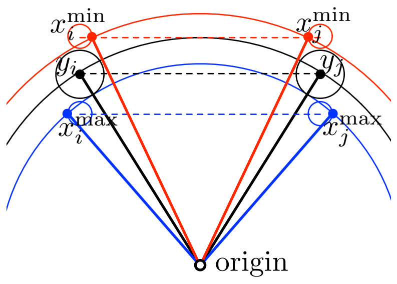


Figure B.1: Illustration of the bounds on $\angle_{x_i,x_j}$ when $||y_i - y_j|| \leq 1/2$. The angle $\angle_{x_i^{\max},x_j^{\max}}$ is the largest when $||x_i^{\max}|| = ||x_i^{\max}||$ is as small as possible (lies on inner circle) and $||x_i^{\max} - x_j^{\max}||$ is as large as possible (on the outer edges of the disks of diameter $\varepsilon/8$ shifted down from dashed line of length $||y_i - y_j||$. Bounds for $x_i^{\min}$ and $x_j^{\min}$ are derived symmetrically.

of the argument follows along the same lines, by showing that $h_\varepsilon(x)$ is concave in the desired range using that $h''_\varepsilon(x) = g''_{-\varepsilon}(-x)$.

While the upper bound of 0.7 on x is not tight, it is close. The actual bound (evaluated by direct calculation) is slightly over 0.72. □

Proof of Theorem 3. Let $X = \pi_H(Y)$. We consider two cases, (*Short Case*) when $||y_i - y_j|| \leq 1/2$ and (*Long Case*) when $||y_i - y_j|| \in (1/2, 2]$.

Short Case: First consider points $y_i, y_j \in \mathbb{S}^d$ such that $||y_i - y_j|| \leq 1/2$. Note that $||y_i - y_j|| = 2\sin(\angle_{y_i,y_j}/2)$, since $||y_i|| = ||y_j|| = 1$. By JL, we know that there exists a constant c such that

$$(1 - \varepsilon/8)||y_i - y_j|| \leq c||x_i - x_j|| \leq (1 + \varepsilon/8)||y_i - y_j||.$$

We need to compare the angle $\angle_{x_i,x_j}$ with that of $\angle_{y_i,y_j}$. The largest $\angle_{x_i,x_j}$ can be is when $c||x_i|| = c||x_j|| = (1 - \varepsilon/8)$ is as small as possible, and so $||cx_i - cx_j|| =$

$(1+\varepsilon/8)||y_i - y_j||$ is as large as possible. See Figure B.1. In this case, we have

$$\begin{aligned}
(||cx_i|| + ||cx_j||)\sin(\angle_{x_i,x_j}/2) &\leq ||cx_i - cx_j|| \\
2(1-\varepsilon/8)\sin(\angle_{x_i,x_j}/2) &\leq (1+\varepsilon/8)||y_i - y_j|| \\
2(1-\varepsilon/8)\sin(\angle_{x_i,x_j}/2) &\leq (1+\varepsilon/8)2\sin(\angle_{y_i,y_j}/2) \\
\sin(\angle_{x_i,x_j}/2) &\leq \frac{1+\varepsilon/8}{1-\varepsilon/8}\sin(\angle_{y_i,y_j}/2),
\end{aligned}$$

which for $\varepsilon < 4$ implies

$$\sin(\angle_{x_i,x_j}/2) \leq (1+\varepsilon/2)\sin(\angle_{y_i,y_j}/2).$$

Similarly, we can show when $\angle_{x_i,x_j}$ is as small as possible (when $c||x_i|| = c||x_j|| = (1+\varepsilon)$ and $||cx_i - cx_j|| = (1-\varepsilon)||y_i - y_j||$), then

$$(1-\varepsilon/2)\sin(\angle_{y_i,y_j}/2) \leq \sin(\angle_{x_i,x_j}/2).$$

We can also show (via Lemma 3) that since $||y_i - y_j|| \leq 1/2$ implies $\angle_{y_i,y_j} < 0.7$ we have

$$\sin((1-\varepsilon)\angle_{y_i,y_j}) \leq (1-\varepsilon/2)\sin(\angle_{y_i,y_j})$$

and

$$(1+\varepsilon/2)\sin(\angle_{y_i,y_j}) \leq \sin((1+\varepsilon)\angle_{y_i,y_j}).$$

Thus, we have

$$\begin{array}{rcccl}\sin((1-\varepsilon)\angle_{y_i,y_j}/2) & \leq & \sin(\angle_{x_i,x_j}/2) & \leq & \sin((1+\varepsilon)\angle_{y_i,y_j}/2) \\ (1-\varepsilon)\angle_{y_i,y_j}/2 & \leq & \angle_{x_i,x_j}/2 & \leq & (1+\varepsilon)\angle_{y_i,y_j}/2 \\ (1-\varepsilon)\angle_{y_i,y_j} & \leq & \angle_{x_i,x_j} & \leq & (1+\varepsilon)\angle_{y_i,y_j}.\end{array}$$

Long Case: For $||y_i - y_j|| \in (1/2, 2]$, we consider 6 additional points $y_{i,j}^{(h)} \in \mathbb{S}^{d+1}$ (for $h \in [1:6]$) equally spaced between y_i and y_j on the shortest great circle connecting them. Let $\hat{Y}$ be the set Y plus all added points $\{y_{i,j}^{(h)}\}_{h=[1:6]}$. Note that $|\hat{Y}| = O(n^2)$, so by JL we have that

$$(1-\varepsilon/8)||y_i - \hat{y}_{i,j}|| \leq c||x_i - \hat{x}_{i,j}|| \leq (1+\varepsilon/8)||y_i - \hat{y}_{i,j}||.$$

For notational convenience let $y_i = y_{i,j}^{(0)}$ and $y_j = y_{i,j}^{(7)}$. Since for $||y_i - y_j|| \in (1/2, 2]$ then $||y_{i,j}^{(h)} - y_{i,j}^{(h+1)}|| \leq 1/2$, for $h \in [0:6]$. This follows since the geodesic length of the great circular arc through y_i and y_j is at most π, and $\pi/7 < 1/2$. Then the chordal distance for each pair $||y_{i,j}^{(h)} - y_{i,j}^{(h+1)}||$ is upper bounded by the geodesic distance. Furthermore, by invoking the short case, for any pair

$$(1-\varepsilon)\angle_{y_{i,j}^{(h)},y_{i,j}^{(h+1)}} \leq \angle_{x_{i,j}^{(h)},x_{i,j}^{(h+1)}} \leq (1+\varepsilon)\angle_{y_{i,j}^{(h)},y_{i,j}^{(h)}}.$$

Then since projections preserve coplanarity (specifically, the points 0 and $y_{i,j}^{(h)}$ for $h \in [0:7]$ are coplanar, hence 0 and $x_{i,j}^{(h)}$ for $h \in [0:7]$ are coplanar), we can add together the bounds on angles which all lie on a single great circle.

$$\begin{aligned}(1-\varepsilon)\angle_{y_i,y_j} &\leq (1-\varepsilon)\sum_{h=0}^{6}\angle_{y_{i,j}^{(h)},y_{i,j}^{(h+1)}}\\ &\leq \sum_{h=0}^{6}\angle_{x_{i,j}^{(h)},x_{i,j}^{(h+1)}}\\ &\leq (1+\varepsilon)\sum_{h=0}^{6}\angle_{y_{i,j}^{(h)},y_{i,j}^{(h+1)}}\\ &\leq \min\{\pi,(1+\varepsilon)\angle_{y_i,y_j}\}\end{aligned}$$

and thus by $\angle_{x_i,x_j}=\sum_{h=0}^{6}\angle_{x_{i,j}^{(h)},x_{i,j}^{(h+1)}}$ implies

$$(1-\varepsilon)\angle_{y_i,y_j}\leq\angle_{x_i,x_j}\leq\min\{\pi,(1+\varepsilon)\angle_{y_i,y_j}\}.$$

□

Appendix C

List of Publications

Table C.1: List of publications from the work completed

Contribution	**Publication**
Geometric analysis of conjugate priors and its applications to hybrid model	[Agarwal and Hal Daumé III, 2010] (**Best Student Paper)**
Semi-supervised hybrid model using conjugate prior	[Agarwal and Daumé III, 2009]
Generative kernels for exponential family distributions	[Agarwal and Daumé III, 2011]
A unified framework for generalized MDS	[Agarwal et al., 2010b]
Sensor Network Localization for Moving Sensors	[Agarwal et al., 2012]
Adaptive Sampling for Large Scale MDS	In preparation

Bibliography

Agarwal, A. and Daumé III, H. (2009). Exponential family hybrid semi-supervised learning. In *International Joint Conference on Artificial Intelligence*, Pasadena, CA. 142, 143, 145, 155, 205

Agarwal, A. and Daumé III, H. (2011). Generative kernels for exponential families. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 164, 175, 205

Agarwal, A., Daumé III, H., Phillips, J. M., and Venkatasubramanian, S. (2012). Sensor Network Localization for Moving Sensors. In *The Second IEEE ICDM Workshop on Data Mining in Networks (To appear)*. IEEE. 205

Agarwal, A. and Hal Daumé III (2010). A geometric view of conjugate priors. *Machine Learning Journal*, 81:99–113. 205

Agarwal, A., Phillips, J. M., and Venkatasubramanian, S. (2010a). Universal multi-dimensional scaling. In *KDD*. 59, 73

Agarwal, A., Phillips, J. M., and Venkatasubramanian, S. (2010b). Universal multi-dimensional scaling. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1149–1158. ACM. 205

Agarwal, P. K., Har-Peled, S., and Yu, H. (2007). On embeddings of moving points in Euclidean space. In *Proceedings 23rd Symposium on Computational Geometry*. 50, 57

Alfakih, A. (2001). On rigidity and realizability of weighted graphs. *Linear Algebra Appl.*, 325(1-3):57–70. 190

Alfakih, A. Y. (2005). On the set of realizations of edge-weighted graphs in Euclidean spaces. Research Report, Mathematics and Statistics, University of Windsor. 190

Amari, S.-I. (1990). *Differential-Geometrical Methods in Statistics (Lecture Notes in Statistics 28)*. Springer. 2, 3, 4

Amari, S.-I. and Nagaoka, H. (2001). *Methods of Information Geometry (Translations of Mathematical Monographs)*. American Mathematical Society. 2, 3, 5, 145, 154, 167

Artač, A., Jogan, M., and Leonardis, A. (2002). Incremental PCA for on-line visual learning and recognition. In *16th International Conference on Pattern Recognition*. 60, 64, 73, 77, 81

Bajaj, C. (1986). Proving geometric algorithm non-solvability: An application of factoring polynomials. *J. Symb. Comput.*, 2:99–102. 21

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with Bregman divergences. *JMLR*, 6:1705–1749. 194

BELK, M. and CONNELLY, R. (2007). Realizability of graphs. *Discrete Comput. Geom.*, 37(2):125–137. 190

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*. 67

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Le Roux, N., and Ouimet, M. (2004). Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *NIPS*, volume 16. 67

Berg, C., Christensen, J., and Ressel, P. (1984). *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions.* Springer. 169, 171

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by probability distributions. *Bull. Calcutta Math Soc.*, 7:401–406. 172

Biswas, P., Liang, T.-C., Toh, K.-C., , Ye, Y., and Wang, T.-C. (2006). Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*, 3:360–371. 95, 97, 105

Biswas, P., Toh, K.-C., and Ye, Y. (2008). A distributed SDP approach for large-scale noisy anchor-free graph reailzation with applications to molecular conformation. *SIAM J. Sci. Comput.*, 30(3):1251–1277. 97

Biswas, P. and Ye, Y. (2004). Semidefinite programming for ad hoc wireless sensor network localization. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 46–54, New York, NY, USA. ACM. 95, 97

Biswas, P. and Ye, Y. (2006). A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization. *Multiscale optimization methods and applications*, pages 69–84. 97

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022. 10, 50

Blumenthal, L. (1953). *Theory and applications of distance geometry*. Clarendon Press Oxford. 7

Borg, I. and Groenen, P. J. F. (2005). *Modern Multidimensional Scaling*. Springer. 21, 59

Bose, P., Maheshwari, A., and Morin, P. (2003). Fast approximations for sums of distances, clustering and the Fermat–Weber problem. *Comput. Geom. Theory Appl.*, 24(3):135–146. 31

Bouchard, G. (2007). Bias-variance tradeoff in hybrid generative-discriminative models. In *Proceedings of the Sixth International Conference on Machine Learning and Applications*, pages 124–129, Washington, DC, USA. IEEE Computer Society. 124, 127, 136, 156

Bouchard, G. and Triggs, B. (2004). The tradeoff between generative and discriminative classifiers. In *IASC International Symposium on Computational Statistics*, pages 721–728, Prague. 125, 136

Brandes, U. and Pich, C. (2007). Eigensolver methods for progressive multidimensional scaling of large data. In *GD'06: Proceedings of the 14th international conference on Graph drawing*, pages 42–53, Berlin, Heidelberg. Springer-Verlag. 47, 60, 66, 73, 77

Brimberg, J. and Love, R. F. (1993). Global convergence of a generalized iterative procedure for the minisum location problem with ℓ_p distances. *Operations Research*, 41(6):1153–1163. 31, 33

Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2006). Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proc. National Academy of Sciences (PNAS)*, 103(5):1168–1172. 9

Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2008). *Numerical Geometry of Non-Rigid Shapes*. Springer. 9, 22, 49, 59

Brown, L. D. (1986). Fundamentals of statistical exponential families with applications in statistical decision theory. *Lecture Notes-Monograph Series*, 9:pp. i–iii+v–vii+ix–x+1–279. 4

Buja, A., Swayne, D. F., Littman, M. L., Dean, N., Hofmann, H., and Chen, L. (2008). Data visualization with multidimensional scaling. *Journal of Computational and Graphic Statistics*, 17:444–472. 60, 63

Buss, S. R. and Fillmore, J. P. (2001). Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics*, 20:95–126. 21

Canu, S., Grandvalet, Y., Guigue, V., and Rakotomamonjy, A. (2005). SVM and kernel methods matlab toolbox. Perception SystÃÍmes et Information, INSA de Rouen, Rouen, France. 174

Carter, M. W., Jin, H. H., Saunders, M. A., and Ye, Y. (2006). Spaseloc: An adaptive subproblem algorithm for scalable wireless sensor network localization. *SIAM J. on Optimization*, 17(4):1102–1128. 95, 97, 98

Cayton, L. and Dasgupta, S. (2006). Robust Euclidean embedding. In *ICML*. 22, 25, 26, 28, 37, 61, 63, 83

Chen, L. and Buja, A. (2009). Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the Americal Statistical Association*, 104:209–219. 22, 60, 63, 191

Chen, P., Chen, Y., and Rao, M. (2008). Metrics defined by Bregman divergences. *Comm. in Mathematics Sciences*, 6(4):915–926. 170

Collins, M., Dasgupta, S., and Schapire, R. E. (2001). A generalization of principal component analysis to the exponential family. In *NIPS 14*. MIT Press. 146

Connelly, R. (2005). Generic global rigidity. *Discrete Comput. Geom.*, 33(4):549–563. 190

Connelly, R. (2009). Tensegrities and global rigidity. Technical report, Dept. of Math., Cornell University, Ithaca, NY. 190

Costa, J., Patwari, N., and Hero III, A. (2006). Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):39–64. 99

Cox, T. F. and Cox, M. A. A. (2000). *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC. 9, 21, 26, 27, 59

Cozman, F. G., Cohen, I., Cirelo, M. C., and PolitÃľcnica, E. (2003). Semi-supervised learning of mixture models. In *20th International Conference on Machine Learning*, pages 99–106. 123

Cuturi, M., Fukumizu, K., and Vert, J.-P. (2005). Semigroup kernels on measures. *J. Mach. Learn. Res.*, 6:1169–1198. 164, 165, 169, 173

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893. 10, 50, 97

Davis, T. A. and Hu, Y. (to appear). The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*. 60, 63

de Leeuw, J. (1977). Applications of convex analysis to multidimensional scaling. In Barra, J., Brodeau, F., Romier, G., and Van Custem, B., editors, *Recent Developments in Statistics*, pages 133–146. North Holland Publishing Company. 22, 27, 59, 64

de Leeuw, J. and Mair, P. (2009). Multidimensional scaling using majorization: SMACOF in R. Technical Report 537, UCLA Statistics Preprints Series. 22, 27, 37, 49, 59, 64, 73

De Leeuw, J. and Michailidis, G. (1994). Block relaxation algorithms in statistics. *Information systems and data analysis*, pages 308–325. 29

de Silva, V. and Tenenbaum, J. B. (2004). Sparse multidimensional scaling using landmark points. `http://pages.pomona.edu/~vds04747/public/papers/landmarks.pdf`. 60, 65, 68, 70, 73, 76

Dhillon, I. S., Fan, J., and Guan, Y. (2001). Efficient clustering of very large document collections. In R. Grossman, C. Kamath, V. K. and Namburu, R., editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers. Invited book chapter. 6

Dhillon, I. S. and Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175. 6

Druck, G., Pal, C., McCallum, A., and Zhu, X. (2007). Semi-supervised classification with hybrid generative/discriminative methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–289, New York, NY, USA. ACM. 124, 125, 126, 136, 137, 138, 155, 156, 159

Elad, A. E., Keller, Y., and Kimmel, R. (2005). Texture mapping via spherical multi-dimensional scaling. In *Scale-Space*. 49, 54, 63

Faloutsos, C. and Lin, K.-I. (1995). Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, pages 163–174, New York, NY, USA. ACM. 46, 60, 65, 73, 77, 81

Fang, H. and O'Leary, D. (2011). Euclidean distance matrix completion problems. 106

Fletcher, P. T., Venkatasubramanian, S., and Joshi, S. (2009). The Geometric Median on Riemannian Manifolds with Application to Robust Atlas Estimation. *Neuroimage (invited to special issue)*, 45(1):S143–S152. 53

Frank, A. and Asuncion, A. (2010). UCI machine learning repository. 75

Fuglede, B. and Topsoe, F. (2004). Jensen-Shannon divergence and Hilbert space embedding. In *IEEE International Symposium on Information Theory*, pages 31–31. 169

Fujino, A., Ueda, N., and Saito, K. (2007). A hybrid generative/discriminative approach to text classification with additional information. *Inf. Process. Manage.*, 43(2):379–392. 136

Gonzalez, T. F. (1995). A simple LP-free approximation algorithm for the minimum weight vertex cover problem. *Information Processing Letters*, 53(3):129–131. 65, 70

Gower, J. (1982). Euclidean distance geometry. *Mathematical Scientist*, 7(1):1–14. 7

Gray, R., Buzo, A., Gray Jr, A., and Matsuyama, Y. (1980). Distortion measures for speech processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):367–376. 10, 50

Hein, M. and Bousquet, O. (2005). Hilbertian metrics and positive definite kernels on probability. In *AISTATS*. 173

Hendrickson, B. (1992). Conditions for unique graph realizations. *SIAM J. Comput.*, 21:65–84. 189

Hochbaum, D. S. (1982). Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162. 21

Indyk, P. and Matousek, J. (2004). Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press. 28

Jaakkola, T. S. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press. 136, 164, 172

Jebara, T., Kondor, R., and Howard, A. (2004). Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844. 164, 172

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer. 10, 50

Johnson, W. B. and Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into Hilbert space. *Contemporary Mathematics*, 26:189–206. 28, 57

Kalfon, M. (2010). Incremental PCA. `http://stackoverflow.com/questions/2745612/incremental-pca`. 77

Karcher, H. (1977). Riemannian center of mass and mollifier smoothing. *Comm. on Pure and Appl. Math.*, 30:509–541. 21, 49, 53

Kass, R. E. and Vos, P. W. (1997). *Geometrical Foundations of Asymptotic Inference*. Wiley-Interscience. 2, 3, 10, 50, 187

Katz, I. N. (1974). Local convergence in Fermat's problem. *Mathematical Programming*, 6:89–104. 33

Kim, S., Kojima, M., and Waki, H. (2009). Exploiting sparsity in sdp relaxation for sensor network localization. *SIAM J. on Optimization*, 20(1):192–215. 95, 98, 105, 106, 107

Krislock, N. (2010). *Semidefinite facial reduction for low-rank euclidean distance matrix completion*. PhD thesis, University of Waterloo. 95, 97, 98, 103, 105, 106, 107

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to nonmetric hypothesis. *Psychometrika*, 29:1–27. 22, 59

Kruskal, J. B. and Wish, M. (1978). Multidimensional scaling. In Uslander, E. M., editor, *Quantitative Applications in the Social Sciences*, volume 11. Sage Publications. 21, 27, 59

Kuhn, H. W. (1973). A note on Fermat's problem. *Mathematical Programming*, 4:98–107. 33

Lafferty, J. and Lebanon, G. (2005). Diffusion kernels on statistical manifolds. *J. Mach. Learn. Res.*, 6:129–163. 164, 172

Lasserre, J. A., Bishop, C. M., and Minka, T. P. (2006). Principled hybrids of generative and discriminative models. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 87–94, Washington, DC, USA. IEEE Computer Society. 123, 124, 125, 126, 136, 142, 143, 145, 155, 156, 159

Lebanon, G. (2005). *Riemannian Geometry and Statistical Machine Learning*. PhD thesis, School of Computer Science, Carnegie Mellon University. 188

Lebanon, G. and Lafferty, J. (2004). Hyperplane margin classifiers on the multinomial manifold. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 66, New York, NY, USA. ACM. 2, 3, 10, 50, 187

Liang, P. and Jordan, M. I. (2008). An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *ICML*. 163, 164, 175

Liberti, L., Lavor, C., Maculan, N., and Mucherino, A. (2012). Euclidean distance geometry and applications. *Arxiv preprint arXiv:1205.0349*. 7, 8, 103, 189

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137. 71

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2). 10, 50

Lunga, D. and Ersoy, O. (2011). Spherical classification of remote sensing data. *Transactions on Machine Learning and Data Mining*, 4(2):75–95. 7

Magen, A. (2002). Dimensionality reductions that preserve volumes and distance to affine spaces, and their algorithmic applications. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, Lecture Notes In Computer Science; Vol. 2483. 50, 51, 57

Marshall, A. W. and Olkin, I. (1979). *Inequalities: Theory of Majorization and Its Applications*. Academic Press. 27, 64

Martins, A. F. T., Figueiredo, M. A. T., Aguiar, P. M. Q., Smith, N. A., and Xing, E. P. (2008). Nonextensive entropic kernels. In *ICML*. 164, 173

Mccallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI Workshop on "Learning for Text Categorization"*. 131

McCallum, A., Pal, C., Druck, G., and Wang, X. (2006). Multi-conditional learning: Generative/discriminative training for clustering and classification. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 433–439. 136

Moore, D., Leonard, J., Rus, D., and Teller, S. (2004). Robust distributed network localization with noisy range measurements. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 50–61, New York, NY, USA. ACM. 99

Muthukrishnan, S. (2005). Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2). 71

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press. 123, 125, 131, 163, 178

Nielsen, O. B. (1978). *Information and Exponential Families in Statistical Theory*. John Wiley, New York. 4, 195

Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, V39(2):103–134. 123

Ostresh, L. M. (1978). On the convergence of a class of iterative methods for solving the Weber location problem. *Operations Research*, 26:597–609. 33

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190. 10, 50

Pietersz, R. and Groenen, P. (2004). Rank reduction of correlation matrices by majorization. Econometric institute report, Erasmus University Rotterdam, Econometric Institute. 49

Platt, J. (2005). Fastmap, MetricMap, and Landmark MDS are all Nyström algorithms. In *AIStats*, pages 261–268. 65, 67, 74, 75, 77

Pless, R. and Simon, I. (2002). Embedding images in non-flat spaces. In *Proc. of the International Conference on Imaging Science, Systems, and Technology*. 22, 49

Pong, T. and Tseng, P. (2009). Robust edge-based semidefinite programming relaxation of sensor network localization. Technical report, U. of Washington. 95, 98, 105, 106, 107, 111, 191

Porkaew, K., Chakrabarti, K., and Mehrotra, S. (1999). Query refinement for content based multimedia retrieval in mars. In *IN MARSâĂİ. MULTIMEDIA COMPUTING AND SYSTEMS*, pages 747–751. 75

Preparata, F. P. and Shamos, M. I. (1985). *Computational geometry: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA. 6

Raina, R., Shen, Y., Ng, A. Y., and McCallum, A. (2003). Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*. MIT Press. 136

Rockafellar, R. T. (1996). *Convex Analysis (Princeton Mathematical Series)*. Princeton University Press. 194

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. 67

Sarlós, T. (2006). Improved approximation algorithms for large matrices via random projections. In *Proceedings 47th Annual IEEE Symposium on Foundations of Computer Science*. 50

Schoenberg, I. J. (1938). Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536. 169, 171

Silva, V. D. and Tenenbaum, J. B. (2003). Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press. 47, 67

Snoussi, H. and Mohammad-Djafari, A. (2002). Information geometry and prior selection. 162

Spivak, M. (1999). *A Comprehensive Introduction to Differential Geometry*, volume One. Publish or Perish, Inc., Houston, Texas, third edition. 5

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. 67

Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419. 22, 27, 59, 73

Tseng, P. (2007). Second-order cone programming relaxation of sensor network localization. *SIAM J. on Optimization*, 18(1):156–185. 95, 98

Vitter, J. S. (1985). Sampling with a reservoir. *ACM Trans. on Mathematical Software*, 11:37–57. 70

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305. 195

Wang, J. T.-L., Wang, X., Lin, K.-I., Shasha, D., Shapiro, B. A., and Zhang, K. (1999). Evaluating a class of distance-mapping algorithms for data mining and clustering. In *Proc. 5th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 307–311, New York, NY, USA. ACM. 47, 60, 65, 73, 77

Wang, Z., Zheng, S., Ye, Y., and Boyd, S. (2008). Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM J. Optim.*, pages 655–673. 97, 106, 107

Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Math. J.*, 43:355–386. 31, 33

Yianilos, P. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 311–321. Society for Industrial and Applied Mathematics. 7

Young, G. and Householder, A. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22. 22, 29, 37, 38, 59

Zhang, J. (2004). Divergence function, duality, and convex analysis. *Neural Comput.*, 16(1):159–195. 154, 167

Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved Nytröm low-rank approximation and error analysis. In *ICML*. 65, 70, 71

Zhong, S. (2005). Efficient online spherical k-means clustering. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 3180–3185. IEEE. 6

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison. 123