DEPARTMENT OF STATISTICS

University of Wisconsin

1300 University Ave.

Madison, WI 53706

TECHNICAL REPORT NO. 1124

July 19, 2006

Regularized Nonparametric Logistic Regression and Kernel Regularization [1]

Fan Lu

flu@stat.wisc.edu

http://www.stat.wisc.edu/~flu

[1] This research partially supported by NSF Grant DMS 0505636, NIH Grant EY09946 and ONR Grant N00014-06-1-0095.

REGULARIZED NONPARAMETRIC LOGISTIC REGRESSION AND KERNEL REGULARIZATION

By

Fan Lu

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Statistics)

at the

UNIVERSITY OF WISCONSIN – MADISON

2006

Abstract

Regularization methods consist of a category of commonly used techniques to obtain robust solutions to ill-posed problems such as nonparametric regression and classification. In recent years, methods of regularization have also been successfully introduced to address some other classical problems in statistics, e.g. model/variable selection and dimension reduction. This thesis is composed of two major parts, both of which are within the framework of regularization methods.

In the first part of this thesis, we are interested in the physics problem of detecting high energy signal neutrino events. We propose a modification to the traditional nonparametric penalized likelihood approach, to take into account the usage of importance sampling techniques in the generation of the training data from computer experiments. We try to estimate the multivariate logit function of the signal neutrino events in order to find the most powerful decision boundary at a certain significance level to optimally separate signal from background neutrinos. For simulated normal data, we compare this approach with a non-standard support vector machine (SVM) approach. The results suggest that in the case of weighted binary data, logistic regression is more appropriate than SVM in terms of finding individual level curves of the logit function. We also propose a diagnostic plot to check the goodness of fit of the result when the truth is unknown.

In the second part of this thesis, we are interested in problems involving dissimilarity data. It is often possible to use expert knowledge or other sources of information to obtain dissimilarity measures for pairs of objects, which serve as pseudo-distances between the objects. When dissimilarity information is available, there are two different types of problems of interest. The first is to estimate the full position configuration for all objects in a low dimensional space while respecting the data. This is usually for the purposes of visualizing the data and/or conducting further statistical analysis, such as clustering or classification. Multidimensional scaling (MDS), which is still an active research area, has been traditionally used to tackle this problem. In the second type of problems, the high dimensional data points are assumed to lie on a low dimensional manifold and the goal is to unfold the manifold in order to recover the underlying intrinsic low dimensional structure. We develop a novel, unified framework called *Kernel Regularization* which can be utilized to solve either problem. Mathematical programming techniques are used to obtain the solutions accurately and efficiently. The proposed method can naturally accommodate real-world situations, where for example, the dissimilarity information is crude, noisy, incomplete (some dissimilarity measurements are missing), inconsistent or associated with weights representing different level of confidence in each measurement. For the regularized kernel estimation (RKE) formulations proposed, dimension reduction is naturally promoted for the estimated kernel, which is not only noise-robust but also advantageous for further data processing like clustering and classification. The method is applied to proteins sequences data with some interesting findings.

Acknowledgments

First and foremost I would like to express my deepest gratitude to my advisor Grace Wahba, for her guidance and support throughout my pursuit of the PhD degree in Statistics at University of Wisconsin-Madison. Without her enthusiasm and encouragement, my thesis work could not have been completed. I have learned tremendously from her knowledge, brilliance and strive for excellence. Most importantly, she teaches me how to be not just a statistician but a scientist. It has been my great honor and privilege to have had the opportunity to work closely and learn from her.

The work in this thesis is the product of a collaboration with a number of researchers. In particular, I had the greatest luck of collaborating with Stephen J. Wright, one of the best optimization researchers in the world. I learned theories of nonlinear programming in his class using his textbook. Our development of Regularized Kernel Estimation could not be possible without his key insights and helpful discussions. I also want to thank Sündüz Keleş for her efforts and patience within our collaborated project. She set up for me the best example of a young researcher. My collaboration with Steve and Sündüz has not only been the most fruitful one for me, but also a very enjoyable one on the personal level. I am grateful to their influence on me about passions in science.

I would like to thank Yi Lin for his inspiration on the Manifold Learning

problem. Many discussions with him on a variety of topics have been very enjoyable and benefited me a great deal. I wish I could have spend more time learning from him before he took his leave. I would also like to thank Gary Hill for bringing Grace and me the problem of neutrino signal detection which has significant scientific importance. It has been a very pleasant collaboration with him, which reconnected me to my undergraduate physics background.

I am grateful to all the members of my thesis committee, Stephen J. Wright, Yi Lin, Sündüz Keleş, Kjell Doksum and Grace Wahba. I would also like to thank my references, Stephen J. Wright, Yi Lin, Sündüz Keleş, and Grace Wahba for writing reference letters and putting up good words on behalf of me. Without their help, I couldn't be focusing on writing this thesis.

Our Thursday group meeting has been a great experience during the course of my study. I should thank the hosts Professors Wahba and Lin and the members of the current group: John Carew, Hyonho Chun, Weiliang Shi, and Hyonho Chun. Some former group members: Chenlei Leng, Ming Yuan, and Xianhong Xie should be thanked as well. I should not forget the fellow students, the professors, and the staff members in the statistics department. During my stay in Madison, they helped me in various ways and made my life and study enjoyable.

I am very fortunate to have my wife Meng Chen accompany me all through this journey. I owe her a great deal for her love, unconditional support and constructive criticism. I owe much more than what I can put into words to my dear parents Quan Lu and Shu Zhang for giving me life, providing me a very pleasant environment to grow and helping me to become who I am. I dedicate this thesis to them for this couldn't be achieved without them being part of my life.

List of Figures

List of Tables

Contents

Chapter 1

General Overview

1.1 Overview

In this thesis, we will derive new regularization methods for different types of data.

For weighted two-class labeled data, we propose a modified version of the traditional nonparametric penalized likelihood method to accommodate the new feature — weights in the logistic regression problem. The proposed data driven method consists of tuning, training and testing procedures. We will study the performance of the procedure using both simulated normal data and neutrino computer experiment data.

For noisy dissimilarity data, we derive a new regularization framework to learn the information in the form of a kernel. We propose different loss functions and kernel regularization functions for different kernel learning tasks to accommodate different purposes or data assumptions. Two types of kernel learning problems involving dissimilarity data — multidimensional scaling type of problem and manifold unfolding problem are studied in this thesis. In both types of problems, our proposed kernel regularization functions promote dimensional reduction.

1.2 Outline of the Thesis

The rest of the thesis is organized as follows. Chapter 2 concerns the neutrino signal detection problem. We first briefly introduce the statistical version of the problem without too much physics background. Then we derive our modified penalized (log)likelihood approach to solve the problem. After that we illustrate the computation procedures to fit our model. Next, we show simulation results on multivariate normal data, where we also compare our method with a non-standard support vector machine approach. Finally, we show results from applying our method on data generated by computer experiment of neutrino events. We also propose a way to check goodness of fit of our method when the truth is unknown.

In Chapter 3, we study the problem of kernel learning with dissimilarity information. After a brief literature review, we lay out our framework of Regularized Kernel Estimation (RKE) before introducing the numeric technique called *convex conic programming*, which is necessary for solving our optimization problems formulated in later sections and in Chapter 4. We then propose our RKE formulation for multidimensional scaling type of problem and we show how to convert it to a convex conic programming problem. Before we discuss more technical details of implementing our RKE method, we define and solve newbie (out-of-sample extension) problem for augmenting estimated kernels with new data coming. Finally, we show RKE results on a simulated example and real protein dissimilarity data.

In Chapter 4, we study the manifold unfolding problem adopting the kernel approach. After a brief literature review of this recent research topic, we derive our specific manifold-unfolding RKE formulation with a different loss function and a different kernel regularization function compared to what we have in Chapter 3. With material in Chapter 3 discussed, it is straightforward to show how to convert our RKE formulation for manifold unfolding to also a convex conic problem for global solution. We demonstrate the good properties of our method via several simulation studies.

The thesis will be concluded with some remarks in Chapter 5.

Chapter 2

Signal Probability Estimation with Penalized Likelihood Method on Weighted Data

2.1 Summary

In this chapter we consider the problem faced by astrophysicists where high energy signal neutrinos must be separated from overwhelming background events. We propose a modification to the usual penalized likelihood approach, to take account of the usage of importance sampling techniques in the generation of the simulated training data. Each simulated multivariate data point has two associated weights, which define its contribution to the signal or background count. We wish to find the most powerful decision boundary at a certain significance level to optimally separate signal from background neutrinos. In this modified penalized likelihood method, the estimation of the logit function involves two major optimization steps and the use of KL (Kullback-Leibler) distance criterion for model tuning. We compare this approach with a non-standard SVM (support vector machine) approach. Results on simulated multivariate normal data and simulated neutrino data are presented. For the neutrino data, since the truth is unknown, we show a way to check whether the proposed method is working properly.

2.2 Introduction

A neutrino is a particle that has no charge and almost no mass. Neutrinos may be produced in the center of active galaxies or from highly energetic objects like γ-ray bursts or black holes. Physicists are trying to use the giant device called AMANDA (Antarctic Muon and Neutrino Detector Array) buried deep in the Antarctic ice cap to detect certain neutrino signals within comparatively overwhelming background noise (Andrés & other 119 coauthors 2001, Ahrens & other 112 coauthors AMANDA collaboration). In computer experiments simulating neutrinos passing through AMANDA, distributions of signal and background are generated by an importance sampling procedure which generates events described by multiple feature variables. Each simulated neutrino can represent both signal and background by assignment of an importance sampling weight. The task is to find the most powerful decision boundary at a certain significance level to distinguish signal neutrino from background neutrino. For a detailed description of the problem and data, see Hill et al. (2003). Because of the curse of dimensionality, usual Monte Carlo methods are not practical. We propose a modified penalized log-likelihood approach to solve the usual multivariate problem with this weighted simulated data. Though this study is motivated by the simulated neutrino data, the proposed method can be applied to other multivariate weighted data.

2.3 Modified Penalized Likelihood Estimation

2.3.1 Penalized Likelihood Method for Labeled Data

Let x be a possibly multidimensional vector of event observables derived from a reconstructed event. Let $h_s(x)$ be the probability density function for signal vectors and $h_b(x)$ be the probability density function for background vectors, and let π_s and π_b be prior probabilities of a signal and background observation, respectively. Then the posterior probability that x is a signal vector is $p(x) = \pi_s h_s(x)/(\pi_b h_b(x) + \pi_s h_s(x))$. The logit $f(x)$ is defined as $\log[p(x)/(1-p(x))] \equiv \theta + \log[h_s(x)/h_b(x)]$, where $\theta = \log(\pi_s/\pi_b)$. We will estimate the logit $f(x)$ for a particular (implicit) value of θ, but since the end result is to obtain level curves of f, the particular value of θ is not important for the calculations. A modified form of the penalized likelihood estimate (Wahba 1990*a*, 2002*a*, Wahba et al. 1995*b*) will be used.

Let y_i be a random variable that is 1 (signal) with probability $p(x_i)$ and 0 (background) with probability $1 - p(x_i)$. So the observed data are actually class labels. Then the likelihood of a single observation y_i is: $\mathcal{L} = p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$. The negative log likelihood of (independent) data $y_1, \ldots, y_n$ is then, in terms of the logit given by

$$Q(y, f) = \sum_{i=1}^{n} \left[\log\left(1 + e^{f(x_i)}\right) - y_i f(x_i)\right]. \qquad (2.3.1)$$

We want to find $f \cong \sum c_k B_k \in H_K$ (a reproducing kernel Hilbert space (RKHS), see Aronszajn (1950*a*), Wahba (1990*a*, 2002*a*)) which minimizes the penalized log-likelihood:

$$I_\lambda(c) = Q(y, f) + \lambda \parallel f \parallel^2_{H_K}, \qquad (2.3.2)$$

where B_k's are basis functions in H_K and $\| \cdot \|_{H_K}$ is the function norm in H_K.

This is essentially the penalized log likelihood estimate of f proposed in O'Sullivan, Yandell and Raynor (1986) , and is in common use in some fields. Under rather general conditions, which include a proper choice of λ, penalized log likelihood estimates are known to converge to the 'true' f as the sample size becomes large (Cox & O'Sullivan 1990). RKHS are discussed in Aronszajn (1950*a*) and their use in statistical model building in Wahba (1990*a*) and elsewhere. A wide variety of these spaces is available. An RKHS is characterized by a unique positive definite function $K(\cdot, \cdot)$, and once K is chosen, the exact minimizer of $I_\lambda(c)$ is known to be in the span of a certain set of basis functions determined from K (Kimeldorf & Wahba 1971). In Section 2.3.2 below we will select a particular K, known to be a good general-purpose choice, and use an approximating subset of this set of basis functions. Estimating f rather than p

directly gives a strictly convex optimization problem whose gradient and Hessian are simple to compute, which makes the numerical analysis easier and thus suitable for very large data sets. It is possible to estimate p directly but this estimate is harder to compute in large data sets and is believed to be not as accurate.

2.3.2 Penalized Likelihood Method For Weighted Data

The form of the negative log likelihood in equation (1) applies where simulated training data are distributed as $h_b(x)$ and $h_s(x)$ through sampling directly from the generating distributions $\Phi_s(\tilde{E})$ and $\Phi_b(\tilde{E})$, then processing the events $\tilde{E}$ through a simulation chain which mimics the tracks seen by the AMANDA detector array and the process which extracts variables x from the simulated tracks. Here, $\tilde{E}$ means a vector of generating parameters, e.g., neutrino energy, position and arrival direction. However, the results were expected to have extremely long tails which is the region of interest, so an importance sampling scheme has been developed. x vectors were generated according to a convenient sampling distribution $g(\tilde{E})$, and pushed through the detector geometry and variable (x) extraction. For each x_i so obtained, two weights were assigned, $w_s(x_i) = \Phi_s(\tilde{E}_i)/g(\tilde{E}_i)$ for signal and $w_b(x_i) = \Phi_b(\tilde{E}_i)/g(\tilde{E}_i)$ for background. $w_s(x_i) + w_b(x_i)$ plays the role as an estimate of relative frequency of x_i in signal + background while $w_s(x_i)/(w_s(x_i) + w_b(x_i))$ plays the same role for the probability of signal given x_i, and similarly for background. The weights satisfy

$\sum_{i=1}^{n} w_s(x_i) = N_s$ and $\sum_{i=1}^{n} w_b(x_i) = N_b$, where N_s and N_b are the predicted numbers of events from the weighted simulation.

Now, if we had multiple unbiased observations at some x_i as $y_{ij}, j = 1, \ldots, m(i)$, the likelihood of these observations is: $\mathcal{L} = p(x_i)^{\sum_{j=1}^{m(i)} y_{ij}}(1-p(x_i))^{\sum_{j=1}^{m(i)}(1-y_{ij})}$. If the samplings at x_i are biased, then the exponent sums are weighted by $w_s(x_i)$ and $w_b(x_i)$ respectively leading to a modified likelihood

$$Q(w, f) = \sum_{i=1}^{n} \sum_{y_i=0}^{1} \{w_{y_i}[\ log(1 + e^{f(x_i)}) - y_i f(x_i)]\}, \tag{2.3.3}$$

where $w_{y_i} = w_s(x_i)$ for $y_i = 1$ and $w_{y_i} = w_b(x_i)$ for $y_i = 0$. The incorporation of weighted events is thus simply accounted for by weighting the terms in the logarithmic likelihood sum. Further, we can substitute $w_s(x_i)$ and $w_b(x_i)$ to obtain an alternative form of the likelihood

$$Q(w, f) = \sum_{i=1}^{n} \{w_t(x_i)[\log(1 + e^{f(x_i)}) - \tilde{p}(x_i) f(x_i)]\}, \tag{2.3.4}$$

where $w_t(x_i) = w_s(x_i) + w_b(x_i)$ and $\tilde{p}(x_i) = w_s(x_i)/w_t(x_i)$.

Notice that, our extension of the penalized likelihood method to weighted data by defining $Q(w, f)$ in equation (2) as in equation (4) is a natural generalization of the original formulation since equation (4) will reduce to equation (2.3.1) in the case of labeled data, which we consider as a special case of weighted data with $(w_s = 1, w_b = 0)$ representing ‘1’ class and $(w_s = 0, w_b = 1)$ representing ‘0’ class.

2.4 Implementation of the Modified Penalized Likelihood Method

After getting the modified penalized likelihood formulation, we now can move on to look for a 'good' estimate of $f(x)$ whose level curves can be obtained. In our implementation, we use radial basis functions plus constant and linear terms. So,

$$f(x) = \beta_0 + \beta^T x + \sum_{k=1}^{N} c_k K_\sigma(x, x_{i_k}), \tag{2.4.1}$$

where $K_\sigma(\cdot, \cdot)$ is the Gaussian kernel with isotropic variance σ^2, N is the total number of basis functions and the $x_{i_k}, k = 1, \ldots, N$, will be chosen as a subset of the $x_i, i = 1, \ldots, n$, as described below. Thus, f will be specified as long as all coefficients, i.e., β_0, β and the c_k's are determined (note that β is a vector). By letting $\lambda \parallel f \parallel_{H_K}^2 = \lambda \sum_{k,l=1}^{N} c_k c_l K_\sigma(x_{i_k}, x_{i_l})$, we put a penalty only on the c_k's leaving constant and linear terms unpenalized.

We used a sequence of data driven procedures to fit the model in the sense that we let the data choose the 'best' combination of smoothing parameter λ, scale parameter σ and number of basis functions N. For a given weighted multi-dimensional data set, we can first transform each variable to make them of comparable scale. Though these preprocessing procedures are not always necessary, they often improve the performance of our algorithm. Then, the entire data set is randomly divided into three subsets of some preset sizes: a training set, a tuning set and a testing set. After that, we randomly, but according to weights (large-weighted simulated data points have higher chance to be selected), choose a modest sized set of N x_{i_k}'s which determines basis functions as a subset of the training set. We solve the minimization problem on a coarse 2-D parameter grid of λ (usually on a log scale) and σ^2 using the training set. For each parameter pair (each point on the $\log \lambda - \sigma$ grid), an iterative Newton-Raphson algorithm is used to solve this convex minimization problem (Wahba 1990*a*). After the algorithm converges, we calculate the Kullback-Leibler (KL) distance between the tuning set and the fitted model. This is essentially just the first term of $I_\lambda(c)$ for tuning simulated data with f replaced by $\hat{f}$. We then find the best parameter combination based on the KL distance over the coarse grid. Starting from there, a direct-searching simplex method (Lagarias et al. 1998) is used to search for a locally best parameter combination according to the KL distance criterion (Ferris et al. 2004). This whole procedure described above is repeated using $2N$ bases, then $4N$ bases and so on, until the improvement on the KL distance is smaller than some preset threshold. We use the coefficients corresponding to the then-best combination of parameters to construct our final estimate of the logit function. Next, the testing set is used to check the goodness of fit of this final model. Finally, the level curves of $p(x)$ are determined and plotted. These level curves are appropriate for use in conjunction with the approaches in Hill & Rawlins (2003), Feldman & Cousins (1998). Finally, the real data can be analyzed by applying the thresholds given by the level curves of $p(x)$, see Ahrens & other 112 coauthors (AMANDA collaboration).

2.5 Results on Simulated Multivariate Normal Data

Instead of using the neutrino data, for which we don't know the truth, we first test our algorithm on simulated multivariate data. For plotting convenience we only show a two dimensional example. Our algorithm has been tested extensively on higher dimensional simulated data (in particular 5-D, which is the expected dimension of the neutrino data) with success. We generate a random sample of x's from a 2-D uniform distribution (which plays the role of g) over a square. We then associate with each x_i two distinct 2-D Gaussian density values ($w_s(x_i)$ for signal, $w_b(x_i)$ for background), giving a simulated data set consisting of the x_i's and their associated w_b's and w_s's. The sum of the two 2-D Gaussian distributions is shown in Figure 2.5.2(a).

For a particular run with sample size 1000 (among which we randomly pick 400 for training and another 400 for tuning), the result for the level curve corresponding to $p = 0.9$ is shown in Figure 2.5.1. Since we know the truth here, the data points are colored green if the true p is less than or equal to 0.9 and black otherwise. The red line is the level curve found by our algorithm, which is visually almost identical to the true level curve (see Figure 2.5.2(c)). We also implemented a nonstandard support vector machine (SVM) (Lin et al. 2002) (the parameters are tuned in a similar way to our modified penalized likelihood method) here for comparison. To use a nonstandard SVM to find the level surface corresponding to $e^{f(x)} = p(x)/(1 - p(x)) = r$, it is not hard to extend the usual SVM formulation to the following weighted regularization problem:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{y=-1,1} w_{iy}c_y[(1 - yf(x_i))_+] + \lambda \parallel f \parallel_{H_k}^2$$

where

$$w_{iy} = \begin{cases} w_b(x_i) & \text{if} \quad y = -1; \\ w_s(x_i) & \text{if} \quad y = 1, \end{cases}$$

$$c_y = \begin{cases} r & \text{if} \quad y = -1; \\ 1 & \text{if} \quad y = 1, \end{cases}$$

and $(\tau)_+ = \tau$ if $\tau > 0$ and 0 otherwise. We minimize this nonstandard SVM criterion while tuning the smoothing parameter and scale parameter through iteratively calling the well-known SVM software SVM^{light} (version 4.0), which gives the blue line in Figure 2.5.1 as the estimated decision boundary.

It is worth mentioning that, our proposed penalized likelihood method estimates the logit function over the domain of the observed x's, hence it is able to give all level curves of the logit (and thus p) simultaneously, while SVM classifiers are targeted at one level curve at a time, i.e. they are meaningful only for the classification boundary. This point may be understood via Figure 2.5.1 of Lee et al. (2004*b*). SVM classifiers come into their own when the classes are (nearly) separable, but in our application that is not the case. In Figure 2.5.2, we show further results from our 2-D example. The color bar beside plot(b)

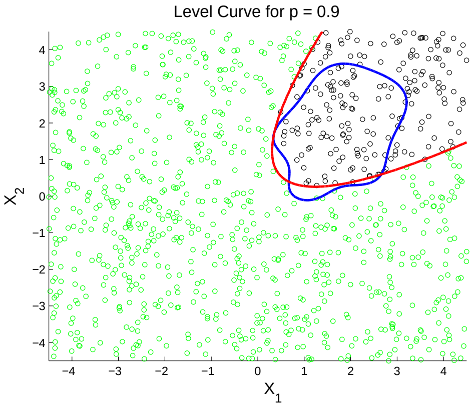

Figure 2.5.1: A 2-D EXAMPLE: Red line is the level curve estimated by our penalized likelihood method; Blue line is the level curve estimated by nonstandard SVM; Black points are data points with the true $p \geq 0.9$; Green points are data points with the true $p < 0.9$.

codes the relative importance $w_s + w_b$ (in log scale) for each data point, and the estimated p for the 200 data points in the test set is plotted against the true p. (The 200 points from the test set are particularly dense near 0 and 1.) We can see that the estimated probabilities (obtained from the estimated logit) match the true probability very well except for several points of very small importance that are colored in blue. Furthermore, by comparing the true and estimated level curve plots in (c) and (d), we can conclude that we are estimating the true logit function very well over the domain of the data.

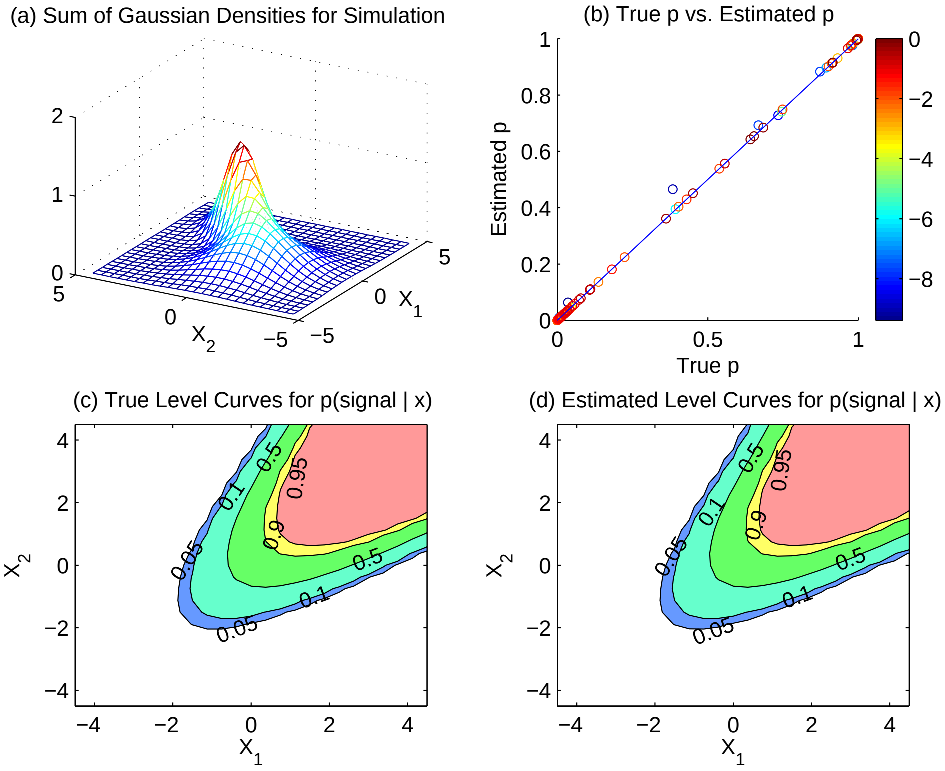

Figure 2.5.2: Results from the modified penalized likelihood method for a 2-D EXAMPLE: (a) sum of the two Gaussians used for weights; (b) p-p plot with the color bar coding the sum of the signal and background weights in log scale; (c) true level curves of p=0.05, 0.1, 0.5, 0.9 and 0.95; (d) estimated level curves of p=0.05, 0.1, 0.5, 0.9 and 0.95.

2.6 Results on Simulated Neutrino Data

2.6.1 Logit Function Estimation

The simulated neutrino data consists of five variables along with the weights (Ahrens et al. 2004). The variables are based on parameters derived from a maximum likelihood reconstruction of the particle track in the detector. For detailed information on the detector, reconstruction, variables and analysis procedure, see Ahrens et al. (2004). We had 10,000 simulated neutrino events, and divided them into 40% training, 40% tuning and 20% test sets. The five variables are first rescaled using their own sample weighted standard deviation after a log transformation. Then, we run our algorithm with the transformed data. We transform them back when plotting the results. The estimated logit function enables us to estimate the probability of an event being a signal neutrino at any point (within the domain of the data) in the 5-D observational space. We show a level curve plot of a 2-D cross section of that 5-D space in Figure 2.6.1, where other three variables have been fixed at their sample medians.

2.6.2 Checking the Goodness of the Estimate

Astrophysicists use very complex computer programs to simulate the observations of neutrinos passing through the AMANDA detector. Even though the parameters related to the simulation are known, the probability of an event being a signal neutrino in the 5-D space is only estimated at the data points where

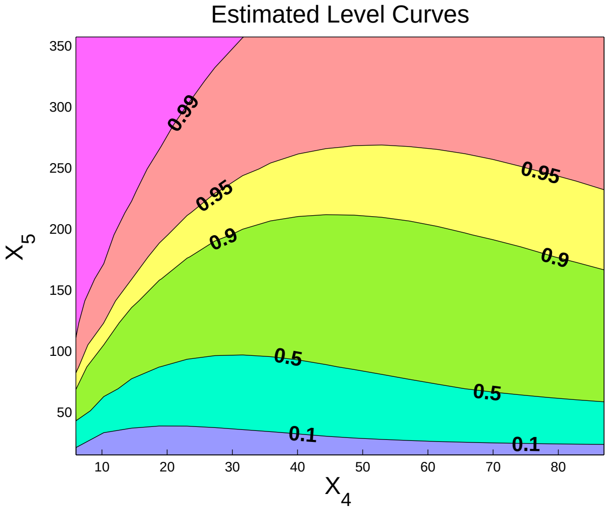

Figure 2.6.1: Level curves: $p = 0.1, 0.5, 0.9, 0.95$ and 0.99 on a 2-D cross section in the 5-D observational space of the neutrino data. $N = 500$ basis functions were used.

the simulation took place. We can't construct a plot as in Figure 2.5.2(b) since we don't know the truth. We can only check whether the estimates reasonably reflect the simulated information.

We try to evaluate the estimated probability surface using a method in common practice among physicists. We take all the x's whose estimated probability of signal falls into one of ten equally spaced bins from 0 to 1. Then for each bin we calculate the ratio of the sum of all the signal weights of the x's to the sum of all the signal + background weights of those x's. Call this the level-binned observed p. We plot the result for the jth bin against the midpoint of the bin, i.e., for the bin [0.4, 0.5], we plot the level-binned observed p against 0.45, and similarly for the other bins. If we estimate the probability reasonably well, based on the simulated, weighted data, we should have these ten points falling close to the 45-degree line, which is what we observe in Figure 2.6.2 for our neutrino data. The color coding of the points represents the relative signal + background weights of the x's in each bin, on a log scale.

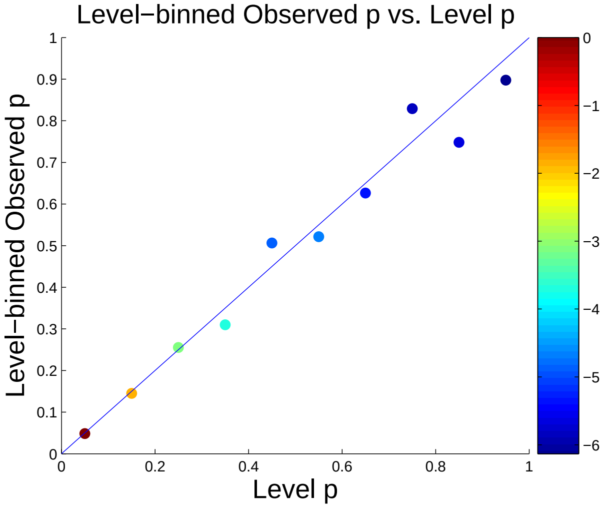

Figure 2.6.2: x-axis: Midpoints of 10 probability bins. y-axis: ratio of the sum of signal weights to the sum of signal + background weights for data points in each bin.

2.7 Conclusions

We have developed a feasible and effective computational method to obtain modified penalized likelihood estimates for signal detection probability in the context of five dimensional data as might be extracted from tracks observed by the AMANDA neutrino detector, where the data simulator has employed importance sampling. We showed result obtained by implementing the proposed method on a simulated neutrino data set before describing a way to check the goodness of our estimation. We have also compared the penalized likelihood method to the nonstandard support vector machine, similarly tuned, on a simulated multi-dimensional problems similar to the AMANDA problem, which can be characterized as having a certain amount of overlap between the signal and background distributions. In the examples of that nature, with significant overlap of the signal and background, the penalized likelihood method is competitive to the comparable nonstandard SVM for classification purposes.

Chapter 3

A Framework for Kernel Regularization with Application to Protein Clustering

3.1 Summary

We develop and apply a novel framework which is designed to extract information in the form of a positive definite kernel matrix from possibly crude, noisy, incomplete, inconsistent dissimilarity information between pairs of objects, obtainable in a variety of contexts. Any positive definite kernel defines a consistent set of distances, and the fitted kernel provides a set of coordinates in Euclidean space which attempts to respect the information available, while controlling for complexity of the kernel. The resulting set of coordinates are highly appropriate for visualization and as input to classification and clustering algorithms. The framework is formulated in terms of a class of optimization problems which can be solved efficiently using modern convex cone programming software. The

power of the method is illustrated using simulation study and in the context of protein clustering based on primary sequence data. An application to the globin family of proteins resulted in a readily visualizable 3D sequence space of globins, where several sub-families and sub-groupings consistent with the literature were easily identifiable.

3.2 Introduction

It has long been recognized that symmetric positive definite kernels (hereinafter "kernels") play a key role in function estimation (Aronszajn (1950*b*), Kimeldorf & Wahba (1971)), clustering and classification, dimension reduction, and other applications. Such kernels can be defined on essentially any conceivable domain of interest (Wahba (1990*b*)), originally function spaces and more recently, finite (but possibly large) collections of trees, graphs, images, DNA and protein sequences, microarray gene expression chips, and other objects. A kernel defines a distance metric between pairs of objects in the domain that admits an inner product. Thus they play a key role in the implementation of classification algorithms (via support vector machines (SVMs)) and clustering (via k-means algorithms, for example), along with their more classical role in function approximation and estimation, and the solution of ill-posed inverse problems Wahba (1977). Since the mid 90s, when the key role of these kernels became evident in SVMs (Wahba (1999), Evgeniou et al. (2000), Cristianini & Shawe-Taylor

(2000)), a massive literature has grown related to the use and choice of kernels in many domains of application, including, notably, computational biology (Schölkopf et al. (2004)). A google search as of the date of this writing gave over *three million* hits on "Kernel Methods" along with an ad from Google soliciting job applications from computer scientists!

Mathematically defined kernels, for example, spline kernels, radial basis functions and related positive definite functions defined on Euclidean space, have long been the workhorses in the field, generally with one or a few free parameters estimated from the data; see, for example Wahba (2002*b*). A recent work Lanckriet et al. (2004) proposes estimating a kernel by optimizing a linear combination of prespecified kernels via a semidefinite programming approach. The reader may connect with the recent literature on kernel construction and use in a variety of contexts by going to the NIPS2004 website (`http://books.nips.cc/nips17.html`) or the book Shawe-Taylor & Cristianini (2004).

It is frequently possible to use expert knowledge or other information to obtain dissimilarity scores for pairs of objects, which serve as pseudo-distances between the objects. There are two problem types of interest. The first is to estimate full relative position information for a (training) set of objects in a space of preferably low dimension in order to visualize the data or to conduct further processing - typically, classification or clustering. One traditional approach for this purpose is multidimensional scaling (MDS) (Buja & Swayne (2002)), which continues to be an active research area. The second problem is to place new objects in the space, given some dissimilarity information between them and some members of the training set, in the coordinate space of the training set.

In this chapter we propose regularized kernel estimation (RKE), a unified framework for solving both problems by fitting a positive definite kernel from possibly crude, noisy, incomplete, inconsistent, weighted, repetitious dissimilarity information, in a fully nonparametric approach, by solving a convex optimization problem with modern convex cone programming tools. The basic idea is to solve an optimization problem which trades off goodness of fit to the data and a complexity (shrinkage) penalty on the kernel which is used to fit the data - analogous to the well known bias-variance tradeoff in the spline and ill-posed inverse literature, but not exactly the same. Within this framework, we provide an algorithm for placing new objects in the coordinate space of the training set. The method can be used instead of MDS to provide a coherent set of coordinates for the given objects in few or many dimensions, without problems with local minima or (some) missing data. It can also be used to solve problems discussed in Lanckriet et al. (2004), but in a fully nonparametric way.

The feasibility of the RKE approach is demonstrated in the context of protein sequence clustering, by applying the method to global pairwise alignment scores of the heme-binding protein family of globins. In this example, we are already able to visualize the known globin subfamilies from a 3D plot of the training sequence coordinates that are obtained by the regularized kernel estimate. Furthermore, apparent sub-clusterings and outliers of the known globin subfamilies from the 3D plot reveal interesting observations consistent with the literature. Clustering of protein sequences from a family to identify subfamilies or clustering and classification of protein domains to determine protein function present one major application area for the novel framework presented here. However, we envision many more applications involving clustering and classification tasks in biological and non-biological data analysis, some of these are discussed in Section 3.7.

In Section 3.3, we present the general formulation of the problem and define the family of Regularized Kernel Estimates. Section 3.4 describes the formulation of RKE problems and the problem of placing test data in the coordinate space of training data as general convex cone problems. Also included is a brief discussion on tuning the parameters of the estimation procedure. Section 3.5 describes a simulation study. Section 3.6 presents an application to the globin protein family to identify subfamilies and discusses the biological implication of the results. Examples of placing test proteins in the coordinate system of training protein sequences are illustrated here. We conclude with a summary and discussion of future work in Section 3.7.

3.3 Dissimilarity Information and RKE

Given a set of N objects, suppose we have obtained a measure of dissimilarity, d_{ij}, for certain object pairs (i,j). We introduce the class of Regularized Kernel Estimates (RKEs), which we define as solutions to optimization problems of the following form:

$$\min_{K \in S_N} \sum_{(i,j)\in\Omega} L\big(w_{ij}, d_{ij}, \hat{d}_{ij}(K)\big) + \lambda J(K), \tag{3.3.1}$$

where S_N is the convex cone of all real nonnegative definite matrices of dimension N, Ω is the set of pairs for which we utilize dissimilarity information, and L is some reasonable loss function, convex in $\hat{d}_{ij}$, where $\hat{d}_{ij}$ is the dissimilarity induced by K. J is a convex kernel penalty (regularizing) functional, and λ is a tuning parameter balancing fit to the data and the penalty on K. The w_{ij} are weights that may, if desired, be associated with particular (i,j) pairs. The natural induced dissimilarity, which is a real squared distance admitting of an inner product, is $\hat{d}_{ij} = K(i,i) + K(j,j) - 2K(i,j) = B_{ij} \cdot K$, where $K(i,j)$ is the (i,j) entry of K and B_{ij} is a symmetric matrix of dimension N with all elements 0 except $B_{ij}(i,i) = B_{ij}(j,j) = 1$, $B_{ij}(i,j) = B_{ij}(j,i) = -1$. The inner (dot) product of two matrices of the same dimensions is defined as: $A \cdot B = \sum_{i,j} A(i,j) \cdot B(i,j) \equiv \text{trace}(A^T B)$. There are essentially no restrictions on the set of pairs other than requiring that the graph of the objects with pairs connected by edges be connected. A pair may have repeated observations, which just yield an additional term in (3.3.1) for each separate observation. If the pair set induces a connected graph, then the minimizer of (3.3.1) will have no local minima.

Although it is usually natural to require the observed dissimilarity information $\{d_{ij}\}$ to satisfy $d_{ij} \geq 0$ and $d_{ij} = d_{ji}$, the general formulation above does not require these properties to hold. The observed dissimilarity information may be incomplete (with the restriction noted), it may not satisfy the triangle inequality, or it may be noisy. It also may be crude, as for example when it encodes a small number of coded levels such as "very close", "close", "distant", and "very distant".

In this work we consider two special cases of the formulation (3.3.1), the first for its use in the application to be discussed in detail.

3.4 Numerical Methods for RKE

In Section 3.3, we define a general formulation of the RKE problem. In this section, we describe a specific formulation of the general form in Section 3.3, based on a linearly weighted l_1 loss, and use the trace function in the regularization term to promote dimension reduction. The resulting problem is as follows:

$$\min_{K \succeq 0} \sum_{(i,j)\in\Omega} w_{ij} |d_{ij} - B_{ij} \cdot K| + \lambda \operatorname{trace}(K). \tag{3.4.1}$$

We show how this formulation can be posed as a convex conic optimization problem and also describe a "newbie" formulation in which the known solution to (3.4.1) for a set of N objects is augmented by the addition of one more object together with its dissimilarity data. A variant of (3.4.1), in which a quadratic loss function is used in place of the l_1 loss function, is described in the Appendix A.1.

The reason that we use trace as the kernel regularization function is very intuitive. We want to obtain low rank kernels as RKE solutions. But rank is not a nice function to optimize with since it is discontinuous and non-convex. So we use trace, which is a continuous and linear (thus convex) function of the kernel, as a simple approximation to the rank. Another intuition is the idea of LASSO (Tibshirani (1996)). Since we want to promote the sparsity among eigenvalues of the estimated kernel, the LASSO idea suggests to regularize the l_1 norm of the eigensequence vector (i.e. trace, since the all eigenvalues of a kernel are non-negative) instead of the l_0 norm (i.e. rank).

Although choosing different weighting schemes will be an interesting problem, in this work we let $w_{ij} = 1$, $\forall (i,j) \in \Omega$. However, it is worth mentioning that computationally, it won't cost anything extra to add the w_{ij} term.

3.4.1 General Convex Cone Problem

We specify here the general convex cone programming problem. This problem, which is central to modern optimization research, involves some unknowns that are vectors in Euclidean space and others that are symmetric matrices. These unknowns are required to satisfy certain equality constraints and are also required to belong to cones of a certain type. The cones have the common feature that they all admit a self-concordant barrier function, which allows them to be solved by interior-point methods that are efficient in both theory and practice (Nesterov & Nemirovskii (1993)).

To describe the cone programming problem, we define some notation. Let $\mathcal{R}^p$ be Euclidean p-space, and let P_p be the nonnegative orthant in $\mathcal{R}^p$, that is, the set of vectors in $\mathcal{R}^p$ whose components are all nonnegative. We let Q_q be the second-order cone of dimension q, which is the set of vectors $x = \big(x(1), \ldots, x(q)\big) \in \mathcal{R}^q$ that satisfy the condition $x(1) \geq [\sum_{i=2}^{q} x(i)^2]^{1/2}$. We define S_s to be the cone of symmetric positive semidefinite $s \times s$ matrices of real numbers. Inner products between two vectors are defined in the usual way and we use the dot notation for consistency with the matrix inner product notation. The general convex cone problem is then:

$$\min_{X_j, x_i, z} \sum_{j=1}^{n_s} C_j \cdot X_j + \sum_{i=1}^{n_q} c_i \cdot x_i + g \cdot z \tag{3.4.2}$$

$$\text{s.t.} \sum_{j=1}^{n_s} A_{rj} \cdot X_j + \sum_{i=1}^{n_q} a_{ri} \cdot x_i + g_r \cdot z = b_r, \ \ \forall_r$$

$$X_j \in S_{s_j} \ \forall_j; \ \ x_i \in Q_{q_i} \ \forall_i; \ \ z \in P_p.$$

Here, C_j, A_{rj} are real symmetric matrices (not necessarily positive semidefinite) of dimension s_j, $c_i, \ a_{ri} \in \mathcal{R}^{q_i}$, $g, \ g_r \in \mathcal{R}^p$, $b_r \in \mathcal{R}^1$.

The solution of a convex cone problem can be obtained numerically using publicly available software such as SDPT3 (Tütüncü et al. (2003)) and DSDP5 (Benson & Ye (2004)).

3.4.2 RKE with l_1 Loss

To convert the problem of equation (3.4.1) into a convex cone programming problem, without loss of generality, we let Ω contain m distinct (i, j) pairs, which we index with $r = 1, 2, \ldots, m$. Define I_N to be the N-dimensional identity matrix and $e_{m,r}$ to be vector of length $2m$ consisting of all zeros except for the rth element being 1 and $(m + r)$th element being -1. If we denote the rth element of Ω as $\big(i(r), j(r)\big)$, and with some abuse of the notation let $i = i(r)$, $j = j(r)$ and $w \in P_{2m}$ with $w(r) = w(r + m) = w_{i(r),j(r)}$, $r = 1, \ldots, m$, we can formulate the problem of equation (3.4.1) as follows:

$$\begin{gathered} \min_{K \succeq 0, u \geq 0}\ w \cdot u + \lambda I_N \cdot K \\ \text{s.t. } d_{ij} - B_{ij} \cdot K + e_{m,r} \cdot u = 0, \ \ \forall_r, \\ K \in S_N, \ \ u \in P_{2m}. \end{gathered} \tag{3.4.3}$$

Though simple, it is non-trivial for us to show in the following theorem that by solving the convex conic problem above, we will always get what we desire. The proof of this theorem can be found in the Appendix A.2.

Theorem 3.4.1. *The optimization problems (3.4.1) and (3.4.3) have the same global minimum which, because of the convexity of problem (3.4.3), is equal to every local minimum of (3.4.3).*

The convexity (see Appendix A.2) of problem (3.4.3) gives us the computational convenience in obtaining the global minimum of problem (3.4.1) through convex programming. Even though, problem (3.4.3) is not guaranteed to be strictly convex, we are guaranteed to have achieved (numerically) the global minimum of problem (3.4.1) if the local minimum conditions for (3.4.3) are verified, since every local minimum of a convex problem coincides with its global minimum. These multiple local minima if exist, consist a flat (with the same value) convex region.

This formulation involves semi-definite cone and linear cones. We can also formulate it as a pure semi-definite programming (SDP) problem, which after minor modification can be adopted for the RKE problem with square loss and trace penalty.

3.4.3 "Newbie" Formulation

We now consider the situation in which a solution K_N of (3.4.1) is known for some set of N objects. We wish to augment the optimal kernel (by one row and column), without changing any of its existing elements, to account for a new object (a newbie). That is, we wish to find a new "pseudo-optimal" kernel $\tilde{K}_{N+1}$ of the form:

$$\tilde{K}_{N+1} = \begin{bmatrix} K_N & b \\ b^T & c \end{bmatrix} \succeq 0, \tag{3.4.4}$$

(where $b \in \mathcal{R}^N$ and c is a scalar) that solves the following optimization problem:

$$\begin{aligned} &\min\nolimits_{c \geq 0, b} \textstyle\sum_{i \in \Psi} w_i \; |d_{i,N+1} - B_{i,N+1} \cdot K_{N+1}| \\ &\text{s.t. } \; b \in \text{Range}(K_N), \;\; c - b^T K_N^+ b \geq 0, \end{aligned} \tag{3.4.5}$$

where K_N^+ is the pseudo-inverse of K_N and Ψ is a subset of $\{1, 2, \ldots, N\}$ of size t. The quantities w_i, $i \in \Psi$ are the weights assigned to the dissimilarity data for the new point. The constraints in this problem are the necessary and sufficient conditions for $\tilde{K}_{N+1}$ to be positive semidefinite.

Suppose that K_N has rank $p < N$ and let $K_N = \Gamma\Lambda\Gamma^T$, where $\Gamma_{N\times p}$ is the orthogonal matrix of non-zero eigenvectors and Λ is the $p \times p$ matrix of positive eigenvalues of K_N. By introducing the variable $\tilde{b}$ and setting $b = \Gamma\Lambda^{1/2}\tilde{b}$, we can ensure that the requirement $b \in \text{Range}(K_N)$ is satisfied. We also introduce the scalar variable $\tilde{c}$, and enforce $c \geq \tilde{c}^2$ by requiring that

$$Z \overset{\text{def}}{=} \begin{bmatrix} 1 & \tilde{c} \\ \tilde{c} & c \end{bmatrix} \in S_2. \tag{3.4.6}$$

Using these changes of variable, the condition $c - b^T K_N^+ b \geq 0$ is implied by the condition

$$x \overset{\text{def}}{=} [\tilde{c} \;\; \tilde{b}^T]^T \in Q_{p+1}.$$

Further we define the $N \times (p+1)$ matrix $\Sigma \overset{\text{def}}{=} [0_N \; 2\Gamma\Lambda^{1/2}]$, where 0_N is the zero vector of length N, and let $\Sigma_{i\cdot}$ be the row vector consisting of the $p+1$ elements of row i of Σ. We use $K_N(i,i)$ to denote the iith entry of K_N and define the weight vector $w \in P_{2t}$ with components $w(r) = w(t+r) = w_{i(r)}$, $r = 1, \ldots, t$. We then replace problem (3.4.5) by the following equivalent convex cone program:

$$\min_{Z \succeq 0, u \geq 0, x} \quad w \cdot u \tag{3.4.7}$$

$$\begin{aligned}
\text{s.t.} \qquad & \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot Z = 1, \\
& \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \cdot Z - \begin{bmatrix} 1 \\ 0_p \end{bmatrix} \cdot x = 0, \\
d_{i,N+1} - K_N(i,i) - & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot Z + \Sigma_{i\cdot} \cdot x + e_{t,r} \cdot u = 0, \quad \forall_{r=1,2,\ldots,t}, \\
& Z \in S_2, \quad x \in Q_{p+1}, \quad u \in P_{2t},
\end{aligned}$$

where $i = i(r)$ as before. Note that the constraints on Z ensure that it has the form (3.4.6).

The positive semi-definite constraint on Z and second-order cone constraint on $x = [\tilde{c} \;\; \tilde{b}^T]^T$ gives $c \geq \tilde{c}^2 \geq \tilde{b} \cdot \tilde{b}$, which by definition of c and $\tilde{b}$ is exactly the positive semi-definite condition of K_{N+1} in the "newbie" problem. And similarly we know we are doing the right thing by the following theorem.

Theorem 3.4.2. *The optimization problems (3.4.5) and (3.4.7) have the same global minimum which, because of the convexity of problem (3.4.7), is equal to every local minimum of (3.4.7).*

There are some interesting things about this newbie problem. One interesting observation is the fact that by augmenting the RKE kernel K_N for "training" data to accommodate one more newbie (then, more and more), the rank of the kernel will either increase by one if $c - b^T K_N^+ b > 0$ or remain unchanged $c - b^T K_N^+ b = 0$. Thus the quantity $(c - b^T K_N^+ b)$ can serve as a indicator whether or not a newbie “brings” something “new”. If the training data is Representative enough, we should expect this quantity to be close to machine zero (numerically). Another observation is that if we do have $c - b^T K_N^+ b > 0$ significantly, by projecting the newbie solution for (3.4.7) back to the lower rank space (expanded by K_N), we almost always won’t get the restricted newbie solution (require the augmented kernel to be the same rank as K_N, i.e., $c - b^T K_N^+ b = 0$). This observation actually shows that our kernel approach to this newbie problem is non-trivially innovative because we suspect that the problem will become nonconvex if we force $c - b^T K_N^+ b = 0$ instead of $c - b^T K_N^+ b \geq 0$. Both observations are more precisely described in two lemmas in the Appendix A.5 and A.6.

3.4.4 Choosing Elements of Ω

If the dissimilarity information is symmetric (i.e., $d_{ij} = d_{ji}$), we can choose Ω to be the subset of $\{(i, j) : i < j\}$ for which information is available. However, the codes we use for solving formulation (3.4.3) (SDPT3 and DSDP5) require $O(m^2)$ storage (where m is the size of Ω), which is prohibitive for the application we describe in Section 3.6. Hence, we define Ω by randomly selecting a subset of the available dissimilarity information in a way that ensures that each object i appears with roughly the same frequency among the (i, j) pairs of Ω. Specifically, for each i, we choose a fixed number k of pairs (i, j) with $j \neq i$ (we call the objects j “buddies” of i) and add either (i, j) or (j, i) to the set Ω, reordering if necessary to ensure that the first index of the pair is smaller than the second. (It is possible that (j, i) has been placed in Ω at an earlier stage.) We choose the parameter k sufficiently large that the solution of (3.4.3) does not vary noticeably with different random subsets.

The "newbie" formulation (3.4.5) is comparatively inexpensive to solve, so we take Ψ to be the complete set of objects for which dissimilarity information $d_{i,N+1}$ is available.

3.4.5 Eigenanalysis, Tuning, Truncation

The left five panels of Figure 3.4.1 illustrate the effect of varying λ on the eigenvalues of the regularized estimate of K obtained by solving (3.4.3). The data is from the example to be discussed later in Section 3.6, with $N = 280$ objects and $k = 55$ "buddies" for each of the N objects. Note that the vertical scale is in units of $\log_{10} \lambda$. As λ increases the smaller eigenvalues begin to shrink, although in this example there is a very broad range of values of λ, spanning several orders of magnitude, where the sensitivity to λ is barely visible. At $\lambda = 10^{-8}$ the condition number of K is about 10^3. As λ goes much past 200 in this example, the penalty on K dominates and the dissimilarity information in the data is suppressed.

It is desirable to have a kernel with rank as low as possible while still respecting the data to an appropriate degree. Even if the rank of the regularized kernel estimate is not low, a low rank approximation obtained by setting all but a relatively small number of the largest eigenvalues to zero might retain

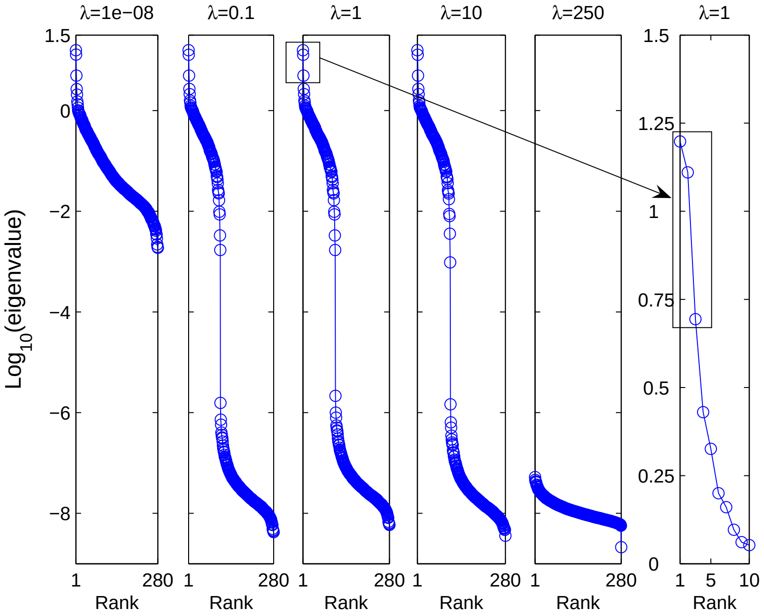

Figure 3.4.1: Left five panels: log scale eigensequence plots for five values of λ. As λ increases, smaller eigenvalues begin to shrink. Right panel: first ten eigenvalues of the $\lambda = 1$ case displayed on a larger scale.

enough information to provide an efficient way of doing classification or clustering. In the work described here, as well as in various simulation studies, we started with a very small positive λ, increased λ in a coarse log scale, and then experimented with retaining various numbers of eigenvalues to get a low rank approximation to K. The rightmost panel in Figure 3.4.1 shows the first 10 eigenvalues for the $\lambda = 1$ case, in an expanded log scale. Natural breaks appear after both the second and the third eigenvalues. Setting all the eigenvalues of K after the largest p to 0 results in the νth coordinates of the jth object as $x_j(\nu) = \sqrt{\lambda_\nu}\phi_\nu(j), \nu = 1, 2, \ldots, p$, where the λ_ν, ϕ_ν are the first p eigenvalues and eigenvectors of K and $\phi_\nu(j)$ is the j component of ϕ_ν. We remark that the coordinates of the N objects are always centered at the origin since the RKE estimate of K always has the constant vector as a 0 eigenvector. This auto-centering effect is a byproduct of trace regularization. When tuning parameter λ is positive, the RKE kernel will be centered automatically. This can be easily understood though thinking the following: If we only shift the origin away from the center (i.e. sample mean) of a group of data points in Euclidean space, the resulting gram matrix will have the same loss but bigger trace than the gram matrix for the centered configuration. Therefore, only centered kernel can be RKE minimizer.

In the example discussed later in Section 3.6 with four classes of labeled objects, the choice of $\lambda = 1$ and $p = 3$ provided plots with a clear, informative clustering on the labels, that was verified from the science of the subject matter. We note that using the estimated K or a low rank version of it as the kernel in an SVM will result in linear classification boundaries in the object coordinates, (piecewise linear in the case of the multicategory SVM (MSVM) of Lee et al. (2004*a*)). It will be seen in the plots for labeled objects in Section 3.6 that piecewise linear classification boundaries in $p = 3$ coordinates would apparently do quite well. However, that will not necessarily always be the case, and a more flexible workhorse kernel in the p object coordinates can be used. The MSVM in Lee et al. (2004*a*) comes with a cross validation based method for choosing the MSVM tuning parameter(s) in a labeled training set. In principle, the parameters λ and p here can be incorporated in that method or other related methods, as additional MSVM parameters. Further examination of principled methods of choosing λ and p along with MSVM tuning parameter(s) will be deferred to later work.

3.5 Applying RKE to Simulated Data

3.5.1 Procrustes Measures

For the simulated data, the truth is known. A reasonable measure of the distance/dissimilarity between two kernel matrices is needed to characterize the goodness of fit for different estimates. In some related literature, it is called Procrustes measure.

A suitable measure proposed in Sibson (1978) is based on the positional differences after matching two gram matrix under translation, rotation and reflection. Suppose A and B are two centered gram matrices, then the measure is calculated as follows:

$$G(A, B) = trace(A) + trace(B) - 2trace(A^{1/2}BA^{1/2})^{1/2}. \tag{3.5.1}$$

The normalized version of this measure is simply:

$$\gamma_p(A, B) = G(A, B)/(trace(A)trace(B))^{1/2}. \tag{3.5.2}$$

Alternatively, if we care only about the pairwise distance information, we can introduce another normalized measure:

$$\gamma_d(A, B) = \sum_{i<j} |\hat{d}_{ijA} - \hat{d}_{ijB}| / \sum_{i<j} \frac{1}{2}(\hat{d}_{ijA} + \hat{d}_{ijB}), \tag{3.5.3}$$

Table 3.5.1: Procrustes Measure between Result and Truth for different λs

	$\lambda = 0.1$	$\lambda = 400$	$\lambda = 420$	$\lambda = 450$
γ_p	0.090	0.0089	0.0081	0.8693
γ_d	0.0309	0.0269	0.0253	0.9032

where $\hat{d}_{ijA}$ and $\hat{d}_{ijB}$ are pairwise squared distance between object i and j, induced by A and B respectively.

3.5.2 An Example of Simulated Clusters

We simulated three clusters in the two-dimensional Euclidean space. The data points, 63 of them in total, are random samples from three distinct bivariate normal distributions. To check the ability of our method to recover the clustering structure under noise, we obtained the dissimilarity data using the following procedure. We first added two noisy coordinates to each data point. These two noisy coordinates follow two independent normal distributions with relatively small variances. The squared Euclidean distances between all pairs of data points, i.e. $d_{ij}s$ in our notation, were then binned into 10 equal sized bins over the interval from the minimum to the maximum of those positive $d_{ij}s$. The value of each d_{ij} was then replaced by the center value of the bin to which it belongs. It is an analog of the scenario where only ranks are provided as the distance/dissimilarity measure. The noisy $d_{ij}s$ were then treated as observed dissimilarity data. Note that, the binning procedure here can introduce very non-Euclidean noise.

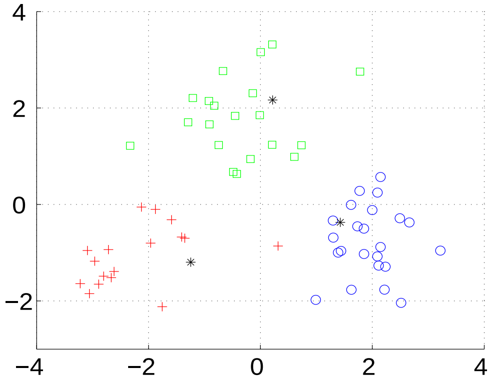

Figure 3.5.1: Noisy Clusters: Original data. Black stars are three left-out "newbies" from three clusters respectively. The size of this plot has been intentionally shrunk to be comparable to the corresponding plots in Figure 2 on the next page.

Since the sample size is small in this simulated example, we didn't follow the neighborhood construction procedure discussed in Section 3.4.4. Instead, we used all distinct pairwise squared distances. We also saved aside one data point from each cluster to test our "newbie" algorithm. The RKE and newbie formulations we used here for this example are square-loss formulations (A.1.1) and (A.1.3) in the Appendix A.1. The original clusters are displayed in Figure 3.5.1, with different colors and symbols for different clusters. The true newbie positions are marked with black stars. The same colors and marks are used for the RKE recovered configurations in the upper right and lower right plots of Figure 3.5.2. In Figure 3.5.2, the upper two plots are RKE results with $\lambda = 0.1$, while the lower two are RKE results with $\lambda = 400$. As we can see both the recovered configurations and the newbie positions are fairly close to the truth. However, the estimated kernel with $\lambda = 0.1$ has many eigenvalues besides the

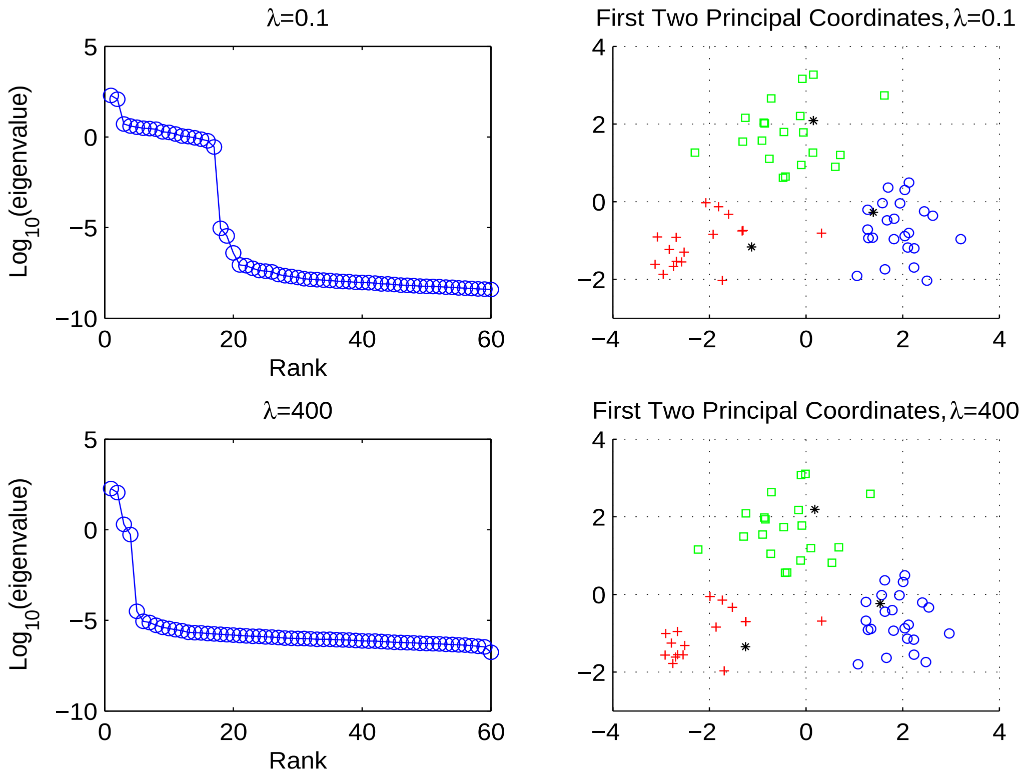

Figure 3.5.2: Noisy Clusters: Effect of λ on the Regularized Kernel Estimate using (A.1.2). The two upper plots are RKE results with $\lambda = 0.1$. The upper left one is the eigensequence plot. The upper right one is the plot of principal coordinates for recovered configuration and "newbies". The two lower plots are RKE results with $\lambda = 400$.

most significant two, corresponding to small noisy dimensions; while for the estimated kernel with $\lambda = 400$, most of these noisy eigenvalues “dropped” to machine zero. That clearly showed the desired effect of the trace regularization term promoting dimension reduction. Moreover, when $\lambda = 0.1$, we got the Procrustes measures $\gamma_p = 0.090$ and $\gamma_d = 0.0309$, while when $\lambda = 400$, we had $\gamma_p = 0.0089$ and $\gamma_d = 0.0269$. This suggests that through tuning λ in a proper way, we can improve the accuracy of the estimated kernel. The “optimal” λ obviously lies within certain appropriate range on the positive real axis because if we make the λ too big, it will eventually suppress every eigenvalue close to machine zero and the RKE kernel will become a kernel of all noise when the Procrustes measures blow up (See Table (3.5.2)).

3.6 Protein Clustering and Visualization with RKE

3.6.1 Background

One of the challenging problems of contemporary biology is inferring molecular functions of unannotated proteins. A widely used successful method of protein function prediction is based on sequence similarity. Statistically significant sequence similarity, which is typically based on a pairwise alignment score between two proteins, forms the basis for inferring the same function. Two major related problems exist for predicting function from sequence. The first problem is the clustering of large number of unlabeled protein sequences into subfamilies for the purpose of easing database searches and grouping similar proteins together. The second problem is concerned with assigning new unannotated proteins to the closest class, given the labeled or clustered training data. There is a substantial amount of literature addressing these two problems. Krogh et al. (1994) employs profile hidden Markov models (HMMs) for both problems. Clustering of proteins is obtained by a mixture of profile HMMs whereas assignment of new protein sequences to the clusters/classes is based on the likelihood of the new sequence under each of the cluster specific HMMs. Later, Jaakkola et al. (2000) addresses the second problem by first obtaining an explicit vector of features (Fisher scores) for each protein sequence and then utilizing a variant of SVMs, using a kernel called the Fisher kernel for classification purposes. The feature vector for each protein sequence is based on the likelihood scores of the input sequence evaluated at the corresponding maximum likelihood estimates of the HMM parameters fitted on the training data. More recently, Liao & Noble (2003) similarly uses SVMs for protein classification. However, in contrast to obtaining a feature vector by likelihood scores, they define a feature vector for each protein sequence as a vector of its pairwise sequence similarity scores to all other proteins. Alternatively, Leslie et al. (2004) represents protein sequences as vectors in a high-dimensional feature space using a string-based feature map and train an SVM based on these vectors using a mismatch kernel. These latter works clearly illustrate the advantage of representing each protein sequence by a high-dimensional feature vector in some coordinate system and the power of kernel methods for protein classification. The RKE methodology presented here provides an efficient way to represent each protein sequence by a feature vector in a chosen coordinate system by utilizing the pairwise dissimilarity between protein sequences.

3.6.2 Data

We illustrate the utility of RKE methodology using a dataset of globins that was first analyzed in Krogh et al. (1994) by a profile HMM approach. The dataset, distributed with the HMMER2 software package (Eddy (1998)), has a total of 630 globin sequences. The globin family is a large family of heme-containing proteins with many sub-families. It is mainly involved in binding and/or transportation of oxygen. For illustration purposes, we randomly choose 280 sequences from these data so that three large sub-classes of the globin family (alpha chains, beta chains, myoglobins) are included along with a heterogeneous class containing various types of chains. This selection resulted in a total of 112 "alpha-globins", 101 "beta-globins", 40 "myoglobins", and 27 "globins" (the heterogeneous class). The proportion of sequences in each class were taken to be proportional to the class sizes in the original dataset.

3.6.3 Implementation of RKE

We used the RKE formulation of Section 3.4.2 for this application. The `Bioconductor` package `pairseqsim` (Gentleman et al. (2004)) was used to obtain global pairwise alignment scores for all pairs of $N = 280$ sequences. This procedure gave a total of $N(N-1)/2 = 39060$ similarity scores, which we then normalized to map into the interval $[0, 1]$. We used one minus each of these numbers as the dissimilarity measure for each pair of sequences. During this process, alignment parameters were taken to be equal to the BLAST server (Atschul et al. (1990)) defaults. To construct the active index pair set Ω, we used the procedure described in Section 3.4.4 with $k = 55$ randomly chosen buddies for each protein sequence. The set Ω thus contained approximately 14000 sequence pairs, corresponding to about 36% of the size of the complete index set. Replicated runs with $k = 55$ proved to be nearly indistinguishable, as judged by examination of eigenvalue and 3D plots and the Procrustes measure γ_d as defined in Section 3.5.1 (γ_d is typically less than 5% for each pairwise comparison). The tuning parameter λ is set to 1 in the plots that follow later in this section.

3.6.4 Visualization of the Globin Sequence Space and Results

Figure 3.6.1 displays the 3D representation of the sequence space of 280 globins. This figure shows that the first three coordinates of the protein sequence space, corresponding to three largest eigenvalues, is already quite informative. The

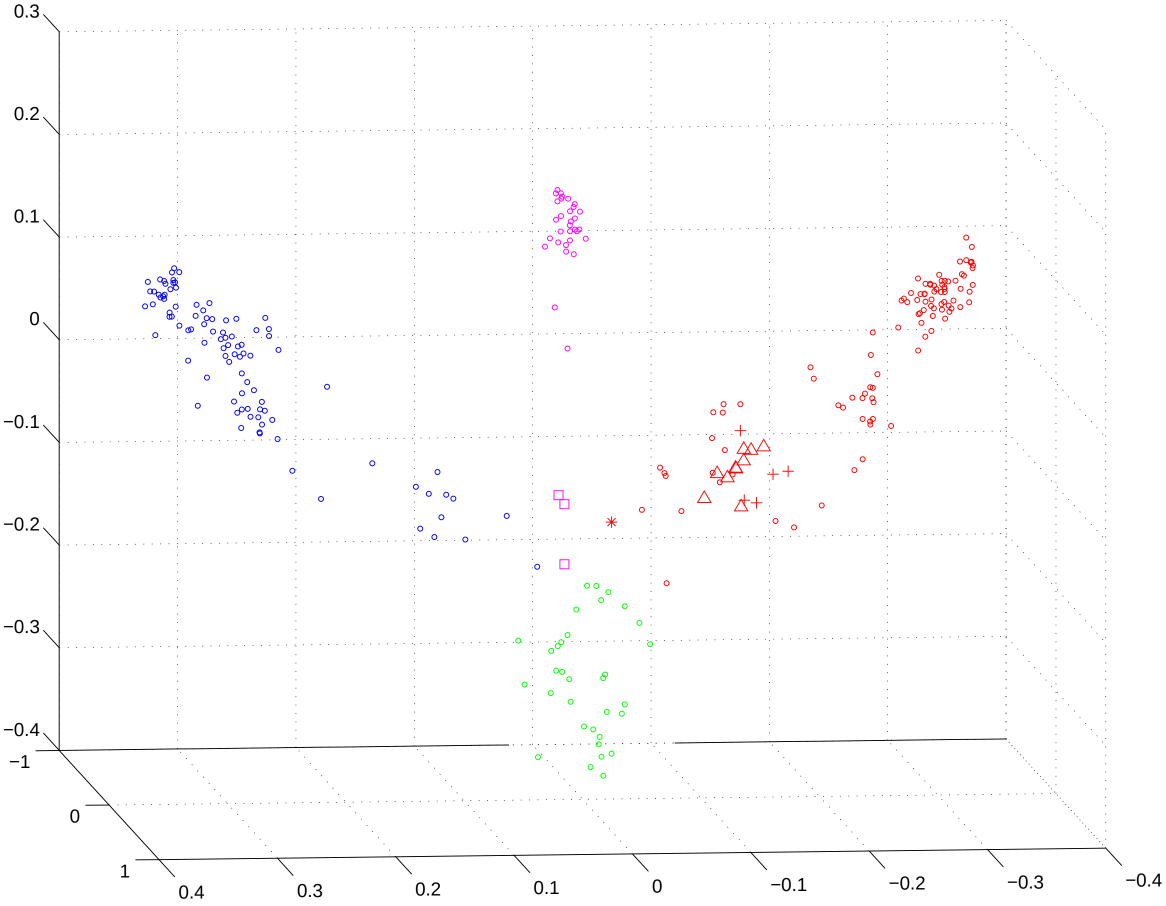

Figure 3.6.1: *3D representation of the sequence space for 280 proteins from the globin family.* Different subfamilies are encoded with different colors: Red symbols are alpha-globin subfamily, blue symbols are beta-globins, purple symbols represent myoglobin subfamily, and green symbols, scattered in the middle, are a heterogeneous group encompassing proteins from other small subfamilies within the globin family. Here, hemoglobin zeta chains are represented by the symbol of red +, fish myoglobins are marked by the symbol of purple □, and the diverged alpha-globin `HBAM_RANCA` is shown by the symbol of red *. Hemoglobin alpha-D chains, embedded within the alpha-globin cluster, are highlighted using the the symbol red △.

four main classes of the globin family are visually identifiable: The four colors red, blue, purple, and green represent alpha-globins, beta-globins, myoglobins, and globins, respectively.

Further investigation of this 3D plot reveals several interesting results. First, we observe that the five hemoglobin zeta chains, namely `HBAZ_HORSE`, `HBAZ_HUMAN`, `HBAZ_MOUSE`, `HBAZ_PANTR`, `HBAZ_PIG`, shown by red +, are located close to each other and embedded within the alpha-globin cluster. Zeta-globin chains are alpha-like polypeptides and are synthesized in the yolk sac of the early embryo. It is well known that human zeta-globin polypeptide is more closely related to other mammalian embryonic alpha-like globins (i.e., zeta-globins) than to human alpha globins (Clegg & Gagnon (1981)). Furthermore, the zeta-globin gene in humans is a member of the alpha-globin gene cluster. Second we note that `HBAM_RANCA`, which is represented by red * and is a hemoglobin alpha-type chain, seems to be isolated from the rest of the alpha-globin sequences. A possible explanation might be found in the structure of this protein. Maeda & Fitch (1982) notes that the gene encoding this protein appeared through a gene duplication of hemoglobin and this took place near the time of the duplication that generated the alpha and beta chains. Our third observation is that the myoglobins `MYG_MUSAN`, `MYG_THUAL`, and `MYG_GALJA`, denoted by purple □, which are all fish myoglobins (*Mustelus antarcticus* (Gummy shark), *Thunnus albacares* (Yellowfin tuna), *Galeorhinus japonicus* (Shark)) - appear to be slightly separated from the rest of the myoglobin cluster. This is quite a remarkable observation because fish myoglobins are known to be structurally distinct from the mammalian myoglobins (Cashon et al. (1997)) and the RKE method nicely highlights this distinction on the basis of primary sequence data only. The 3D plot also reveals sub-clusters in the alpha-globin cluster. For example, all the 10 hemoglobin alpha-D chains (shown by red △ in Figure 3.6.1) are clustered together within the alpha-globin cluster.

In a recent work, Hou et al. (2005) provided a 3D plot of the protein structure space of 1898 chains. These authors utilized multi-dimensional scaling to project protein structures to a lower dimensional space based on the pairwise structural dissimilarity scores derived from 3D structures of proteins. Our application of RKE to the globin family, which is a few levels down from the top level of the protein structure hierarchy considered by Hou et al. (2005), provide an analogous 3D plot for the sequence space of the globin family. It is quite encouraging that sub-protein domains of this family are readily distinguishable from the 3D embedding of the protein sequences. It is also worth mentioning that our current application is concerned only with pairwise sequence similarity, which can be obtained efficiently. However, clustering at the higher levels of the protein structure hierarchy is known to benefit enormously from 3D structural similarities and we plan to perform clustering at that level in future work (see Section 3.7 for details).

3.6.5 Classification of New Protein Sequences

We next illustrate how the newbie algorithm can be used to visualize unannotated protein sequences in the coordinate space of training data obtained by RKE. We used the following sequences as our test data: (1) `HBAZ_CAPHI`: hemoglobin zeta chain from goat *Capra hircus*; (2) `HBT_PIG`: Hemoglobin theta chain from pig *Sus scrofa*. Figure 3.6.2 displays the positions of these two test sequences with respect to 280 training sequences. We observe that `HBAZ_CAPHI` (black circle) clusters nicely with the rest of the hemoglobin zeta chains, whereas `HBT_PIG` (black star), which is an embryonic beta-type chain, is located closer to beta-globins. Additionally, we also used 17 Leghemoglobins (black triangles) as test data and found that these cluster tightly within the heterogeneous globin group. This observation is consistent with the results of Krogh et al. (1994), who also found a heterogeneous globin cluster with a tight sub-class of Leghemoglobins among their seven clusters obtained by a mixture of HMMs. These results indicate that RKE together with newbie algorithm provide a powerful means for clustering and classifying proteins.

3.7 Discussions

We have described a framework for estimating a regularized kernel (RKE methodology) from general dissimilarity information via the solution of a convex cone

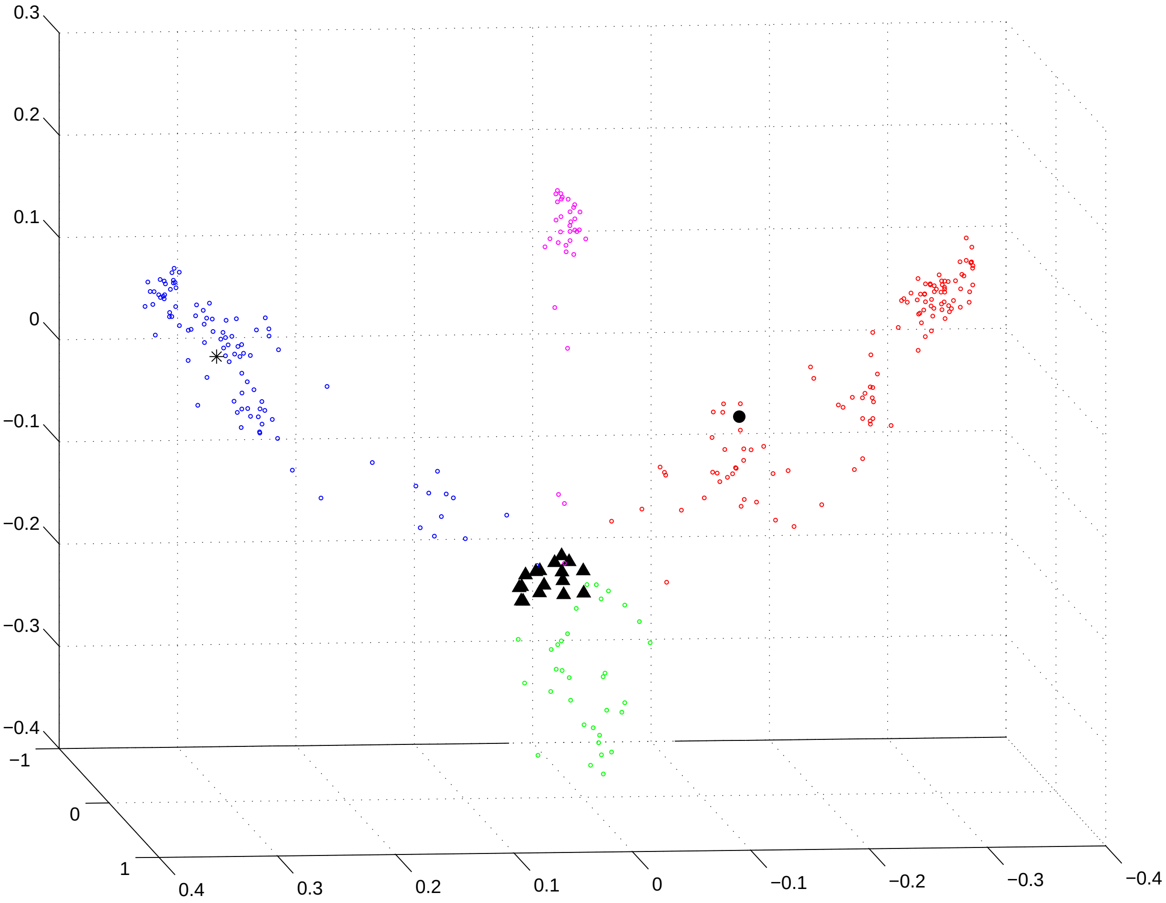

Figure 3.6.2: *Positioning test globin sequences in the coordinate system of 280 training sequences from the globin family.* The newbie algorithm is used to locate one Hemoglobin zeta chain (black circle), one Hemoglobin theta chain (black star), and seventeen Leghemoglobins (black triangles) into the coordinate system of the training globin sequence data.

optimization problem. We have presented an application of the RKE methodology (including the "newbie" algorithm) to homology detection in the globin family of proteins. The most striking result here is perhaps the fact that a simple 3D plot is sufficient for visual identification of the subfamily information. However, in other applications, the plot coordinates (or higher dimensional coordinate vectors obtained by retaining more eigenvalues) may be used to build an automatic classification algorithm via the (principled) multicategory support vector machine (MSVM) (Lee et al. (2004*a*)). That algorithm comes with a tuning method, it partitions the attribute space into regions for each training category, and it also comes with a method for signaling "none of the above". Multicategory penalized likelihood estimates may also be used if there is substantial overlap of the data from different classes (Wahba (2002*b*), Lin (1998), Wahba et al. (1995*a*), Zhu & Hastie (2004)).

A much harder problem in the context of protein classification and clustering is remote homology detection, that is, detecting homology in the presence of low sequence similarity. Since our framework accommodates an arbitrary notion of dissimilarities, we can easily take advantage of various types of dissimilarities such as presence or absence of discrete sequence motifs (Ben-Hur & Brutlag (2003)) and dissimilarities based on the primary, secondary, and tertiary structure (Tang et al. (2003)), and obtain optimal kernels from each piece of information or data set. Without using labeled training sets, relations between a pair of kernels from different sources of information (or their lower rank approximations) can be quantified in various ways. A simple example is a measure of correlation: $\sum_{ij} \hat{d}^{s/2}_{ij\alpha}\hat{d}^{s/2}_{ij\beta}/((\sum_{ij} \hat{d}^{s}_{ij\alpha})^{1/2}(\sum_{ij} \hat{d}^{s}_{ij\beta})^{1/2})$ where α and β index the different sources of information and s is a real number to be chosen. With labeled data, these kernels can further be examined and combined in an optimal way, as for example in Lanckriet et al. (2004), in the context of classification. As emphasized above, a striking feature of the presented methodology is the fact that it can exploit any type of dissimilarity measure, and data sets with missing information. These properties are clearly beneficial in biological data analysis, since many biologically relevant dissimilarities may not naturally result in positive semidefinite kernels (pairwise alignment scores, for example) which are essential for powerful classification methods such as SVMs.

Homology detection is one type of computational biology problem for which our framework offers rigorous, flexible tools. However, there are many other computational biology applications which can naturally be handled within this framework. Clustering of transcription factor position weight matrices (binding profiles) is one such application. With the increasing growth of transcription factor binding site databases, such as Sandelin et al. (2004), a need for characterizing the space of DNA binding profiles and for developing tools to identify the class of newly estimated/studied profiles is emerging. A characterization of all available experimentally verified binding profiles such as in Sandelin et al. (2004) might provide invaluable information regarding the possible class of binding profiles. Such information can then be utilized in supervised motif finding methods such as Keleş et al. (2003). A natural dissimilarity measure for binding profiles is the Kullback-Leibler divergence. Clustering of the experimentally verified binding profiles based on a regularized kernel estimate of such dissimilarity measure might group binding profiles in a way that is consistent with the DNA binding domains of the transcription factors. We envision that this might generate a "protein binding profile space", as the work of Hou et al. (2005) generated a "protein structure space".

Future work of interest includes both extension of the methodology and extension of the applications; in biology, the clustering of proteins at the top level of the protein hierarchy; and in other contexts, medical images in particular. We can also investigate other choices of loss and penalty functionals to achieve better dimension reduction.

Chapter 4

Robust Manifold Unfolding with Kernel Regularization

4.1 Summary

We describe a robust method to unfold a low-dimensional manifold embedded in high-dimensional Euclidean space based on only pairwise distance information (possibly noisy) from the sampled data on the manifold. Our method is derived as one special extension of the recently developed framework called Kernel Regularization, which is originally designed to extract information in the form of a positive definite kernel matrix from possibly crude, noisy, incomplete, inconsistent dissimilarity information between pairs of objects. The special formulation is transformed into an optimization problem that can be solved globally and efficiently using modern convex cone programming techniques. The geometric interpretation of our method will be discussed.

4.2 Introduction

The dimensionality reduction problem appears in many research fields, where scientists try to conduct exploratory analysis or visualization of multivariate data. One special scenario happens often, when the goal is to find a meaningful/expected low-dimensional structure behind high-dimensional observations, or more precisely, to recover a low-dimensional parameterization of high-dimensional data assuming the data all lie on a low-dimensional manifold. In several recent papers (Tenenbaum et al. (2000), Roweis & Saul (2000), Belkin & Niyogi (2003), Donoho & Grimes (2003), Weinberger et al. (2004)) a large family of algorithms has been proposed to solve this particular type of dimensionality reduction problem (hereinafter, manifold-unfolding problem), in the spirit of reconstructing the manifold structure globally, but respecting only local information from the observed data. It is also well known (Weinberger et al. 2004, Ham et al. 2004, Schölkopf et al. 1998, Williams 2001) that the solution to the manifold-unfolding problem is closely related to finding a symmetric positive definite kernel (hereinafter "kernel"). Recall that an $N \times N$ kernel obtained from data relating N objects may be used to assign Euclidean coordinates to the N objects in some $p < N$ Euclidean space.

In a recent work Lu et al. (2005) (see Chapter 3), a novel framework called Kernel Regularization is proposed to extract information in the form of a kernel matrix from possibly crude, noisy, incomplete, inconsistent dissimilarity or distance-like information between pairs of objects, while controlling a certain complexity measure of the kernel. A special formulation of the framework was applied to configure a set of proteins globally in a Euclidean space based on pairwise dissimilarity information only (see Chapter 3). The configuration (in the form of a kernel) obtained can then be used by any clustering or classification algorithm for further inference if visualization is not the only purpose. Kernel regularization can be used replace of the roles of two traditional techniques, multidimensional scaling (MDS) and principal component analysis (PCA), in a unified fashion using the special formulation in Chapter 3. However, the manifold-unfolding problem is different from the problem targeted in Chapter 3, where global information needs to be preserved.

In this chapter, we describe a kernel approach to solve the manifold-unfolding problem using another variation/formulation of the kernel regularization framework. We adopt the same essential idea as in Tenenbaum et al. (2000), Roweis & Saul (2000), Belkin & Niyogi (2003), Donoho & Grimes (2003), Weinberger et al. (2004) which is, loosely speaking, to respect local information while flattening the global structure. Nevertheless, our method is more flexible than the others in terms of the assumptions on the data. All previous methods assume (at least implicitly) that there is no noise associated with the observed data, while our method allows the existence of noise. One scenario of the noisy-version manifold-unfolding problem is when the observed scattered data points can fall off the 'true' underlying manifold. More precisely, one can assume the direct distance between observed data points and the target manifold follows a compactly supported distribution with zero mean and relatively small variance compared to the global spread of the original manifold . Another possible source of noise is measurement error for either point coordinates or pairwise distances. A related issue here is that to the extent that the desired solution to the manifold-unfolding problem is translation and rotation invariant, a reasonable method should depend only on pairwise distance information. The procedures in algorithms proposed in Roweis & Saul (2000), Belkin & Niyogi (2003), Donoho & Grimes (2003) use more than pairwise distances (at least, procedure-wise), while implementation of our method, like the algorithms in Tenenbaum et al. (2000), Weinberger et al. (2004), needs only pairwise distances. It is also worth mentioning that, when the distance/dissimilarity information is actually noisy, it might also be non-Euclidean (e.g., the triangle inequality can be violated). Then, in that case, the algorithm in Weinberger et al. (2004) will try to solve an infeasible optimization problem. Nonetheless, our method can naturally handle this noisy situation. Moreover, our algorithm is insensitive to the non-convex case investigated in Donoho & Grimes (2003), which causes the algorithms proposed in Tenenbaum et al. (2000), Roweis & Saul (2000) to fail. So our method is robust for the manifold-unfolding problem in the sense that it can handle both noisy and non-convex data.

This chapter is organized as follows. In Section 4.3, we review the Kernel Regularization framework before proposing a new formulation of it in order to solve the (noisy) manifold-unfolding problem. We also discuss the geometric interpretation of different formulations. In Section 4.4, we show some simulation results from implementing our method. Finally, we conclude in Section 3.7 with a summary and discussion of future work.

4.3 Regularized Kernel Embedding

4.3.1 Framework of Kernel Regularization

In the same spirit of all regularization methods, the Kernel Regularization method is designed to estimate a target, in our case, a kernel, from observed information, while controlling a certain complexity measure of the resulting estimate to prevent overfitting. The most general framework can be expressed as the following optimization problem:

$$\min_{K \in S_n} L(data, K) + \lambda J(K), \tag{4.3.1}$$

where S_N is the convex cone of all real nonnegative definite matrices of dimension N and L is some reasonable loss function on K. J is a kernel penalty (regularizing) functional, and λ is a tuning parameter balancing fit to the data and the penalty on K. The choice of L obviously depends on the functional/distributional relationship (given or from model assumptions) between the observed data and target kernel, which is usually straightforward after the underlying problem is clear. On the other hand, a reasonable J can only be found after one understands/defines the complexity of the estimated kernel properly for a particular problem. Moreover, computational convenience should be considered when choosing L and J. In most cases, we want use L and J which makes the resulting optimization problem convex.

The Kernel Regularization framework proposed in Chapter 3 is motivated by the need to extract useful information from various kinds of dissimilarity information. Given a set of N objects, suppose we have obtained a measure of dissimilarity, d_{ij}, for certain object pairs (i, j). So the dissimilarity measure becomes the proxy to construct the loss function for the target kernel. Also, trace is chosen to be the kernel regularizing function J in order to promote dimension reduction in this case. One special formulation of Kernel Regularization framework in this scenario is the following:

$$\min_{K \in S_N} \sum_{(i,j)\in\Omega} w_{ij} |d_{ij} - B_{ij} \cdot K| + \lambda \operatorname{trace}(K), \tag{4.3.2}$$

where Ω is the set of pairs for which we utilize dissimilarity information and the w_{ij}s are weights that may, if desired, be associated with particular (i, j) pairs. The natural induced dissimilarity, which is a real squared distance admitting of an inner product, is $\hat{d}_{ij} = K(i, i) + K(j, j) - 2K(i, j) = B_{ij} \cdot K$, where $K(i, j)$ is the (i, j) entry of K and B_{ij} is a symmetric matrix of dimension N with all elements 0 except $B_{ij}(i, i) = B_{ij}(j, j) = 1$, $B_{ij}(i, j) = B_{ij}(j, i) = -1$. The inner (dot) product of two matrices of the same dimensions is defined as: $A \cdot B = \sum_{i,j} A(i, j) \cdot B(i, j) \equiv \operatorname{trace}(A^T B)$.

There are essentially no restrictions on the set of pairs other than requiring that the graph of the objects with pairs connected by edges be connected. A pair may have repeated observations, which just yield an additional term in

(4.3.1) for each separate observation. If the pair set induces a connected graph, then the minimizer of (4.3.1) will have no local minima.

4.3.2 Deriving the Regularized Kernel Embedding Formulation

Denote the observed squared distance between x_i and x_j by d_{ij}. Let $(i, j) \in \Omega$ if x_j is one of the k-nearest neighbors of x_i, according to this observed squared distance.

We formulate the problem as finding a new positioning of the N points in $\mathcal{R}^p$ so that

(i) the repositioning respects local distance;

(ii) the repositioned points lie in a subspace of $\mathcal{R}^p$ with as low a dimension as possible.

Denote the points after repositioning as y_i, $i = 1, ..., N$. That is, the positioning moves the original point x_i to y_i, $i = 1, ..., N$. These N points are still in $\mathcal{R}^p$, but we hope they lie in a low dimensional subspace in $\mathcal{R}^p$. Denote the distance between y_i and y_j in $\mathcal{R}^p$ by r_{ij}. Notice that the positioning that satisfies these two conditions is not unique. In fact, for any positioning that satisfies (i) and (ii), rotating the coordinate system or shifting the points by a common vector results in a different positioning that also satisfies (i) and (ii). To take care of the shifting problem, we require that the repositioned points are centered at 0_p, the origin of the coordinate system. That is, $\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{N} y_i = 0_p$. As will be seen later, we shall take care of the rotation problem by formulating the problem in terms of the reproducing kernel matrix generated by the repositioned points, instead of the repositioned points themselves. The reproducing kernel matrix K is the N by N matrix with the element $K(i,j) = (y_i, y_j)$, where $(\cdot,\cdot)$ is the Euclidean inner product in $\mathcal{R}^p$. Notice this matrix is invariant under rotation of the points y_i.

Condition (i) requires that

$$r_{ij}^2 \approx d_{ij}, \quad \forall (i,j) \in \Omega. \tag{4.3.3}$$

Now notice that among all possible positioning that satisfies condition (i), the positioning that meets condition (ii) is one in which the distance between pairs of the repositioned points in $\mathcal{R}^p$ are maximized. This is most easily seen in the broken stick example coming up later, but it is also easy to see in general. Therefore we try to maximize $\sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij}^2$ subject to (4.3.3). To balance the fidelity to local distances and the maximization of distance between all pairs, we use a penalty approach and try to minimize:

$$\sum_{(i,j)\in\Omega} (d_{ij} - r_{ij}^2)^2 - \lambda \sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij}^2, \tag{4.3.4}$$

where $\lambda > 0$ is a tuning parameter. Alternatively, we can try to minimize

$$\sum_{(i,j)\in\Omega} |d_{ij} - r_{ij}^2| - \lambda \sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij}^2, \tag{4.3.5}$$

Adopting the matrix inner product definition defined in Section (4.3.1)

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=1}^{N} r_{ij}^2 &= \sum_{i=1}^{N}\sum_{j=1}^{N} K(i,i)+K(j,j)-2K(i,j)\\ &= 2NI\cdot K-2\sum_{i=1}^{n}\sum_{j=1}^{n} K(i,j)\\ &= 2NI\cdot K-2E\cdot K=2(NI-E)\cdot K,\end{aligned}$$

where I is the N-dimensional identity matrix and E is the N by N matrix with all elements being 1. Plugging $r_{ij}^2 = K(i,i)+K(j,j)-2K(i,j)$ into (4.3.4) or (4.3.5), the problem becomes: find K positive semidefinite to minimize

$$\sum_{(i,j)\in\Omega} (d_{ij}-K(i,i)-K(j,j)+2K(i,j))^2 - 2\lambda(NI-E)\cdot K, \tag{4.3.6}$$

or

$$\sum_{(i,j)\in\Omega} |d_{ij}-K(i,i)-K(j,j)+2K(i,j)| - 2\lambda(NI-E)\cdot K. \tag{4.3.7}$$

We can also add weights w_{ij} into the first summation. There is an additional constraint needed to guarantee that the points are centered at 0_p. It is easy to show that if K is positive semidefinite, then $\bar{y}=0_p$ is equivalent to $Ke=0_p$, where e is the p by one vector whose elements are all ones, and this constraint can be added to the above optimization problems. This constrained minimization problem can be recast as a convex cone programming problem and there are efficient algorithms developed in the convex optimization community for solving this type of problems. Notice that under the $Ke=0_p$ constraint the objective function can be further simplified a little since if K is positive semidefinite, then $Ke=0_p$ is equivalent to $E\cdot K=0$. Once the matrix K is obtained, the dimension of the subspace of the repositioned points is $p = rank(K)$. Alternatively, the constraint $Ke = 0_p$ may be omitted and K centered later. This will be discussed further below. We can use the spectral decomposition, i.e., $K = \Gamma' D \Gamma$ where Γ is p by N consisting of p rows of eigenvectors of K, and D is the p by p diagonal matrix of non-zero eigenvalues $\{\lambda_\nu\}$, to get the principal coordinates $Y = D^{1/2}\Gamma$ with columns of Y to be y_i's.

One thing that we need to be careful about is that the neighbor set Ω should not be too small. Otherwise the objective function may diverge to $-\infty$. The necessary and sufficient condition to avoid this situation is that the edges in Ω construct a connected graph for all data points.

Now, it is easy to see that (4.3.7) differs from the formulation (4.3.2) only in the choice of kernel regularization function $J(K)$. In (4.3.7), $J(K) = trace(K) = I \cdot K$, while in (4.3.2) $J(K) = -2(NI - E) \cdot K$. It is worth pointing out that they are both linear thus convex in K.

4.3.3 Regularized Kernel Embedding Formulation for l_1 Loss

To convert the problem of equation (4.3.7) into a convex cone programming problem, without loss of generality, we let Ω contain m distinct (i, j) pairs, which we index with $r = 1, 2, \ldots, m$. Let $N \times N$ matrices I and E be defined as before. Define $e_{m,r}$ to be vector of length $2m$ consisting of all zeros except for the rth element being 1 and $(m + r)$th element being -1. If we denote the rth element of Ω as $\big(i(r), j(r)\big)$, and with some abuse of the notation let $i = i(r)$, $j = j(r)$ and $w \in P_{2m}$ with $w(r) = w(r+m) = w_{i(r),j(r)}, \;\; r = 1, \ldots, m$, we can formulate the problem of equation (4.3.2) as follows:

$$\begin{aligned} &\min\nolimits_{K \succeq 0, u \geq 0} \; w \cdot u - 2\lambda(NI - E) \cdot K \\ &\text{s.t.} d_{ij} - B_{ij} \cdot K + e_{m,r} \cdot u = 0, \;\; \forall_r, \\ &K \in S_N, \;\; u \in P_{2m}. \end{aligned} \tag{4.3.8}$$

The solution to (4.3.8) might not be centered. To obtain sensible principal coordinates of the corresponding configuration using the spectral decomposition, we can center the solution kernel following a simple procedure. Define a to be the column with ith entry the average of the ith column of K, c to be the scalar as the mean of all elements in K and e to be vector of suitable dimension consisting of all 1's. A kernel K can be centered simply by: $K_{centered} = K - ae^T - ea^T + cE$. An alternative way to handle this centering step, as discussed before, is to directly impose the centering condition $E \cdot K = 0$, which is actually a linear constraint and can be directly incorporated into the convex cone formulation (4.3.8). Then, we can also simplify the kernel regularization function from $-2(NI - E) \cdot K$ to a reduced form $-2(NI) \cdot K$. However in our experiment with the examples here, the optimization problem without the $E \cdot K = 0$ constraint converges faster.

The Regularized Kernel Embedding formulation with square loss can also be easily obtained after simple modification of the square formulation in the Appendix A.1.1.

4.3.4 ‘Newbie’ Formulation

A very useful ‘newbie’ algorithm was developed in Chapter 3 to find the coordinates for new data points (newbies) within the previously constructed configuration. The corresponding newbie problem is essentially the minimization of the sum of losses involving the newbie only. We adopt the same idea here for the manifold-unfolding problem except we restrict the summation even further to a reasonable neighborhood of the newbie. The neighborhood construction problem will be discussed in general in the following section. However the algorithm remains the same.

4.3.5 Choosing Neighbors

Choosing neighbors for each sampled data point is a very important/tricky step for almost all methods including ours that are in the spirit of ‘thinking globally while fitting locally’. However, it is not discussed in detail in previous papers. A simple way, which is adopted by most algorithms, is to choose k nearest neighbors for all data points. Then the neighbor-choosing problem degenerates into a neighborhood-size-choosing problem. In exploratory studies, where the truth is not known, the only thing one can do might be to start from a ‘suitable’ neighborhood size based some prior knowledge or intuition, and then vary the size to see how the results change. As we discussed previously, the neighborhood size has to be big enough so that the neighbor edges and all points consist of a connected graph. On the other hand, if the neighborhood size is too big, we are respecting more than just the local structure, and also, the computation cost usually goes up quickly. A good choice of k and the sensitivity of the results to k depend on the density and distribution of the sampled points on the manifold. In previous papers, with dense enough samples for the examples, the authors simply choose moderate neighborhood sizes. In this paper, we also adopt this approach in the simulated examples.

Nonetheless, in some cases, fixing a neighborhood-size might not be a good approach to setup a connected graph especially when the sampling is very uneven across the underlying manifold or the manifold has very different curvature from place to place. For the formulation proposed in this work, we have another, possibly more stable way to tackle this issue. We can impose a compactly supported kernel around each data point to generate weights for all other points. Only those points who get non-zero weights become candidates to be neighbors for a particular data point, and their weights will be used to multiply the corresponding loss terms in (4.3.6) and (4.3.7). A threshold number can also to set so that every point only keeps no more than that number of closest neighbors from all candidates. A suitable bandwidth of the kernel can be selected based on the average closest-neighbor distance. The intuition of this approach is to give higher confidence to the distances between closer neighbors.

4.3.6 Parameter λ

The tuning parameter λ controls a balance between the twin goals we want to achieve – as λ increases, the average squared distance between points far apart is allowed to increase, thus enhancing "flattening", while as λ decreases, the solution is driven towards more closely respecting the observed local structure.

For an exploratory study, where the truth is not known, a sequence of λs within an appropriate range (usually in *log* scale) will give different results. Then prior knowledge may help to choose a good λ. For example, if it is known that there is not much noise within the data and a low dimensional embedding is preferred, one can gradually increase λ to get rid of insignificant dimensions until the sum of the losses exceed some limit.

However, if this manifold-learning task is just a part of bigger problem, e.g., clustering or classification, where we know the truth for training data, it will be natural to tune λ simultaneously with other possible tuning parameters, using standard tuning techniques like cross validation.

4.4 Unfolding Simulated Examples

4.4.1 Unfolding the Swiss Roll with a Window Punched Out

The first simulated example is a Swiss roll manifold with a rectangular window punched out close to the center. This example was used in Donoho and Grimes (2003) to show how a non-convex feature (the punched-out window) can cause some previous methods like ISOMAP (Tenenbaum et al. 2000) and LLE (Roweis & Saul 2000) to fail.

The following results are obtained on a random sample of 770 points, each point with 6 neighbors. There is no noise in this example. Figure 4.4.1 gives the scatter plot of original data points sampled on the manifold (except within the punched-out window). Figure 4.4.2 is the true parameterization, and its "rolled-up" version gives Figure 4.4.1. Figure 4.4.3 is the solution to our formulation with (4.3.8), with the tuning parameter $\lambda = 7e - 7$. In Figure 4.4.4, the eigensequence of the corresponding solution kernel is plotted in descending order on a log scale. We can clearly see the fact that the first two eigenvalues stand out significantly in magnitude compared with the rest of the eigenvalues, indicating a 2D embedding. (The last eigenvalue in Figure 4.4.4 is the computer version of the zero eigenvalue that goes with the constant function.) The principal coordinates in Figure 4.4.3 are constructed using these two significant eigenvalues and corresponding eigenvectors.

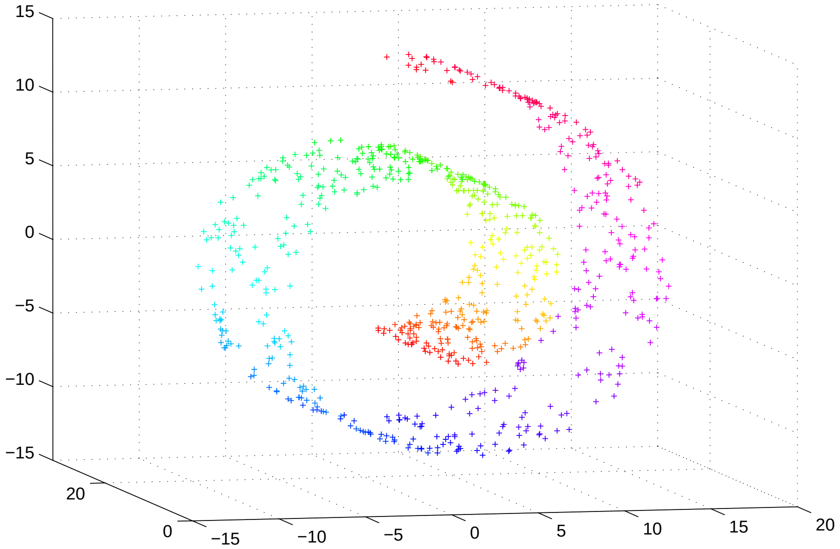

Figure 4.4.1: Swiss Roll: Scatter plot of original data points.

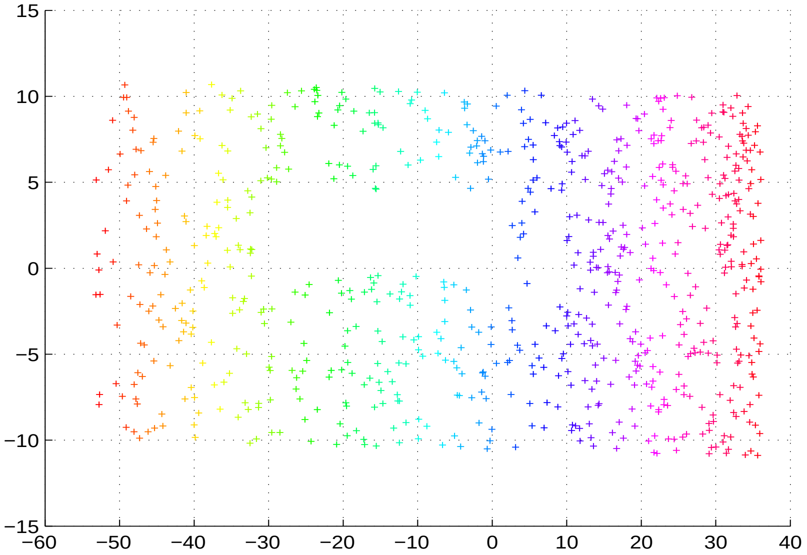

Figure 4.4.2: Swiss Roll: True parameterization. Unrolled version of Figure 4.4.1.

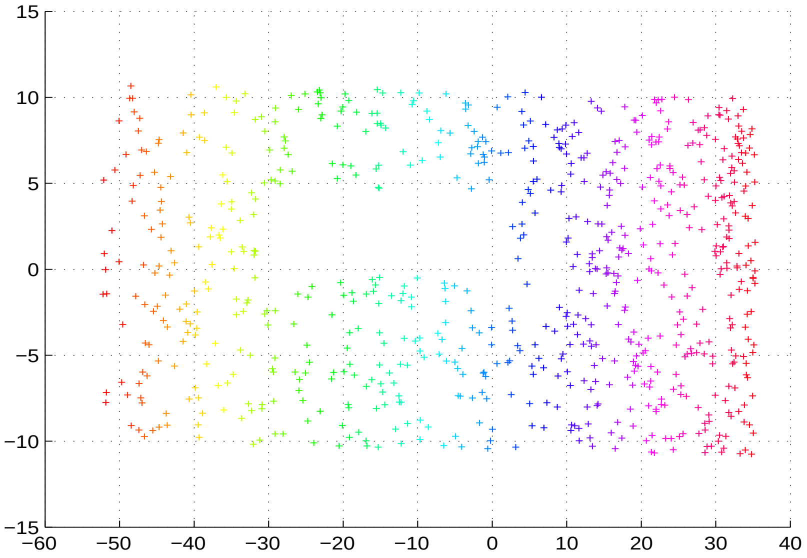

Figure 4.4.3: Swiss Roll Unrolled: Regularized Kernel Embedding using (4.3.8), $\lambda = 7e - 7$, first two principal coordinates.

4.4.2 Unfolding the Noisy Wisconsin Roll

This example is specially designed to show the robustness of our method, especially compared with the method proposed recently in Weinberger et al. (2004), which has a basic idea very similar to ours. We consider two types of noise, which are imposed on the pairwise distances between neighbors after the all neighbors are selected. In this example, the data points are sampled on a 'Wisconsin roll', which is a Swiss roll except there is a window in the shape of letter 'W' punched out (thus no points can be sampled with in it) which can be seen clearly if the roll is flatten out.

To impose the first type of noise, twenty percent of the selected pairwise

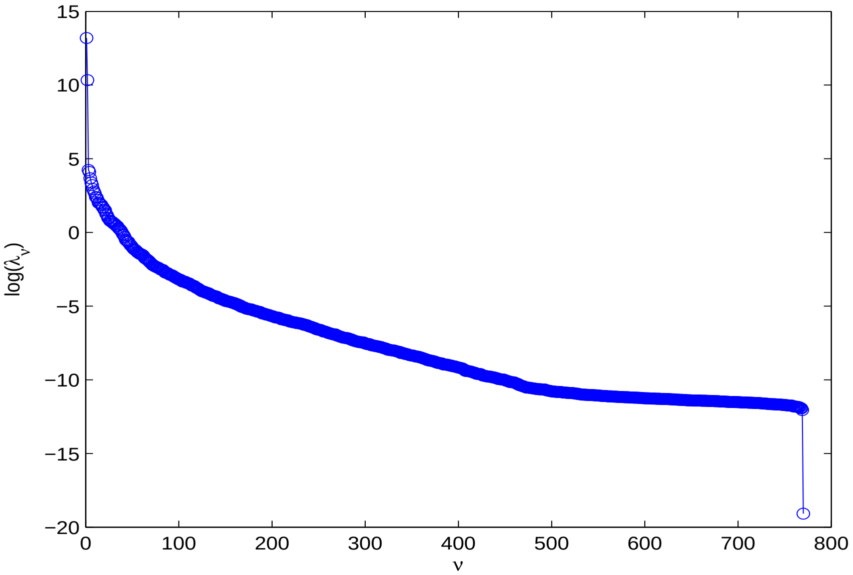

Figure 4.4.4: Swiss Roll: Eigensequence of the solution kernel, $\lambda = 7e-7$. Note log scale.

distances are multiplied by a uniform random number over the interval from 0.85 to 1.15. The second type of noise is introduced to all chosen $d_{ij}s$ (between chosen neighbors) by binning them into 15 equal sized bins over the interval from the minimum to the maximum among these $d_{ij}s$. The value of each d_{ij} is then replaced by the center of the bin that it belongs to. It is an analog of the scenario where only ranks are provided as the distance/dissimilarity measure.

A random sample of 861 points was used for this example with the neighborhood size set to be $k = 6$. In both noisy situations, our method successfully (with λ in a proper range) converges to a global optimum with only two significant dimensions. See eigensequence plots Figure 4.4.7 and Figure 4.4.9. The

Table 4.4.1: Procrustes Measure between Result and Truth

	1st type of noise case	2nd type of noise case
γ_p	0.0055	0.0030
γ_d	0.0154	0.0112

Procrustes measure shows our solution is very close to the truth (See Table 4.4.2), although the recovered embeddings shown in Figure 4.4.6 and Figure 4.4.8 are distorted a little bit from the truth (see Figure 4.4.5) due to the imposed noise.

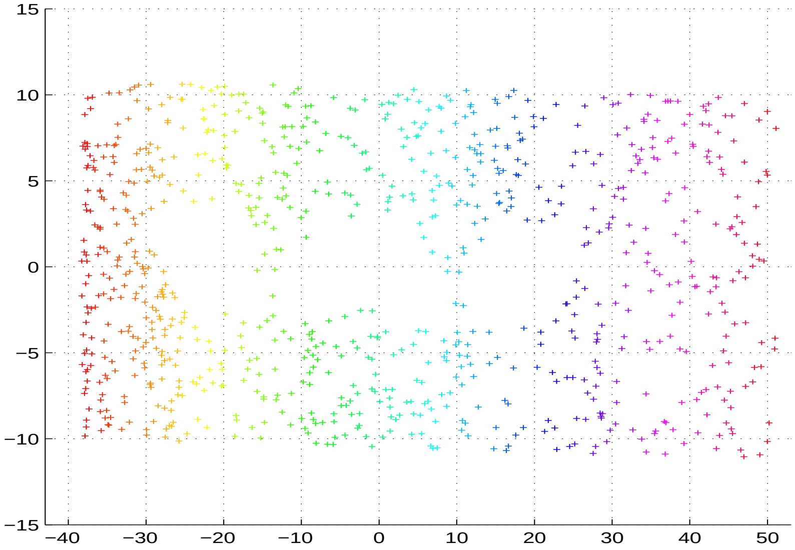

Figure 4.4.5: Wisconsin Roll: True parameterization. Observations come from rolled up version after adding noise.

On the contrary, the algorithm in Weinberger et al. (2004) fails to converge

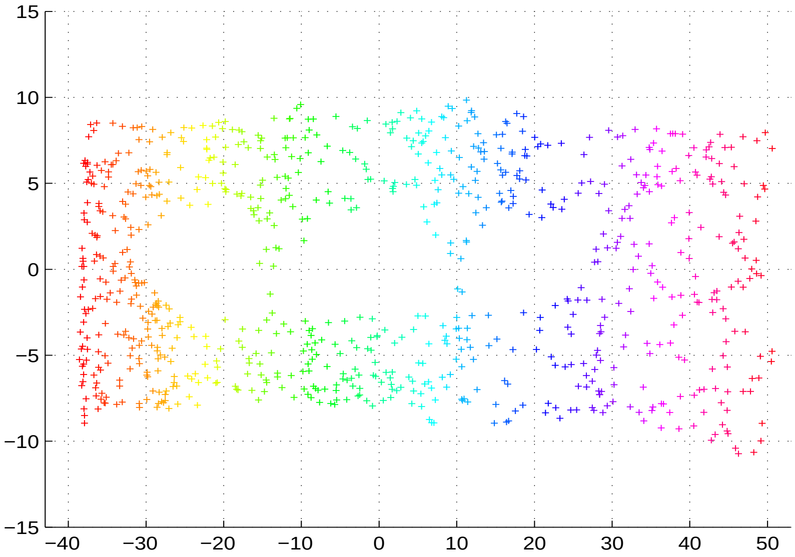

Figure 4.4.6: Wisconsin Roll with first type of noise, unrolled. Regularized Kernel Embedding using (4.3.8), $\lambda = 0.002$, first two principal coordinates.

because it tries to solve an infeasible primal problem for which the dual is unbounded. For the solvers we used, DSDP5 reported " DSDP: Dual Unbounded, Primal Infeasible" and SDPT3 reported "Stop: primal problem is suspected of being infeasible". These results are expected, because when a certain level of noise is directly imposed on the distance information, it is very likely that no Euclidean metric can fit the noisy distance data (for instance if the triangle inequality is violated somewhere). Then problem set-up in Weinberger et al. (2004) is infeasible in the sense that no solution can satisfy all the constraints simultaneously.

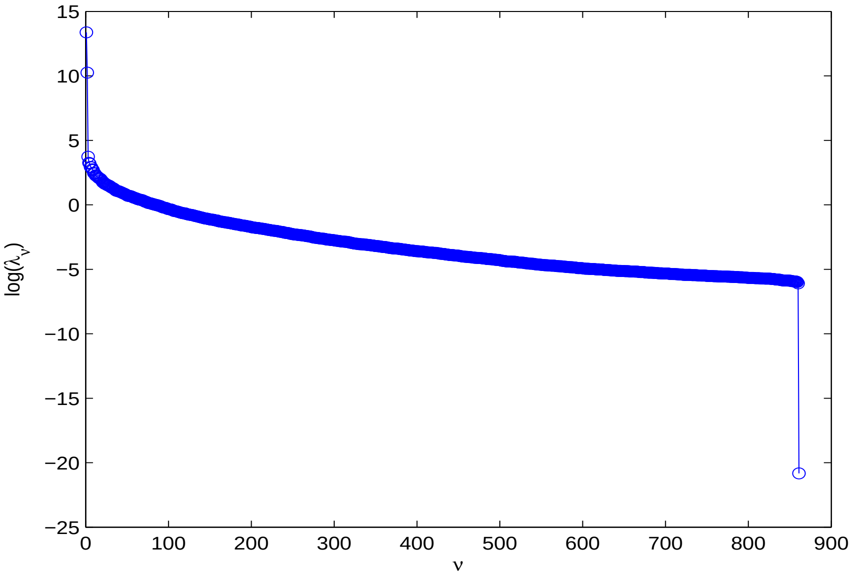

Figure 4.4.7: Wisconsin Roll: Eigensequence of the solution kernel, first type of noise, $\lambda = 0.002$.

4.4.3 Unfolding a Broken Stick

In this section we describe a toy example for the purpose of highlighting the difference between our method and the method proposed in Weinberger et al. (2004). The primary difference between the two methods is that for the method in Weinberger et al. (2004) local distances are enforced rigidly while here we relax that requirement. We want to show that this relaxation can be very important for manifold-unfolding problems even in the cases without noise.

The data points are randomly sampled on two branches of a 'broken stick' (see Figure 4.5.1). One branch is from the origin to the point $(1, 1)$ and the other is from $(1, 1)$ to $(2, 0)$. We force the sample to include the point $(1, 1)$.

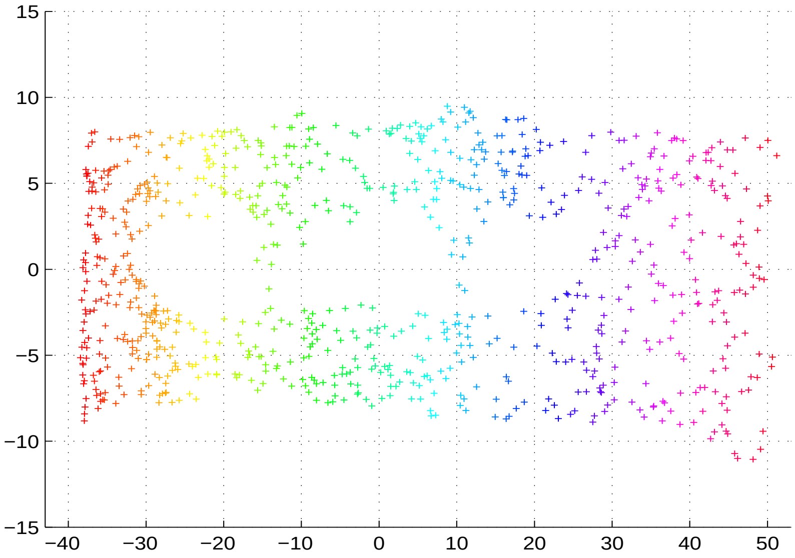

Figure 4.4.8: Wisconsin Roll with second type of noise, unrolled. Regularized Kernel Embedding using (4.3.8), $\lambda = 0.0025$, first two principal coordinates.

The manifold unfolding goal here is to flatten out the stick. If any of the pairs for which distance is selected to fit, has one member from the left branch and the other member from the right branch (for example, see the black line in Figure 4.5.1), then the method in Weinberger et al. (2004) will not be able to flatten the stick. For our method, a small λ will not flatten the stick either, but a sufficiently large λ will. The result from employing the method in Weinberger et al. (2004) with $k = 5$ is almost visually indistinguishable from the plot in Figure 4.5.1. With $k = 5$ and λ too small ($\lambda = 1e-5$) , our method also fails to flatten the stick but recovers the original broken stick. Two outstanding eigenvalues are obtained as can be seen in the upper left corner of Figure 4.5.2. However, with

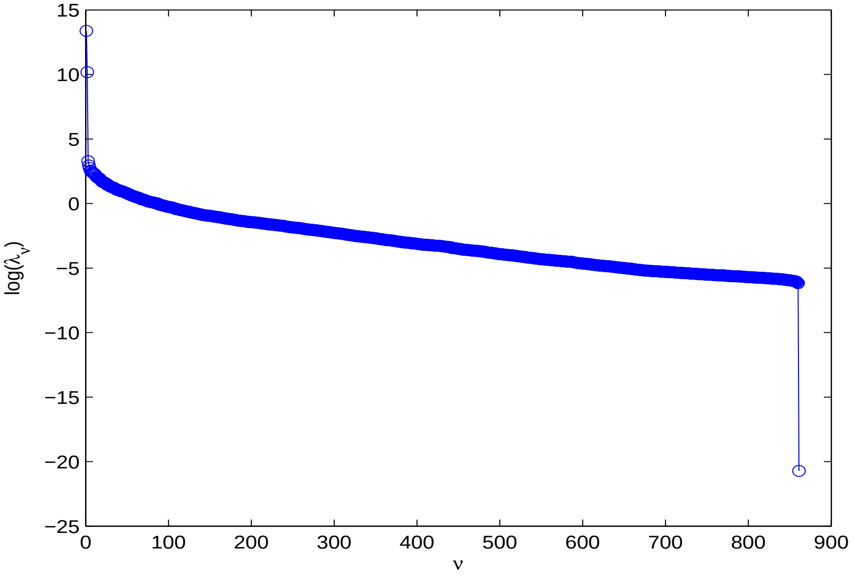

Figure 4.4.9: Wisconsin Roll: Eigensequence of the solution kernel, second type of noise, $\lambda = 0.0025$.

λ sufficiently large ($\lambda = 0.3$) we see only one outstanding eigenvalue, and so we obtain the one dimensional flattened stick on the lower right corner of Figure 4.5.2. As expected, within our regularized kernel embedding framework, the smoothness/dimensionality is controlled by the smoothing/tuning parameter λ.

4.5 Discussions

In this chapter, we developed a robust manifold learning method as a variation of the RKE framework proposed in Chapter 3. It is worth mentioning that, if we choose to impose the centering constraint $E \cdot K = 0$ (although we can do without this) in problems (4.3.6) and (4.3.7), the kernel regularization function

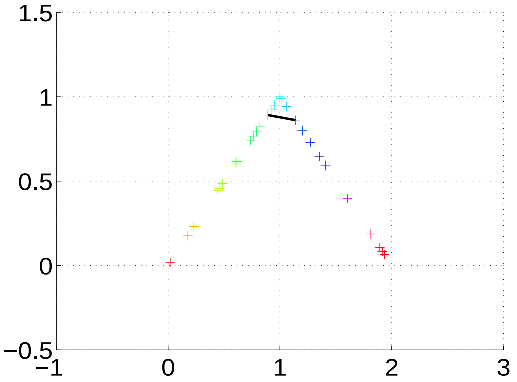

Figure 4.5.1: Broken Stick: Original data. Heavy black line joins a pair of points with the members from different sides of the break.

for manifold unfolding becomes $J(K) = -2(NI - E) \cdot K = -2NI \cdot K = -2N\text{trace}(K)$. Interestingly, in Chapter 3, the kernel regularization function we use to promote dimension reduction is trace instead of the negative trace (with a constant multiplier) here. So, different signs in front of trace actually both promote dimension reduction but in different scenarios.

More interesting problems come up when there are multiple source of information that are believed to share the same underlying low-dimensional structure. Our method can be naturally extended to that case. Also, it is often unrealistic to assume the given distance information is actually Euclidean. Then, a non-metric variation of our method, i.e., only rank information among all distances will be used, can be very useful. Last but not the least, we will explore the weighting scheme as we discussed in Section (4.3.5) to select neighbors in order to achieve higher stability and robustness.

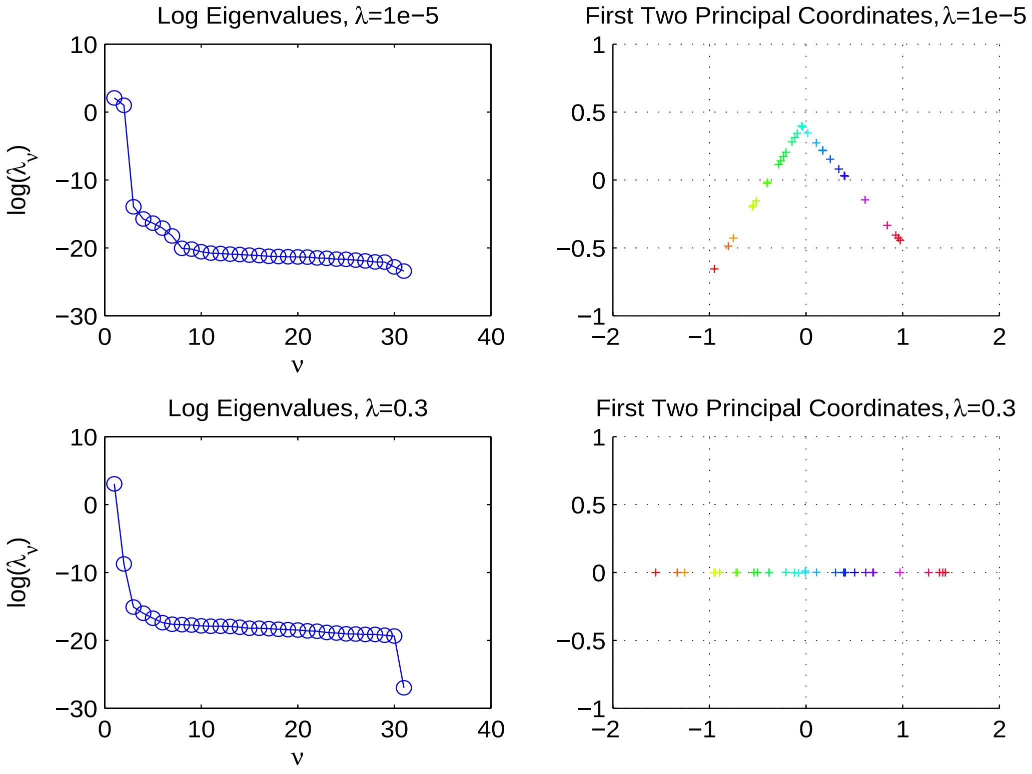

Figure 4.5.2: Broken Stick: Effect of λ on the Regularized Kernel Embedding using (4.3.8). Small λ does not flatten the stick, but a larger λ does.

Chapter 5

Concluding Remarks

Regularization methodology has proved to be a powerful tool in a variety of statistical practices, traditionally in preventing overfitting of nonparametric models and more recently in model/variable selection and dimention reduction. Especially for massive data sets which are very typical challenges to statisticians nowadays, where oftentimes, the statistical inference problems are buried inside data mining tasks, regularization methods are not only elegantly concise but also robust, given that the computation burdern can be well handled. On the other hand, with the fast advance of computer technology (hardwares) and computation algorithms (softwares), an impossible computation problem yesterday might be a very easy one tomorrow.

In this thesis, we investigate the methods of regularized estimation in three different settings. The first one is a modification/generalization of traditional nonparametric penalized likelihood method. The second and the third are two special applications of the RKE framework proposed in this thesis to dissimilarity data. In penalized logistic regression, our work shows that the regularization method is very flexiable in accomodating new features of the data and providing robust estimation. Our success in extending the idea of regularization to dissimilarity data in kernel learning also reveals that the art of good estimation/prediction lies in the good balance of fitting a model to the observations and controlling the model complexity.

Our novel work in kernel regularization can be extended in serveral different directions. One interesting future direction is to modify the RKE formulation in Chapter 3 and/or Chapter 4 to a non-metric version, meaning that only the ranks instead of the exact values of observed dissimilarity measures will be used to estimate the target kernel. Non-metric RKE will be more flexible to fit the data especially when the observed dissimilarity measure is very non-Euclidean. Another interesting but more distinct future direction would be to extend current RKE framework to solve feature-space kernel learning problems. What we show in this thesis is all within 'object space', that is the kernel we try to estimate is characterizing only the relations between objects. One can think about the situation where each object is associated with a representing feature vector, and the goal becomes to estimate a kernel in the feature space that connects the features to the relations within objects.

Appendix A

Derivation and Proof

A.1 Formulations with Square Loss Functions

We describe here the formulations of the regularized kernel estimation and newbie problems when a square loss function is used in place of an l_1 loss function. We acknowledge the help of Kim-Chuan Toh who gave us a suggestion which substantially improved our original formulation of the square loss algorithm, in terms of computational complexity.

A.1.1 RKE Formulation

The sum-of-squares variant of Formulation (3.4.1) is

$$\min_{K \succeq 0} \sum_{(i,j) \in \Omega} w_{ij}(d_{ij} - B_{ij} \cdot K)^2 + \lambda \operatorname{trace}(K). \tag{A.1.1}$$

Let $\delta_{ij} = \sqrt{w_{ij}}(d_{ij} - B_{ij} \cdot K)$. Then (A.1.1) can be rewritten as:

$$\begin{array}{c} \min_{K \succeq 0} \sum_{(i,j) \in \Omega} \delta_{ij}^2 + \lambda \operatorname{trace}(K) \\ \text{s.t. } B_{ij} \cdot K + \frac{1}{\sqrt{w_{ij}}} \delta_{ij} = d_{ij}, \;\; (i,j) \in \Omega. \end{array}$$

We use $r = 1, 2, \ldots, m$ to index the elements of Ω, as in Section 3.2. To convert the problem into a convex conic formulation, we introduce a variable $y = [\tilde{a}, \delta_1, \delta_2, \ldots, \delta_m]$ to be in a second-order cone and we define e_r to be the constant vector of length $m + 1$ consisting of all zeros except for the $(r + 1)$th element being 1. We further let $X = \begin{bmatrix} 1 & \tilde{a} \\ \tilde{a} & a \end{bmatrix} \in S_2$ and 0_l be the zero vector of length l. Finally, we obtain the following formulation for the problem above:

$$\min \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot X + \lambda I_N \cdot K \tag{A.1.2}$$

$$\begin{aligned} \text{s.t. } B_{ij} \cdot K + \tfrac{1}{\sqrt{w(r)}} e_r \cdot y &= d_{ij}, \ \forall_r \\ \begin{bmatrix} 1 \\ 0_m \end{bmatrix} \cdot y - \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \cdot X &= 0, \\ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot X &= 1, \\ K \succeq 0, \quad X \succeq 0, \quad y \in Q_{m+1}, \end{aligned}$$

where $K \in S_N$ and $X \in S_2$ and $(i, j) = (i(r), j(r))$ as in Section 3.2. This conic formulation, unlike Formulation (3.4.3), involves both semidefinite cone and second-order cone. However, because of the number of constraints differs only by a constant 2 (i.e., independent on the size of the problem) from Formulation (3.4.3), the space complexity of Formulation (A.1.2) is similar to Formulation (3.4.3). In simulation experiments, we obtained similar results using the l_1 and square loss formulations as λ varies.

A.1.2 Newbie Formulation

The newbie problem with least-squares objective is

$$\begin{aligned} &\min \textstyle\sum_{i\in\Psi} w_i(d_{i,N+1} - B_{i,N+1}\cdot K_{N+1})^2 \\ &\text{s.t. } \; b \in \text{Range}(K_N), \;\; c - b^T K_N^{+} b \geq 0, \end{aligned} \tag{A.1.3}$$

where $c \in \mathcal{R}$ and $b \in \mathcal{R}^p$. After performing the same transformations as in Section 3.3 and defining variables $\delta_{i,N+1}$, y and X similarly as above, we obtain the following formulation of this problem as a convex conic program:

$$\begin{array}{rcl} \min \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot X & & \\ \text{s.t.} \;\; \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot Z & = & 1, \\ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot X & = & 1, \\ \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \cdot Z - \begin{bmatrix} 1 \\ 0_p \end{bmatrix} \cdot x & = & 0, \\ \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} \cdot X - \begin{bmatrix} 1 \\ 0_t \end{bmatrix} \cdot y & = & 0, \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot Z + K_N(i,i) - \Sigma_{i\cdot} \cdot x - \frac{1}{\sqrt{w(r)}} e_r \cdot y & = & d_{i,N+1}, \;\; \forall_r, \\ Z \succeq 0, \;\; X \succeq 0, & & \end{array}$$

where $Z \in S_2, \;\; X \in S_2, \;\; x \in Q_{p+1}, \;\; y \in Q_{t+1}$.

A.2 Proof of Theorem 3.4.1

Proof. Let $[s^T t^T]^T = u$ in (3.4.3), where s and t are non-negative vectors of length m. As in section 3.4.2, we denote the rth element of Ω as $\big(i(r), j(r)\big)$, and with some abuse of the notation let $i = i(r)$, $j = j(r)$ and $w \in P_{2m}$ with $w(r) = w(r+m) = w_{i(r),j(r)}$, $\;r = 1, \ldots, m$, we can rewrite the constraints in (3.4.3) as:

$$\begin{gathered} d_{ij} - B_{ij} \cdot K = t(r) - s(r), \;\; \forall_r, \\ K \in S_N, \; s \in P_m, \; t \in P_m. \end{gathered}$$

Since s and t are non-negative vectors, we have:

$$\begin{aligned} t(r) + s(r) &= |t(r) + s(r)| \\ &\geq ||t(r)| - |s(r)|| \qquad \text{(A.2.1)} \\ &= |d_{ij} - B_{ij} \cdot K|, \quad \forall_r. \end{aligned}$$

Summing both sides of (A.2.1) over r gives the following inequality between the first item of the minimization objects in (3.4.1) and (3.4.3) respectively when all constraints are satisfied: $w \cdot u \geq \sum_{(i,j)\in\Omega} w_{ij} |d_{ij} - B_{ij} \cdot K|$. Therefore the global minimum of (3.4.3) is no smaller than that of (3.4.1). On the other hand, since t and s (i.e., u) are dummy variables that can vary freely (with the non-negative constraints), the equality in (A.2.1) is always achievable by letting $t(r) = d_{ij} - B_{ij} \cdot K$ and $s(r) = 0$ when $d_{ij} - B_{ij} \cdot K > 0$ or $s(r) = B_{ij} \cdot K - d_{ij}$ and $t(r) = 0$ when $d_{ij} - B_{ij} \cdot K \leq 0$, $r = 1, \ldots, m$. So, the global minimum of (3.4.3) is also no bigger than that of (3.4.1). Hence, optimization problems (3.4.1) and (3.4.3) have the same global minimum.

Moreover, since optimization problem (3.4.3) is a convex cone programming problem bounded from below (with $w \in P_{2m}$, zero is an obvious lower bound). Every local minimum of it is equal to the its unique global minimum and thus also the global minimum of (3.4.1). So computationally, it will be convenient to solve for any local (which is also global) minimum of optimization problem (3.4.3).

A.3 A sketch of the proof for the l_2 version of Theorem 3.4.1

Notice that, when all constraints of the optimization problem (A.1.1) are satisfied, $y = \begin{bmatrix} \tilde{a} \\ \delta \end{bmatrix} \in Q_{m+1}$ where δ is the m-dimensional vector $\{\delta_{i(r),j(r)}, r = 1, \ldots, m\}$ as defined before, and $X = \begin{bmatrix} 1 & \tilde{a} \\ \tilde{a} & a \end{bmatrix} \in S_2$.

Similarly as we prove Theorem 3.4.1, we first prove the global minimum of (A.1.1) is no smaller than that of (A.1.2) by observing that $a \geq \tilde{a}^2 \geq \sum_{r=1}^{m} \delta(r)^2$, where the first inequality is the direct result of positive semidefinite constraint on X and the second follows the second order cone constraint on y. We then prove the global minimum of (A.1.1) is no bigger than that of (A.1.2) by showing both inequalities are achievable, obviously, which completes our proof.

A.4 Proof of Theorem 3.4.2

Proof. First, we remark that the equivalence between condition (3.4.4) and the constraints of the optimization problem (3.4.5) is simply the generalized version of Schur complement Lemma (see Boyd & Vandenberghe (2004)).

Now, since the optimization target of Newbie problem is nothing but the first item of that of RKE formulation (3.4.1), we will not repeat the similar part as in the proof of Theorem A.2, but focus on the proof of the constraints part, which is very straightforward.

We first notice that, the condition $b \in \text{Range}(K_N)$ with the spectral decomposition of K_N being defined as $K_N = \Gamma\Lambda\Gamma^T$, is equivalent to the existence of a real vector $\tilde{b}$ such that $b = \Gamma\Lambda^{1/2}\tilde{b}$. We then use this equality to change the variable in the inequality: $c - b^T K_N^+ b \geq 0$, which becomes $c - \tilde{b}^T\tilde{b} \geq 0$. But that is guaranteed because $c \geq \tilde{(}c)^2 \geq \tilde{b}^T\tilde{b}$, where the first inequality is the result of positive semidefinite constraint on Z as $Z \overset{\text{def}}{=} \begin{bmatrix} 1 & \tilde{c} \\ \tilde{c} & c \end{bmatrix} \in S_2$ and the second inequality follows the second order cone constraint on x as $x \overset{\text{def}}{=} [\tilde{c} \;\; \tilde{b}^T]^T \in Q_{p+1}$ in optimization problem (3.4.7). Thus we have proved that the constraints of (3.4.5) are satisfied in (3.4.7). The rest we need to prove that optimization problems (3.4.5) and (3.4.7) have the same global minimum follows similarly as the proof of Theorem A.2. And again, since optimization problem (3.4.7) is a convex cone programming problem bounded from below (with $w \in P_{2t}$). Every local minimum of it is equal to the its unique global minimum and thus also the global minimum of (3.4.5).

We will skip the proof for the l_2 version of Theorem 3.4.2, for which all we need are the already in this and previous section.

A.5 Newbie Lemma and Proof

Lemma A.5.1. *Let b and c be the global solution of the optimization problem (3.4.5). Let $\tilde{K}_{N+1}$ be the extended kernel defined as in (3.4.4). Suppose* $\text{Rank}(K_N) = p < N$. *Then,* $\text{Rank}(\tilde{K}_{N+1})$ *can only be equal to p or $p+1$. It equals p if and only if $c - b^T K_N^+ b = 0$. It equals $p+1$ if and only if $c - b^T K_N^+ b > 0$.*

Proof. Since $c - b^T K_N^+ b \geq 0$ is a necessary condition for $\tilde{K}_{N+1}$ to be nonnegative semidefinite (see the remark at the beginning of the proof of Theorem A.4), we only need to prove the 'if' direction .

We first notice that since $b \in \text{Range}(K_N)$, $\text{Rank}([K_N \quad b]) = p$. Also, by the definition of pseudo-inverse (generalized inverse), $b^T - b^T K_N^+ K_N = 0_N^T$. (Again, 0_p is zero vector of dimension p.) Thus, if $c - b^T K_N^+ b = 0$, we have $[b^T \quad c] - b^T K_N^+ [K_N \quad b] = 0_{N+1}^T$, which implies $[b^T \quad c] \in \text{Range}([K_N \quad b])$. Hence, $\text{Rank}(\tilde{K}_{N+1}) = p$. If $c - b^T K_N^+ b = \delta > 0$, since we have shown $[b^T \quad b^T K_N^+ b] \in \text{Range}([K_N \quad b])$, it all boils down to whether $[0_N^T \quad \delta] \in \text{Range}([K_N \quad b])$. Assume that is true. Then, there exist a N-dimensional vector g such that $g^T [K_N \quad b] = [0_N^T \quad \delta]$. Thus $g^T K_N = 0_N^T$, which means g belongs to the null space of K_N. Remember that $b \in \text{Range}(K_N)$. So we should have $g^T b = 0$

which is a contradiction to part of the assumption that $g^T b = \delta > 0$. Therefore the assumption is wrong and $[b^T \quad c] \notin \text{Range}([K_N \quad b])$, which implies $\text{Rank}(\tilde{K}_{N+1}) = \text{Rank}([K_N \;\; b]) + 1 = p + 1$. This completes our proof.

A.6 Projection Lemma and Proof

An interesting feature of our kernel approach to the Newbie problem is that, when the solution actually introduce an extra dimension to form the extended kernel (i.e.$c - b^T K_N^+ b > 0$), if we simply project the newbie back into the space expanded by K_N, almost always that projection is not the optimal solution for the newbie problem restricted in the space expanded by K_N. The following lemma states this observation (in the case of l_2 loss) more precisely.

Lemma A.6.1. *Let $\{x_{i,j}\}$ be the known coordinates of N points, where $i = 1, \ldots, N$ indexes the points and $j = 1, \ldots, p$ indexes the dimensions. Let $*$ be the dimension index of a newbie with $\{d_{*,i},\ i = 1, \ldots, N\}$ denoting the observed dissimilarity measures (corresponding to squared Euclidean distances) between the newbie and the existing N 'training' points. Denote the unknown coordinate of the newbie as $x_{*,j}$, and the fitted dissimilarity $\hat{d}_{*,i} = \sum_{j=1}^{P}(x_{*,j} - x_{i,j})^2$, where $P = p$ if we restrict the newbie to be in the original p-dimensional space, or $P = p + 1$ if we allow the newbie to introduce a new dimension while padding the coordinates of the existing N points each will a zero. We redefine the newbie problem as: Finding $\{x_{*,j},\ j = 1, \ldots, P\}$ that minimizes the l_2 loss $\sum_{i=1}^{N}(d_{*,i} -$*

$\hat{d}_{*,i})^2$.

Let $\{\hat{x}_{*,j},\ j = 1, \ldots, p+1\}$ *be the solution to the newbie problem with* $P = p+1$. *Suppose* $\hat{x}_{*,p+1} \neq 0$. *Then* $\sum_{i=1}^{N}(\hat{x}_{*,j} - x_{i,j}) = 0$, *for* $j = 1, \ldots, p$, *i.e., the condition that* $\hat{x}_{*,j},\ j = 1, \ldots, p$ *is the sample mean of the existing* N *points is a necessary for* $\{\hat{x}_{*,j},\ j = 1, \ldots, p\}$ *to be the solution to the newbie problem with* $P = p$. *Obviously, this condition is very unlikely in real situations.*

Proof. Suppose $\{\hat{x}_{*,j},\ j = 1, \ldots, p+1\}$ is the solution to the newbie problem with $P = p+1$ and $\hat{x}_{*,p+1} = c \neq 0$. Let $\hat{d}_{*,i} = \sum_{j=1}^{p+1}(x_{*,j} - x_{i,j})^2$ with $x_{i,p+1} = 0,\ i = 1, \ldots, N$ (padded zeros). Then first order condition for this to be optimal solution states:

$$\frac{\partial}{\partial x_{*,j}} \sum_{i=1}^{N}(d_{*,i} - \hat{d}_{*,i})^2 = \sum_{i=1}^{N}(d_{*,i} - \hat{d}_{*,i})(x_{*,j} - x_{i,j}) = 0, \quad j = 1, \ldots, p+1. \tag{A.6.1}$$

If the projection $\{\hat{x}_{*,j},\ j = 1, \ldots, p\}$ is the solution to the newbie problem with $P = p$, we write down the first order condition similarly as:

$$\sum_{i=1}^{N}(d_{*,i} - (\hat{d}_{*,i} - c^2))(x_{*,j} - x_{i,j}) = 0, \quad j = 1, \ldots, p. \tag{A.6.2}$$

Bring (A.6.1) to (A.6.2) for $j = 1, \ldots, p$, we have:

$$\begin{aligned} & \sum_{i=1}^{N}(d_{*,i} - (\hat{d}_{*,i} - c^2))(x_{*,j} - x_{i,j}) \\ = \;& \sum_{i=1}^{N}(d_{*,i} - \hat{d}_{*,i})(x_{*,j} - x_{i,j}) + c^2 \sum_{i=1}^{N}(x_{*,j} - x_{i,j}) \\ = \;& c^2 \sum_{i=1}^{N}(x_{*,j} - x_{i,j}) \\ = \;& 0, \quad j = 1, \ldots, p. \end{aligned}$$

The second equality is due to (A.6.1). Now, since $c^2 > 0$, we must have $\sum_{i=1}^{N}(\hat{x}_{*,j} - x_{i,j}) = 0$, for $j = 1, \ldots, p$, which completes the proof.

Appendix B

Some Computer Codes

B.1 RKE for Multidimensional Scaling with l_1 loss

```
function [D,K]=lrkei(lambda,d,H,delta,method,sw,sw1,ni)


% absolute loss penalty Regularized Kernel Estimate
% Written by Fan LU, Department of Statistics, UW-Madison


% lambda : the scalar coefficient in front of the trace penalty on the kernel
% d : given dissimilarity matrix n-by-n, missing entries are zeros
%      we usually assume (though not necessary for the algorithm):
%      1) d(i,j)>=0 for all i~=j; 2) d(i,i)=0; 3) d(i,j)=d(j,i)
% H: H is both indicator matrix and weight matrix of the same size as d
%     the sum of loss part in the RKE formulation includes only the entries
%     of d for which, the corresponding entries in H are nonzero. And the
%     nonzero entries in H will also serve as the weights for inividual
%     loss within the sum of losses. So, for the missing entries in d, the
%     corresponding entries in H must be zero.  Also because of 2) 3)
%     above, H is usually a (defaul: down) triangular matrix excluding
%     the main diagnal. Uniform weighting scheme will correspond to
%     assign all observed dissimilarity entry a weight of 1
% sw: 1: using given starting point (X0, Z0 stored in start.m), 0: otherwise,
%      default: 0
% sw1: 1: using hidden conditon to guarantee the self-centering conditions,
%      0: not, default: 0
% method: sdpt3 or dsdp5, default: sdpt3
% ni: # of iterative log(det) minimization, default 1 (no iteration, as in
%       PNAS paper), see Boyd reference
% delta: a small positive scalar to add to main diagonal elements a non-full
%        rank nonegative-definite matrix for inversion. Default: machine
%        precision

%% Checking the inputs
[n,m]=size(d);
if n˜=m
    error(’Need squre distance matrix’);
end
disp([’# of points= ’,num2str(n)])
disp(['lambda= ',num2str(lambda)])


%% Setting default arguments

if nargin <8

    ni=1; %% Default: no log(determinant) miniziation steps

    if nargin <7

        sw1=0; %% Default: no hidden condition necessary of MDS type of RKE

        if nargin<6

            sw=0; %% Default: using solver default starting point

            if nargin<5

                method='sdpt3'; %% Default method is 'SDPT3'

                if nargin<4

                    delta=eps; %% Default: machine precision

                    if nargin<3

                        [x,y]=meshgrid(1:n); %% Default: fit all down

                        H=zeros(n);          %% triangle d_ij elements

                        H(d>0)=1;

                        H(y<=x)=0;

                    end

                end

            end

        end
    end
end

if any(size(H)~=[n,n])
    error(’Weight Matrix must be the same dim of distance matrix’);
end

[iind,jind,Hij]=find(H); %%find nonzero values, zero d_ij won’t be fitted
nHij=length(Hij);
disp([’# of distinct pairs= ’,num2str(nHij)])
nHij2=nHij*2;

%% Preparing convex conic problem 3.4.3 in thesis in SDPT3 solver format
%% Preparing optimization target coefficients in SDPT3 format
Hij=(Hij(:))’;
Hij=[Hij;Hij];
Hij=Hij(:);
%% for target C*X
C{1}=sparse(1:(nHij2+n),1:(nHij2+n),[Hij;lambda/(1+delta)*ones(n,1)]);
nC=size(C,1)-n+1;

%% Preparing constraints coefficients in SDPT3 format
```

```
b=zeros(nHij,1);
%% b is the right side of the constraints: Avec^T*X=b
%% the length of b is the number of constraints
%% For example, for the formulation 3.4.3, b
%% will be of length m, (but we better not set b as zero
%% vector, but containing d_ij values which are observed constants)
%% See also the SDPT3 reference

iA2=[];
for ij=1:nHij
    i=iind(ij);
    j=jind(ij);

%% NOTE i>j (down triangle!!!)
    if i>j
         iA2=[iA2,[i*(i+1)/2,(i-1)*i/2+j,j*(j+1)/2]];
    else
         iA2=[iA2,[i*(i+1)/2,(j-1)*j/2+i,j*(j+1)/2]];
    end

    b(ij)=d(i,j);%% Note: But if h(ij)=0,, the constraint doesn't exist
end
```

```
%% Preparing hidden conditions in SDPT3 format

if sw1|lambda==0 %%if sw1==1 or lambda==0 use hidden condition

    disp('Using hidden condition')%%whether explicitly restrict K to be centered

    Avec{1}=sparse( 2:3:(3*nHij-1),1:nHij,ones(1,nHij)*sqrt(.5),...

            3*nHij,nHij+1);

    jA2=1:nHij; jA2=repmat(jA2,3,1); jA2=jA2(:); jA2=jA2';

    d2=n*(n+1)/2;

    iA2=[iA2,1:d2];

    jA2=[jA2,(nHij+1)*ones(1,d2)];

    d2=ones(1,d2)*2;

    ij=0;

    for i=1:n

        ij=ij+i;

        d2(ij)=sqrt(2);

    end


    Avec{1}=[Avec{1};sparse(iA2,jA2,[repmat([1,-sqrt(2),1],1,nHij),...

    d2],n*(n+1)/2,nHij+1)];

    b(nHij+1)=0;

else

   disp('Not using hidden condition')
   Avec{1}=sparse( 2:3:(3*nHij-1),1:nHij,ones(1,nHij)*sqrt(.5),3*nHij,nHij);
   jA2=1:nHij; jA2=repmat(jA2,3,1);
   Avec{1}=[Avec{1};sparse(iA2,jA2(:),repmat([1,-sqrt(2),1],1,nHij),...
     n*(n+1)/2,nHij)];
   %% Avec is essentially the coefficients of the constaints: Avec^T*X=b;
   %% the number of columns are the number of constraints (thus the length
   %% of b). But every original matrix coefficient has to be first converted
   %% to the vector format as svec(X)=[x11,sqrt(2)x12,x22,sqrt(2)x13,...,xnn]’
   %% See SDPT3 and DSDP5 references
end
%% End of Hidden conditions

%% SDP block sizes and types (see SDPT3 reference in PNAS paper)
blk{1,1} = ’s’; %% SDP cones
blk{1,2} = [2*ones(1,nHij),n]; %% block structure: nHij 2*2, and one n*n
%% note that I put all variables in one block strucured matrix


%% Clear redundant variables to save memory for convex conic solver
clear iind jind Hij
clear i j ij iA2 jA2 sw1 nHij2 nHij
if sw  %% if sw==1 use special starting point
```

```
    disp('Using optimal starting point')

    load('start','X0','Z0')

    %% feeding in sqlp (SDPT3 solver) the prepared b, blk, Avec, C

    [obj,X,y,Z,info,runhist] = sqlp(blk,Avec,C,b,[],X0,zeros(length(b),...

     1),Z0);

else

    disp('Using suggested starting point')


    if method(1)=='s'|method(1)=='S'

        disp('Will use solver:  SDPT3') %%using solver SDPT3

        whos, pack

%% Calling solver SDPT3 (install SDPT3, see SDPT3 reference)

         %% feeding in sqlp (SDPT3 solver) the prepared b, blk, Avec, C

        [obj,X,y,Z,info,runhist,Xiter,yiter,Ziter] = sqlp(blk,Avec,C,b);

        if info(1)>0

            X=Xiter; y=yiter; Z=Ziter;

        end

        clear Xiter yiter Ziter

        K=X{1}(nC:end,nC:end);


        %% log(determinant) iterations

        for i=2:ni
            disp(' ')
            disp(['Iteration # ',num2str(i),' .......'])
            C{1}(nC:end,nC:end)=lambda*inv(K+delta*eye(n));
            whos, pack
            [obj,X,y,Z,info,runhist,Xiter,yiter,Ziter] = ...
            sqlp(blk,Avec,C,b,[],X,y,Z);
            if info(1)>0
                X=Xiter; y=yiter; Z=Ziter;
            end
            clear Xiter yiter Ziter
            K=X{1}(nC:end,nC:end);
        end
    else
        disp('Will use solver:  DSDP5')  %%using solver DSDP5
        %% DSDP5 solver provides program readsdpt3.m to convert
        %% SDPT3 problem format to DSDP5 format
        [AC,b]=readsdpt3(blk,Avec,C,b);
        clear blk Avec C sw
        whos, pack
        %% Calling solver DSDP5 (install DSDP5, see DSDP5 reference)
        [STAT,y,X]= dsdp(b,AC);
        K=dmat(X{1}((end+1-n*(n+1)/2):end));
```

```
        %% log(determinant) iterations
        for i=2:ni
            disp(' ')
            disp(['Iteration # ',num2str(i),' .......'])
            AC{1,3}(:,end)=lambda*dvec(inv(K+delta*eye(n)));
            whos, pack
            [STAT,y,X]= dsdp(b,AC,[],y);
            clear STAT
           K=dmat(X{1}((end+1-n*(n+1)/2):end));
        end
    end
end

%% Check the result
%% Eigenvalue eigenvector decomposition
n1=n-1;n2=n-2;
[eigv,tem]=eig(full(K));
tem=diag(tem);
%% first 3 principal coordinates
eigv(:,n)=eigv(:,n)*sqrt(tem(n));
eigv(:,n1)=eigv(:,n1)*sqrt(tem(n1));
eigv(:,n2)=eigv(:,n2)*sqrt(tem(n2));
%% plotting eigen values
figure
plot (log10(tem(n:-1:1)),'-o')
title(['Log_{10} Eigen values in descending order, \lambda=',...
 num2str(lambda)])
ylabel('log_{10}(e.v.)')
xlabel('Rank of e.v.s')
disp('Largest 11 eigen values')
tem(n-10:n)
fprintf('\nSum of eigen values'' squre roots= %e \n',sum(sqrt(abs(tem))));

%% RKE K induced D
tem=diag(K)*ones(1,n);
D=tem+tem'-2*K;
D(H==0)=0;
D=sparse(D);

%% Checking the centering and symmetric condition on RKE K
disp(' ');
disp('Checking the symmetry of result Kernel');
if all((K-K')<1e-10)
```

```
    disp(’K is symmetric’);
else
    disp(’K is not symmetric’)
end

disp(’ ’);
disp(’Checking the normality of result Kernel’);
if all(sum(K)<1e-5)&all(sum(K,2)<1e-5)
    disp(’K is normal’);
else
    disp(’K is not normal’)
end

%% Printing the values of total loss and penalty on the trace for RKE K
fprintf(’\nSum of loss= %e’, full(sum(sum(H.*((abs(D-d)).^2))))  );
fprintf(’\nPenalty on trace (with lambda)= %e \n’, full(lambda*trace(K)));
```

B.2 Newbie algorithm with l_1 loss

```
function [dh,x,ris]=embed3v3L(hdname,d,lambda,H,r)
%hdname: file name of the saved and to-save variable space
%d: observed dissimilarity data for newbies against training set
%lambda: the lambda used for the RKE
%H: as in RKE code
%r: dimensions preserved in newbie algorithm, if r is integer greater than 1
%    proportion of the trace preserved if r is between 0 and 1.

[nnew,N]=size(d);
x=[];ris=[];

if  isempty(lambda)
    load( [’temp’,hdname],’K’,’V’,’E’,’r’)
else
    if prod(size(lambda))==1
        %% Retrieve stored RKE kernels for certain lambda
        sl=sign(lambda);
        lambda=abs(lambda);
        nlambda=num2str(lambda);
        clambda=nlambda;
        clambda(strfind(nlambda,’.’))=’d’;
        clambda(strfind(nlambda,'-'))='m';

        if sl<0
            load(hdname,'fname',['k',clambda])
            eval(['K=k',clambda,';'])
        else
            load(hdname,'fname',['K',clambda])
            eval(['K=K',clambda,';'])
        end
    else
        K=lambda; %% Use lambda to pass unstored K
    end
    %% Eigenvalue eigenvector decomposition
    [V,E]=eig(K);
    [tem,ie]=sort(abs(diag(E)));
    E=real(E(ie,ie));
    V=real(V(:,ie));

    oE=E;
    K=(diag(K))';

    if nargin<5
```

```
        r=.999; %% Default percentage of trace preserved
    end
    if r<1
        tem=diag(E);
        %% Find the number of eigenvalues (from the biggest)
        %% need to be preserved
        r=find( (cumsum(tem(N:-1:1))/sum(K))>r,1,'first');
    end
    tem=N:-1:N-r+1;
    E=-2*[zeros(N,1),V(:,tem)*sqrt(E(tem,tem))];
    save(['temp',hdname],'K','V','E','r')
    %% Careful the abuses of K and E for saving memory
end

if length(K)~=N
    error('Wrong length of provided dissimilarity!')
end
disp(['Embedding ',int2str(nnew),' new objects'])

if nargin<4|isempty(H)
    H=ones(nnew,N);
end
```

```
H(d==0)=0;

%% Set SDPT3 option values
OPTIONS.vers           = 1;
OPTIONS.gam            = 0;
OPTIONS.predcorr       = 1;
OPTIONS.expon          = [1 1];
OPTIONS.gaptol         = 1e-8;
OPTIONS.inftol         = 1e-13;
OPTIONS.steptol        = 1e-12;
OPTIONS.maxit          = 50;
OPTIONS.printyes       = 0;
OPTIONS.scale_data     = 0;
OPTIONS.randnstate     = 0;
OPTIONS.spdensity      = 0.5;
OPTIONS.rmdepconstr    = 0;
OPTIONS.cachesize      = 256;
OPTIONS.smallblkdim    = 15;

for io=1:nnew
    [iind,jind,Hij]=find(H(io,:)); %find nonzero weights
    nHij=length(Hij);
```

```
disp(['# of distinct pairs= ',num2str(nHij)])
nHij1=nHij+1;
nHij2=nHij*2;

%% Preparing convex conic problem 3.4.7 in thesis in SDPT3 solver format
%% Prepare coefficients for variable matrices in optimization target
%% in SDPT3 format
Hij=(Hij(:))';
Hij=[Hij;Hij];
Hij=Hij(:);
C{1}=sparse(2:(nHij2+2),2:(nHij2+2),[1;Hij]);
C{2}=sparse(zeros(r+1,1));

%% Prepare constraints coefficients in SDPT3 format
b=zeros(nHij+2,1);
b(1)=1;
b(3:end)=d(io,jind)-K(jind);
%% b is the right side of the constraints: Avec^T*X=b
%% the length of b is the number of constraints
%% For example, for the formulation 3.4.3, b
%% will be of length m, (but we better not set b as zero
%% vector, but containing d_ij values which are observed constants)
%% See also the SDPT3 reference


Avec{1}=sparse([1:2,3*ones(1,nHij),5:3:(3*nHij1)],[1:nHij+2,3:nHij+2],...
    [1,sqrt(.5),ones(1,nHij),ones(1,nHij)*sqrt(.5)],3*nHij1,nHij+2);
Avec{2}=[zeros(r+1,1),[-1;zeros(r,1)],(E(jind,:))'];
%% Avec is essentially the coefficients of the constaints: Avec^T*X=b;
%% the number of columns are the number of constraints (thus the length
%% of b). But every original matrix coefficient has to be first converted
%% to the vector format as svec(X)=[x11,sqrt(2)x12,x22,sqrt(2)x13,...,xnn]'
%% See SDPT3 and DSDP5 references



%% Block sizes and types in SDPT3 format
blk{1,1} = 's'; %% sdp cones
blk{1,2} = 2*ones(1,nHij1); %% nHij1 # of 2*2 cones
blk{2,1} = 'q'; %% second order cone
blk{2,2} = r+1; %% one for size r+1

%% Clear redundant variables to save memory
clear iind jind Hij tem
clear nHij2 nHij nHij1
```

```
    whos, pack %% Check memory condition
    %% Calling solver SDPT3 (install SDPT3, see SDPT3 reference)
    %% feeding in prepared blk, Avec, C, b
    [obj,X,y,Z,info,runhist] = sqlp(blk,Avec,C,b,OPTIONS);

    tem=X{2}(2:end);
    %% Take out part of the output that is of interest
    %% which is the \~{c} in equation (3.4.6), which is
    %% also the embeded coordinates
    x=[x,tem];
    c=(X{2}(1))^2;
    %% recording the residues c-b^T*K^+*b
    ris=[ris,[c-tem'*tem;X{1}(2,2)-c]];
    clear  blk Avec C b
    clear obj X y Z info runhist
end

% Calculating the fitted dissimilarity (d hat)
load( ['temp',hdname],'K','E')
dh=repmat((K'+c),1,nnew)-E(:,2:end)*x;
```

Bibliography

Ahrens, J. & other 112 coauthors (AMANDA collaboration) (2003), 'Limits on diffuse fluxes of high energy extraterrestrial neutrinos with the amanda-b10 detector', *Physical Review Letters* **90**, 251101.

Ahrens, J. & other 119 coauthors (AMANDA collaboration) (2004), 'Muon track reconstruction and data selection techniques in amanda', *Nuclear Inst. and Methods in Physics Research* **A524**, 169. Reconstruction algorithms are described in section 3 and the event variables are detailed in section 6.2.

Andrés, E. & other 119 coauthors (2001), 'Observation of high-energy neutrinos using cerenkov detectors embedded deep in antarctic ice', *Nature* **410**, 441.

Aronszajn, N. (1950*a*), 'Theory of reproducing kernels', *Transactions of the American Mathematical Society* **68**, 337–404.

Aronszajn, N. (1950*b*), 'Theory of reproducing kernels', *Transactions of the American Mathematical Society* **68**, 337–404.

Atschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. (1990), 'A basic local alignment search tool', *Journal of Molecular biology* **215**, 403–410.

Belkin, M. & Niyogi, P. (2003), 'Laplacian eigenmaps for dimensionality reduction and data representation', *Neural Computation* **15(6)**, 1373–1396.

Ben-Hur, A. & Brutlag, D. (2003), 'Remote homology detection: a motif based approach', *Bioinformatics* **19**(1), i26–i33.

Benson, S. J. & Ye, Y. (2004), DSDP5: A software package implementing the dual-scaling algorithm for semidefinite programming, Technical Report ANL/MCS-TM-255, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL.

Boyd, S. & Vandenberghe, L. (2004), *Convex Optimization*, Cambridge University Press.

Buja, A. & Swayne, D. (2002), 'Visualization methodology for multidimensional scaling', *Journal of Classification* **19**, 7–43.

Cashon, R., Vayda, M. & Sidell, B. D. (1997), 'Kinetic characterization of myoglobins from vertebrates with vastly different body temperatures', *Comparative Biochemistry and Physiology* **117B**, 613–620.

Clegg, J. B. & Gagnon, J. (1981), 'Structure of the zeta chain of human embryonic hemoglobin', *Proceedings of the National Academy of Sciences* **78**(10), 6076–6080.

Cox, D. & O'Sullivan, F. (1990), 'Asymptotic analysis of penalized likelihood and related estimators', *Annals of Statistics* **18**, 1676–1695.

Cristianini, N. & Shawe-Taylor, J. (2000), *An Introduction to Support Vector Machines*, Cambridge University Press.

Donoho, D. L. & Grimes, C. E. (2003), 'Hessian eigenmaps: locally linear embedding techniques for highdimensional data', *Proceedings of the National Academy of Arts and Sciences* **100**, 5591–5596.

Eddy, S. R. (1998), 'Profile hidden markov models', *Bioinformatics* **14**, 755–763.

Evgeniou, T., Pontil, M. & Poggio, T. (2000), 'Regularization networks and support vector machines', *Advances in Computational Mathematics* **13**, 1–50.

Feldman, G. J. & Cousins, R. D. (1998), 'Unified approach to the classical statistical analysis of small signals', *Physical Review D* **57**, 3873.

Ferris, M., Voelker, M. & Zhang, H. H. (2004), 'Model building with likelihood basis pursuit', *Journal of Optimization Methods and Software* **19(5)**, 577–594.

Gentleman, R. C., Carey, V. J., Bates, D. J., Bolstad, B. M., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G. K., Tierney, L., Yang, Y. H. & Zhang, J. (2004), 'Bioconductor: Open software development for computational biology and bioinformatics', *Genome Biology* **5**(10), doi:10.1186/gb–2004–5–10–r80.

Ham, J., Lee, D. D., Mika, S. & Schölkopf, B. (2004), 'A kernel view of the dimensionality reduction of manifolds', *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)* pp. 369–376.

Hill, G. C. & Rawlins, K. (2003), ‘Unbiased cut selection for optimal upper limits in neutrino detectors: the model rejection potential technique’, *Astroparticle Physics* **19**, 393.

Hill, G. C., Lu, F., Desiati, P. & Wahba, G. (2003), ‘Optimizing the limit setting potential of a multivariate analysis using the bayes posterior ratio’, *Proceedings of PHYSTAT2003 at SLAC* pp. 218–221.

Hou, J., Jun, S., Zhang, C. & Kim, S. (2005), ‘Global mapping of the protein structure space and application in structure-based inference of protein function’, *Proceedings of the National Academy of Sciences* **102**(10), 3651–3656.

Jaakkola, T., Diekhans, M. & Haussler, D. (2000), ‘A discriminative framework for detecting remote protein homologies’, *Journal of Computational Biology* **7**(1), 95–114.

Keleş, S., van der Laan, M. J., Dudoit, S., Xing, B. & Eisen, M. B. (2003), ‘Supervised detection of regulatory motifs in DNA sequences’, *Statistical Applications in Genetics and Molecular Biology.* Article 5.

Kimeldorf, G. & Wahba, G. (1971), ‘Some results on Tchebycheffian spline functions’, *Journal of Mathematical Analysis and Applications* **33**, 82–95.

Krogh, A., Brown, M., Mian, I. S., Sjlander, K. & Haussler, D. (1994), ‘Hidden markov models in computational biology: Applications to protein modeling’, *Journal of Molecular Biology* **235**, 1501–1531.

Lagarias, J. C., Reeds, J. A., Wright, M. H. & Wright, P. E. (1998), 'Properties of the nelder-mead simplex method in low dimensions', *SIAM Journal of Optimization* **9**, 112–147.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. E. & Jordan, M. (2004), 'Learning the kernel matrix with semidefinite programming', *Journal of Machine Learning Research* **5**, 27–72.

Lee, Y., Lin, Y. & Wahba, G. (2004*a*), 'Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data', *Journal of The American Statistical Association* **99**, 67–81.

Lee, Y., Lin, Y. & Wahba, G. (2004*b*), 'Multicategory vector machines, theory and application to the classification of microarray data and satellite radiance data', *Journal of The American Statistical Association* **99**, 67–81.

Leslie, C., Eskin, E., Cohen, A., Weston, J. & Noble, W. S. (2004), 'Mismatch string kernels for discriminative protein classification', *Bioinformatics* **20**(4), 467–476.

Liao, L. & Noble, W. S. (2003), 'Combining pairwise sequence similarity and support vector machines for remote protein homology detection', *Journal of Computational Biology* **10**, 857–868.

Lin, X. (1998), Smoothing Spline Analysis of Variance for Polychotomous Response Data, PhD thesis, Department of Statistics, Uiversity of Wisconsin, Madison WI. Also Statistics Dept TR 1003 available via Grace Wahba's website.

Lin, Y., Lee, Y. & Wahba, G. (2002), 'Vector machines for classification in nonstandard situations', *Machine Learning* **46**, 191–202.

Lu, F., Keles, S., Wright, S. & Wahba, G. (2005), 'Framework for kernel regularization with application to protein clustering', *Proceedings of the National Academy of Sciences* **102**, 12332–12337.

Maeda, N. & Fitch, W. M. (1982), 'Isolation and amino acid sequence of a monomeric hemoglobin in heart muscle of the bullfrog, *Rana catesbeiana*', *Journal of Biological Chemistry* **257**(6), 2806–2815.

Nesterov, Y. & Nemirovskii, A. (1993), *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM Studies in Applied Mathematics, v. 13.

O'Sullivan, F., Yandell, B. & Raynor, W. (1986), 'Automatic smoothing of regression functions in generalized linear models', *Journal of The American Statistical Association* **81**, 96–103.

Roweis, S. T. & Saul, L. K. (2000), 'Nonlinear dimensionality reduction by locally linear embedding', *Science* **290**, 2323–2326.

Sandelin, A., Alkema, W., Engström, P., Wasserman, W. & Lenhard, B. (2004), 'JASPAR: an open access database for eukaryotic transcription factor binding profiles', *Nucleic Acids Research* **32**(1), D91–D94.

Schölkopf, B., Smola, A. J. & Muller, K.-R. (1998), “nonlinear component analysis as a kernel eigenvalue problem’, *Neural Computation* **10**, 1299–1319.

Schölkopf, B., Tsuda, K. & Vert, J.-P. (2004), *Kernel Methods in Computational Biology*, MIT Press.

Shawe-Taylor, J. & Cristianini, N. (2004), *Kernel Methods for Pattern Analysis*, Cambridge University Press.

Sibson, R. (1978), ‘Studies in the robustness of multidimensional scaling: Procrustes statistics’, *Journal of the Royal Statistical Society Series, B* **40**, 234–238.

Tang, C. L., Xie, L., Koh, I. Y. Y., Posy, S., Alexov, E. & Honig, B. (2003), ‘On the role of structural information in remote homology detection and sequence alignment: New methods using hybrid sequence profiles’, *Journal of Molecular Biology* **334**(5), 1043–1062.

Tenenbaum, J. B., de Silva, V. & Langford, J. C. (2000), ‘A global geometric framework for nonlinear dimensionality reduction’, *Science* **290**, 2319–2323.

Tibshirani, R. J. (1996), ‘Regression shrinkage and selection via the lasso’, *Journal of the Royal Statistical Society Series, B* **58**, 267–288.

Tütüncü, R. H., Toh, K. C. & Todd, M. J. (2003), ‘Solving semidefinite-quadratic-linear programs using SDPT3’, *Mathematical Programming* **95**(2), 189–217.

Wahba, G. (1977), 'Practical approximate solutions to linear operator equations when the data are noisy', *SIAM Journal on Numerical Analysis* **14**, 651–667.

Wahba, G. (1990*a*), *Spline Models for Observational Data*, SIAM. CBMS-NSF Regional Conference Series in Applied Mathematics, v. 59.

Wahba, G. (1990*b*), *Spline Models for Observational Data*, SIAM. CBMS-NSF Regional Conference Series in Applied Mathematics, v. 59.

Wahba, G. (1999), Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV, *in* B. Schölkopf, C. Burges & A. Smola, eds, 'Advances in Kernel Methods-Support Vector Learning', MIT Press, pp. 69–88.

Wahba, G. (2002*a*), 'Soft and hard classification by reproducing kernel hilbert space methods', *Proceedings of the National Academy of Sciences* **99**, 16524–16530.

Wahba, G. (2002*b*), 'Soft and hard classification by reproducing kernel Hilbert space methods', *Proceedings of the National Academy of Sciences* **99**, 16524–16530.

Wahba, G., Gu, C., Wang, Y. & Chappell, R. (1995*a*), Soft classification, a. k. a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance, *in* D. Wolpert, ed., 'The Mathematics of Generalization, Santa

Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XX', Addison-Wesley, Reading, MA, pp. 329–360.

Wahba, G., Wang, Y., Gu, C., Klein, R. & Klein, B. (1995*b*), 'Smoothing spline anova for exponential families, with application to the wisconsin epidemiological study of diabetic retinopathy', *Annals of Statistics* **23**, 1865–1895.

Weinberger, K. Q., Sha, F. & Saul, L. K. (2004), 'Learning a kernel matrix for nonlinear dimensionality reduction', *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)* pp. 839–846.

Williams, C. K. I. (2001), On a connection between kernel pca and metric multidimensional scaling, *in* T. G. D. T. K. Leen & V. Tresp, eds, 'Advances in Neural Information Processing Systems', Vol. 13, MIT Press, pp. 675–681.

Zhu, J. & Hastie, T. (2004), 'Classification of gene microarrays by penalized logistic regression', *Biostatistics* **5**, 329–340.