Learning via Hilbert Space Embedding of Distributions

by

Le Song

A thesis submitted to
The School of Information Technologies
The University of Sydney
for the degree of
DOCTOR OF PHILOSOPHY

June 1, 2008

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

_____

Le Song

Sydney

June 1, 2008

To My Parents

Acknowledgements

Many individuals and institutions contributed in many different ways to the completion of this thesis. I am deeply grateful for their support, and thankful for the unique chances they offered to me. These supports include scholarship from National ICT Australia, and travel funding and facilities from the University of Sydney and the Australian National University.

My research has profited a lot from interacting with some of the best researchers in my field. I am thankful to all of them: Yasemine Altun, Karsten Borgwardt, Tim Dwyer, Peter Eades, Kenji Fukumizu, Evian Gordon, Arthur Gretton, Seokhee Hong, Bernhard Schölkopf, Alexander Smola, Vishy Vishwanasan and Lea Williams.

I'd also like to thank my fellow students for all the discussions and time. This has helped me gain deeper understanding of the subjects I studied. I am grateful to fellow students from the University of Sydney: Adel Ahmed, Kelvin Cheng, David Fung, Damian Merrick, Kathryn Merrick, Collin Murray, Xiaobin Shen, Yingxin Wu, Ben Yip and Lanbo Zheng; and to fellow students from the Australian National University: Justin Bedo, Quoc Viet Le, Dima Kamenesky, Choonhui Teo, Owen Thomas, Tim Sears, Qingfeng Shi, Jin Yu and Xinhua Zhang.

I especially thank Prof. Peter Eades, Dr. Masahiro Takatsuka and Dr. Soekhee Hong for their advice in the early stage of my PhD studies, and their later generosity to allow me to pursue my own interests.

I feel strongly indebted to my thesis supervisor Prof. Alex Smola. Without his visionary supervision I could never achieve what I have now.

More than to anyone else, I owe to the love and support of my families. Despite all the PhD studies, you are for ever the best of my life.

Contents

List of Figures

List of Tables

Abstract

In this thesis, we propose a framework for learning based on Hilbert space embedding of distributions. By embedding the distributions via the kernel mean map, we are able to compare distributions by computing their distance in the reproducing Hilbert spaces. We show that learning via mean map leads to both good generalization ability and finite sample convergence.

Using this distance between distributions, we are able to tackle a wide range of learning problems under a common framework. Very often this new view leads to simpler and more effective algorithms in various learning problems. In particular, this thesis focuses on a measure of dependence based on the mean map, and we show that it can tackle a raft of learning problems from which we singled out four concrete examples and discussed in details in the thesis:

- Independence measure and test for structured and heterogeneous data.
- Feature selection via dependence for supervised learning scenario.
- Clustering via dependence with additional metric on labels.
- Dimensionality reduction via dependence with side information.

We also show that learning via Hilbert space embedding/dependence subsumes many existing algorithms as special cases. By elucidating the differences and connections of these algorithms, we are able to provide useful guidelines for practitioners in various applications.

This embedding approach for distribution analysis offers us a principled drop-in replacement for information theoretic approaches. We believe it will have wide applications in near future.

CHAPTER 1

Introduction

Information theoretic approaches [30, 4] have long been dominant in the context of analysis of probability distributions (eg. [123, 132, 83]). For instance, mutual information has been used in a wide range of applications such as independent component analysis [132], feature selection [156] and clustering [123]. Information theoretic approaches, however, have by and large a common issue: to estimate quantities such as entropy or mutual information, usually density estimation is needed as an intermediate step. For a direct estimation of these quantities, sophisticated space partitioning and/or bias correction strategies (eg. [99]) are usually needed. To a certain extent, these drawbacks have limited the practical applicability of information theoretic approaches, especially to data of high dimension and small sample size.

The goal of this thesis is to study kernel methods that sidestep these problems. We will discuss a framework of methods which allow us to compute distances between distributions without the need for intermediate density estimation. Moreover, these methods allow algorithm designers to specify which properties of a distribution are most relevant to their problems. This framework works by embedding distributions into the Hilbert spaces, and it often leads to algorithms which are simpler and more effective than information theoretic methods in a broad range of applications.

At the heart of this embedding approach is the mean map, $\mu[\Pr_{\mathbf{x}}] := \mathbb{E}_{\mathbf{x}}[k(\mathbf{x}, \cdot)]$, via the kernel $k(\mathbf{x}, \mathbf{x}')$. The mean map is attractive since, with universal kernels [131], each distribution can be uniquely represented as an element in the RKHS, and then comparing distributions can be achieved by simply manipulating elements in the RKHS [124]. The major advantages of this embedding approach for distribution analysis are three-fold: first, it does not requires intermediate density estimation; second, the empirical mean map has a good convergence guarantee to its population counterpart; and third, prior knowledge can be readily incorporated using kernels. This powerful concept has been exploited successfully for applications such as two-sample test and covariate shift correction [58, 73].

In this thesis, we will use this embedding approach to design a new measure of statistical dependence — we will use the RKHS embedding distance between the joint distribution, $\Pr_{\mathbf{xy}}$, and the product of its marginals, $\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$, as such a measure [124], ie.

$$\Delta := \left\| \mu[\Pr_{\mathbf{xy}}] - \mu[\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}] \right\|_{\mathcal{H}}^2$$

Δ can be used as a principled drop-in replacement for mutual information as a dependence measure and in a wide range of other learning problems.

As a dependence measure, Δ has many nice properties besides those inherited from the embedding approach. For instance, under the assumption of iid. sampling, it is equivalent to the Hilbert-Schmidt norm of the cross-covariance operator and it has a simple empirical

estimate $m^{-2}\operatorname{tr}(\mathbf{HKHL})$ which relies only on the kernel matrices $\mathbf{K}$ and $\mathbf{L}$ for observations $\mathbf{x}$ and $\mathbf{y}$ respectively. Furthermore, by formulating Δ as U-statistics, we obtain a novel test statistic for independence. This new statistic is applicable to structured data such as strings and texts. Most interestingly, it also works for data of heterogeneous sources, such as independence between images and text captions.

As a tool for learning, dependence/relevance is also a natural objective to be optimized in many situations. For instance, we may want to pinpoint several genes from a large pool of candidates such that they are most relevant to cancer diagnosis; we may want to label a set of images such that the labels have good relevance to the underlying categories; or we may want to construct a low dimensional vector presentation of documents such that it is most relevant to the coauthor information in the documents. More specifically, we show that learning via dependence, as expressed by Δ, not only leads to new algorithms but also subsumes many existing algorithms as special cases:

- Feature selection can be formulated as maximizing Δ between the *chosen* features and the given labels. Here we propose a greedy backward elimination procedure for this task (BAHSIC) [130]. This procedure is applicable to regression as well as binary and multi-class datasets. A plethora of feature selectors in microarray analysis, such as Pearson's correlation and signal-to-noise ratio, are special cases of BAHSIC. By showing their connections, BAHSIC provides guidelines on how to choose them (for instance, it helps bioinformaticians on gene selection from microarray data [127]).

- Clustering can be formulated as maximizing Δ between the *generated* cluster labels and the data. Here we propose an iterative procedure (CLUHSIC) [129] which applies to novel clustering problems with structured labels (for instance, a hierarchy of facial images). Furthermore, it provides a unifying theory for the geometric (eg. k-means), spectral and statistical view of clustering. This connection also leads to a new perturbation bound for k-means clustering, which has practical implications on how to choose a kernel for clustering tasks.

- Dimensionality reduction can be formulated as maximizing Δ between the *reduced* representation of the data and the external goal. Here we propose an algorithm that takes side information into account: it reduces the dimensionality of the data via semidefinite programming, while maximizes the dependence between the reduced presentation of the data and the side information (MUHSIC) [128]. Such situation arises, for instance, in text documents where coauthorship is the side information. MUHSIC recovers the popular maximum variance unfolding technique [149] when no side information is available.

Besides the above successes of this embedding approach, we can actually replace information theoretic quantities in many other scenarios as well (for instance, in sensor placement problem [63]). Furthermore, treating various algorithms under a unifying framework and elucidating their connections also benefits practitioners in their specific applications.

1.1 Thesis Structure

The rest of this thesis will be organized into seven chapters. The lineage of the chapters is shown in Figure 1.1. The main contents of each chapter are summarized below:

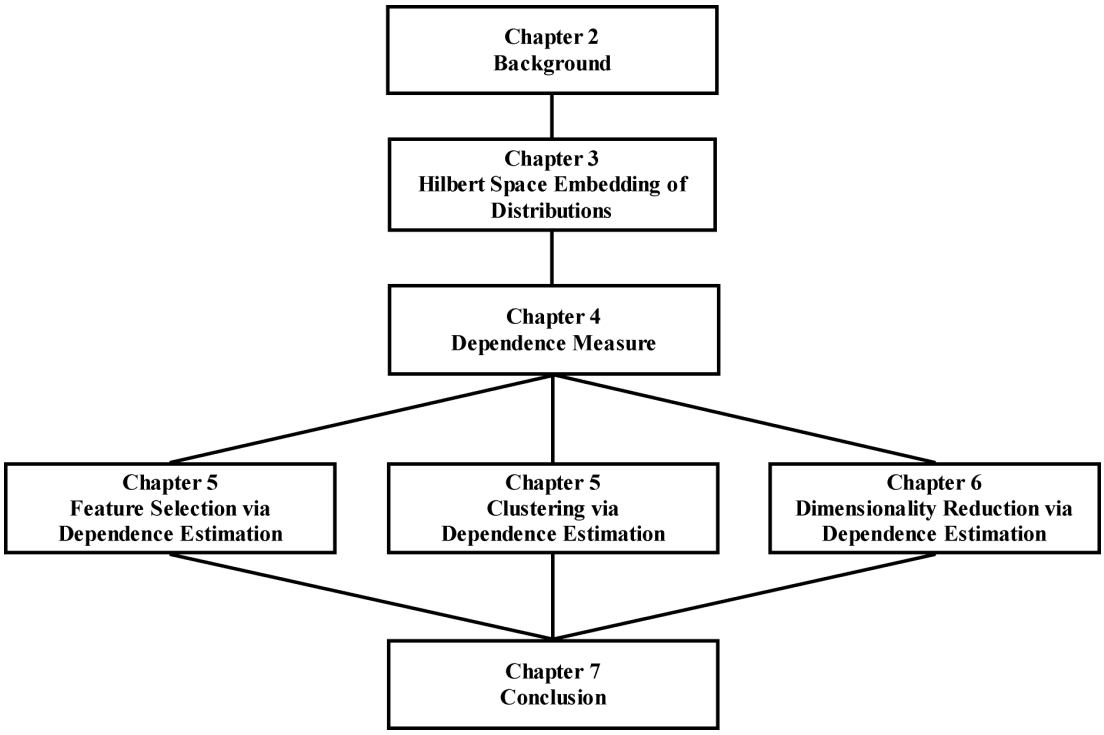


Figure 1.1: Structure of this thesis.

Chapter 2. Background In this chapter, we will cover background knowledge needed for our later theory development. This includes a brief introduction to kernel methods with Support Vector Machines as a motivating example. Then we collect a list of essential properties of kernels. In particular, we put emphasis on the reproducing kernel Hilbert spaces and universal kernels. These two aspects of kernels will play key roles in our Hilbert space embedding approach for distribution analysis. Then we will present basic concepts from exponential family model of distributions. The exponential families are closely connected to kernel methods: the sufficient statistics in an exponential model can be identified as feature maps of kernels. This allows us to draw a connection between our embedding approach and the marginal polytope used in graphical models. Next we will present basic techniques from statistical learning theory for studying the concentration of statistics and the complexity of functions. These techniques are useful for analyzing the convergence of the empirical mean map to its population counterpart. Last we will describe two families of statistics, U-statistics and V-statistics. We will design our later tests for two-sample problem and independence based on U-statistics and V-statistics. Many nice properties of U-statistics and V-statistics can then be transferred directly into our test statistics.

Chapter 3. Hilbert Space Embedding of Distributions In this chapter, we will discuss our Hilbert space embedding approach for distribution analysis in details. We will use the mean map as the key embedding for distribution. We show that the empirical mean map has many nice properties, such as fast convergence to its population counterpart, and one-to-one mapping for distributions when the universal kernels are used. Furthermore, the mean map has the flexibility of choosing a kernel. This allows us to incorporate prior knowledge into the

analysis of distributions and build a connection with the marginal polytope used in graphical models. Most interestingly, the mean map provides us a principled way for comparing the difference between distributions. This is achieved by computing the Hilbert space embedding distance between the distributions. Based on this distance measure, we can formulate many learning problems under a common framework. For instance, we can perform two-sample tests, covariate shift correction, density estimation and independence tests using the mean maps. In particular, we propose a new dependence measure which spawns a whole branch of methods for various supervised and unsupervised learning problems. In chapter 4, 5, 6 and 7, we will focus on learning via this dependence measure.

Chapter 4. Dependence Measure In this chapter, we will first discuss the strengths and weaknesses of four dependence measures from the literature, ie. Pearson's correlation coefficient, mutual information, kernel canonical correlation and constrained covariance. Then we propose our new dependence measure based on the Hilbert space distance between distributions as a better alternative. This new dependence measure is very general. By choosing the kernels, it can be used for structured datasets and data of heterogenous types. Furthermore, under the assumption of iid. sampling, we recover the Hilbert-Schmidt norm of the cross-covariance operator (HSIC) as a special case. We also derive an unbiased estimator and concentration results for HSIC. By formulating HSIC in term of V-statistics we are able to design powerful yet efficient statistics for testing statistical independence. Our experiments on artificial data and text documents demonstrate its superior performance.

Chapter 5. Feature Selection via Dependence Estimation In this chapter, we propose a framework for feature selection based on maximizing HSIC. In principle, selecting features using HSIC can be carried out using either forward selection, backward elimination or feature weighting. In this chapter, we will focus on backward elimination (BAHSIC), since it provides the best performance among the three. Selecting features using HSIC has many additional advantages. First, it has a good statistical basis; second, it can be directly applied to various data types such as binary, multiclass and regression by simply using a different kernel; and third, it also subsumes many existing feature selectors as special cases. For instance, Pearson's correlation, t-statistic and quadratic mutual information are all special cases of HSIC. In a way, we also provide a unifying theory for a raft of feature selectors: they differ only in their respective preprocessing and kernels used. By elucidating their similarity and difference, we are able to provide guidelines on how to choose feature selectors. We have applied BAHSIC to various benchmark datasets, and compared various members of BAHSIC in a large scale microarray studies. We provide rules of thumb for gene selection from microarray data and biological meaningful explanations for gene expression profiles.

Chapter 6. Clustering via Dependence Estimation In this chapter, we propose a framework for clustering based on HSIC (CLUHSIC): clustering is to generate the labels for the data such that the dependence as measured by HSIC between the generated labels and the data are maximized. This formulation subsumes k-means clustering as a special case, and hence it provides a connection between the geometric and statistical views of clustering. A distinctive feature of our approach is that a kernel can also be applied on the labels. We show that CLUHSIC in its most general form is equivalent to the 0-Extension problem studied in theoretical computer science. This connection offers the 0-Extension problem a statistical interpretation. Furthermore, it suggests that we can borrow discrete approximation algorithms for 0-Extension

into the clustering community. In this chapter, we also design several heuristics for solving the CLUHSIC problem as well. This includes a greedy approach, a spectral relaxation method and a nonnegative matrix factorization method. By viewing clustering as a dependence maximization process, we are able to derive a new perturbation bound for k-means, which has practical implications such as how to choose a kernel for clustering. In our experiments, we demonstrate interesting clustering results for hierarchical and manifold datasets.

Chapter 7. Dimensionality Reduction via Dependence Estimation In this chapter, we propose an algorithm for dimensionality reduction based on HSIC (MUHSIC). The idea is to maximize the dependence between the reduced data and the side information while at the same time preserve the local distances between the observations. The advantage of this new algorithm is that it can take side information into account in a principled way. When there is no side information, our algorithm reduces to a maximum variance unfolding algorithm. In a way we also provide statistical interpretation for the excellent performance of maximum variance unfolding. Furthermore, we study MUHSIC using convex duality and show that the dual of MUHSIC is to learn the weights of a graph with constraints on the graph Laplacian. This dual problem offers us insights into the ability of MUHSIC for dimensionality reduction. We run our algorithm on various datasets ranging from images to text documents. In either case, by incorporating side information we obtain better interpretable visualizations. The comparisons with other dimensionality reduction methods also show favorable results for MUHSIC.

Chapter 8. Conclusion In this chapter, we summarize the main results in this thesis. We also discuss some future directions for our Hilbert space embedding approach for distribution analysis. These include further connection to graphical models and learning via dependence for non-iid. data.

1.2 Thesis Contribution

This thesis includes conceptual and algorithmic contributions to the field of machine learning. These contributions are:

1. A Hilbert space embedding approach for distribution analysis.
2. Measuring statistical dependence based on the Hilbert space distance between the joint distribution and the product of the marginal distributions.
3. A framework of learning via dependence estimation.
4. Feature selection via dependence.
5. Clustering via dependence.
6. Dimensionality reduction via dependence.

The set of publications related to this thesis are listed below:

1. A. Smola, A. Gretton, L. Song and B. Schölkopf. A Hilbert space embedding for distributions. *18th International Conference on Algorithmic Learning Theory*, 2007.

2. A. Gretton, K. Fukumizu, C.H. Teo, L. Song, B. Schölkopf and A. Smola. A kernel statistical test of independence. *Advances in Neural Information Processing Systems 20*, 2007.

3. L. Song, A. Smola, A. Getton, J. Bedo and K. Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Researches*. (accepted)

4. L. Song, J. Bedo, K. Borgwardt, A. Getton and A. Smola. Gene selection via the BAHSIC family of algorithms. *15th Annual International Conference on Intelligent Systems for Molecular Biology*, 2007.

5. L. Song, A. Smola, Arthur Gretton, K. Borgwardt and J. Bedo. Supervised feature selection via dependence estimation. *24th International Conference on Machine Learning*, 2007.

6. L. Song, A. Smola, Arthur Gretton and K. Borgwardt. A dependence maximization view of clustering. *24th International Conference on Machine Learning*, 2007.

7. L. Song, A. Smola, K. Borgwardt and A. Getton. Colored maximum variance unfolding. *Advances in Neural Information Processing Systems 20*, 2007.

8. L. Song, X. Zhang, A. Smola, A. Gretton and B. Schoelkopf. Tailoring density estimation via reproducing kernel moment matching. *25th International Conference on Machine Learning*, 2008.

9. N. Quadrianto, L. Song and A. Smola. Kernelized sorting. Submitted.

10. X. Zhang, L. Song, A. Gretton and A. Smola. Kernel measures of independence for non-*iid* data. Submitted.

Besides the above contributions, the author has also carried out research in other areas such as computational neuroscience and information visualization. For the reason of consistency, these contents are not included in this thesis. The corresponding research contributions for neuroscience community include:

1. S. Kuan, J. Gatt, C. Dobson-Stone, D. Palmer, R. Paul, L. Song, E. Gordon, P. Schofield and L. Williams. A polymorphism of the MAOA gene is associated with emotional brain and behaviour markers of antisocial and psychopathic personality traits. (submitted to the Journal of Neuroscience)

2. L. Williams, D. Palmer, B. Liddell, L. Song and E. Gordon. The 'when' and 'where' of perceiving signals of threat versus non-threat. *NeuroImage*, vol 31, pp. 458–467, 2006.

3. L. Song, and J. Epps. Classifying EEG for brain-computer interfaces: learning optimal filters for dynamical system features. *23rd International Conference on Machine Learning*, 2006.

4. L. Song, and J. Epps. Improving the separability of EEG signals during motor imagery with an efficient circular Laplacian. *31st IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006.

5. L. Song, E. Gordon, and E. Gysels. Phase synchrony rate for the recognition of motor imagery in brain-computer interface. *Advances in Neural Information Processing Systems 18*, 2005.

6. L. Song. Desynchronization network analysis for the recognition of imagined movement. *27th IEEE International Conference of the Engineering in Medicine and Biology Society*, 2005.

and those for information visualization community include:

1. W. Huang, C. Murray, X. Shen, L. Song, Y. X. Wu, and L. Zheng. Visualization and analysis of network motifs. *9th International Conference on Information Visualization*, 2005.

2. A. Ahmed, T. Dywer, S.H. Hong, C. Murray, L. Song, and Y.X. Wu. Visualization and analysis of large and complex scale-free networks. *7th IEEE VGTC Symposium on Visualization*, 2005.

3. L. Zheng, L. Song and P. Eades. Crossing minimization problems of drawing bipartite graphs in two clusters. *4th Asian-Pacific Symposium on Information Visualization*, 2005.

4. A. Ahmed, T. Dywer, S.H. Hong, C. Murray, L. Song, and Y. X. Wu. Wilmascope graph visualization. *10th IEEE Symposium on Information Visualization*, 2004.

1.3 Notation

In this thesis, we will deal mainly with vectorial data in $\mathbb{R}^d$ where $d \in \mathbb{N}$ is the dimension of the space. However, many theories developed in this thesis apply to general Hilbert spaces $\mathcal{H}$. We will use bold lower case character to denote both a vector in $\mathbb{R}^d$ and an element in a Hilbert space. For instance, $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{x} \in \mathcal{H}$. For scalars, we will use lower case character. For instance, $m \in \mathbb{R}$ and $\alpha \in \mathbb{R}$. We will use upper case character $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ to denote a set of m observations, and bold upper case character $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)^\top$ to denote a matrix formed from m observations. Likewise, we bundle the labels into a matrix $\mathbf{Y}$ for $\mathbf{y}_i \in \mathbb{R}^d$, ie. $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)^\top$, or $\mathbf{y}$ for $y_i \in \mathbb{R}$, ie. $\mathbf{y} = (y_1, \ldots, y_m)^\top$. We will index the ijth entry in $\mathbf{X}$ by $\mathbf{X}_{ij}$. Also, we will refer to the ith row and column of $\mathbf{X}$ as $\mathbf{X}_{i\star}$ and $\mathbf{X}_{\star i}$ respectively. For a vector $\mathbf{x}$, we will use $\mathbf{x}(i)$ to denote its ith dimension.

Furthermore, we denote the mean and standard deviation of the jth dimension of $\mathbf{X}$ by $\mu_j = \frac{1}{m}\sum_{i=1}^m \mathbf{X}_{ij}$ and $s_j^2 = \frac{1}{m}\sum_{i=1}^m (\mathbf{X}_{ij} - \mu_j)^2$. For binary classification problems we denote by m_+ and m_- the numbers of positive and negative observations. Moreover, μ_{j+} and μ_{j-} correspond respectively to the means of the positive and negative classes at the jth dimension (and correspondingly for standard deviations s_{j+} and s_{j-}). More generally, let m_y be the number of samples with class label equal to y (this notation is also applicable to multiclass problems). Finally, let $\mathbf{1}_m$ be a vector of all ones with length m and $\mathbf{0}_m$ be a vector of all zeros. In many cases, we will omit the subscript m. The exact dimension of the $\mathbf{1}$ and $\mathbf{0}$ will be clear from the context. When the labels are scalars, we use μ_y and s_y to denote their mean and standard deviation respectively.

CHAPTER 2

Background

In this chapter, we will introduce essential background knowledge necessary for the development of our later theory: Hilbert space embedding for distribution analysis. Our approach draws resources from kernel methods, functional analysis, exponential family of distributions, statistical learning theory and classical statistical literature.

2.1 Kernel Methods

Many machine learning algorithms can be expressed in term of inner products between observations, $\langle \mathbf{x}, \mathbf{x}' \rangle$, or inner products between observation matrices, $\mathbf{X}\mathbf{X}'^{\top}$. For instance, a linear classifier, $f(\mathbf{x})$, can be expressed as $\sum_i \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$; A cluster assignment, Π, can be optimized over the objective $\mathrm{tr}(\Pi \mathbf{X}\mathbf{X}^{\top}\Pi^{\top})$; and dimensionality reduction can be achieved from the principle eigenvectors of $\mathbf{X}\mathbf{X}^{\top}$. A key idea arising from this observation is the so-called *kernel trick* [115]: wherever inner products are used, they are replaced by kernel functions (or kernels, for simplicity). Kernels can be viewed as nonlinear similarity measures between two observations. This is in contrast to a normal inner product which is a linear similarity measure. The kernel trick can readily convert a linear algorithm into a non-linear one by simply using kernels. Non-linear mappings coded in kernels transform the observations into elements in a higher-dimensional space; thus a linear algorithm in the new feature space is equivalent to a non-linear one in the original space. In other words, kernel methods modularize a learning algorithms. While the kernel components can be changed to incorporate different nonlinearity in the data, the same algorithm remains unchanged irrespective of the kernel used. Such reusability has made kernel methods favorable from a software engineering point of view. This has also led to the popularity of kernel methods in a wide range of applications.

2.1.1 Support Vector Machines

Before we introduce the theory of kernels, we would like to use the Support Vector Machine (SVM) as a motivating example for kernel methods. This simple example aim to reveal the basic principle underlying a large family of kernel learning algorithms. The key message is that a linear algorithm can be lifted to handle nonlinear cases by upgrading it with kernels. For a more extensive treatment of various kernel methods, we refer the reader to [115] and the references therein.

Support Vector Machines (SVMs) deal with the following binary classification problem: given a set of observations $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ (eg. $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^d, d \in \mathbb{N}$) and their associated labels $Y = \{y_1, \ldots, y_m\}$ (eg. $y_i \in \mathcal{Y} = \{-1, 1\}$) drawn i.i.d. from the space $\mathcal{X} \times \mathcal{Y}$, the task is to learn a classifier $f : \mathcal{X} \longmapsto \mathcal{Y}$ that predicts the labels of new observations. A linear SVM parameterizes the classifier as a hyperplane and an offset, ie. $f = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$. It

tries to separate the observations using this hyperplane while at the same time ensure that the minimum distance between the hyperplane and the observations (the margin γ) is maximized. These constraints lead to the following large margin algorithm for learning f

$$\begin{aligned} &\underset{\mathbf{w},b,\gamma}{\text{maximize}}\ \gamma \\ &\text{subject to } \forall i, y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq \gamma, \text{ and } \|\mathbf{w}\|^2 = 1. \end{aligned} \tag{2.1}$$

To see how this constrained optimization problem motivates the kernel method, we first derive its dual problem using the Lagrangian below

$$\mathcal{L} = -\gamma - \sum_i \alpha_i \left[y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) - \gamma \right] + \lambda(\|\mathbf{w}\|^2 - 1), \quad \forall i, \alpha_i \geq 0. \tag{2.2}$$

The dual problem is equal to $\inf_{\mathbf{w},b,\gamma} \{\mathcal{L}\}$. Thus we differentiate $\mathcal{L}$ with respect to the primal variables $\mathbf{w}, b, \gamma$, and set the derivatives to zeros

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = -\sum_i \alpha_i y_i \mathbf{x}_i + 2\lambda \mathbf{w} = 0, \tag{2.3}$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_i \alpha_i y_i = 0, \text{ and} \tag{2.4}$$

$$\frac{\partial \mathcal{L}}{\partial \gamma} = -1 + \sum_i \alpha_i = 0. \tag{2.5}$$

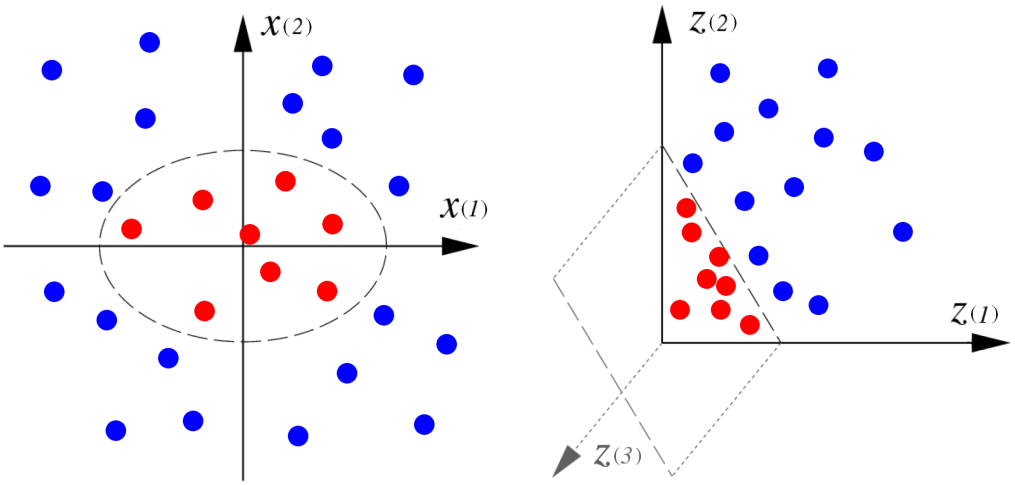
Substituting these relations into (2.2) and then simplifying, we obtain the dual problem

$$\underset{\alpha}{\text{minimize}} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{2.6}$$

$$\text{subject to } \sum_i y_i \alpha_i = 0, \ \sum_i \alpha_i = 1 \text{ and } \forall i, \alpha_i \geq 0. \tag{2.7}$$

Note that this is a convex quadratic optimization problem with respect to the dual variables α_i. To see this, we can express the dual objective in a matrix format, ie. $\boldsymbol{\alpha}^\top \mathbf{A} \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^\top$ and $\mathbf{A}_{ij} = y_i y_i \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. Matrix $\mathbf{A}$ is positive semi-definite and equal to the Hessian of the optimization problem; furthermore, all constraints are affine. Hence the optimization problem is convex [21].

An interesting characteristic of the dual problem is that the observations are completely encapsulated inside the inner products $\langle \mathbf{x}, \mathbf{x}' \rangle$. First, the dual objective depends on the observations only through their inner products. Second, the hyperplane in the classifier is a linear combination of the observations, ie. $\mathbf{w} = \frac{1}{2\lambda} \sum_i \alpha_i y_i \mathbf{x}_i$. Thus both $\langle \mathbf{w}, \mathbf{x} \rangle = \frac{1}{2\lambda} \sum_i \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle$ and the offset $b = y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle$ can be expressed as inner products between observations. This encapsulation provides us nice basis for the kernel trick: we can replace the inner product with a nonlinear similarity measure between observations, ie. $\langle \mathbf{x}, \mathbf{x}' \rangle \Rightarrow k(\mathbf{x}, \mathbf{x}')$. In this case, matrix $\mathbf{A}$ has entries $\mathbf{A}_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$. As long as this new measure maintains the convexity of the underlying optimization problems, ie. $\mathbf{A} \succeq 0$, we can use the same algorithm for both linear and nonlinear problems. As we will discuss later, $\mathbf{A} \succeq 0$ is also a sufficient condition for defining a valid kernel.



(a) Original space (b) Feature space

Figure 2.1: Effects of kernels

2.1.2 Kernel Tricks

In this section, we will illustrate how a kernel affects the behavior of an SVM. As shown in Figure 2.1(a), we have a dataset with observations $\mathbf{x} \in \mathbb{R}^2$. Different colors of the dots represent observations from different classes. A linear classifier (or a line) is unable to perfectly separate the positive classes (red) and the negative classes (blue). One way to separate the classes is to use the ellipsoidal dash line showed in the figure. The question is how can we obtain the ellipsoidal boundary but still using the same optimization routine? This can be achieved by first transforming the observations into linearly separable data and then performing linear classification in the new feature space. To illustrate this, we apply a nonlinear transformation on $\mathbf{x}$, ie. $\phi(\mathbf{x}) : \mathbf{x} \longmapsto z = (\mathbf{x}(1)^2, \mathbf{x}(2)^2, \sqrt{2}\mathbf{x}(1)\mathbf{x}(2))^\top \in \mathbb{R}^3$. In the new feature space (Figure 2.1(b)), the observations from different classes are now linearly separable, and therefore we can use the algorithm for a linear classifier.

As we discussed in section 2.1.1, an SVM applied in the new feature space can be expressed in term of the inner products $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathbb{R}^3}$. If we simplify this expression, we have

$$\begin{aligned} \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathbb{R}^3} &= \mathbf{x}(1)^2\mathbf{x}'(1)^2 + \mathbf{x}(2)^2\mathbf{x}'(2)^2 + 2\mathbf{x}(1)\mathbf{x}(2)\mathbf{x}'(1)\mathbf{x}'(2) \\ &= \langle \mathbf{x}, \mathbf{x}' \rangle^2_{\mathbb{R}^2} . \end{aligned} \tag{2.8}$$

The interesting thing is that to compute inner products $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathbb{R}^3}$, we do not actually need to construct the nonlinear feature map ϕ explicitly. This can be done implicitly by simply computing $\langle \mathbf{x}, \mathbf{x}' \rangle^2_{\mathbb{R}^2}$ instead. This may lead to significant computational savings especially when the feature space is large or of infinite dimension. In this example, we call $\langle \mathbf{x}, \mathbf{x}' \rangle^2_{\mathbb{R}^2}$ a kernel. It is a nonlinear similarity measure between two observations. Equivalently a kernel transforms the observations nonlinearly into a new feature space ϕ and then compare transformed observations linearly in the new space. From this example, we also see that a kernel endows a linear algorithm with extra capacity to handle nonlinear situations. In the next section, we will define kernels formally and introduce some related theories.

2.2 Kernels

Kernel functions (or kernels) $k(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \longmapsto \mathbb{R}$ can be viewed as a special type of similarity measure between two observations. Formally, a kernel $k(\cdot, \cdot)$ is an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ in a feature space $\mathcal{H}$. Given a general set $\mathcal{X}$ and two observations $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$,

$$k(\mathbf{x}, \mathbf{x}') := \left\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \right\rangle_{\mathcal{H}}, \tag{2.9}$$

where $\phi(\mathbf{x}) : \mathcal{X} \longmapsto \mathcal{H}$ is a nonlinear mapping from an observation to its feature space representation. In many cases, $\phi(\mathbf{x})$ seldom needs to be computed explicitly. This may lead to significant computational savings, especially when the feature space $\mathcal{H}$ is of much higher dimensions than $\mathcal{X}$. Note that, for a given kernel, the feature map $\phi(\mathbf{x})$ is usually not unique. For instance, the kernel $\langle \mathbf{x}, \mathbf{x}' \rangle^2$ can be decomposed as either

$$\left\langle (\mathbf{x}(1)^2, \mathbf{x}(2)^2, \sqrt{2}\mathbf{x}(1)x(2))^\top, (\mathbf{x}'(1)^2, \mathbf{x}'(2)^2, \sqrt{2}\mathbf{x}'(1)\mathbf{x}'(2))^\top \right\rangle \text{ or} \tag{2.10}$$

$$\left\langle (\mathbf{x}(1)^2, \mathbf{x}(2)^2, \mathbf{x}(1)\mathbf{x}(2), \mathbf{x}(1)\mathbf{x}(2))^\top, (\mathbf{x}'(1)^2, \mathbf{x}'(2)^2, \mathbf{x}'(1)\mathbf{x}'(2), \mathbf{x}'(1)\mathbf{x}'(2))^\top \right\rangle. \tag{2.11}$$

A natural question is what functions are qualified as kernels? From the definition of kernels, we have so far only one way of verifying this: explicitly construct a feature map ϕ and then test whether the inner product between two images equals to the kernel value. In some cases, however, it may simply be difficult to construct a feature map explicitly. This happens especially when the structure of the data and prior knowledge of a particular application suggest a way of comparing two observations. We want to know whether the function that makes this comparison is a kernel.

2.2.1 Properties of Kernels

Before we describe alternative methods for demonstrating a candidate function is a kernel, we first introduce the following two concepts.

Definition 1 (Gram Matrix) *Given a symmetric function $k : \mathcal{X} \times \mathcal{X} \longmapsto \mathbb{R}$ and observations $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$, then the real symmetric $m \times m$ matrix $\mathbf{K}$ with elements*

$$\mathbf{K}_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.12}$$

for all $1 \leq i, j \leq m$ is called the Gram matrix of k with respect to X.

Definition 2 (Positive Semi-Definite Matrix) *A real symmetric $m \times m$ matrix $\mathbf{K}$ satisfying*

$$\sum_{i,j=1}^{m} c_i c_j \mathbf{K}_{ij} \geq 0 \tag{2.13}$$

for all $c_i \in \mathbb{R}$ is called positive semi-definite.

Theorem 3 (Positive Semi-Definite Kernel) *A symmetric function $k : \mathcal{X} \times \mathcal{X} \longmapsto \mathbb{R}$ is a positive definite kernel (or a kernel), if for any finite set of observations ($\forall m \in \mathbb{N}$ and $\forall \mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{X}$) it gives rise to a positive definite Gram matrix.*

Equivalently this theorem says that, as long as the function k satisfies the finite positive definite property, it has a decomposition of $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$ and hence a kernel. Furthermore, kernels have a set of attractive closure properties. This means that certain operations on kernels still yield kernels. These closure operations will not only help us to verify whether a function is a kernel, but also allow us to create new kernels from simple building blocks. Four primitive closure operations are (1) nonnegative offset, (2) positive linear combination, (3) tensor product and (4) limit of kernels, ie.

(1) If k_1 is a kernel, and $b \geq 0$, then $k_1 + b$ is a kernel.

(2) If k_1 and k_2 are kernels, and $\alpha_1, \alpha_2 \geq 0$, then $\alpha_1 k_1 + \alpha_2 k_2$ is a kernel.

(3) If k_1 and k_2 are kernels, then $k(\mathbf{x}, \mathbf{x}') := k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$ is a kernel.

(4) If $k_1, k_2, \ldots$ are kernels, and $k(\mathbf{x}, \mathbf{x}') := \lim_{n \to \infty} k_n(\mathbf{x}, \mathbf{x}')$ exists for all $\mathbf{x}, \mathbf{x}'$, then k is a kernel.

Based on these primitives, we can derive more complicated closure operations. For instance, a polynomial function $f : \mathbb{R} \longmapsto \mathbb{R}$ with positive coefficients, ie.

$$f(r) = \sum_{i=0}^{d} a_i r^i, \ \forall d \in \mathbb{N} \text{ and } a_i \geq 0 \tag{2.14}$$

is such an operation, since it only involves positive combination of the products of r. From this operation, we can derive the polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + c)^d$ $(c \geq 0, d \in \mathbb{N})$.

A second example is the exponential kernel $k(\mathbf{x}, \mathbf{x}') = \exp(\sigma \langle \mathbf{x}, \mathbf{x}' \rangle)$ $(\sigma > 0)$. We can verify this by expanding the exponential function $\exp(\sigma r)$ around zero, where we obtain an infinite sum of polynomials with positive coefficients, ie.

$$\exp(\sigma r) = \sum_{i=0}^{\infty} \frac{\sigma^i}{i!} r^i. \tag{2.15}$$

A third example is to combine Theorem 3 and the closure operations to define new kernels. For instance,

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') :=& \frac{\exp(2\sigma \langle \mathbf{x}, \mathbf{x}' \rangle)}{\sqrt{\exp(2\sigma \langle \mathbf{x}, \mathbf{x} \rangle) \exp(2\sigma \langle \mathbf{x}', \mathbf{x}' \rangle)}} \\ =& \exp\left(-\sigma \left\| \mathbf{x} - \mathbf{x}' \right\|^2\right) \end{aligned} \tag{2.16}$$

is a kernel. First, we know that $k_1(\mathbf{x}, \mathbf{x}') = \exp(2\sigma \langle \mathbf{x}, \mathbf{x}' \rangle)$ is a kernel according to (2.15); Second, according to Theorem 3, there must exist certain feature space $\mathcal{H}$ and the associated feature map $\phi(\mathbf{x})$ such that $k_1(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}$. If we scale the feature map by $\sqrt{k_1(\mathbf{x}, \mathbf{x})} = \|\phi(\mathbf{x})\|$, we have the new feature map $\tilde{\phi}(\mathbf{x}) := \phi(\mathbf{x}) / \|\phi(\mathbf{x})\|$ which is the feature map for k. Since we can explicitly construct a feature map for k, then k is a kernel by definition.

A fourth example is the inverse square distance kernel defined as

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') :=& \int_0^\infty \exp\left(-\sigma\left(\left\|\mathbf{x}-\mathbf{x}'\right\|^2 + \epsilon\right)\right) \mathrm{d}\sigma \\ =& \frac{1}{\left\|\mathbf{x}-\mathbf{x}'\right\|^2 + \epsilon}, \quad \epsilon > 0. \end{aligned} \tag{2.17}$$

We show that k is a kernel by using the argument that limit of kernels remains a kernel. First, we construct the following sequence k_Δ indexed by $\Delta > 0$

$$\begin{aligned} k_\Delta(\mathbf{x}, \mathbf{x}') :=& \int_0^\Delta \exp\left(-\sigma\left(\left\|\mathbf{x}-\mathbf{x}'\right\|^2 + \epsilon\right)\right) \mathrm{d}\sigma \\ =& \lim_{n\to\infty} \sum_{i=1}^{n} \exp\left(-\frac{i\Delta}{n}\left(\left\|\mathbf{x}-\mathbf{x}'\right\|^2 + \epsilon\right)\right) \frac{\Delta}{n}. \end{aligned} \tag{2.18}$$

Note that each k_Δ is a kernel, since k_Δ by itself is the limit of a sequence of kernels. Second, we take the limit with respect to Δ, then we have $k = \lim_{\Delta\to\infty} k_\Delta$ is also a kernel.

2.2.2 Example Kernels

Besides vectors, kernels have been defined over various domains, such as strings, graphs and dynamical systems (for recent reviews see [119, 117, 71]). These kernels are very often based on prior knowledge or special structure of a particular domain. These non-conventional kernels play a key role in extending the applicability of kernel algorithms to a large variety of data type. Since we will mainly deal with vectorial data in this thesis, we will list a battery of commonly used vectorial kernels in Table 2.1 for our later reference.

Table 2.1: Example kernels

Name	Definition
linear kernel	$\langle \mathbf{x}, \mathbf{x}' \rangle$
polynomial kernel	$\left(\langle \mathbf{x}, \mathbf{x}' \rangle + c\right)^d,\ c \geq 0, d \in \mathbb{N}$
exponential kernel	$\exp\left(\sigma \langle \mathbf{x}, \mathbf{x}' \rangle\right)$
Gaussian kernel	$\exp\left(-\sigma \left\|\mathbf{x}-\mathbf{x}'\right\|^2\right)$
Laplace kernel	$\exp\left(-\sigma \left\|\mathbf{x}-\mathbf{x}'\right\|\right)$
inverse distance kernel	$\frac{1}{\left\|\mathbf{x}-\mathbf{x}'\right\|+\epsilon} = \int_0^\infty \exp\left(-\sigma\left(\left\|\mathbf{x}-\mathbf{x}'\right\| + \epsilon\right)\right) \mathrm{d}\sigma,\ \epsilon > 0$
inverse square distance kernel	$\frac{1}{\left\|\mathbf{x}-\mathbf{x}'\right\|^2+\epsilon} = \int_0^\infty \exp\left(-\sigma\left(\left\|\mathbf{x}-\mathbf{x}'\right\|^2 + \epsilon\right)\right) \mathrm{d}\sigma,\ \epsilon > 0$
delta kernel	$\delta(\mathbf{x}, \mathbf{x}') = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{x}', \\ 0, & \text{otherwise.} \end{cases}$

2.2.3 Reproducing Kernel Hilbert Spaces

As we discussed at the beginning of section 2.2, the feature space for a given kernel is usually not unique. There is, however, a special feature space — the reproducing kernel Hilbert spaces

(RKHS) — which is unique to a given kernel. We will first quote the theorem below before we formally define the RKHS.

Theorem 4 (Moore-Aronszajn) *(Theorem 1.1.1 [59]) Given a kernel $k : \mathcal{X} \times \mathcal{X} \longmapsto \mathbb{R}$, we can construct a unique RKHS with k as its reproducing kernel. Conversely, to every RKHS there corresponds to a unique reproducing kernel k.*

In the theorem, the RKHS is simply a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \longmapsto \mathbb{R}$ spanned by the function map $k(\mathbf{x}, \cdot)$, i.e. $\mathcal{H} = \overline{\text{span}\{k(\mathbf{x}, \cdot)|\mathbf{x} \in \mathcal{X}\}}$. Furthermore, the dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ in this space satisfies the following reproducing properties

$$\langle f, k(\mathbf{x}, \cdot) \rangle_{\mathcal{H}} = f(\mathbf{x}) \text{ for all } f \in \mathcal{H} \text{ and } \mathbf{x} \in \mathcal{X}, \tag{2.19}$$

$$\text{and consequently } \left\langle k(\mathbf{x}, \cdot), k(\mathbf{x}', \cdot) \right\rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}') \text{ for all } \mathbf{x}, \mathbf{x}' \in \mathcal{X}. \tag{2.20}$$

The one-to-one correspondence between k and its associated RKHS is important. This means that by choosing a particular kernel, we uniquely define the class of functions for subsequent analysis.

The concept of the RKHS is essential in characterizing a special class of kernel, called universal kernels. Universal kernels are intimately related to the consistency of an SVM classifier [131] and is commonly referenced in the rest of the thesis.

2.2.4 Universal Kernels

Now we formally define continuous kernels and universal kernels below.

Definition 5 (Continuous Kernel [131]) *Let k be a kernel on $\mathcal{X}$ with feature map $k(\mathbf{x}, \cdot)$. Then k is called a continuous kernel if and only if $k(\mathbf{x}, \cdot)$ is continuous.*

Definition 6 (Universal Kernel [131]) *Let $C(\mathcal{X})$ be the space of continuous bounded functions on compact domain $\mathcal{X}$. A continuous kernel k on domain $\mathcal{X}$ is called universal if the space of all functions induced by k is dense in $C(\mathcal{X})$, i.e. for every function $f \in C(\mathcal{X})$ and every $\epsilon > 0$ there exists a function g induced by k with $\|f - g\|_\infty \leq \epsilon$.*

Sometimes, it may not be convenient to check the universality of a kernel using the definition alone. There are other ways to determine whether a kernel is universal according to the following theorems presented in [131].

Theorem 7 *A kernel $k(\mathbf{x}, \mathbf{x}') = k(\langle \mathbf{x}, \mathbf{x}' \rangle)$ defined on $\mathbb{R}^d$, with power series expansion*

$$k(r) = \sum_{i=0}^{\infty} a_i r^i, \tag{2.21}$$

is a universal kernel if and only if for all i, we have a_i strictly positive.

Theorem 8 *A kernel $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ defined on $\mathbb{R}^d$, with a Fourier transformation*

$$F[k](\boldsymbol{\omega}) = (2\pi)^{-\frac{d}{2}} \int_{\mathbf{r} \in \mathbb{R}^d} e^{i\langle \boldsymbol{\omega}, \mathbf{r} \rangle} k(\mathbf{r}) \mathrm{d}\mathbf{r} \tag{2.22}$$

is a universal kernel if and only if for all $\boldsymbol{\omega}$, we have the Fourier coefficients strictly positive.

Theorem 9 *Let k_1 be a universal kernel on $\mathcal{X}$. Then the following normalization*

$$k(\mathbf{x}, \mathbf{x}') := \frac{k_1(\mathbf{x}, \mathbf{x}')}{\sqrt{k_1(\mathbf{x}, \mathbf{x}) k_1(\mathbf{x}', \mathbf{x}')}} \tag{2.23}$$

defines a universal kernel on $\mathcal{X}$.

From Theorem 7, we see that the exponential kernel $k(\mathbf{x}, \mathbf{x}') = \exp(\sigma \langle \mathbf{x}, \mathbf{x}' \rangle)$ is universal. Gaussian kernel is a normalized exponential kernel, and therefore it is also universal due to Theorem 9. Note that the above results on universal kernels focus on compact subsets of $\mathbb{R}^d$. For general domain, there is a simple condition [19].

Theorem 10 (Positive Definite Kernel is Universal) *A kernel is universal, if for arbitrary sets of distinct points it induces strictly positive definite kernel matrices.*

Using this theorem, we can show that the following string kernel is universal [19].

Theorem 11 (Universal String Kernel) *Let $\mathcal{X}$ be a finite set of strings. Then any string kernel of the form $k(\mathbf{x}, \mathbf{x}') = \sum_{a \in \mathcal{X}} w_a \#_a(\mathbf{x}) \#_a(\mathbf{x}')$ is universal, where $\#_a(\cdot)$ computes the number of occurrence of substring a and $w_a > 0$ for all $a \in \mathcal{X}$.*

For graphs unfortunately no universal kernel exists which are efficiently computable. This can be understood via a self-contradiction, as appeared in the reasoning of [52]: first, a necessary condition for a kernel to be universal is that its feature map $\phi(\mathbf{x})$ has to be injective; second, if such a kernel exists, we can detect the isomorphism between two graphs $\mathbf{x}$ and $\mathbf{x}'$ by simply computing their Hilbert space distance $\|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|^2 = k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}')$. However, graph isomorphism is an NP-hard problem [51], which means that computing a universal kernel on graphs is also NP-hard.

Universal kernels are important since many results regarding the analysis of distributions are stated with respect to $C(\mathcal{X})$ and we would like to translate them into results on Hilbert spaces. Two most important properties of universal kernels are [131]

(1) Every universal kernel separates all compact subsets;

(2) Every feature map of a universal kernel is injective.

The first property means that, given any finite subset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$, the classes corresponding to every possible label assignment can always be correctly separated by a function in the RKHS. The second property means that a unique observation will be represented as a unique element in the RKHS. In another word, universal kernels preserve the identity of an observation in the RKHS. This second property plays a key role in our later approach for Hilbert space embedding of distribution.

2.3 Exponential Family Model of Distributions

There is a family of distributions that are intimately related to the kernels, called the exponential family. In this family, the potential functions or sufficient statistics can be identified with the feature maps of the kernels. Furthermore, this family has some other nice properties pertaining to our later theory for Hilbert space analysis of distributions.

2.3.1 Definitions and Basic Facts

Formally, given a kernel $k : \mathcal{X} \times \mathcal{X} \longmapsto \mathbb{R}$ with associated reproducing kernel Hilbert space $\mathcal{H}$, the corresponding exponential family model of distributions can be parameterized by θ as

$$\Pr_{\mathbf{x}}(\theta) := \exp\left(\langle \theta, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} - A(\theta)\right). \tag{2.24}$$

Here $k(\mathbf{x}, \cdot)$ is called potential functions or sufficient statistics, and $\theta \in \mathcal{H}$ is the canonical or natural parameter. The quantity $A(\theta)$ is called log-partition function or moment generating function. It is defined as

$$A(\theta) := \log\left(\int_{\mathcal{X}} \exp\left(\langle \theta, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}}\right) \mathrm{d}\mathbf{x}\right), \tag{2.25}$$

which has the effect of normalizing the distribution properly. Furthermore, the derivatives of $A(\theta)$ generate the moments of the random variable $k(\mathbf{x}, \cdot)$. For instance,

$$\frac{\partial A(\theta)}{\partial \theta} = \mathbb{E}_\theta\left[k(\mathbf{x}, \cdot)\right] := \int_{\mathcal{X}} k(\mathbf{x}, \cdot) \Pr_{\mathbf{x}}(\theta) \mathrm{d}\mathbf{x}, \tag{2.26}$$

$$\frac{\partial^2 A(\theta)}{\partial \theta^2} = \mathbb{V}_\theta\left[k(\mathbf{x}, \cdot)\right] := \mathbb{E}_\theta\left[k(\mathbf{x}, \cdot) \otimes k(\mathbf{x}, \cdot)\right] - \mathbb{E}_\theta\left[k(\mathbf{x}, \cdot)\right] \otimes \mathbb{E}_\theta\left[k(\mathbf{x}, \cdot)\right], \tag{2.27}$$

where $\otimes$ is the tensor product. Since the Hessian in (2.27) is the variance matrix and hence positive semi-definite, we know that $A(\theta)$ is convex in θ. Very often, $A(\theta)$ is also strictly convex: there exists no $\theta \in \mathcal{H}$ and $\theta \neq 0$ such that $\langle \theta, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}}$ equal to a constant. In this case, the representation of the exponential family model is called minimal, and each θ is uniquely associated with a distribution in the family. Based on the log-partition function, we can also define the set of valid parameters for an exponential family model

$$\Theta := \left\{\theta \in \mathcal{H} \middle| A(\theta) < \infty\right\}. \tag{2.28}$$

Similarly we can also define conditional distribution of $\mathbf{y} \in \mathcal{Y}$ given $\mathbf{x} \in \mathcal{X}$, ie. $\Pr_{\mathbf{y}|\mathbf{x}}(\theta)$. Here we use a joint kernel $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \longmapsto \mathbb{R}$ to generate the reproducing kernel Hilbert space $\mathcal{H}$ and the sufficient statistics $k((\mathbf{x}, \mathbf{y}), \cdot)$, and we have

$$\Pr_{\mathbf{y}|\mathbf{x}}(\theta) := \exp\left(\langle \theta, k((\mathbf{x}, \mathbf{y}), \cdot)\rangle_{\mathcal{H}} - A_{\mathbf{x}}(\theta)\right). \tag{2.29}$$

Note that the log-partition function in this case is an integral over the domain $\mathcal{Y}$ only, ie.

$$A_{\mathbf{x}}(\theta) := \log\left(\int_{\mathcal{Y}} \exp\left(\langle \theta, k((\mathbf{x}, \mathbf{y}), \cdot)\rangle_{\mathcal{H}}\right) \mathrm{d}\mathbf{y}\right). \tag{2.30}$$

2.3.2 Examples of Exponential Families

Many commonly used distributions, such as Bernoulli and Gaussian distribution, are from exponential families. A comprehensive list of exponential families can be found in [27]. Here we use a univariate Gaussian distribution with domain $x \in \mathbb{R}$ as an example

$$\Pr_x = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{2.31}$$

Its corresponding potential function is $\left(x, -\frac{1}{2}x^2\right)^\top$ which gives us the natural parameter $\theta = (\theta_1, \theta_2)^\top \in \mathbb{R} \times (0, +\infty)$ and the exponential family $\Pr_x(\theta)$

$$\Pr_x(\theta) = \exp\left(\left\langle \theta, \left(x, -\frac{1}{2}x^2\right)^\top \right\rangle - A(\theta)\right). \tag{2.32}$$

In this model, the natural parameter θ_1 and θ_2 will correspond to $\frac{\mu}{\sigma^2}$ and $\frac{1}{\sigma^2}$ respectively. The log-partition function can be computed as

$$A(\theta) = \log\left(\int_{-\infty}^{+\infty} \exp\left(\left\langle \theta, \left(x, -\frac{1}{2}x^2\right)^\top \right\rangle\right) \mathrm{d}x\right) = \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\theta_2) + \frac{1}{2}\frac{\theta_1^2}{\theta_2}. \tag{2.33}$$

A second example is the Bernoulli distribution, which corresponds to tossing a coin once with probability p for heads and probability $1-p$ for tails. The probability of the outcome can be modeled as

$$\Pr_x = p^x(1-p)^{1-x}, \quad x \in \{0, 1\}. \tag{2.34}$$

The exponential family presentation of (2.34) has a simple potential function x with natural parameter $\theta \in \mathbb{R}$. Thus we have

$$\Pr_x(\theta) = \exp\left(\langle \theta, x\rangle - A(\theta)\right), \tag{2.35}$$

where θ corresponds to $\log p(1-p)$ and the log-partition function is

$$A(\theta) = \log\left(\sum_{x\in\{0,1\}} \exp\left(\langle \theta, x\rangle\right)\right) = \log\left(1 + e^\theta\right). \tag{2.36}$$

2.3.3 Marginal Polytope

The marginal polytope is the image of the space of all probability distributions $\mathcal{P}$ under the expectation of a particular potential function $k(\mathbf{x}, \cdot) \in \mathcal{H}$. Formally it is defined as

$$\mathcal{M} := \left\{ \mu[\Pr_{\mathbf{x}}] \,\middle|\, \exists \Pr_{\mathbf{x}} \in \mathcal{P},\ \mu[\Pr_{\mathbf{x}}] := \int_{\mathcal{X}} k(\mathbf{x}, \cdot) \Pr_{\mathbf{x}} \mathrm{d}\mathbf{x} = \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x}, \cdot)\right] \right\}. \tag{2.37}$$

Since the space $\mathcal{P}$ is a convex set, then the marginal polytope, constructed from a linear map of $\Pr_{\mathbf{x}} \in \mathcal{P}$, is also convex. Furthermore, we also define a mapping by restricting the expectation to a family of exponential distributions with the same potential function as $k(\mathbf{x}, \cdot)$

$$\Lambda(\theta) := \int_{\mathcal{X}} k(\mathbf{x}, \cdot) \Pr_{\mathbf{x}}(\theta) \mathrm{d}\mathbf{x} = \mathbb{E}_\theta\left[k(\mathbf{x}, \cdot)\right], \tag{2.38}$$

where $\Pr_{\mathbf{x}}(\theta) = \exp\left(\langle \theta, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} - A(\theta)\right)$. Since $\Pr_{\mathbf{x}}(\theta) \in \mathcal{P}$, then $\Lambda(\theta)$ is also a member of the marginal polytope, ie. $\Lambda(\theta) \in \mathcal{M}$. Also note that $\Lambda(\theta)$ is equal to the derivative of the log partition function, ie. $\Lambda(\theta) = \partial A(\theta)/\partial\theta$ in (2.26). Furthermore, the mapping $\Lambda(\theta)$ has the following interesting properties [145].

Theorem 12 *The mapping $\Lambda(\theta) : \mathcal{H} \longmapsto \mathcal{M}$ is onto the relative interior of $\mathcal{M}$, ie. $\Lambda(\theta) = \mathrm{ri}(\mathcal{M})$.*

Theorem 13 *The mapping $\Lambda(\theta) : \mathcal{H} \longmapsto \mathcal{M}$ is one-to-one if and only if the exponential representation is minimal.*

The significance of Theorem 12 is that for any $\mu \in \mathrm{ri}(\mathcal{M})$, there exists an exponential family distribution which is the preimage of μ. Note that normally an exponential family describes only a subset of all distributions, for a given μ there may exist distributions other than a member of the exponential family which are also mapped to μ. The significance of Theorem 13 is that for each $\mu \in \mathrm{ri}(\mathcal{M})$, we can find a unique member of the exponential family as its preimage. In a sense the mapping $\mu[\Pr_{\mathbf{x}}]$ partitions the space of distributions $\mathcal{P}$ into sets of equivalent classes, with each class being indexed by a member of the exponential family and mapped to the same $\mu \in \mathcal{M}$. Due to these nice properties and its dual relation to the entropy functions, marginal polytopes have become a standard tool in deriving efficient algorithms for approximate inference in graphical models [145, 108]. We refer the reader to [145] for more details.

2.4 Statistical Stability

We would like to introduce some useful tools for studying the statistical stability of learning algorithms. It is desirable for a learning algorithm to capture statistically stable patterns from training observations and generalize to future observations. Such behavior can be quantified by the concentration of random variables characterizing the detected pattern. Basically, a random variable that is concentrated is very likely to assume values close to its expectation and values away from the expectation become exponentially unlikely. We will describe several commonly used concentration inequality in this section for our later use.

Besides concentration inequalities, we will also introduce some basic concepts of a family of statistics, called U-statistics. We can think of U-statistics as a generalization of the sample mean. Typically, U-statistics have nice consistency properties and asymptotical distributions. We will use U-statistics to design statistical tests based on our Hilbert space embedding approach for distribution analysis.

2.4.1 Concentration Inequalities

A common assumption in analyzing the stability of learning algorithms is that observations are drawn independently and identically (iid.) according to a fixed distribution. One of the best-known concentration theorem is due to McDiarmid [92].

Theorem 14 (McDiarmid [92]) *Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be independent random variables taking values in $\mathcal{X}$, and assume that $f : \mathcal{X}^m \longmapsto \mathbb{R}$ satisfies*

$$\sup_{\mathbf{x}_1,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_m;\hat{\mathbf{x}}_i} |f(\mathbf{x}_1,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_m) - f(\mathbf{x}_1,\ldots,\hat{\mathbf{x}}_i,\ldots,\mathbf{x}_m)| \leq c_i \tag{2.39}$$

for all $1 \leq i \leq m$. Then for all $\epsilon > 0$

$$\Pr\left\{f(\mathbf{x}_1,\ldots,\mathbf{x}_m) - \mathbb{E}_X\left[f(\mathbf{x}_1,\ldots,\mathbf{x}_m)\right] \geq \epsilon\right\} \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^m c_i^2}\right). \tag{2.40}$$

Basically, this theorem says that if we can upper bound the variation of a function f with respect to its parameter $\mathbf{x}_i$, then function value evaluated at a random sample will not deviate too far away from its expectation. More specifically, the probability for a deviation of ϵ decays at exponential rate, ie. $O(\exp(-\epsilon))$. Alternatively, given a small probability $\delta \in (0, 1)$, we know that with probability $1 - \delta$, the deviation of f from its expectation is bounded by

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_m) - \mathbb{E}_X \left[f(\mathbf{x}_1, \ldots, \mathbf{x}_m) \right] \leq \sqrt{\frac{\ln(1/\delta) \sum_{i=1}^m c_i^2}{2}}. \tag{2.41}$$

As a special case of McDiarmid's theorem, Hoeffding derived the concentration inequality for the mean of a set of scalar random variables [70].

Theorem 15 (Hoeffding [70]) *Let $X = \{x_1, \ldots, x_m\}$ be independent random variables satisfying $x_i \in [a, b] \subset \mathbb{R}$, and $s_m := \frac{1}{m} \sum_{i=1}^m x_i$, then it follows that*

$$\Pr \left\{ s_m - \mathbb{E}_X \left[s_m \right] \geq \epsilon \right\} \leq \exp \left(-\frac{2m\epsilon^2}{(b-a)^2} \right). \tag{2.42}$$

U-statistics, which we are to introduce in section 2.5, are generalization of the mean statistic. Therefore Theorem 15 also provides a basis for the concentration of U-statistics.

2.4.2 Rademacher Average

Note that both Theorem 14 and 15 apply only to the case of a fixed function f. Learning algorithms typically optimize over a class of functions. Therefore, we also need to characterize their stability when we are allowed to choose from a class of functions. Clearly the more flexible the function class, the more likely we will learn a spurious pattern from the observations. In other words, when we consider the stability, we need to take the complexity of the function class into account. In particular, we will introduce such a complexity measure called Rademacher average [14].

Definition 16 (Rademacher Average) *For a sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ drawn iid. from a distribution $\Pr_{\mathbf{x}}$ on $\mathcal{X}$, and a real-valued function class $\mathcal{F}$ with domain $\mathcal{X}$, the empirical Rademacher average of $\mathcal{F}$ is the random variable*

$$\hat{R}_m(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \left| \frac{2}{m} \sum_{i=1}^m \sigma_i f(\mathbf{x}_i) \right| \right], \tag{2.43}$$

where $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_m\}$ are independent uniform $\{\pm 1\}$-valued (Rademacher) random variables. The corresponding population Rademacher average of $\mathcal{F}$ is

$$R_m(\mathcal{F}) = \mathbb{E}_X \left[\hat{R}_m(\mathcal{F}) \right] = \mathbb{E}_X \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \left| \frac{2}{m} \sum_{i=1}^m \sigma_i f(\mathbf{x}_i) \right| \right]. \tag{2.44}$$

The Rademacher average measures the ability of a function class to fit random labels. For instance, for a reproducing kernel Hilbert space $\mathcal{H}$ where $\|f\|_{\mathcal{H}} \leq 1$ for all $f \in \mathcal{H}$, we have

the Rademacher average

$$
\begin{aligned}
R_m(\mathcal{H}) &= \mathbb{E}_X \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \frac{2}{m} \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right| \right] \\
&= \mathbb{E}_X \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \left\langle f, \frac{2}{m} \sum_{i=1}^{m} \sigma_i k(\mathbf{x}_i, \cdot) \right\rangle \right| \right] && (2.45) \\
&= \frac{2}{m} \mathbb{E}_X \mathbb{E}_{\boldsymbol{\sigma}} \left[\left\| \sum_{i=1}^{m} \sigma_i k(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}} \right] && (2.46) \\
&\leq \frac{2}{m} \mathbb{E}_X \left[\left(\mathbb{E}_{\boldsymbol{\sigma}} \left[\sum_{i,j=1}^{m} \sigma_i \sigma_j k(\mathbf{x}_i, \mathbf{x}_j) \right] \right)^{1/2} \right] && (2.47) \\
&= \frac{2}{m} \mathbb{E}_X \left[\sqrt{\operatorname{tr}(\mathbf{K})} \right], && (2.48)
\end{aligned}
$$

where $\mathbf{K}$ is the kernel matrix corresponding to sample X. In step 2.45, we use the reproducing properties of the kernel. In step 2.46, we use the definition of dual norm. In step 2.47, we use the concavity of the square root. In step 2.48, we use the fact that $\mathbb{E}_{\boldsymbol{\sigma}} [\sigma_i \sigma_j]$ vanishes except when $i = j$.

Intuitively, choosing a function class with low Rademacher average lowers the chance of detecting spurious pattern. Hence the stability of learning algorithms is intimately related to the Rademacher average. For instance, let f be a bounded function, then its empirical average and expected value are tied together via the following inequality [14].

Theorem 17 *Let a sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be drawn from a distribution $\Pr_{\mathbf{x}}$ on $\mathcal{X}$, and $\mathcal{F}$ be a function class mapping from $\mathcal{X}$ to $[0, 1]$. Then for all $\delta > 0$, with probability at least $1 - \delta$, every $f \in \mathcal{F}$ satisfies*

$$
\begin{aligned}
\mathbb{E}_{\mathbf{x}} [f(\mathbf{x})] &\leq \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) + R_m(\mathcal{F}) + \sqrt{\frac{\ln(2/\delta)}{2m}} && (2.49) \\
&\leq \frac{1}{m} \sum_{i=1}^{m} f(x_i) + \hat{R}_m(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}. && (2.50)
\end{aligned}
$$

We will include the proof of this theorem in this thesis, since the techniques used in the proof can help us better understand the theory we develop later. The proof is similar as it appears in [14].

Proof For a fixed function $f \in \mathcal{F}$, we have

$$
\mathbb{E}_{\mathbf{x}} [f(\mathbf{x})] \leq \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) + \sup_{h \in \mathcal{F}} \left(\mathbb{E}_{\mathbf{x}} [h(\mathbf{x})] - \frac{1}{m} \sum_{i=1}^{m} h(\mathbf{x}_i) \right). \tag{2.51}
$$

First we bound the second term in the right hand side of (2.51) using McDiarmid's inequality. Since the range of f is within $[0, 1]$, the variation of this term is at most $1/m$ if we only vary the value of a single observation. We can then apply the bound we presented in (2.41) and set $c_i = \frac{1}{m}$. Hence, given a small probability $\delta/2 \in (0, 1)$, we know that with probability

$1-\delta/2$, its deviation from the expectation is bounded by

$$\sup_{h\in\mathcal{F}}\left(\mathbb{E}_{\mathbf{x}}\left[h(\mathbf{x})\right]-\frac{1}{m}\sum_{i=1}^{m}h(\mathbf{x}_i)\right)\leq\mathbb{E}_X\left[\sup_{h\in\mathcal{F}}\left(\mathbb{E}_{\mathbf{x}}\left[h(\mathbf{x})\right]-\frac{1}{m}\sum_{i=1}^{m}h(\mathbf{x}_i)\right)\right]+\sqrt{\frac{\ln(2/\delta)}{2m}}, \tag{2.52}$$

which also means for any $f\in\mathcal{F}$

$$\mathbb{E}_{\mathbf{x}}\left[f(\mathbf{x})\right]\leq\frac{1}{m}\sum_{i=1}^{m}f(\mathbf{x}_i)+\mathbb{E}_X\left[\sup_{h\in\mathcal{F}}\left(\mathbb{E}_{\mathbf{x}}\left[h(\mathbf{x})\right]-\frac{1}{m}\sum_{i=1}^{m}h(\mathbf{x}_i)\right)\right]+\sqrt{\frac{\ln(2/\delta)}{2m}}. \tag{2.53}$$

Now we bound the middle term in (2.53) by introducing an independent ghost sample $\tilde{X}$ and Rademacher variables $\boldsymbol{\sigma}$

$$\begin{aligned}
&\mathbb{E}_X\left[\sup_{h\in\mathcal{F}}\left(\mathbb{E}_{\mathbf{x}}\left[h(\mathbf{x})\right]-\frac{1}{m}\sum_{i=1}^{m}h(\mathbf{x}_i)\right)\right]\\
&=\mathbb{E}_X\left[\sup_{h\in\mathcal{F}}\mathbb{E}_{\tilde{X}}\left[\frac{1}{m}\sum_{i=1}^{m}h(\tilde{\mathbf{x}}_i)-\frac{1}{m}\sum_{i=1}^{m}h(\mathbf{x}_i)\right]\right] &(2.54)\\
&\leq\mathbb{E}_X\mathbb{E}_{\tilde{X}}\left[\sup_{h\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^{m}\left(h(\tilde{\mathbf{x}}_i)-h(\mathbf{x}_i)\right)\right] &(2.55)\\
&=\mathbb{E}_X\mathbb{E}_{\tilde{X}}\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^{m}\sigma_i\left(h(\tilde{\mathbf{x}}_i)-h(\mathbf{x}_i)\right)\right] &(2.56)\\
&\leq 2\mathbb{E}_X\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{F}}\left|\frac{1}{m}\sum_{i=1}^{m}\sigma_i h(\mathbf{x}_i)\right|\right] &(2.57)\\
&=R_m(\mathcal{F}).
\end{aligned}$$

Note that the ghost sample $\tilde{X}$ is applied in step (2.54). In step (2.55), we use the convexity of $\sup$ function. In step (2.56), we use the fact that X and $\tilde{X}$ are samples of the same size drawn iid. from the same distribution. Hence if we randomly exchange the observations between X and $\tilde{X}$ and average over all possible exchanges, the overall expectation remains the same. This proves the result in (2.49).

Last, we apply McDiarmid's inequality again and bound the difference between $R_m(\mathcal{F})$ and $\hat{R}_m(\mathcal{F})$ for probability $\delta/2$. Then we obtain the result in (2.50). ■

2.5 U-Statistics and V-statistics

Most of the materials in this section can be found in [118]. Suppose we have a sample $X=\{\mathbf{x}_1,\ldots,\mathbf{x}_r\}$ of size r drawn iid. from a distribution $\Pr_{\mathbf{x}}$. U-statistics concern an unbiased estimation of a parameter θ of $\Pr_{\mathbf{x}}$ using X. Suppose there is some function $h(\mathbf{x}_1,\ldots,\mathbf{x}_r)$ which is an unbiased estimator of θ, ie.

$$\theta=\mathbb{E}_X\left[h(\mathbf{x}_1,\ldots,\mathbf{x}_r)\right]. \tag{2.58}$$

h is called a kernel of the estimator.[1] For a kernel h that is not symmetric in its arguments, we can always make it symmetric by the following average

$$\frac{1}{r!}\sum_{\pi^r} h(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_r}), \tag{2.59}$$

where the summation ranges over all permutations π^r of $(1,\ldots,r)$. Furthermore, the smallest integer r for which (2.58) holds is called the degree of θ.

When we have a sample $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_m\}$ of size m larger than r, we can then construct a U-statistic in the following way.

Definition 18 (U-statistic) *Given a kernel h of degree r and a sample X of size $m \geq r$, the corresponding U-statistic for estimation of θ is obtained by averaging the kernel h symmetrically over the observations*

$$U := \frac{1}{(m)_r}\sum_{\mathbf{i}_r^m} h(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_r}), \tag{2.60}$$

where the summation ranges over r indices drawn without replacement from $(1,\ldots,m)$, and $(m)_r$ is the Pochhammer symbol

$$(m)_r := \frac{m!}{(m-r)!} = r!\binom{m}{r}. \tag{2.61}$$

Note that in this definition, we do not require the kernel h to be symmetric in its arguments. The symmetrization is absorbed into the overall summation. If the kernel h is already symmetric, the U-statistics can be equivalently written as

$$U = \binom{m}{r}^{-1}\sum h(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_r}), \tag{2.62}$$

where the summation now is only over all combination of indices drawn from $(1,\ldots,m)$. In this case we can drop the normalization by $r!$ due to the symmetrization of the kernel h. Corresponding to a U-statistic, there is the following associated V-statistic.

Definition 19 (V-statistic) *Given a kernel h of degree r and a sample X of size $m \geq r$, the V-statistic is*

$$V := \frac{1}{m^r}\sum_{i_1=1}^{m}\cdots\sum_{i_r=1}^{m} h(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_r}). \tag{2.63}$$

The difference in a V-statistic is that the summation here is over all indices drawn *with* replacement from $(1,\ldots,m)$. V-statics usually lead to biased estimator of θ.

Many well-known statistics can be expressed as U-statistics. The first example is the mean parameter μ of a scalar variable x with distribution $\Pr_x$. In this case, $\mu = \int x \mathrm{d}\Pr_x$ has a

[1] Note that the kernels for the U-statistics are different from the kernels we used in kernel methods.

kernel $h(x) = x$, and the corresponding U-statistic is

$$U = \frac{1}{m}\sum_{i=1}^{m} x_i = \bar{x}. \tag{2.64}$$

Another example is the variance, ie. $\sigma^2 = \int (x-\mu)^2 \mathrm{d}\Pr_x$. In this case, the kernel is $h(x_1, x_2) = \frac{x_1^2 + x_2^2 - 2x_1x_2}{2}$ and the corresponding U-statistic is

$$U = \frac{2}{m(m-1)} \sum_{1 \le i < j \le m} h(x_i, x_j) = \frac{1}{m-1}\left(\sum_{i=1}^{m} x_i^2 - m\bar{x}^2\right) = s^2. \tag{2.65}$$

While the corresponding V-statistic is the biased estimator of the variance, ie.

$$V = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} h(x_i, x_j) = \frac{1}{m}\left(\sum_{i=1}^{m} x_i^2 - m\bar{x}^2\right). \tag{2.66}$$

As we mentioned in section 2.4.1, U-statistics are generalization of the mean statistic. Therefore, we can also derive the following concentration inequality for U-statistics similarly to Theorem 15 [70].

Theorem 20 (Hoeffding [70]) *Let U be a U-statistic as defined in equation (2.60) with kernel h of degree r. Let X be a sample of size $m \ge r$. If $h \in [a, b]$, then for all $\epsilon > 0$*

$$\Pr\left\{U - \mathbb{E}_X[U] \ge \epsilon\right\} \le \exp\left(-\frac{2\lfloor m/r \rfloor \epsilon^2}{(b-a)^2}\right). \tag{2.67}$$

Another property of a U-statistic is that it has minimum variance [118].

Theorem 21 *Let S be an unbiased estimator of θ based on a sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ drawn iid. from the distribution $\Pr_{\mathbf{x}}$. Then the corresponding U-statistic is unbiased and has minimum variance, ie.*

$$\mathbb{V}[U] \le \mathbb{V}[S]. \tag{2.68}$$

We can further characterize the variance of a U-statistic. Suppose that the kernel h satisfying $\mathbb{E}_X\left[h^2(X)\right] < \infty$. For $1 \le c \le r$, we define

$$h_c(\mathbf{x}_1, \ldots, \mathbf{x}_c) = \mathbb{E}_{\mathbf{x}_{c+1}} \cdots \mathbb{E}_{\mathbf{x}_r}\left[h(\mathbf{x}_1, \ldots, \mathbf{x}_c, \mathbf{x}_{c+1}, \ldots, \mathbf{x}_r)\right] \tag{2.69}$$

by taking the expectation of h with respect to the last $r - c$ variables. Furthermore, define $\zeta_0 = 0$, and for $1 \le c \le r$, define

$$\zeta_c = \mathbb{V}\left[h_c(\mathbf{x}_1, \ldots, \mathbf{x}_c)\right]. \tag{2.70}$$

Then we have the following characterization of the variance of the U-statistics.

Lemma 22 *The variance of U is given by*

$$\mathbb{V}[U] = \binom{m}{r}^{-1} \sum_{c=1}^{r} \binom{r}{c} \binom{m-r}{r-c} \zeta_c. \tag{2.71}$$

With this characterization of the variance, we can examine the asymptotic distribution of a U-statistic. Depending on whether ζ_1 is vanishing or not, the asymptotic distribution is different. For $\zeta_1 > 0$, the asymptotic distribution is a Gaussian; while for $\zeta_1 = 0$, the asymptotic distribution is an infinite sum of $\chi^2(1)$ random variables. These results are stated formally in the following two theorems [118].

Theorem 23 *If* $\mathbb{E}\left[h^2\right] < \infty$ *and* $\zeta_1 > 0$*, then* U *is asymptotically normal with mean* θ *and variance* $\frac{r^2}{m}\zeta_1$*, i.e.*

$$(U - \theta) \longrightarrow \mathcal{N}(0, \frac{r^2}{m}\zeta_1) \quad \textit{a.s.} \quad m \longrightarrow \infty. \tag{2.72}$$

Theorem 24 *If* $\mathbb{E}\left[h^2\right] < \infty$ *and* $\zeta_1 = 0 < \zeta_2$*, then*

$$(U - \theta) \longrightarrow \frac{r(r-1)}{2m} \sum_{i=1}^{\infty} \lambda_i(\chi_i^2(1) - 1) \quad \textit{a.s.} \quad m \longrightarrow \infty. \tag{2.73}$$

where $\chi_i^2(1)$ *are iid.* $\chi^2(1)$ *random variables, and* λ_i *are the solution to the eigenvalue problem*

$$\int_{\mathcal{X}} \left(h_2(\mathbf{x}, \mathbf{x}') - \theta\right) \psi_i(\mathbf{x}) \mathrm{d}\Pr_{\mathbf{x}'} = \lambda_i \psi_i(\mathbf{x}) \tag{2.74}$$

and ψ_i *are the corresponding eigenfunctions.*

In practice, the infinite sum of the $\chi^2(1)$ is hard to compute. In this case, the distribution can be approximated via resampling the observations, fitting Pearson curves [58], or fitting a Gamma distribution [60].

CHAPTER 3

Hilbert Space Embedding of Distributions

In this chapter, we will explain a framework for distribution analysis via the mean map. The mean map has attractive properties such as injectivity and fast sample convergence. Furthermore, learning via the mean map also leads to interesting new algorithms.

3.1 Introduction

Kernel methods have become widespread and have gained popularity in the context of supervised learning [115, 107]. However, in the context of testing, estimation, and analysis of probability distributions, information theoretic approaches have long been dominant [30, 4, 123, 99, 83]. Their recent applications include, for instance, construction of graphical models [83], feature extraction [123], and independent component analysis [132]. These methods, however, have by and large a common issue: to estimate quantities such as the mutual information, entropy, or Kullback-Leibler divergence, sophisticated space partitioning and/or bias correction strategies are required [99, 132].

In this section we will introduce kernel methods for estimating distances between distributions *without* the need for intermediate density estimation. Moreover, they allow algorithm designers to specify which properties of a distribution are most relevant to their problems by using kernels. Very often this embedding approach to distribution representation and analysis leads to algorithms which are simpler and more effective than information theoretic methods in a broad range of applications.

3.2 Mean Map

Suppose we have a kernel $k : \mathcal{X} \times \mathcal{X} \longmapsto \mathbb{R}$ with associated reproducing kernel Hilbert space $\mathcal{H}$. At the heart of our approach are the following two mappings

$$\mu[\Pr_{\mathbf{x}}] := \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x}, \cdot)\right] \tag{3.1}$$

and its empirical estimate

$$\mu[X] := \frac{1}{m}\sum_{i=1}^{m} k(\mathbf{x}_i, \cdot). \tag{3.2}$$

Here $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is assumed to be drawn independently and identically distributed (iid.) from $\Pr_{\mathbf{x}}$. If the (sufficient) condition $\mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x}, \mathbf{x})\right] < \infty$ is satisfied, then $\mu[\Pr_{\mathbf{x}}]$ is an element of the Hilbert space (as is, in any case, $\mu[X]$). By virtue of the reproducing property

of $\mathcal{H}$, we have

$$\langle f, \mu[\Pr_{\mathbf{x}}] \rangle = \mathbb{E}_{\mathbf{x}} [f(\mathbf{x})] \text{ and} \tag{3.3}$$

$$\langle f, \mu[X] \rangle = \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i). \tag{3.4}$$

That is, we can compute expectations and empirical means of f with respect to $\Pr_{\mathbf{x}}$ and X, respectively, by taking its inner products with $\mu[\Pr_{\mathbf{x}}]$ and $\mu[X]$ in the RKHS.

These two mappings embed distributions and their samples into the RKHS. Then natural questions arise: what is the advantage of this approach to distribution representation? what is the finite sample convergence of the mean map to its population counterpart? and how can we use this approach in the context of machine learning? We will answer these questions in the following sections.

3.2.1 Properties of Mean Map

The representations $\mu[\Pr_{\mathbf{x}}]$ and $\mu[X]$ are attractive for the following reasons [49, 58].

Theorem 25 *If the kernel k is universal, then the mean map $\mu : \Pr_{\mathbf{x}} \longmapsto \mu[\Pr_{\mathbf{x}}]$ is one-to-one.*

A formal proof for this theorem can be found in [58]. Here we will present some intuitive arguments instead. Basically the proof builds upon the connection between the space of continuous functions and the reproducing kernel Hilbert spaces of universal kernels. It contains three key ingredients: first, the space of continuous functions separate all distributions [43, Lemma 9.3.2]

Lemma 26 *Let $\Pr_{\mathbf{x}}$ and $\Pr_{\tilde{\mathbf{x}}}$ be two probability distribution defined on $\mathcal{X}$. Then $\Pr_{\mathbf{x}} = \Pr_{\tilde{\mathbf{x}}}$ if and only if $\mathbb{E}_{\mathbf{x}} [f(\mathbf{x})] = \mathbb{E}_{\tilde{\mathbf{x}}} [f(\tilde{\mathbf{x}})]$ for all $f \in C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of continuous bounded functions on $\mathcal{X}$.*

Second, this result is translated into one in the reproducing kernel Hilbert spaces $\mathcal{H}$. To do this, we express Lemma 26 as the following form

$$0 \equiv \mathbb{E}_{\mathbf{x}} [f(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}} [f(\tilde{\mathbf{x}})], \quad \forall f \in C(\mathcal{X}) \tag{3.5}$$

Using the reproducing properties of $\mathcal{H}$ in (3.3) and the fact that $\mathcal{H}$ is dense in $C(\mathcal{X})$, we have

$$0 \equiv \langle f, \mathbb{E}_{\mathbf{x}} [k(\mathbf{x}, \cdot)] - \mathbb{E}_{\tilde{\mathbf{x}}} [k(\tilde{\mathbf{x}}, \cdot)] \rangle, \quad \forall f \in \mathcal{H} \tag{3.6}$$

Third, since the optimum is identically zero for all $f \in \mathcal{H}$, we know that the two mean maps $\mathbb{E}_{\mathbf{x}} [k(\mathbf{x}, \cdot)]$ and $\mathbb{E}_{\tilde{\mathbf{x}}} [k(\tilde{\mathbf{x}}, \cdot)]$ have to be identical if and only if $\Pr_{\mathbf{x}} = \Pr_{\tilde{\mathbf{x}}}$.

Besides the one-to-one mapping of the mean map, we also have fast convergence of empirical mean map $\mu[X]$ to its population quantity [3, Theorem 15]. Denote by B_m the Rademacher average [14] associated with $\Pr_{\mathbf{x}}$ and $\|f\|_{\mathcal{H}} \leq 1$ via

$$B_m := \mathbb{E}_X \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \frac{2}{m} \sum_{i=1}^{m} \sigma_i f(\mathbf{x}_i) \right| \right]. \tag{3.7}$$

The following theorem ensures that $\mu[X]$ is a good proxy for $\mu[\Pr_{\mathbf{x}}]$, provided the Rademacher average is well behaved [3].

Theorem 27 *Assume that $f \in [0,1]$ for all $f \in \mathcal{H}$ with $\|f\|_{\mathcal{H}} \leq 1$. Given a sample $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ drawn iid. from the distribution $\Pr_{\mathbf{x}}$, with probability at least $1-\delta$, we have*

$$\|\mu[\Pr_{\mathbf{x}}] - \mu[X]\|_{\mathcal{H}} \leq \frac{2}{m}\mathbb{E}_{\mathbf{x}}\left[\sqrt{\operatorname{tr}\mathbf{K}}\right] + \sqrt{\frac{\ln(2/\delta)}{2m}}. \tag{3.8}$$

Proof The proof relies on a dual relation between the norm and the following optimization problem

$$\begin{aligned}\sup_{\|f\|_{\mathcal{H}}\leq 1} \mathbb{E}_{\mathbf{x}}\left[f(\mathbf{x})\right] - \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}_i) &= \sup_{\|f\|_{\mathcal{H}}\leq 1} \langle f, \mu[\Pr_{\mathbf{x}}] - \mu[X]\rangle \\ &= \|\mu[\Pr_{\mathbf{x}}] - \mu[X]\|_{\mathcal{H}}.\end{aligned} \tag{3.9}$$

While for (3.9), we use the bound presented in Theorem 17. Suppose the optimal solution of (3.9) is denoted as f^*, then we have

$$\begin{aligned}\sup_{\|f\|_{\mathcal{H}}\leq 1} \mathbb{E}_{\mathbf{x}}\left[f(\mathbf{x})\right] - \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}_i) &= \mathbb{E}_{\mathbf{x}}\left[f^*(\mathbf{x})\right] - \frac{1}{m}\sum_{i=1}^{m} f^*(\mathbf{x}_i) \\ &\leq B_m + \sqrt{\frac{\log(2/\delta)}{2m}} = \frac{2}{m}\mathbb{E}_{\mathbf{x}}\left[\sqrt{\operatorname{tr}\mathbf{K}}\right] + \sqrt{\frac{\log(2/\delta)}{2m}}.\end{aligned} \tag{3.10}$$

■

Note that there is a strong connection between Theorem 27 and uniform convergence results commonly used in Statistical Learning Theory [82, 58]. This is captured in the theorem below

Theorem 28 *Let $\mathcal{F}$ be a unit ball in the reproducing kernel Hilbert space $\mathcal{H}$, and $\mathbf{x}_1, \ldots, \mathbf{x}_m$ be drawn iid. from $\Pr_{\mathbf{x}}$. Then the deviation between empirical means and expectations for any $f \in \mathcal{F}$ is bounded*

$$\sup_{f\in\mathcal{F}}\left|\mathbb{E}_{\mathbf{x}}\left[f(\mathbf{x})\right] - \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}_i)\right| = \|\mu[\Pr_{\mathbf{x}}] - \mu[X]\|_{\mathcal{H}}.$$

Bounding the probability that this deviation exceeds some threshold ϵ is one of the key problems of statistical learning theory. This means that we have at our disposition a large range of tools typically used to assess the quality of estimators. The key difference is that while those bounds are typically used to bound the deviation between empirical and expected means under the assumption that the data *are* drawn from the same distribution, we will use the bounds to motivate new learning algorithms. For instance, we can use it to test whether two distributions are the same in Section 3.3.1, and in Sections 3.3.2 and 3.3.4 to motivate strategies for approximating particular distributions.

This is analogous to what is commonly done in the univariate case: the Glivenko-Cantelli lemma allows one to bound deviations between empirical and expected means for functions of bounded variation, as generalized by the work of Vapnik and Chervonenkis [143]. However, the Glivenko-Cantelli lemma also leads to the Kolmogorov-Smirnov statistic comparing distributions by comparing their cumulative distribution functions.

3.2.2 Distance between Distributions

There are many notions of distance between distributions in the statistical literature (eg. [106, 54]). However, in the machine learning community, the most commonly used measure is the Kullback-Leibler (KL) divergence [30]. The KL divergence between two distributions $\Pr_{\mathbf{x}}$ and $\Pr_{\tilde{\mathbf{x}}}$ is the expected value of their log ratio with respect to $\Pr_{\mathbf{x}}$, ie.

$$\mathrm{KL}\left(\Pr_{\mathbf{x}} \parallel \Pr_{\tilde{\mathbf{x}}}\right) := \int_{\mathcal{X}} \log\left(\frac{\mathrm{d}\Pr_{\mathbf{x}}}{\mathrm{d}\Pr_{\tilde{\mathbf{x}}}}\right) \mathrm{d}\Pr_{\mathbf{x}} . \tag{3.11}$$

Note that KL divergence is not a metric between distributions, since it is not symmetric in its arguments. Exchanging the order of the arguments results in different values of the KL divergence. Another drawback of KL divergence is that either parametric or nonparametric density estimation has to be performed before its value can be estimated. Recent efforts on estimating the KL divergence have been focused on sophisticated space partitioning and/or biased correction strategies (eg. [99, 103]).

Here we define a distance measure between distributions $\Pr_{\mathbf{x}}$ and $\Pr_{\tilde{\mathbf{x}}}$, simply by letting

$$D(\Pr_{\mathbf{x}}, \Pr_{\tilde{\mathbf{x}}}) := \|\mu[\Pr_{\mathbf{x}}] - \mu[\Pr_{\tilde{\mathbf{x}}}]\|_{\mathcal{H}} . \tag{3.12}$$

This measure is not new although it has been derived in different ways and comes under different names (eg. integral probability metric and probability metric with ζ-structure [159, 97, 96]). The novelty of our definition is that we derive the distance from the view of the Hilbert space embedding of distributions. This new view also allows us to study further properties of the measure and its applications in machine learning which have not been previously explored.

Theorem 27 tells us that we do not need to have access to actual distributions in order to compute $D(\Pr_{\mathbf{x}}, \Pr_{\tilde{\mathbf{x}}})$ approximately — as long as $B_m = O(m^{-\frac{1}{2}})$, a finite sample from the distributions will yield error of $O(m^{-\frac{1}{2}})$. The following theorem shows that the deviation between the empirical estimate denoted as $D(X, \tilde{X})$ and the population quantity $D(\Pr_{\mathbf{x}}, \Pr_{\tilde{\mathbf{x}}})$ is bounded.

Theorem 29 *Let X and $\tilde{X}$ be samples of size m drawn from distribution $\Pr_{\mathbf{x}}$ and $\Pr_{\tilde{\mathbf{x}}}$ respectively. Then for all $\delta \in (0, 1)$, with probability at least $1 - \delta$, we have*

$$D(\Pr_{\mathbf{x}}, \Pr_{\tilde{\mathbf{x}}}) \leq D(X, \tilde{X}) + R_m(\mathcal{H}, \Pr_{\mathbf{x}}) + R_m(\mathcal{H}, \Pr_{\tilde{\mathbf{x}}}) + 2\sqrt{\frac{\ln(2/\delta)}{2m}} \tag{3.13}$$

$$\leq D(X, \tilde{X}) + \frac{2}{m}\mathbb{E}_X\left[\sqrt{\mathrm{tr}\,(\mathbf{K})}\right] + \frac{2}{m}\mathbb{E}_{\tilde{X}}\left[\sqrt{\mathrm{tr}(\tilde{\mathbf{K}})}\right] + 2\sqrt{\frac{\ln(2/\delta)}{2m}}. \tag{3.14}$$

Proof First we simply make use of the definition of $D(\Pr_{\mathbf{x}}, \Pr_{\tilde{\mathbf{x}}})$ and the triangle inequality

$$\|\mu[\Pr_{\mathbf{x}}] - \mu[\Pr_{\tilde{\mathbf{x}}}]\|_{\mathcal{H}} \leq \|\mu[\Pr_{\mathbf{x}}] - \mu[X]\|_{\mathcal{H}} + \left\|\mu[X] - \mu[\tilde{X}]\right\|_{\mathcal{H}} + \left\|\mu[\Pr_{\tilde{\mathbf{x}}}] - \mu[\tilde{X}]\right\|_{\mathcal{H}} . \tag{3.15}$$

Then we apply the bound in Theorem 17 and we prove inequality (3.13) in the theorem. For the second part, we apply bound on the Rachemacher average as presented in (2.48). This completes the proof. ■

The above characterization of $D(\Pr_{\mathbf{x}}, \Pr_{\tilde{\mathbf{x}}})$ allows us to use it as a drop-in replacement wherever information theoretic quantities would have been used instead, e.g. for the purpose of determining whether two sets of observations have been drawn from the same distribution. The advantage is that $D(X, \tilde{X})$ requires no density estimation and it is a true distance measure. The latter advantage follows readily from the fact we are using the RKHS distance to define the measure.

3.2.3 Choosing the Hilbert Space

Identifying probability distributions with elements of Hilbert spaces is not new: see e.g. [68]. However, this leaves the question of which Hilbert space to employ. We could informally choose a space with a kernel equal to a Delta function $k(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x}, \mathbf{x}')$, in which case the operator μ would simply be the identity map (which restricts us to probability distributions with square integrable densities).

The latter is in fact what is commonly done on finite domains (hence the L_2 integrability condition is trivially satisfied). For instance, [145] effectively use the Kronecker Delta $\delta(\mathbf{x}, \mathbf{x}')$ as their feature map. The use of kernels has additional advantages: we need not deal with the issue of representation of the sufficient statistics or whether such a representation is minimal (i.e. whether the sufficient statistics actually span the space).

Whenever we have knowledge about the class of functions $\mathcal{F}$ we would like to analyze, we should be able to trade off simplicity in $\mathcal{F}$ with better approximation behavior in $\mathcal{P}$. For instance, assume that $\mathcal{F}$ contains only linear functions, then we only need to use linear kernels for the analysis. In this case, μ maps $\mathcal{P}$ into the space of all expectations of $k(\mathbf{x}, \cdot) = \langle \mathbf{x}, \cdot \rangle$. Consequently, one may expect better convergence (in terms of the constants in the deviation bound) of $\mu[X]$ to $\mu[\Pr_{\mathbf{x}}]$.

3.3 Learning via Mean Map

While the previous description may be of interest on its own, it is in application to areas of statistical machine learning that its relevance becomes apparent. In the following sections, we will provide overviews over several learning algorithms based on Hilbert space embedding of distributions. These algorithms include two-sample test, covariate shift correction, set kernels and density estimation. Furthermore, we will use this embedding approach for distribution analysis to design a new measure of statistical dependence in the next section. This new dependence measure allows us to provide a further framework for various supervised and unsupervised learning problems.

3.3.1 Two-Sample Test

Since we know that $\mu[X] \to \mu[\Pr_{\mathbf{x}}]$ with a fast rate (given appropriate behavior of B_m), we may compare data drawn from two distributions $\Pr_{\mathbf{x}}$ and $\Pr_{\mathbf{y}}$, with associated samples X and Y, to test whether both distributions are identical; that is, whether $\Pr_{\mathbf{x}} = \Pr_{\mathbf{y}}$. For this purpose, recall that we defined $D(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}}) = \|\mu[\Pr_{\mathbf{x}}] - \mu[\Pr_{\mathbf{y}}]\|_{\mathcal{H}}$. Using the reproducing property of an RKHS we may show [58] that

$$D^2(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}}) = \mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[k(\mathbf{x}, \mathbf{x}')\right] - 2\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[k(\mathbf{x}, \mathbf{y})\right] + \mathbb{E}_{\mathbf{y},\mathbf{y}'}\left[k(\mathbf{y}, \mathbf{y}')\right], \tag{3.16}$$

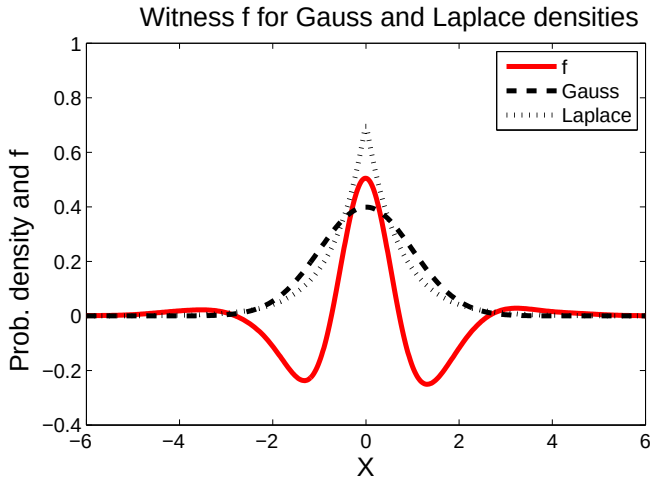


Figure 3.1: Illustration of the function maximizing the mean discrepancy in the case where a Gaussian is being compared with a Laplace distribution. Both distributions have zero mean and unit variance. The function f that witnesses the difference in feature means has been scaled for plotting purposes, and was computed empirically on the basis of 2×10^4 samples, using a Gaussian kernel with $\sigma = 0.5$.

where $\mathbf{x}'$ is an independent copy of $\mathbf{x}$, and $\mathbf{y}'$ an independent copy of $\mathbf{y}$. An unbiased empirical estimator of $D^2(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}})$ can be written as a U-statistic [118],

$$\hat{D}^2(X, Y) := \frac{1}{m(m-1)} \sum_{i \neq j} h((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j)), \tag{3.17}$$

where the kernel of the U-statistic is defined as

$$h((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) := k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{y}') - k(\mathbf{y}, \mathbf{x}') + k(\mathbf{y}, \mathbf{y}'). \tag{3.18}$$

An equivalent interpretation, also in [58], is that we find a function that maximizes the difference in expectations between probability distributions. The resulting problem may be written

$$D(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}}) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{y}}[f(\mathbf{y})]. \tag{3.19}$$

This latter setting in (3.19) lends $D(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}})$ the name maximum mean discrepancy (MMD). Furthermore, the maximizer f (witness function) provides us extra insights on how the difference between $\Pr_{\mathbf{x}}$ and $\Pr_{\mathbf{y}}$ is detected. To illustrate this, we plot the witness function f in Figure 3.1, when $\Pr_{\mathbf{x}}$ is Gaussian and $\Pr_{\mathbf{y}}$ is Laplace, using a Gaussian kernel. This function is straightforward to obtain, since the solution to Eq. (3.19) can be written $f(x) = \langle \mu[\Pr_{\mathbf{x}}] - \mu[\Pr_{\mathbf{y}}], k(x, \cdot) \rangle$.

The following two theorems give uniform convergence and asymptotic results, respectively. The first theorem is a straightforward application of the concentration inequality for U-statistics in Theorem 20 [70, p. 25].

Theorem 30 *Assume that the kernel k is nonnegative and bounded by 1. Then with probability*

at least $1 - \delta$ *the deviation*

$$|D^2(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}}) - \hat{D}^2(X, Y)| \leq 4\sqrt{\log(2/\delta)/m}. \tag{3.20}$$

Proof In $\hat{D}^2(X, Y)$, the kernel h for the U-statistic has a degree $r = 2$ and h is bounded between $[-2, 2]$. Plug these conditions into Theorem 20 we prove the claim. ■

Note that an alternative uniform convergence bound is provided in [58], based on McDiarmid's inequality [92]. The second theorem appeared as [58, Theorem 8], and describes the asymptotic distribution of $\hat{D}^2(X, Y)$. When $\Pr_{\mathbf{x}} \neq \Pr_{\mathbf{y}}$, this distribution is given by [118, Section 5.5.1]; when $\Pr_{\mathbf{x}} = \Pr_{\mathbf{y}}$, it follows from [118, Section 5.5.2] and [7, Appendix].

Theorem 31 *We assume* $\mathbb{E}\left[h^2\right] < \infty$. *When* $\Pr_{\mathbf{x}} \neq \Pr_{\mathbf{y}}$, $\hat{D}^2(X, Y)$ *converges in distribution to a Gaussian according to*

$$\left(\hat{D}^2(X, Y) - D^2(\Pr_{\mathbf{x}}, \Pr_{\mathbf{y}})\right) \longrightarrow \mathcal{N}\left(0, \frac{\sigma_u^2}{m}\right) \quad \text{a.s.} \quad m \longrightarrow \infty, \tag{3.21}$$

where $\sigma_u^2 = 4\left(\mathbb{E}_{\mathbf{z}}\left[\left(\mathbb{E}_{\mathbf{z}'}\left[h(\mathbf{z}, \mathbf{z}')\right]\right)^2\right] - \left(\mathbb{E}_{\mathbf{z},\mathbf{z}'}\left[h(\mathbf{z}, \mathbf{z}')\right]\right)^2\right)$ *and* $\mathbf{z} := (\mathbf{x}, \mathbf{y})$, *uniformly at rate* $1/\sqrt{m}$. *When* $\Pr_{\mathbf{x}} = \Pr_{\mathbf{y}}$, *the U-statistic is degenerate, meaning* $\mathbb{E}_{\mathbf{z}'}\left[h(\mathbf{z}, \mathbf{z}')\right] = 0$. *In this case,* $\hat{D}^2(X, Y)$ *converges in distribution according to*

$$\hat{D}^2(X, Y) \longrightarrow \frac{1}{m}\sum_{i=1}^{\infty} \lambda_i \left(g_i^2 - 2\right) \quad \text{a.s.} \quad m \longrightarrow \infty, \tag{3.22}$$

where $g_i \sim \mathcal{N}(0, 2)$ *i.i.d.,* λ_i *are the solutions to the eigenvalue equation*

$$\int_{\mathcal{X}} \tilde{k}(\mathbf{x}, \mathbf{x}')\psi_i(\mathbf{x}) \mathrm{d}\Pr_{\mathbf{x}} = \lambda_i \psi_i(\mathbf{x}'), \tag{3.23}$$

and $\tilde{k}(\mathbf{x}_i, \mathbf{x}_j) := k(\mathbf{x}_i, \mathbf{x}_j) - \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x}_i, \mathbf{x})\right] - \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x}, \mathbf{x}_j)\right] + \mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[k(\mathbf{x}, \mathbf{x}')\right]$ *is the centered RKHS kernel.*

We illustrate the MMD density by approximating it empirically for both $\Pr_{\mathbf{x}} = \Pr_{\mathbf{y}}$ (also called the null hypothesis, or H_0) and $\Pr_{\mathbf{x}} \neq \Pr_{\mathbf{y}}$ (the alternative hypothesis, or H_1). Results are plotted in Figure 3.2. We may use this theorem directly to test whether two distributions are identical, given an appropriate finite sample approximation to the $(1 - \alpha)$th quantile of (3.22). In [58], this was achieved via two strategies: by using the bootstrap [10], and by fitting Pearson curves using the first four moments [76, Section 18.8].

While uniform convergence bounds have the theoretical appeal of making no assumptions on the distributions, they produce very weak tests. We find the test arising from Theorem 31 performs considerably better in practice. In addition, [19] demonstrates that this test performs very well in circumstances of high dimension and low sample size (i.e. when comparing microarray data), as well as being the only test currently applicable for structured data such as distributions on graphs. Moreover, the test can be used to determine whether records in databases may be matched based on their statistical properties. In a later chapter of this thesis, we will also apply it to extract features with the aim of *maximizing* discrepancy between sets of observations.

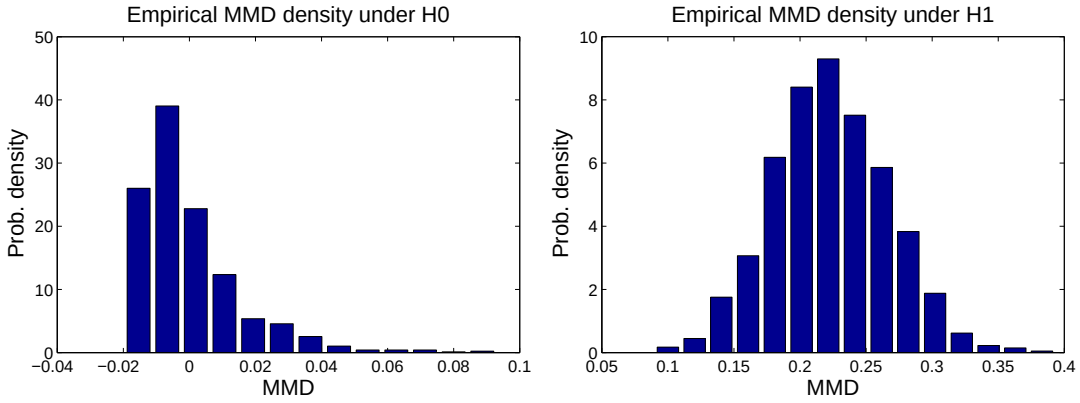


Figure 3.2: Left: Empirical distribution of the MMD under H_0, with $\Pr_{\mathbf{x}}$ and $\Pr_{\mathbf{y}}$ both Gaussians with unit standard deviation, using 50 samples from each. **Right:** Empirical distribution of the MMD under H_1, with $\Pr_{\mathbf{x}}$ a Laplace distribution with unit standard deviation, and $\Pr_{\mathbf{y}}$ a Laplace distribution with standard deviation $3\sqrt{2}$, using 100 samples from each. In both cases, the histograms were obtained by computing 2000 independent instances of the MMD.

3.3.2 Covariate Shift Correction and Local Learning

A second application of the mean operator arises in situations of supervised learning where the training and test sets are drawn from different distributions, i.e. $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is drawn from $\Pr_{\mathbf{x}}$ and $X' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{m'}\}$ is drawn from $\Pr_{\mathbf{x}'}$. We assume, however, that the labels $y \in \mathbb{R}$ are drawn from the same *conditional* distribution $\Pr_{y|\mathbf{x}}$ on both the training and test sets. The goal in this case is to find a weighting of the training set such that minimizing a reweighted empirical error on the training set will come close to minimizing the expected loss on the test set. That is, we would like to find weights $(\beta_1, \ldots, \beta_m)^\top$ for X with $\sum_i \beta_i = 1$.

If $\Pr_{y|\mathbf{x}}$ is a rapidly changing function of x, or if the loss measuring the discrepancy between y and its estimate is highly non-smooth, this problem is difficult to solve. However, under regularity conditions spelled out in [73], one may show that by minimizing

$$\Delta := \left\| \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \cdot) - \mu[X'] \right\|_{\mathcal{H}} \tag{3.24}$$

subject to $\beta_i \geq 0$ and $\sum_i \beta_i = 1$, we will obtain weights which achieve this task. The idea here is that the expected loss with the expectation taken over $y|\mathbf{x}$ should not change too quickly as a function of $\mathbf{x}$. In this case we can use points $\mathbf{x}_i$ "nearby" to estimate the loss at location $\mathbf{x}'_j$ on the test set. Hence we are re-weighting the empirical distribution on the training set X such that the distribution behaves more like the empirical distribution on X'.

Note that by re-weighting X we will assign some observations a higher weight than $\frac{1}{m}$. This means that the statistical guarantees can no longer be stated in terms of the sample size m. One may show [73], however, that $\|\boldsymbol{\beta}\|_2^{-2}$ now behaves like the effective sample size. Instead of minimizing Δ, it pays to minimize $\Delta^2 + \lambda \|\boldsymbol{\beta}\|_2^2$ subject to the above constraints. It is easy to show using the reproducing property of $\mathcal{H}$ that this corresponds to the following quadratic

program

$$\begin{aligned} &\underset{\boldsymbol{\beta}}{\text{minimize}}\ \frac{1}{2}\boldsymbol{\beta}^\top (\mathbf{K} + \lambda \mathbf{I})\,\boldsymbol{\beta} - \boldsymbol{\beta}^\top \mathbf{l} \\ &\text{subject to } \beta_i \geq 0 \text{ and } \sum_i \beta_i = 1. \end{aligned} \tag{3.25}$$

Here $\mathbf{K}_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ denotes the kernel matrix and $\mathbf{l}_i := \frac{1}{m'} \sum_{j=1}^{m'} k(\mathbf{x}_i, \mathbf{x}'_j)$ is the expected value of $k(\mathbf{x}_i, \cdot)$ on the test set X', i.e. $\mathbf{l}_i = \langle k(\mathbf{x}_i, \cdot), \mu[X'] \rangle$.

Experiments show that solving (3.25) leads to sample weights which perform very well in covariate shift. Remarkably, the approach can even outperform "importance sampler" weights, i.e. weights β_i obtained by computing the ratio $\beta_i = \Pr_{\mathbf{x}'}(\mathbf{x}_i) / \Pr_{\mathbf{x}}(\mathbf{x}_i)$. This is surprising, since the latter provide unbiased estimates of the expected error on X'.

In the case where X' contains only a single observation, i.e. $X' = \{\mathbf{x}'\}$, the above procedure leads to estimates which try to find a subset of observations in X and a weighting scheme such that the error at $\mathbf{x}'$ is approximated well. In practice, this leads to a local sample weighting scheme, and consequently an algorithm for local learning [20]. The key advantage here, however, is that we do not need to define the shape of the neighborhood in which we approximate the error at $\mathbf{x}'$. Instead, this is automatically taken care of via the choice of the Hilbert space $\mathcal{H}$ and the location of $\mathbf{x}'$ relative to X.

3.3.3 Kernels on Sets

Up to now we used the mapping $X \longmapsto \mu[X]$ to estimate the distance between two distributions (or their samples). However, since $\mu[X]$ itself is an element of an RKHS we can define a kernel on sets of objects, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $X' = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_{m'}\}$, directly via

$$k(X, X') := \langle \mu[X], \mu[X'] \rangle = \frac{1}{mm'} \sum_{i,j}^{m,m'} k(\mathbf{x}_i, \mathbf{x}'_j). \tag{3.26}$$

In other words, $k(X, X')$, and by analogy $k(\Pr_{\mathbf{x}}, \Pr_{\mathbf{x}'}) := \langle \mu[\Pr_{\mathbf{x}}], \mu[\Pr_{\mathbf{x}'}] \rangle$, define kernels on sets and distributions respectively, and also between sets and distributions analogously. The intuition here is that the unordered objects in a set possesses some common property and can be treated as observations drawn iid. from a distribution. If we have multisets and sample weights for instances we may easily include this in the computation of $\mu[X]$. It turns out that (3.26) is exactly the set kernel proposed by [53], when dealing with multiple instance learning. This can be useful, for instance, for medical applications where an examination or test has been carried out multiple times for a certain patient, and we want to group these data together into a set and treat them as a whole.

We can also define universal kernels on sets by using universal kernels on the set elements. The key idea here is that each set is embedded as a unique element in the RKHS and hence the kernel on sets inherits all properties of the underlying RKHS. Formally, we have the following corollary:

Corollary 32 *If k is universal, given n sets of objects, $X_1, \ldots, X_n$, the kernel matrix $\mathbf{K}$ where $\mathbf{K}_{ij} := \langle \mu[X_i], \mu[X_j] \rangle$ has full rank as long as the sets are not identical.*

Similar statement also holds for kernels on distributions. Note, however, that the set/distribution

kernel may not be ideal for all multiple instance problems: in the latter one assumes that at least a *single instance* has a given property, whereas for the use of (3.26) one needs to assume that at least a certain *fraction of instances* have this property.

3.3.4 Density Estimation

We may also use the connection between mean operators and empirical means for the purpose of estimating densities. In fact, [41, 3, 42] show that this may be achieved in the following fashion

$$\begin{aligned} &\underset{\Pr_{\mathbf{x}}}{\text{maximize}}\ H(\Pr_{\mathbf{x}}) \\ &\text{subject to}\ \|\mu[X] - \mu[\Pr_{\mathbf{x}}]\|_{\mathcal{H}} \leq \epsilon. \end{aligned} \tag{3.27}$$

Here H is an entropy-like quantity (e.g. Kullback-Leibler divergence, Csiszar divergence, Bregmann divergence, Entropy, Amari divergence) that is to be maximized subject to the constraint that the expected mean should not stray too far from its empirical counterpart. In particular, one may show that this approximate maximum entropy formulation is the dual of a maximum-a-posteriori estimation problem.

In the case of conditional probability distributions, it is possible to draw connections with many popular estimation algorithms, such as Gaussian Process classification, regression, and conditional random fields. The key idea in this context is to identify the sufficient statistics in generalized exponential families with the map $\mathbf{x} \longmapsto k(\mathbf{x}, \cdot)$ into a reproducing kernel Hilbert space.

In problem (3.27) we try to find the optimal $\Pr_{\mathbf{x}}$ over the entire space of probability distributions on $\mathcal{X}$. This can be an exceedingly costly optimization problem, in particular in the nonparametric setting. For instance, computing the normalization of the density itself may be intractable, in particular for high-dimensional data. In this case we may content ourselves with finding a suitable mixture distribution such that $\|\mu[X] - \mu[\Pr_{\mathbf{x}}]\|_{\mathcal{H}}$ is minimized with respect to the mixture coefficients. The diagram below summarizes our approach

$$\text{density}\ \Pr_{\mathbf{x}} \longrightarrow \text{sample}\ X \longrightarrow \text{empirical mean}\ \mu[X] \longrightarrow \text{estimate via}\ \mu[\widehat{\Pr}_{\mathbf{x}}]. \tag{3.28}$$

The connection between $\mu[\Pr_{\mathbf{x}}]$ and $\mu[X]$ follows from Theorem 27. To obtain a density estimate from $\mu[X]$ assume that we have a set of candidate densities $\Pr^i_{\mathbf{x}}$ on $\mathcal{X}$. We want to use these as basis functions to obtain $\widehat{\Pr}_{\mathbf{x}}$ via

$$\widehat{\Pr}_{\mathbf{x}} = \sum_i^M \beta_i \Pr^i_{\mathbf{x}}, \tag{3.29}$$

where $\sum_{i=1}^M \beta_i = 1$ and $\beta_i \geq 0$. In other words we wish to estimate $\Pr_{\mathbf{x}}$ by means of a mixture model with mixture densities $\Pr^i_{\mathbf{x}}$. The goal is to obtain good estimates for the coefficients β_i and to obtain performance guarantees which specify how well $\widehat{\Pr}_{\mathbf{x}}$ is capable of estimating $\Pr_{\mathbf{x}}$

in the first place. This is possible using a very simple optimization problem

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \ & \left\| \mu[X] - \mu[\widehat{\Pr}_{\mathbf{x}}] \right\|_{\mathcal{H}}^2 \\ \text{subject to} \ & \sum_{i=1}^{M} \beta_i = 1 \text{ and } \beta_i \geq 0. \end{aligned} \tag{3.30}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)^\top$. To ensure good generalization performance we can add a regularizer $\Omega[\boldsymbol{\beta}]$ to the optimization problem, such as $\frac{1}{2}\|\boldsymbol{\beta}\|^2$. It follows using the expansion of $\widehat{\Pr}_{\mathbf{x}}$ in (3.29) that the resulting optimization problem can be reformulated as a quadratic program via

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \ & \frac{1}{2}\boldsymbol{\beta}^\top (\mathbf{Q} + \lambda \mathbf{I}) \boldsymbol{\beta} - \mathbf{l}^\top \boldsymbol{\beta} \\ \text{subject to} \ & \sum_{i=1}^{M} \beta_i = 1 \text{ and } \beta_i \geq 0. \end{aligned} \tag{3.31}$$

Here $\lambda > 0$ is a regularization constant, and the quadratic matrix $\mathbf{Q} \in \mathbb{R}^{M \times M}$ and the vector $\mathbf{l} \in \mathbb{R}^M$ are given by

$$\mathbf{Q}_{ij} = \left\langle \mu[\Pr_{\mathbf{x}}^i], \mu[\Pr_{\mathbf{x}}^j] \right\rangle = \mathbb{E}_{\mathbf{x} \sim \Pr_{\mathbf{x}}^i} \mathbb{E}_{\mathbf{x}' \sim \Pr_{\mathbf{x}'}^j} \left[k(\mathbf{x}, \mathbf{x}') \right] \text{ and} \tag{3.32}$$

$$\mathbf{l}_j = \left\langle \mu[X], \mu[\Pr_{\mathbf{x}}^j] \right\rangle = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{\mathbf{x}' \sim \Pr_{\mathbf{x}'}^j} \left[k(\mathbf{x}_i, \mathbf{x}') \right]. \tag{3.33}$$

By construction $\mathbf{Q} \succeq 0$ is positive semidefinite, hence the quadratic program (3.31) is convex. For a number of kernels and mixture terms $\Pr_{\mathbf{x}}^i$ we are able to compute $\mathbf{Q}, \mathbf{l}$ in closed form.

Since $\widehat{\Pr}_{\mathbf{x}}$ is an empirical estimate it is quite unlikely that $\widehat{\Pr}_{\mathbf{x}} = \Pr_{\mathbf{x}}$. This raises the question of how well expectations with respect to $\Pr_{\mathbf{x}}$ are approximated by those with respect to $\widehat{\Pr}_{\mathbf{x}}$. This can be answered by an extension of the Koksma-Hlawka inequality [102].

Lemma 33 *Let* $\epsilon := \left\| \mu[X] - \mu[\widehat{\Pr}_{\mathbf{x}}] \right\|_{\mathcal{H}}$. *Under the assumptions of Theorem 27 we have that, given* $\delta \in (0, 1)$, *with probability at least* $1 - \delta$,

$$\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \mathbb{E}_{\mathbf{x} \sim \Pr_{\mathbf{x}}} [f(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim \widehat{\Pr}_{\mathbf{x}}} [f(\mathbf{x})] \right| \leq B_m + \sqrt{\frac{\ln(2/\delta)}{2m}} + \epsilon. \tag{3.34}$$

Proof Using the reproducing property of $\mathcal{H}$, we have

$$\mathbb{E}_{\mathbf{x} \sim \Pr_{\mathbf{x}}} [f(\mathbf{x})] = \langle f, \mu[\Pr_{\mathbf{x}}] \rangle \text{ and} \tag{3.35}$$

$$\mathbb{E}_{\mathbf{x} \sim \widehat{\Pr}_{\mathbf{x}}} [f(\mathbf{x})] = \left\langle f, \mu[\widehat{\Pr}_{\mathbf{x}}] \right\rangle. \tag{3.36}$$

Hence the left hand side of (3.34) is equal to $\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \left\langle f, \mu[\Pr_x] - \mu[\widehat{\Pr}_x] \right\rangle \right|$, which is given by the norm $\left\| \mu[\Pr_x] - \mu[\widehat{\Pr}_x] \right\|_{\mathcal{H}}$. Using triangle inequality, the assumption on $\mu[\widehat{\Pr}_x]$, and Theorem 27, we complete the proof. ■

This means that we have good control over the behavior of expectations of random variables, as

long as they belong to "smooth" functions on $\mathcal{X}$ — the uncertainty increases with their RKHS norm.

The above technique is useful when it comes to representing distributions in message passing and data compression. Rather than minimizing an information theoretic quantity, we can choose a Hilbert space which accurately reflects the degree of smoothness required for any subsequent operations carried out by the estimate. For instance, if we are only interested in linear functions, an accurate match of the first order moments will suffice, without requiring a good match in higher order terms.

3.4 Learning via Dependence

Another branch of applications of the embedding approach for distribution analysis arises from measuring whether two random variables $\mathbf{x}$ and $\mathbf{y}$ are independent. Assume that pairs of observations $(\mathbf{x}_i, \mathbf{y}_i)$ are jointly drawn from some distribution $\Pr_{\mathbf{xy}}$. We wish to determine whether this distribution factorizes.

Having a measure of (in)dependence between random variables is a very useful tool in data analysis. One application is in independent component analysis [29], where the goal is to find a linear mapping of the observations $\mathbf{x}_i$ to obtain mutually independent outputs. One of the first algorithms to gain popularity was InfoMax, which relies on information theoretic quantities [86]. Recent developments using cross-covariance or correlation operators between Hilbert space representations have since improved on these results significantly [11, 59, 61]; in particular, a faster and more accurate quasi-Newton optimization procedure for kernel ICA is given in [120]. In the following we will see that one of the above kernel independence measures can be derived from the mean operators instead.

Besides independent component analysis, a measure of independence can be used for many other learning tasks, such as feature selection, clustering and dimensionality reduction. This connection relies on the fact that many existing learning tasks can be cast into problems of dependence maximization/minimization. We will show that this new view not only provides a unifying framework for many existing algorithms, but also suggests interesting new algorithms such as colored maximum variance unfolding. In the next few sections, we will present overview of this learning via dependence framework. It will serve as a road map for the current and future work. In the following chapters of this thesis, we will single out several examples from this framework and discuss them in details.

3.4.1 Dependence Measure

We begin by defining the Hilbert space embedding of the joint distribution $\Pr_{\mathbf{xy}}$ and the product of its marginals $\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$

$$\mu[\Pr_{\mathbf{xy}}] := \mathbb{E}_{\mathbf{xy}} \left[v((\mathbf{x}, \mathbf{y}), \cdot) \right] \text{ and} \tag{3.37}$$

$$\mu[\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}] := \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}} \left[v((\mathbf{x}, \mathbf{y}), \cdot) \right]. \tag{3.38}$$

Here we assumed that $\mathcal{V}$ is an RKHS over $\mathcal{X} \times \mathcal{Y}$ with kernel $v((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$. If $\mathbf{x}$ and $\mathbf{y}$ *are* dependent, the equality $\mu[\Pr_{\mathbf{xy}}] = \mu[\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}]$ will not hold. Hence we may use the distance between the two embeddings in the RKHS as a measure of dependence, ie.

$$\Delta := \left\| \mu[\Pr_{\mathbf{xy}}] - \mu[\Pr_{\mathbf{x}} \times \Pr_{\mathbf{y}}] \right\|_{\mathcal{V}}. \tag{3.39}$$

Now assume that joint kernel decomposes $v((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')$, i.e. the RKHS $\mathcal{V}$ is a direct product $\mathcal{H}\otimes\mathcal{G}$ of the RKHSs on $\mathcal{X}$ and $\mathcal{Y}$. In this case it is easy to see that

$$\begin{aligned}\Delta^2 &= \|\mathbb{E}_{\mathbf{xy}}\left[k(\mathbf{x},\cdot)l(\mathbf{y},\cdot)\right] - \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x},\cdot)\right]\mathbb{E}_{\mathbf{y}}\left[l(\mathbf{y},\cdot)\right]\|_{\mathcal{V}}^2 \\ &= \mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x}'\mathbf{y}'}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right] - 2\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{x}'\mathbf{y}'}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right] \\ &\quad + \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{x}'}\mathbb{E}_{\mathbf{y}'}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right].\end{aligned} \tag{3.40}$$

The latter, however, is exactly what [59] show to be the Hilbert-Schmidt norm of the cross-covariance operator between RKHSs: this is zero if and only if $\mathbf{x}$ and $\mathbf{y}$ are independent, for universal kernels. Furthermore, we have the following two results which we are going to discuss in detail in the next chapter:

- Denote by $\mathcal{C}_{\mathbf{xy}}$ the cross-covariance operator between random variables $\mathbf{x}$ and $\mathbf{y}$ with joint distribution $\Pr_{\mathbf{xy}}$. Let the functions on $\mathcal{X}$ and $\mathcal{Y}$ be draw from the reproducing kernel Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$ respectively. Then the Hilbert-Schmidt norm $\|\mathcal{C}_{\mathbf{xy}}\|_{\mathrm{HS}}$ of the cross-covariance operator equals Δ.
- Denote by $\mathbf{K}$ and $\mathbf{L}$ the kernel matrices on X and Y respectively. Moreover, denote by $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$ the projection matrix onto the subspace orthogonal to the vector with all entries set to 1. Then $\frac{1}{m^2}\operatorname{tr}(\mathbf{HKHL})$ is an estimate of Δ^2 with bias $O(m^{-1})$. With high probability the deviation from Δ^2 is $O(m^{-\frac{1}{2}})$.

Note that if $v((\mathbf{x},\mathbf{y}),\cdot)$ does *not* factorize we obtain a more general measure of dependence. In particular, we might not care about all types of interaction between x and y to an equal extent, and use an ANOVA kernel (a computationally efficient recursions for computing ANOVA kernel can be found from [25, 142]). More importantly, this representation will allow us to deal with *structured* random variables which are *not* drawn independently and identically distributed, such as time series. However, in the following sections, we will focus on the case where observations are drawn iid. and we will show learning via the quantity $\operatorname{tr}(\mathbf{HKHL})$ already gives us many useful insights.

3.4.2 Supervised Learning

(Kernel) Fisher Discriminant Analysis We try to project the observations $\mathbf{X}$ via $\mathbf{Xw}$ such that the dependence between the projected data and the binary labels $\mathbf{y}$ are maximized

$$\underset{\mathbf{w}}{\text{maximize}}\ \operatorname{tr}\left(\mathbf{Xww}^\top\mathbf{X}^\top\mathbf{Hyy}^\top\mathbf{H}\right) \tag{3.41}$$

$$\text{subject to}\ \|\mathbf{w}\| = 1, \tag{3.42}$$

where we use linear kernel on the labels and set $\mathbf{y}$ as

$$\mathbf{y}_i = \begin{cases} \frac{1}{m_+}, & \mathbf{x}_i \text{ from positive class,} \\ -\frac{1}{m_-}, & \text{otherwise.}\end{cases} \tag{3.43}$$

In this case, it is equivalent to maximize the mean between positive class and negative class of the projected data, ie. $(\mu_+ - \mu_-)^2$. This is nothing but Fisher discriminant analysis. If we carry out the projection in reproducing kernel Hilbert spaces $\mathcal{H}$ instead, $\mathbf{w}$ can be written as a linear combination of the observations, ie. $\mathbf{w} = \mathbf{X}^\top\boldsymbol{\alpha}$. We then obtain the following kernelized

version of Fisher discriminant analysis

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{maximize}} \ \operatorname{tr}\left(\boldsymbol{\alpha}^\top \mathbf{K}\mathbf{H}\mathbf{y}\mathbf{y}^\top \mathbf{H}\mathbf{K}\boldsymbol{\alpha}\right) \\ &\text{subject to } \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} = 1. \end{aligned} \tag{3.44}$$

(Kernel) Regression The difference between regression and binary case is that regression usually deals with continuous-valued labels. In this case, we can still use the formulation in (3.41), except that we can also apply more complicated kernel on the labels. For instance we can apply a Gaussian kernel on the labels.

Supervised Feature Selection We try to find a subset of features in $\mathbf{x}$ such that the dependence between the selected features and the labels are maximized. Suppose the full set of feature be T and a subset be $S \subseteq T$. Let $\mathbf{x}_S$ be $\mathbf{x}$ restricted to the features in S, and the corresponding kernel and kernel matrix be $k_S(\mathbf{x}_S, \mathbf{x}'_S)$ and $\mathbf{K}_S$ respectively, then feature selection can be cast in the dependence maximization framework in the following way

$$\underset{S \subseteq T}{\text{maximize}} \ \operatorname{tr}\left(\mathbf{K}_S \mathbf{H}\mathbf{L}\mathbf{H}\right). \tag{3.45}$$

For binary feature selection problem, we can simply use linear kernel and set $\mathbf{y}$ as in (3.43). For multiclass problem we can use the delta kernel $\delta_{\mathbf{y},\mathbf{y}'}$ for the labels. For regression case, we can simply apply a Gauss kernel on the continuous-valued labels.

Kernelized Sorting problem We try to find the relative ordering between two sets of observations such that the dependence between the re-ordered observations is maximized. This is equivalent to maximize the correspondence between two sets of objects, by permuting the order of the objects in one set. Suppose the permutation matrix is Π, the problem can be formulated as

$$\begin{aligned} &\underset{\Pi}{\text{maximize}} \ \operatorname{tr}\left(\Pi^\top \mathbf{K}\Pi\mathbf{H}\mathbf{L}\mathbf{H}\right) \\ &\text{subject to } \Pi\mathbf{1} = \Pi^\top\mathbf{1} = \mathbf{1}, \ \Pi_{ij} \in \{0,1\}. \end{aligned} \tag{3.46}$$

This is a cleaner way of formulating kernelized sorting in [75]. The optimization problem in this case, however, is a hard quadratic assignment problem in general.

Information Bottleneck We try to recode the observations $\mathbf{x}$ into new presentations $\tilde{\mathbf{x}}$, by channel all information through a third variables $\mathbf{y}$ [122]. In other words, we try to extract certain amount of information $\tilde{\mathbf{x}}$ from $\mathbf{x}$ and this information is maximally relevant to $\mathbf{y}$. Let $\mathbf{K}$, $\tilde{\mathbf{K}}$ and $\mathbf{L}$ be the corresponding kernel matrix for $\mathbf{x}$, $\tilde{\mathbf{x}}$ and $\mathbf{y}$ respectively, we can first learning the kernel matrix $\tilde{\mathbf{K}}$ via the following optimization problem

$$\begin{aligned} &\underset{\tilde{\mathbf{K}}}{\text{maximize}} \ \operatorname{tr}\left(\tilde{\mathbf{K}}\mathbf{H}\mathbf{L}\mathbf{H}\right) \\ &\text{subject to } \operatorname{tr}\left(\tilde{\mathbf{K}}\mathbf{H}\mathbf{K}\mathbf{H}\right) = c, \ \tilde{\mathbf{K}} \succeq \mathbf{0}. \end{aligned} \tag{3.47}$$

Then the compressed representation can be calculated from the eigendecomposition of $\tilde{\mathbf{K}}$.

Supervised Dimensionality Reduction In some cases, we have side information for our disposal when we perform dimensionality reduction. Furthermore, we also try to approximately keep the distance between neighboring observations in the original space and reduced space. Suppose the side information is coded in kernel matrix $\mathbf{L}$, the reduced presentation is coded in kernel matrix $\mathbf{K}$, and the distances between the original observations are d_{ij}. We can use the following dependence maximization scheme to perform dimensionality reduction

$$\begin{aligned} &\underset{\mathbf{K}}{\text{maximize}} \ \text{tr}\left(\mathbf{KHLH}\right) \\ &\text{subject to } \mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{jj} = d_{ij}^2, \ \forall (i,j) \in \mathcal{N}, \end{aligned} \tag{3.48}$$

where $\mathcal{N}$ denotes a nearest neighbor graph and $(i,j) \in \mathcal{N}$ means observations $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors (ie. an edge in graph $\mathcal{N}$).

3.4.3 Unsupervised Learning

(Kernel) Principal Component Analysis We try to project the data $\mathbf{X}$ via $\mathbf{Xw}$ such that the dependence between the projected data and the original data is maximized. Furthermore, we require that $\mathbf{w}$ be unit norm, then we have the following optimization problem

$$\begin{aligned} &\underset{\mathbf{w}}{\text{maximize}} \ \text{tr}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{HXX}^\top \mathbf{HXw}\right) \\ &\text{subject to } \|\mathbf{w}\| = 1, \end{aligned} \tag{3.49}$$

where $\mathbf{C} := \mathbf{X}^\top \mathbf{HX}$ is exactly the covariance matrix. Thus the solution of the above optimization problem is equal to finding the eigenvectors of $\mathbf{C}^2$. Since the eigenvectors of $\mathbf{C}^2$ is the same as those of $\mathbf{C}$, the dependence maximization problem in (3.49) becomes principal component analysis. If we carry out the projection in the RKHS, we then have kernel principal component analysis as in [116].

(Kernel) k-means Clustering and Spectral Clustering We try to generate discrete labels for the observations, such that the dependence between the labels and the observations are maximized. Suppose we have c distinct labels, and the assignment of observations into labels is encoded in an assignment matrix $\Pi_{m \times c}$. Furthermore, suppose the kernel matrix for the distinct labels be $\mathbf{A}$, then we can cluster the data by optimizing over the assignment

$$\begin{aligned} &\underset{\Pi}{\text{maximize}} \ \text{tr}\left(\mathbf{HKH}\Pi\mathbf{A}\Pi^\top\right) \\ &\text{subject to } \Pi^\top \mathbf{1} = \mathbf{1}, \Pi_{ij} \in \{0,1\}. \end{aligned} \tag{3.50}$$

We will show later that if we choose $\mathbf{A}$ properly we can actually recover many existing clustering algorithms as special cases. For instance, if $\mathbf{A} = \text{diag}\left(\frac{1}{m_1}, \ldots, \frac{1}{m_c}\right)$, where m_i is the number of observations assigned to label i, then the problem (3.50) is equivalent to k-means clustering. With other $\mathbf{A}$s, we have a more general clustering algorithm. Furthermore, for spectral clustering, we basically apply graph kernels for kernel k-means [125].

Unsupervised Feature Selection We try to find a subset of features in $\mathbf{x}$ such that the dependence between the selected features and the original data are maximized. This problem has been studied by combining Gaussian processes and information theoretic quantities [83].

Here we will use our dependence measure to formulate this problem instead. To avoid selecting correlated features, we also need to encourage independence between the selected features. Suppose the full set of feature be T and a subset be $S \subseteq T$. Let $\mathbf{x}_S$ be $\mathbf{x}$ restricted to the features in S, and the corresponding kernel and kernel matrix be $k_S(\mathbf{x}_S, \mathbf{x}'_S)$ and $\mathbf{K}_S$ respectively, then unsupervised feature selection can be cast in the dependence maximization framework in the following way

$$\underset{S \subseteq T}{\text{maximize}} \ \operatorname{tr}(\mathbf{K}_S \mathbf{H} \mathbf{K} \mathbf{H}) - \lambda \sum_{i,j \in S} \operatorname{tr}\left(\mathbf{K}_{\{i\}} \mathbf{H} \mathbf{K}_{\{j\}} \mathbf{H}\right), \tag{3.51}$$

where λ controls the amount of penalty for correlated features. If we select features incrementally, we can then use simpler expression for such penalization. Suppose the current set of selected features be S, then the next feature to be included can be obtained from

$$\underset{i \in T - S}{\operatorname{argmax}} \ \operatorname{tr}\left(\mathbf{K}_{S \cup \{i\}} \mathbf{H} \mathbf{K} \mathbf{H}\right) - \lambda \operatorname{tr}\left(\mathbf{K}_{\{i\}} \mathbf{H} \mathbf{K}_S \mathbf{H}\right). \tag{3.52}$$

Dimensional Reduction First, we can use the setting in (3.48) for dimensionality reduction. In unsupervised case, we do not have label information; therefore we set $\mathbf{L} = \mathbf{I}$. This corresponds exactly to the maximum variance unfolding approach [149], which seeks a reduced representation that retains maximal diversity between observations.

Other dimensionality reduction techniques, such as locally linear embedding (LLE) and Isomap, are also intimately related to dependence maximization. This connection has been drawn in [66] which shows that LLE and Isomap essentially performs principal component analysis on particular kernel matrices.

3.5 Summary

We have shown that Hilbert space embeddings of distributions are a powerful tool to deal with a broad range of estimation problems, including two-sample tests, covariate shift, kernels on sets, and density estimation. Furthermore, we can also define useful dependence measure via Hilbert space embeddings. This leads to a unifying framework for various supervised and unsupervised learning problems. In the next few chapters, we will focus on the dependence measure and discuss several concrete examples of this new learning framework. These examples include feature selection, clustering and supervised dimensionality reduction algorithms.

CHAPTER 4

Dependence Measure

In this chapter, we will use the Hilbert space embedding approach to define a new measure of dependence. We show that under the assumption of iid. sampling, this new measure recovers Hilbert-Schmidt Independence Criterion (HSIC) as a special case. We will also design estimators for HSIC, and study their concentration and asymptotic distribution. Last, we will design statistical tests of independence based on HSIC. We demonstrate the advantages of this new test in our experiments.

4.1 Introduction

The concept of statistical dependence concerns the covariations between two random variables. Intuitively, a random variable is dependent on another random variable, if its variations can be explained by those of another. Measuring statistical dependence is useful in many real world applications. For instance, a dependence measure can help answer questions like whether the price of one stock is linked to another; whether the global temperature is connected to the amount of carbon dioxide emission; or whether the activities of one biological unit is synchronized with those of another.

Measuring statistical dependence has long been a field of statistical analysis [110]. Mutual information, for instance, is a well-known dependence measure [30]. Recently, there has been considerable interest in using functions in the RKHS to define dependence measure. This was first started by Bach and Jordan [11], who used a regularized estimate of the spectral norm of the correlation operator between two RKHS (KCC). They showed that this measure significantly outperforms traditional measures in independence component analysis. Other work that builds on KCC includes the constrained covariance (COCO) [61]. Instead of the correlation operator, COCO employed the covariance operator. In this chapter, we will introduce a dependence measure based on the Hilbert space embedding distance between distributions. We will also show that, under the assumption of iid. sampling, this measure is equivalent to the squared Hilbert-Schmidt norm of the covariance operator (HSIC) [59]. This measure has many nice properties such as the simple empirical estimate for iid. data and its fast convergence to the population counterpart. Before we further explain these dependence measures, we will explain the concept of statistical dependence in more details.

4.2 Measures of Dependence

Two random variables $\mathbf{x}$ and $\mathbf{y}$ are statistically independent, if and only if their joint distribution ($\Pr_{\mathbf{xy}}$) factorizes into the product of their marginal distributions ($\Pr_{\mathbf{x}}$ and $\Pr_{\mathbf{y}}$), ie. $\Pr_{\mathbf{xy}} = \Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$. We will denote $\mathbf{x} \perp \mathbf{y}$ if $\mathbf{x}$ and $\mathbf{y}$ are independent. When the joint distribution

does not factorize, we call $\mathbf{x}$ and $\mathbf{y}$ are dependent. To measure the strength of dependence, we usually construct a summary statistic, $A(\mathbf{x}, \mathbf{y})$, to quantify the difference between $\Pr_{\mathbf{xy}}$ and $\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$. Renyi [110] gave a list of desirable properties for a measure of statistical dependence between random variables, as listed in Table 4.1. In practice, however, it is usually difficult to find a measure that satisfies all these properties while at the same easy to compute.

Note that, for measuring *in*dependence, normally we care most about property (4) and the relative amplitude of the test statistics. Very often we can drop the normalization requirement (3) and (5). For a general measure of dependence, however, all five properties are desirable. We will present several dependence measures before we define our new dependence measure.

Table 4.1: Desirable properties of a dependence measure

	Property	Notes
(1)	$A(\mathbf{x}, \mathbf{y})$ is well defined	For all nonconstant variables
(2)	$A(\mathbf{x}, \mathbf{y}) = A(\mathbf{y}, \mathbf{x})$	Symmetric
(3)	$0 \leq A(\mathbf{x}, \mathbf{y}) \leq 1$	Normalized to range $[0, 1]$
(4)	$A(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} \perp \mathbf{y}$	Necessary and sufficient condition
(5)	$A(\mathbf{x}, \mathbf{y}) = 1 \Leftrightarrow \mathbf{y} = f(\mathbf{x})$ or $\mathbf{x} = g(\mathbf{y})$	Maximum at deterministic relation

4.2.1 Pearson's Correlation Coefficient

Pearson's correlation coefficient measures linear dependence between two univariate random variables x and y. Assume that x and y are linearly related via

$$y = a \cdot x + b + \epsilon, \tag{4.1}$$

where $a, b \in \mathbb{R}$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise with unknown variance. Pearson's correlation coefficient is defined as

$$\operatorname{corr}(x, y) = \frac{\operatorname{cov}(x, y)}{\sqrt{\operatorname{var}(x)\operatorname{var}(y)}} \tag{4.2}$$

with the covariance and variance

$$\operatorname{cov}(x, y) = \mathbb{E}_{xy}\left[(x - \mu_x)\,(y - \mu_y)\right], \tag{4.3}$$

$$\operatorname{var}(x) = \mathbb{E}_{x}\left[(x - \mu_x)\,(x - \mu_x)\right], \tag{4.4}$$

where $\mu_x := \mathbb{E}[x]$ and $\mu_y := \mathbb{E}[y]$. Pearson's correlation is limited in many aspects. It only satisfies properties (1), (2) and (3) required for a dependence measure. In term of property (4) and (5), Pearson's correlation can only measure linear relation between univariate Gaussian random variables, and it also relies heavily on the assumption of Gaussian noise. If these assumptions are violated, Pearson's correlation may not be able to detect the dependence.

4.2.2 Mutual Information

Mutual information $I(\mathbf{x}, \mathbf{y})$ is a more general measure of dependence and does not rely on any model assumption. It is defined via the Kullback-Leibler divergence between the joint

distribution $\Pr_{\mathbf{xy}}$ and the product $\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$ of its marginals [30]

$$I(\mathbf{x}, \mathbf{y}) := \int_{\mathcal{X}\times\mathcal{Y}} \log\left(\frac{\mathrm{d}\Pr_{\mathbf{xy}}}{\mathrm{d}\Pr_{\mathbf{x}}\,\mathrm{d}\Pr_{\mathbf{y}}}\right) \mathrm{d}\Pr_{\mathbf{xy}}\,. \tag{4.5}$$

By convention, $0 \cdot \log\left(\frac{0}{0}\right) = 0$ such that mutual information is well-defined (property (1)). Furthermore, mutual information is symmetric in its arguments, and it is a necessary and sufficient condition for independence (property (2) and (4)). Mutual information, however, is not normalized to the range $[0, 1]$. It is maximized when there exists deterministic relation between $\mathbf{x}$ and $\mathbf{y}$. The maximum is the entropy of either $\mathbf{x}$ or $\mathbf{y}$ which is in general not equal to 1.

The definition of mutual information does not rely on any assumption on the type of distribution or the type of dependence. We expect it to have wide applicability in theory. In practice, however, such generality also brings problems. The distribution of the random variables are usually not easy to estimate especially for high dimensional data. Furthermore, such generality is usually not necessary in many cases. For instance, if we want to investigate whether second order dependence exists between random variables $\mathbf{x}$ and $\mathbf{y}$ (suppose both variables are drawn from $\mathbb{R}^d$). For mutual information, we still need to estimate both the joint and marginal distribution before we can assess the strength of the dependence. In this case, all information needed for testing their dependence will be contained in the cross-covariance matrix

$$\mathcal{C}_{\mathbf{xy}} := \mathbb{E}_{\mathbf{xy}}[\mathbf{x}\mathbf{y}^\top] - \mathbb{E}_{\mathbf{x}}[\mathbf{x}]\mathbb{E}_{\mathbf{y}}[\mathbf{y}^\top]. \tag{4.6}$$

If there exists no second order dependence between $\mathbf{x}$ and $\mathbf{y}$, then all entries in the cross-covariance matrix should be zero. In this case, we can simply use the Frobenius norm of $\mathcal{C}_{\mathbf{xy}}$ to summarize the degree of dependence between $\mathbf{x}$ and $\mathbf{y}$. Alternatively, given the singular values σ_i of $\mathcal{C}_{\mathbf{xy}}$, we can compute it via

$$\|\mathcal{C}_{\mathbf{xy}}\|^2_{\mathrm{Frob}} = \mathrm{tr}\left(\mathcal{C}_{\mathbf{xy}}\mathcal{C}^\top_{\mathbf{xy}}\right) = \sum_i \sigma_i^2. \tag{4.7}$$

This quantity is zero if and only if there exists no second order dependence between x and y. From this example, we see that, by incorporating prior knowledge, dependence estimation becomes much easier; we can sidestep density estimation and still obtain an evaluation of the dependence.

The statistic in (4.7) is limited in several respects, however, of which we mention two: first, dependence can exist in forms other than that detectable via covariance (and even when a second order relation exists, the full extent of the dependence between $\mathbf{x}$ and $\mathbf{y}$ may only be apparent when nonlinear effects are included). Second, the restriction to subsets of $\mathbb{R}^d$ excludes many interesting kinds of variables, such as strings and class labels. Therefore, recent research on dependence measure tries to generalize the notion of covariance (or correlation) to nonlinear relationships, and to a wider range of data types. In the following sections, we will first describe two kernel measures of dependence before we introduce our new measure based on Hilbert space embedding of distributions.

4.2.3 Kernel Canonical Correlation

The idea of kernel canonical correlation dates back to Renyi [110]. He states that $\mathbf{x}$ and $\mathbf{y}$ are independent if and only if $\mathrm{cov}(f(\mathbf{x}), g(\mathbf{y})) = 0$ for any bounded continuous function f and g, ie. $f \in C(\mathcal{X}), g \in C(\mathcal{Y})$. Kernel canonical correlation (KCC) specializes this result by

considering functions in the RKHS. Formally, let k and l be kernels for random variables $\mathbf{x}$ and $\mathbf{y}$ respectively, and their associated reproducing kernel Hilbert spaces be $\mathcal{F}$ and $\mathcal{G}$ respectively. The kernel canonical correlation between $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$\begin{aligned}\mathrm{KCC} &:= \sup_{\|f\|_{\mathcal{F}}\leq 1, \|g\|_{\mathcal{G}}\leq 1} \mathrm{corr}\left(\langle f, k(\mathbf{x},\cdot)\rangle, \langle g, l(\mathbf{y},\cdot)\rangle\right) && (4.8)\\ &= \sup_{\|f\|_{\mathcal{F}}\leq 1, \|g\|_{\mathcal{G}}\leq 1} \frac{\mathrm{cov}\left(\langle f, k(\mathbf{x},\cdot)\rangle, \langle g, l(\mathbf{y},\cdot)\rangle\right)}{\sqrt{\mathrm{var}\left(\langle f, k(\mathbf{x},\cdot)\rangle\right)\mathrm{var}\left(\langle g, l(\mathbf{y},\cdot)\rangle\right)}}. && (4.9)\end{aligned}$$

KCC is well-defined, symmetric with respect to $\mathbf{x}$ and $\mathbf{y}$, and normalized to range $[0, 1]$ (property (1), (2) and (3)). The type of dependence that can be detected by KCC is intimately related to the kernels chosen. As we discussed in the Chapter 2, the kernels uniquely determines the RKHS and hence the class of induced functions. If we are only interested in second order dependence, we can simply use linear kernels for both k and l. Linear kernels define the space of linear functions which is a subspace of $C(\mathcal{X})$. If the set of functions induced by the RKHS is dense in $C(\mathcal{X})$, KCC is guaranteed to detect dependence of any kind [11]. In particular, universal kernels satisfy this condition and have the flexibility to detect dependence of any kind.

To compute KCC, we first center the observations in the RKHS, ie. $\tilde{k}(\mathbf{x}_i,\cdot) := k(\mathbf{x}_i,\cdot) - \sum_{i=1}^m k(\mathbf{x}_i,\cdot)$, and then consider the functions in the RKHS which lie in the span of the finite observations, ie. $f = \sum_{i=1}^m \alpha_i \tilde{k}(\mathbf{x}_i,\cdot)$ and $g = \sum_{i=1}^m \beta_i \tilde{l}(\mathbf{y}_i,\cdot)$. The empirical estimate of the covariance and variance can be computed as

$$\begin{aligned}&\widehat{\mathrm{cov}}\left(\left\langle f, \tilde{k}(\mathbf{x},\cdot)\right\rangle, \left\langle g, \tilde{l}(\mathbf{y},\cdot)\right\rangle\right) && (4.10)\\ =&\sum_{i=1}^m \left(\sum_{j=1}^m \alpha_j \left\langle \tilde{k}(\mathbf{x}_j,\cdot), \tilde{k}(\mathbf{x}_i,\cdot)\right\rangle \sum_{j=1}^m \beta_j \left\langle \tilde{l}(\mathbf{y}_j,\cdot), \tilde{l}(\mathbf{y}_i,\cdot)\right\rangle\right)\\ =&\boldsymbol{\alpha}^\top \tilde{\mathbf{K}}\tilde{\mathbf{L}}\boldsymbol{\beta},\end{aligned}$$

and similarly $\widehat{\mathrm{var}}\left(\left\langle f, \tilde{k}(\mathbf{x},\cdot)\right\rangle\right) = \boldsymbol{\alpha}^\top \tilde{\mathbf{K}}\tilde{\mathbf{K}}\boldsymbol{\alpha}$. Note that $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ are the kernel matrices for the centered observations. They can be readily computed from the kernel matrices $\mathbf{K}$ and $\mathbf{L}$ via

$$\tilde{\mathbf{K}} = \mathbf{HKH} \quad \text{and} \quad \tilde{\mathbf{L}} = \mathbf{HLH}, \tag{4.11}$$

where $\mathbf{H}$ is the centering matrix defined as $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top$. Therefore, the empirical estimate of KCC is equal to

$$\widehat{\mathrm{KCC}} = \sup_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{R}^m} \frac{\boldsymbol{\alpha}^\top \tilde{\mathbf{K}}\tilde{\mathbf{L}}\boldsymbol{\beta}}{\sqrt{\left(\boldsymbol{\alpha}^\top \tilde{\mathbf{K}}^2\boldsymbol{\alpha}\right)\left(\boldsymbol{\beta}^\top \tilde{\mathbf{L}}^2\boldsymbol{\beta}\right)}}. \tag{4.12}$$

However, a drawback of $\widehat{\mathrm{KCC}}$ is that, for finite number of observations, $\widehat{\mathrm{KCC}}$ can potentially over-fit the data. To show this, we re-parameterize the $\widehat{\mathrm{KCC}}$ optimization problem using $\mathbf{u} = \tilde{\mathbf{K}}\boldsymbol{\alpha} \in \mathcal{U}$ and $\mathbf{v} = \tilde{\mathbf{L}}\boldsymbol{\beta} \in \mathcal{V}$ where $\mathcal{U}$ and $\mathcal{V}$ denote the subspaces of $\mathbb{R}^m$ generated by the

column space of $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ respectively. Then we get

$$\widehat{\mathrm{KCC}} = \sup_{\mathbf{u}\in\mathcal{U},\mathbf{v}\in\mathcal{V}} \frac{\mathbf{u}^\top\mathbf{v}}{\sqrt{(\mathbf{u}^\top\mathbf{u})(\mathbf{v}^\top\mathbf{v})}} = \sup_{\mathbf{u}\in\mathcal{U},\mathbf{v}\in\mathcal{V}} \cos(\mathbf{u},\mathbf{v}), \tag{4.13}$$

which is exactly the cosine of the angle between the two subspaces $\mathcal{U}$ and $\mathcal{V}$. If both $\mathbf{K}$ and $\mathbf{L}$ have full rank, the subspaces spanned by their columns become identical, ie. $\mathbb{R}^m$. This situation happens, for instance, when we use universal kernels for computing $\widehat{\mathrm{KCC}}$. The centering matrix only projects out the subspaces spanned by the vector of all ones, and the remaining subspaces spanned by $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ will still coincide. Therefore we will always obtain a value of 1 for $\widehat{\mathrm{KCC}}$. To avoid this, we need to consider the following regularized version of KCC

$$\widehat{\mathrm{KCC}} = \sup_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{R}^m} \frac{\boldsymbol{\alpha}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\boldsymbol{\beta}}{\sqrt{\left(\boldsymbol{\alpha}^\top\left(\tilde{\mathbf{K}}+\frac{m\epsilon}{2}\mathbf{I}\right)^2\boldsymbol{\alpha}\right)\left(\boldsymbol{\beta}^\top\left(\tilde{\mathbf{L}}+\frac{m\epsilon}{2}\mathbf{I}\right)^2\boldsymbol{\beta}\right)}}. \tag{4.14}$$

$\widehat{\mathrm{KCC}}$ can be equivalently formulated as the following constrained optimization problem

$$\begin{aligned}\widehat{\mathrm{KCC}} = \underset{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{R}^m}{\text{maximize}}\ & \boldsymbol{\alpha}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\boldsymbol{\beta} \\ \text{subject to}\ & \boldsymbol{\alpha}^\top\left(\tilde{\mathbf{K}}+\frac{m\epsilon}{2}\mathbf{I}\right)^2\boldsymbol{\alpha} = 1 \ \text{ and } \ \boldsymbol{\beta}^\top\left(\tilde{\mathbf{L}}+\frac{m\epsilon}{2}\mathbf{I}\right)^2\boldsymbol{\beta} = 1.\end{aligned} \tag{4.15}$$

we can obtain the solution for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as the principal eigenvector of the following generalized eigenvalue problem

$$\begin{pmatrix} 0 & \tilde{\mathbf{K}}\tilde{\mathbf{L}} \\ \tilde{\mathbf{L}}\tilde{\mathbf{K}} & 0 \end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} = \lambda\begin{pmatrix}\left(\tilde{\mathbf{K}}+\frac{m\epsilon}{2}\mathbf{I}\right)^2 & 0 \\ 0 & \left(\tilde{\mathbf{L}}+\frac{m\epsilon}{2}\mathbf{I}\right)^2\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}. \tag{4.16}$$

Thus $\widehat{\mathrm{KCC}}$ is equal to the largest eigenvalue. The drawback of $\widehat{\mathrm{KCC}}$ is that it is not easy to tune the regularization parameter ϵ. Normally, as the number of observations increases, ϵ must approach zero at rate $O(m^{\frac{1}{3}})$ to ensure the consistency of the estimated KCC, and of the associated maximizers f and g [48]. Furthermore, computation of the generalized eigenvalue problem is costly especially when the number of observations are large.

4.2.4 Constrained Covariance

Constrained covariance (COCO) gears towards the test of *in*dependence, and it only satisfies properties (1) (2) and (4). For this purpose, COCO drops the normalization by variances and uses only the functional covariance. Keeping the notations for kernel canonical correlation, we have the definition for COCO [61]

$$\mathrm{COCO} := \sup_{\|f\|_{\mathcal{F}}\leq 1,\|g\|_{\mathcal{G}}\leq 1} \mathrm{cov}\left(\langle f, k(\mathbf{x},\cdot)\rangle, \langle g, l(\mathbf{y},\cdot)\rangle\right). \tag{4.17}$$

From the definition we can see that COCO is well-defined and symmetric. Furthermore, COCO is zero if and only if $\mathbf{x}$ and $\mathbf{y}$ are independent [61]. For finite number of observations, both f and g lie in the span of the observations in the RKHS, ie. $f = \sum_{i=1}^m \alpha_i\tilde{k}(\mathbf{x}_i,\cdot)$ and $g =$

$\sum_{i=1}^{m} \beta_i \tilde{l}(\mathbf{y}_i, \cdot)$. We compute the empirical estimate of COCO as [61]

$$\begin{aligned}\widehat{\mathrm{COCO}} &= \underset{\boldsymbol{\alpha}^\top \tilde{\mathbf{K}} \boldsymbol{\alpha} \leq 1, \boldsymbol{\beta}^\top \tilde{\mathbf{L}} \boldsymbol{\beta} \leq 1}{\text{maximize}} \frac{1}{m} \boldsymbol{\alpha}^\top \tilde{\mathbf{K}} \tilde{\mathbf{L}} \boldsymbol{\beta} \\ &= \max_{\boldsymbol{\alpha}'^\top \boldsymbol{\alpha}' \leq 1, \boldsymbol{\beta}'^\top \boldsymbol{\beta} \leq 1} \frac{1}{m} \boldsymbol{\alpha}'^\top \tilde{\mathbf{K}}^{1/2} \tilde{\mathbf{L}}^{1/2} \boldsymbol{\beta}' \\ &= \frac{1}{m} \left\| \tilde{\mathbf{K}}^{1/2} \tilde{\mathbf{L}}^{1/2} \right\|_2 ,\end{aligned} \tag{4.18}$$

where we have made a replacement of variables as $\boldsymbol{\alpha}' = \tilde{\mathbf{K}}^{1/2} \boldsymbol{\alpha},\ \boldsymbol{\beta}' = \tilde{\mathbf{L}}^{1/2} \boldsymbol{\beta}$. The matrices $\tilde{\mathbf{K}}^{1/2}$ and $\tilde{\mathbf{L}}^{1/2}$ are the matrix square roots of $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ respectively, and $\|\cdot\|_2$ returns the largest singular value of its argument.

The advantage of $\widehat{\mathrm{COCO}}$ over $\widehat{\mathrm{KCC}}$ is that $\widehat{\mathrm{COCO}}$ requires no regularization parameter. This avoids the additional model selection problem in $\widehat{\mathrm{KCC}}$. It has been shown that $\widehat{\mathrm{COCO}}$ achieves better performance than $\widehat{\mathrm{KCC}}$ when used as a contrast function in ICA. The drawback of $\widehat{\mathrm{COCO}}$ is that we need to compute the square root of $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$. If the number of observations is large, this operation is very costly since it requires full eigen-decomposition of $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$.

4.3 Measure based on Hilbert Space Embedding of Distributions

In this section, we will introduce our new dependence measure based on Hilbert-space embedding of distributions. Like COCO, this new measure only satisfies property (1), (2) and (4). However, it is computationally much more efficient than COCO. The key idea is that we embed both the joint distribution and the product of the marginal distributions into the RKHS and then compare their embedding distance in the RKHS. Given two random variables $\mathbf{x}$ and $\mathbf{y}$, we begin by defining the following two embeddings

$$\mu[\Pr_{\mathbf{xy}}] := \mathbb{E}_{\mathbf{xy}} \left[v((\mathbf{x}, \mathbf{y}), \cdot) \right] \text{ and} \tag{4.19}$$

$$\mu[\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}] := \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}} \left[v((\mathbf{x}, \mathbf{y}), \cdot) \right]. \tag{4.20}$$

Here we assume that $\mathcal{V}$ is an RKHS over $\mathcal{X} \times \mathcal{Y}$ with joint kernel $v((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}'))$. If $\mathbf{x}$ and $\mathbf{y}$ *are* independent, the equality $\mu[\Pr_{\mathbf{xy}}] = \mu[\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}]$ should hold. Hence we may use their distance, $\Delta := \|\mu[\Pr_{xy}] - \mu[\Pr_x \Pr_y]\|_{\mathcal{V}}$, as a measure of dependence. Expanding Δ^2 using the kernel, we have

$$\begin{aligned}\Delta^2 =& \left\| \mathbb{E}_{\mathbf{xy}} \left[v((\mathbf{x}, \mathbf{y}), \cdot) \right] - \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}} \left[v((\mathbf{x}, \mathbf{y}), \cdot) \right] \right\|_{\mathcal{V}}^2 \\ =& \mathbb{E}_{\mathbf{xy}} \mathbb{E}_{\mathbf{x}'\mathbf{y}'} \left[v((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) \right] - 2 \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}} \mathbb{E}_{\mathbf{x}'\mathbf{y}'} \left[v((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) \right] \\ &+ \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}} \mathbb{E}_{\mathbf{x}'} \mathbb{E}_{\mathbf{y}'} \left[v((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) \right].\end{aligned} \tag{4.21}$$

Alternatively, Δ can be defined via an optimization problem in the RKHS

$$\Delta = \sup_{\|f\|_{\mathcal{V}} \leq 1} \mathbb{E}_{\mathbf{xy}}[f(\mathbf{x}, \mathbf{y})] - \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}}[f(\mathbf{x}, \mathbf{y})], \tag{4.22}$$

where we try to find a function f inside a unit ball in the RKHS, such that the difference between the expected value of f under the joint distribution and that under the product of the marginals are maximized. In essence, f can be treated as a classifier between matching pair of observations $(\mathbf{x}, \mathbf{y})$ and randomly assembled pairs $(\mathbf{x}, \mathbf{y}')$. If the classifier can distinguish

them better than random, then there exists some dependence between $\mathbf{x}$ and $\mathbf{y}$. The maximizer f is called the witness function in resemblance to the two-sample test in section 3.3.1. An illustration of the witness function is provided in Figure 4.1. We observe that this is a smooth function which has large magnitude where the joint distribution is most different from the product of the marginals.

In some cases, the joint kernel $v((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}'))$ factorizes into the product of two kernels, ie. $v((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')$. In this case, the RKHS $\mathcal{V}$ is a direct product $\mathcal{H}\otimes\mathcal{G}$ of the RKHSs on $\mathcal{X}$ and $\mathcal{Y}$. Then we can compute Δ^2 as

$$\begin{aligned}\Delta^2 &= \left\|\mathbb{E}_{\mathbf{xy}}\left[k(\mathbf{x},\cdot)l(\mathbf{y},\cdot)\right] - \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x},\cdot)\right]\mathbb{E}_{\mathbf{y}}\left[l(\mathbf{y},\cdot)\right]\right\|_{\mathcal{V}}^2 \\ &= \mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x}'\mathbf{y}'}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right] - 2\mathbb{E}_{\mathbf{xy}}\left[\mathbb{E}_{\mathbf{x}'}\left[k(\mathbf{x},\mathbf{x}')\right]\mathbb{E}_{\mathbf{y}'}\left[l(\mathbf{y},\mathbf{y}')\right]\right] \\ &+ \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{x}'}\left[k(\mathbf{x},\mathbf{x}')\right]\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{y}'}\left[l(\mathbf{y},\mathbf{y}')\right].\end{aligned} \tag{4.23}$$

We will show later that Δ^2 is equal to the squared Hilbert-Schmidt norm of the cross covariance operator between space $\mathcal{X}$ and $\mathcal{Y}$. Therefore, this measure is also called Hilbert-Schmidt Independence Criterion (HSIC) [59]. In some cases, such as using a Gaussian RBF kernel, the empirical estimate of HSIC can also be interpreted as a smoothed difference between the joint empirical characteristic function (ECF) and the product of the marginal ECFs [45, 77]. However, this interpretation does not hold in all cases which include kernels on strings, graphs, and other structured spaces.

Note that if $v((\mathbf{x},\mathbf{y}),\cdot)$ does *not* factorize we obtain a more general measure of independence. In particular, we might not care about all types of interaction between $\mathbf{x}$ and $\mathbf{y}$ to an equal extent, and use an ANOVA kernel [25, 142]. More importantly, this representation will allow us to deal with *structured* random variables which are *not* drawn independently and identically distributed, such as time series. For instance, we can define the following joint feature map for time series $\mathbf{x}$ and $\mathbf{y}$ which takes the dependency between adjacent observations into account

$$v((\mathbf{x},\mathbf{y}),\cdot) = \begin{pmatrix} k(\mathbf{x}_{t-1},\cdot) & \otimes & k(\mathbf{x}_t,\cdot) \\ k(\mathbf{x}_t,\cdot) & \otimes & l(\mathbf{y}_t,\cdot) \\ l(\mathbf{y}_{t-1},\cdot) & \otimes & l(\mathbf{y}_t,\cdot) \end{pmatrix}. \tag{4.24}$$

Another example is for the case of EEG (electroencephalogram) data where we have both spatial and temporal structure in the signal. Few algorithms take full advantage of this when performing independent component analysis [8]. The pyramidal kernel of [113] is one possible choice for measuring dependence in this structured case.

However, many theoretical questions related to dependence measure in structured case have not been settled yet: for instance, what is the estimator, whether we have good convergence of empirical estimator to population counterpart, and how to perform learning via dependence in structured case. In the following sections, we will only discuss the case for iid. data. In this case, we show that the dependence measure is equivalent to the Hilbert-Schmidt norm of the cross-covariance operator (HSIC). We will develop further theory for HSIC in the rest of this chapter and present examples of learning via HSIC in the following chapters.

4.4 Hilbert-Schmidt Independence Criterion (HSIC)

In this section, we will start with the cross-covariance operator, a generalization of the cross-covariance matrix. In accordance with [12, 49], this is a linear operator $\mathcal{C}_{\mathbf{xy}} : \mathcal{G} \longmapsto \mathcal{F}$ such

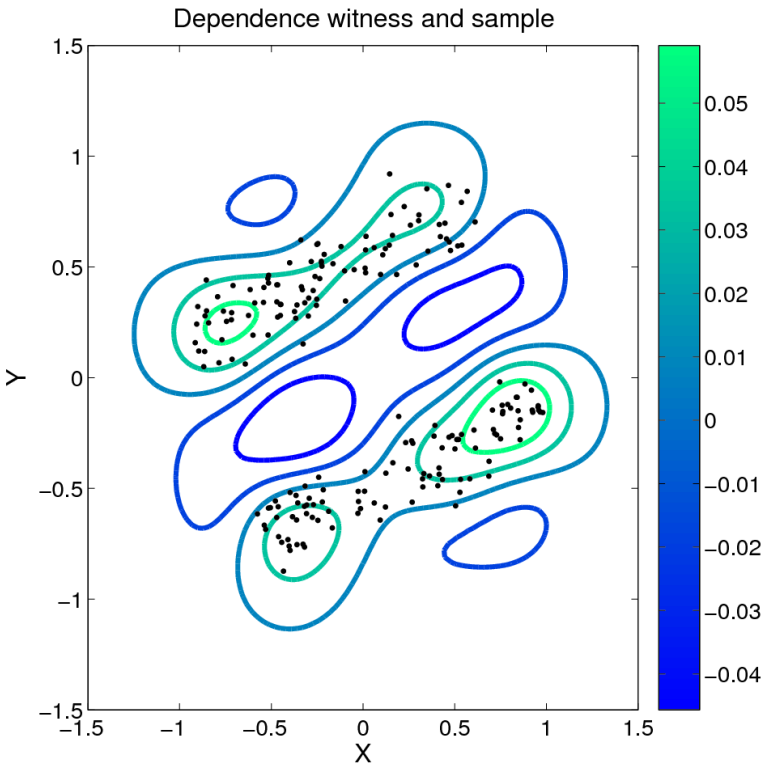


Figure 4.1: Illustration of the function maximizing the discrepancy between $\mathbb{E}_x\mathbb{E}_y[f(x,y)]$ and $\mathbb{E}_{xy}[f(x,y)]$. A sample from dependent scalar random variables x and y is shown in black, and the associated witness function f is plotted as a contour. The latter was computed empirically on the basis of 200 samples, using a Gaussian kernel with $\sigma = 0.2$.

that[1]

$$\begin{aligned} \mathcal{C}_{\mathbf{xy}} &:= \mathbb{E}_{\mathbf{xy}}\left[(k(\mathbf{x},\cdot) - \mu_{\mathbf{x}}) \otimes (l(\mathbf{y},\cdot) - \mu_{\mathbf{y}})\right] \\ &= \mathbb{E}_{\mathbf{xy}}\left[k(\mathbf{x},\cdot) \otimes l(\mathbf{y},\cdot)\right] - \mu_{\mathbf{x}} \otimes \mu_{\mathbf{y}}, \end{aligned} \tag{4.25}$$

where $\mu_{\mathbf{x}} = \mathbb{E}_{\mathbf{x}}[k(\mathbf{x},\cdot)]$ and $\mu_{\mathbf{y}} = \mathbb{E}_{\mathbf{y}}[l(\mathbf{y},\cdot)]$. Here $\otimes$ denotes the tensor product. For $f \in \mathcal{F}$ and $g \in \mathcal{G}$, a tensor product is a linear operator $f \otimes g : \mathcal{G} \longmapsto \mathcal{F}$ formally defined as

$$(f \otimes g)\, h \;:=\; f \left\langle g, h \right\rangle_{\mathcal{G}} \text{ for all } h \in \mathcal{G}. \tag{4.26}$$

Earlier we used Frobenius norm of the cross-covariance matrix to measure second order dependence. Here we need to extend the notion of the Frobenius norm to operators. This leads us to the Hilbert-Schmidt (HS) norm of a linear operator $\mathcal{C} : \mathcal{G} \longmapsto \mathcal{F}$, ie. provided the sum converges,

$$\|\mathcal{C}\|^2_{\mathrm{HS}} \;:=\; \sum_{ij} \left\langle C\mathbf{v}_i, \mathbf{u}_j \right\rangle^2_{\mathcal{F}}, \tag{4.27}$$

[1] Again we abuse the notation here by using the same subscript in the operator $\mathcal{C}_{\mathbf{xy}}$ as in the covariance matrix of (4.6), even though we now refer to the covariance between feature maps.

where $\mathbf{u}_j$ and $\mathbf{v}_i$ are orthonormal bases of $\mathcal{F}$ and $\mathcal{G}$ respectively. For operators with discrete spectrum this amounts to computing the ℓ_2 norm of the singular values. A linear operator is called Hilbert-Schmidt operator if its Hilbert-Schmidt norm exists. The set of Hilbert-Schmidt operators $\mathcal{C} : \mathcal{G} \longmapsto \mathcal{F}$ also form a Hilbert space $\mathcal{H}$ with inner product defined as

$$\langle \mathcal{C}, \mathcal{D} \rangle_{\mathcal{H}} := \sum_{ij} \langle \mathcal{C}\mathbf{v}_i, \mathbf{u}_j \rangle_{\mathcal{F}} \langle \mathcal{D}\mathbf{v}_i, \mathbf{u}_j \rangle_{\mathcal{F}} . \tag{4.28}$$

Then we can compute the HS norm of a tensor product operator as

$$\begin{aligned} \|f \otimes g\|^2_{\mathrm{HS}} &= \langle f \otimes g, f \otimes g \rangle_{\mathcal{H}} = \langle f, (f \otimes g)\, g \rangle_{\mathcal{F}} \\ &= \langle f, f \rangle_{\mathcal{F}} \langle g, g \rangle_{\mathcal{G}} = \|f\|^2_{\mathcal{F}} \|g\|^2_{\mathcal{G}} . \end{aligned} \tag{4.29}$$

We use the squared Hilbert-Schmidt norm of the cross-covariance operator (HSIC), $\|\mathcal{C}_{\mathbf{xy}}\|^2_{\mathrm{HS}}$ as our dependence measure. In terms of kernels, HSIC can be expressed as [59]

$$\begin{aligned} \mathrm{HSIC} &:= \|\mathcal{C}_{\mathbf{xy}}\|^2_{\mathrm{HS}} = \langle \mathcal{C}_{\mathbf{xy}}, \mathcal{C}_{\mathbf{xy}} \rangle_{\mathcal{H}} \\ &= \mathbb{E}_{\mathbf{xy}} \left[\langle k(\mathbf{x}, \cdot) \otimes l(\mathbf{y}, \cdot), k(\mathbf{x}, \cdot) \otimes l(\mathbf{y}, \cdot) \rangle_{\mathcal{H}} \right] - 2\mathbb{E}_{\mathbf{xy}} \left[\langle \mu_{\mathbf{x}} \otimes \mu_{\mathbf{y}}, k(\mathbf{x}, \cdot) \otimes l(\mathbf{y}, \cdot) \rangle_{\mathcal{H}} \right] \\ &+ \langle \mu_{\mathbf{x}} \otimes \mu_{\mathbf{y}}, \mu_{\mathbf{x}} \otimes \mu_{\mathbf{y}} \rangle_{\mathcal{H}} \\ &= \mathbb{E}_{\mathbf{xy}} \mathbb{E}_{\mathbf{x}'\mathbf{y}'} [k(\mathbf{x}, \mathbf{x}') l(\mathbf{y}, \mathbf{y}')] - 2\mathbb{E}_{\mathbf{xy}} [\mathbb{E}_{\mathbf{x}'} [k(\mathbf{x}, \mathbf{x}')] \mathbb{E}_{y'} [l(\mathbf{y}, \mathbf{y}')]] \\ &+ \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{x}'} [k(\mathbf{x}, \mathbf{x}')] \mathbb{E}_{\mathbf{y}} \mathbb{E}_{\mathbf{y}'} [l(\mathbf{y}, \mathbf{y}')]. \end{aligned} \tag{4.30}$$

Note that the expression in (4.30) is the same as the one in (4.23). This means HSIC can also be interpreted as the squared distance between the Hilbert space embedding of $\Pr_{\mathbf{xy}}$ and $\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$. Furthermore, the above expression involves only the expectations over two kernel functions k and l. This is in contrast to the estimation of mutual information where density estimation is usually needed.

HSIC detects arbitrary dependence A nice property of HSIC is that with suitable kernels $\mathrm{HSIC} = 0$ if and only if $\mathbf{x}$ and $\mathbf{y}$ are independent. Hence HSIC is capable of detecting dependence of any kind. This results is proved in Theorem 4 of [59].

Theorem 34 (HSIC **and Independence)** *Let $\mathcal{F}, \mathcal{G}$ be RKHSs with universal kernels k, l on respective compact domains $\mathcal{X}$ and $\mathcal{Y}$ in the sense of [131], then* $\mathrm{HSIC} = 0$ *if and only if* $\mathbf{x}$ *and* $\mathbf{y}$ *are independent.*

Note that non-universal kernels can also be used for HSIC, although they may not guarantee that all dependence is detected. Different kernels incorporate distinctive prior knowledge into the dependence estimation, and they focus HSIC on dependence of a certain type. For instance, a linear kernel requires HSIC to seek only first order dependence, whereas a polynomial kernel of degree b restricts HSIC to test for dependences of degree (up to) b. Clearly HSIC is capable of finding and exploiting dependence of a much more general nature using kernels on graphs, strings, or other discrete domains.

4.5 Empirical Estimates of HSIC

We denote by $Z = (X, Y)$ the set of observations $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$ drawn iid. from $\Pr_{\mathbf{xy}}$. Denote by $\mathbb{E}_Z$ the expectation with respect Z. Moreover, denote by $\mathbf{K}, \mathbf{L} \in \mathbb{R}^{m \times m}$

kernel matrices obtained via $\mathbf{K}_{ij} = k(x_i, x_j)$ and $\mathbf{L}_{ij} = l(y_i, y_j)$. Finally let $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top \in \mathbb{R}^{m\times m}$ be the centering matrix which is a projection onto the space orthogonal to the vector $\mathbf{1}$. We can derive biased estimator of HSIC from these kernel matrices. Furthermore, by removing the bias and forming the U-statistics, we also provide an unbiased estimator of HSIC.

4.5.1 Biased Estimator

We derive a biased estimator of HSIC by estimating the three terms in (4.23) respectively using the kernel matrices $\mathbf{K}$ and $\mathbf{L}$ (Table 4.2).

Table 4.2: Biased estimator of the terms in HSIC.

Population Quantity	Biased Estimator
$\mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x}'\mathbf{y}'}[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')]$	$\frac{1}{m^2}\operatorname{tr}(\mathbf{KL})$
$\mathbb{E}_{\mathbf{xy}}[\mathbb{E}_{\mathbf{x}'}[k(\mathbf{x},\mathbf{x}')]\mathbb{E}_{\mathbf{y}'}[l(\mathbf{y},\mathbf{y}')]]$	$\frac{1}{m^3}\mathbf{1}^\top\mathbf{KL}\mathbf{1}$
$\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{x}'}[k(\mathbf{x},\mathbf{x}')]\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{y}'}[l(\mathbf{y},\mathbf{y}')]$	$\frac{1}{m^4}\mathbf{1}^\top\mathbf{K}\mathbf{1}\mathbf{1}^\top\mathbf{L}\mathbf{1}$

Combining the terms we have the following biased estimator of HSIC

$$\begin{aligned}
\widehat{\mathrm{HSIC}}_b &= \frac{1}{m^2}\operatorname{tr}(\mathbf{KL}) - \frac{2}{m^3}\mathbf{1}^\top\mathbf{KL}\mathbf{1} + \frac{1}{m^4}\mathbf{1}^\top\mathbf{K}\mathbf{1}\mathbf{1}^\top\mathbf{L}\mathbf{1} \\
&= \frac{1}{m^2}\left(\operatorname{tr}(\mathbf{KL}) - \frac{1}{m}\operatorname{tr}\left(\mathbf{KL}\mathbf{1}\mathbf{1}^\top\right) - \frac{1}{m}\operatorname{tr}\left(\mathbf{L}\mathbf{K}\mathbf{1}\mathbf{1}^\top\right) + \frac{1}{m^2}\operatorname{tr}\left(\mathbf{L}\mathbf{1}\mathbf{1}^\top\mathbf{K}\mathbf{1}\mathbf{1}^\top\right)\right) \\
&= \frac{1}{m^2}\left(\operatorname{tr}\left(\mathbf{KL}\left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\right)\right) - \frac{1}{m}\operatorname{tr}\left(\mathbf{K}\mathbf{1}\mathbf{1}^\top\mathbf{L}\left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\right)\right)\right) \\
&= \frac{1}{m^2}\operatorname{tr}\left(\mathbf{K}\left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\right)\mathbf{L}\left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\top\right)\right) \\
&= \frac{1}{m^2}\operatorname{tr}(\mathbf{KHLH})
\end{aligned} \tag{4.31}$$

The bias of $\widehat{\mathrm{HSIC}}_b$ arises from the self-interaction terms, i.e. we still have $O(m)$ terms of the form $\mathbf{K}_{ii}\mathbf{L}_{ii}$, $O(m^2)$ terms of the form $\mathbf{K}_{ii}\mathbf{L}_{ij}$ and $\mathbf{K}_{ij}\mathbf{L}_{ii}$, and $O(m^3)$ terms of the form $\mathbf{K}_{ii}\mathbf{L}_{js}$ and $\mathbf{K}_{js}\mathbf{L}_{ii}$ in the sum. However, the bias decreases as the sample size increases, with a rate of $O(m^{-1})$ [59].

Theorem 35 (Biased estimator of HSIC **[59])** *The estimator*

$$\widehat{\mathrm{HSIC}}_b := \frac{1}{m^2}\operatorname{tr}(\mathbf{KHLH}) \tag{4.32}$$

has bias $O(m^{-1})$*, i.e.* $\mathrm{HSIC} - \mathbb{E}_Z\left[\widehat{\mathrm{HSIC}}_b\right] = O(m^{-1})$.

To address this, we now devise an unbiased estimator by removing these self-interaction terms while ensuring proper normalization.

4.5.2 Unbiased Estimator

Our proposed unbiased estimator has the form

$$\widehat{\mathrm{HSIC}}_u := \frac{1}{m(m-3)}\left(\mathrm{tr}(\tilde{\mathbf{K}}\tilde{\mathbf{L}}) - \frac{2}{m-2}\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1} + \frac{\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1}\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1}}{(m-1)(m-2)}\right), \tag{4.33}$$

where $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ are related to $\mathbf{K}$ and $\mathbf{L}$ by $\tilde{\mathbf{K}}_{ij} = (1-\delta_{ij})\mathbf{K}_{ij}$ and $\tilde{\mathbf{L}}_{ij} = (1-\delta_{ij})\mathbf{L}_{ij}$ (i.e. the diagonal entries of $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$ are set to zero).

Theorem 36 (Unbiased estimator of HSIC) *The estimator $\widehat{\mathrm{HSIC}}_u$ is unbiased, that is, we have $\mathbb{E}_Z\left[\widehat{\mathrm{HSIC}}_u\right] = \mathrm{HSIC}$.*

Proof We prove the claim by constructing unbiased estimators for each term in (4.30). Note that we have three types of expectations, namely $\mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x'y'}}$, a partially decoupled expectation $\mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x'}}\mathbb{E}_{\mathbf{y'}}$, and $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{x'}}\mathbb{E}_{\mathbf{y'}}$ which takes all four expectations independently.

If we want to replace the expectations by empirical averages, we need to take care to avoid using the same discrete indices more than once for independent random variables. In other words, when taking expectations over n independent random variables, we need n-tuples of indices where each index occurs exactly once. We define the sets $\mathbf{i}_n^m$ to be the collections of indices satisfying this property. By simple combinatorics one can see that their cardinalities are given by the Pochhammer symbols $(m)_n = \frac{m!}{(m-n)!}$. Jointly drawn random variables, on the other hand, share the same index.

For the joint expectation over pairs we have

$$\mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x'y'}}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right] = \frac{1}{(m)_2}\mathbb{E}_Z\Big[\sum_{(i,j)\in\mathbf{i}_2^m}\mathbf{K}_{ij}\mathbf{L}_{ij}\Big] = \frac{1}{(m)_2}\mathbb{E}_Z\left[\mathrm{tr}\left(\tilde{\mathbf{K}}\tilde{\mathbf{L}}\right)\right]. \tag{4.34}$$

Recall that we set $\tilde{\mathbf{K}}_{ii} = \tilde{\mathbf{L}}_{ii} = 0$. In the case of the expectation over three independent terms $\mathbb{E}_{\mathbf{xy}}\mathbb{E}_{\mathbf{x'}}\mathbb{E}_{\mathbf{y'}}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right]$ we obtain

$$\frac{1}{(m)_3}\mathbb{E}_Z\Big[\sum_{(i,j,q)\in\mathbf{i}_3^m}\mathbf{K}_{ij}\mathbf{L}_{iq}\Big] = \frac{1}{(m)_3}\mathbb{E}_Z\left[\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1} - \mathrm{tr}\left(\tilde{\mathbf{K}}\tilde{\mathbf{L}}\right)\right]. \tag{4.35}$$

For four independent random variables $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{y}}\mathbb{E}_{\mathbf{x'}}\mathbb{E}_{\mathbf{y'}}\left[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')\right]$,

$$\frac{1}{(m)_4}\mathbb{E}_Z\Big[\sum_{(i,j,q,r)\in\mathbf{i}_4^m}\mathbf{K}_{ij}\mathbf{L}_{qr}\Big] = \frac{1}{(m)_4}\mathbb{E}_Z\left[\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1}\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1} - 4\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1} + 2\,\mathrm{tr}\left(\tilde{\mathbf{K}}\tilde{\mathbf{L}}\right)\right]. \tag{4.36}$$

To obtain an expression for HSIC we only need to take linear combinations using (4.30). Collecting terms related to $\mathrm{tr}\left(\tilde{\mathbf{K}}\tilde{\mathbf{L}}\right)$, $\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1}$, and $\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1}\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1}$ yields

$$\mathrm{HSIC} = \frac{1}{m(m-3)}\mathbb{E}_Z\left[\mathrm{tr}\left(\tilde{\mathbf{K}}\tilde{\mathbf{L}}\right) - \frac{2}{m-2}\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1} + \frac{\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1}\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1}}{(m-1)(m-2)}\right]. \tag{4.37}$$

This is the expected value of $\widehat{\mathrm{HSIC}}_u$. ■

Note that neither $\widehat{\mathrm{HSIC}}_b$ nor $\widehat{\mathrm{HSIC}}_u$ require any explicit regularization parameters, unlike ear-

lier work on kernel canonical correlation. The regularization is encapsulated in the choice of the kernels. Another key feature is that the computation for both $\widehat{\mathrm{HSIC}}_b$ and $\widehat{\mathrm{HSIC}}_u$ is simple: only the kernel matrices are needed. This is in contrast to information theoretic quantities, which requires density estimation and/or sophisticated bias correction strategies [e.g. 99].

4.5.3 Computation

Note that both $\widehat{\mathrm{HSIC}}_b$ and $\widehat{\mathrm{HSIC}}_u$ only need the kernel matrices $\mathbf{K}$ and $\mathbf{L}$. Assume that computing an entry in $\mathbf{K}$ and $\mathbf{L}$ takes constant time, then computing the full matrix takes $O(m^2)$ time. In term of the sample size m, we have the following analysis of the time complexity of $\widehat{\mathrm{HSIC}}_b$ and $\widehat{\mathrm{HSIC}}_u$ (by considering summation and multiplication as atomic operations)

- $\widehat{\mathrm{HSIC}}_b$: Centering $\mathbf{L}$ takes $O(m^2)$ time. Since $\mathrm{tr}(\mathbf{KHLH})$ is equivalent to $\mathbf{1}^\top(\mathbf{K} \circ \mathbf{HLH})\mathbf{1}$, it also takes $O(m^2)$ time. Overall, computing $\widehat{\mathrm{HSIC}}_b$ takes $O(m^2)$ time.
- $\widehat{\mathrm{HSIC}}_u$: Each of the three terms in $\widehat{\mathrm{HSIC}}_u$, namely $\mathrm{tr}(\tilde{\mathbf{K}}\tilde{\mathbf{L}})$, $\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1}\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1}$ and $\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1}$, takes $O(m^2)$ time. Overall, computing $\widehat{\mathrm{HSIC}}_u$ also takes $O(m^2)$ time.

Further speedup is also possible via a low rank approximation of the kernel matrices. Particularly, using incomplete Cholesky decomposition, [59] derive an efficient approximation of $\widehat{\mathrm{HSIC}}_b$. Formally, it can be summarized as the following lemma

Lemma 37 (Efficient Approximation to $\widehat{\mathrm{HSIC}}_b$) *Let $\mathbf{K} \approx \mathbf{A}\mathbf{A}^\top$ and $\mathbf{L} \approx \mathbf{B}\mathbf{B}^\top$, where $\mathbf{A} \in \mathbb{R}^{m\times d_f}$ and $\mathbf{B} \in \mathbb{R}^{m\times d_g}$. Then $\widehat{\mathrm{HSIC}}_b$ can be approximated in $O(m(d_f^2 + d_g^2))$ time.*

Note that in this case the dominant computation comes from the incomplete Cholesky decomposition, which can be carried out in $O(md_f^2)$ and $O(md_g^2)$ time respectively [46].

The three terms in $\widehat{\mathrm{HSIC}}_u$ can be computed analogously. Denote by $\mathbf{D}_\mathbf{K} = \mathrm{diag}(\mathbf{A}\mathbf{A}^\top)$ and $\mathbf{D}_\mathbf{L} = \mathrm{diag}(\mathbf{B}\mathbf{B}^\top)$ the diagonal matrices of the approximating terms. The latter can be computed in $O(md_f)$ and $O(md_g)$ time respectively. We have

$$\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1} = \mathbf{1}^\top(\mathbf{A}\mathbf{A}^\top - \mathbf{D}_\mathbf{K})\mathbf{1} = \|\mathbf{1}^\top\mathbf{A}\|^2 + \mathbf{1}^\top\mathbf{D}_\mathbf{K}\mathbf{1}. \tag{4.38}$$

Computation requires $O(md_f)$ time. The same holds when computing $\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1}$. To obtain the second term we exploit that

$$\begin{aligned}\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1} &= \mathbf{1}^\top(\mathbf{A}\mathbf{A}^\top - \mathbf{D}_\mathbf{K})(\mathbf{B}\mathbf{B}^\top - \mathbf{D}_\mathbf{K})\mathbf{1} \\ &= ((\mathbf{A}(\mathbf{A}^\top\mathbf{1})) - \mathbf{D}_\mathbf{K}\mathbf{1})^\top((\mathbf{B}(\mathbf{B}^\top\mathbf{1})) - \mathbf{D}_\mathbf{L}\mathbf{1}).\end{aligned} \tag{4.39}$$

This can be computed in $O(m(d_f + d_g))$. Finally, to compute the third term we use

$$\begin{aligned}\mathrm{tr}\left(\tilde{\mathbf{K}}\tilde{\mathbf{L}}\right) &= \mathrm{tr}(\mathbf{A}\mathbf{A}^\top - \mathbf{D}_\mathbf{K})(\mathbf{B}\mathbf{B}^\top - \mathbf{D}_\mathbf{L}) \\ &= \|\mathbf{A}^\top\mathbf{B}\|^2_{\mathrm{Frob}} - \mathrm{tr}\left(\mathbf{B}^\top\mathbf{D}_\mathbf{K}\mathbf{B}\right) - \mathrm{tr}\left(\mathbf{A}^\top\mathbf{D}_\mathbf{L}\mathbf{A}\right) + \mathrm{tr}\left(\mathbf{D}_\mathbf{K}\mathbf{D}_\mathbf{L}\right).\end{aligned} \tag{4.40}$$

This can be computed in $O(md_fd_g)$ time. It is the most costly part of all operations, since it takes all interactions between the reduced factorizations of $\mathbf{K}$ and $\mathbf{L}$ into account. Hence we may compute $\widehat{\mathrm{HSIC}}_u$ efficiently (note again that dominant computation comes from the incomplete Cholesky decomposition):

Lemma 38 (Efficient Approximation of $\widehat{\mathrm{HSIC}}_u$) *Let $\mathbf{K} \approx \mathbf{A}\mathbf{A}^\top$ and $\mathbf{L} \approx \mathbf{B}\mathbf{B}^\top$, where $\mathbf{A} \in \mathbb{R}^{m \times d_f}$ and $\mathbf{B} \in \mathbb{R}^{m \times d_g}$. Then $\widehat{\mathrm{HSIC}}_u$ can be approximated in $O(m(d_f^2 + d_g^2))$ time.*

4.5.4 $\widehat{\mathrm{HSIC}}$ is concentrated

$\widehat{\mathrm{HSIC}}_u$, the estimator in (4.33), can be alternatively formulated using U-statistics [69]. This reformulation allows us to derive a uniform convergence bound for $\widehat{\mathrm{HSIC}}_u$. Such a bound has practical importance. For instance, in term of feature selection, this means that the quality of features evaluated using $\widehat{\mathrm{HSIC}}_u$ closely reflects its population counterpart HSIC. Furthremore, the same features should consistently be selected to achieve high dependence if data are repeatedly drawn from the same distribution. We formalize these results below

Theorem 39 (U-statisic of HSIC) *$\widehat{\mathrm{HSIC}}_u$ can be rewritten in terms of a U-statistic*

$$\widehat{\mathrm{HSIC}}_u = \frac{1}{(m)_4} \sum_{(i,j,q,r) \in \mathbf{i}_4^m} h(i,j,q,r), \tag{4.41}$$

where the kernel h of the U-statistic is defined by

$$\begin{aligned} h(i,j,q,r) &= \frac{1}{24} \sum_{(s,t,u,v)}^{(i,j,q,r)} \mathbf{K}_{st} \left(\mathbf{L}_{st} + \mathbf{L}_{uv} - 2\mathbf{L}_{su}\right) \\ &= \frac{1}{6} \sum_{(s \prec t),(u \prec v)}^{(i,j,q,r)} \mathbf{K}_{st} \left(\mathbf{L}_{st} + \mathbf{L}_{uv}\right) - \frac{1}{12} \sum_{(s,t,u)}^{(i,j,q,r)} \mathbf{K}_{st}\mathbf{L}_{su}. \end{aligned} \tag{4.42}$$

Here the first sum represents all $4! = 24$ quadruples (s,t,u,v) which can be selected without replacement from (i,j,q,r). Likewise the sum over (s,t,u) is the sum over all triples chosen without replacement. Finally, the sum over $(s \prec t),(u \prec v)$ has the additional condition that the order imposed by (i,j,q,r) is preserved. That is (i,q) and (j,r) are valid pairs, whereas (q,i) or (r,j) are not.

Proof Combining the three unbiased estimators in (4.34-4.36) we obtain a single U-statistic

$$\widehat{\mathrm{HSIC}}_u = \frac{1}{(m)_4} \sum_{(i,j,q,r) \in \mathbf{i}_4^m} \left(\mathbf{K}_{ij}\mathbf{L}_{ij} + \mathbf{K}_{ij}\mathbf{L}_{qr} - 2\mathbf{K}_{ij}\mathbf{L}_{iq}\right). \tag{4.43}$$

In this form, however, the kernel $h(i,j,q,r) = \mathbf{K}_{ij}\mathbf{L}_{ij} + \mathbf{K}_{ij}\mathbf{L}_{qr} - 2\mathbf{K}_{ij}\mathbf{L}_{iq}$ is not symmetric in its arguments. For instance $h(i,j,q,r) \neq h(q,j,r,i)$. The same holds for other permutations of the indices. Thus, we replace the kernel with a symmetrized version, which yields

$$h(i,j,q,r) := \frac{1}{4!} \sum_{(s,t,u,v)}^{(i,j,q,r)} \left(\mathbf{K}_{st}\mathbf{L}_{st} + \mathbf{K}_{st}\mathbf{L}_{uv} - 2\mathbf{K}_{st}\mathbf{L}_{su}\right), \tag{4.44}$$

where the sum in (4.44) represents all ordered quadruples (s,t,u,v) selected without replacement from (i,j,q,r).

This kernel can be simplified, since $\mathbf{K}_{st} = \mathbf{K}_{ts}$ and $\mathbf{L}_{st} = \mathbf{L}_{ts}$. The first one only contains terms $\mathbf{K}_{st}\mathbf{L}_{st}$, hence the indices (u,v) are irrelevant. Exploiting symmetry we may impose $(s \prec t)$ without loss of generality. The same holds for the second term. The third term remains

unchanged, which completes the proof. ■

We now show that $\widehat{\mathrm{HSIC}}_u$ is concentrated and that it converges to HSIC with rate $O(m^{-1/2})$. This is a slight improvement over the convergence of the biased estimator $\widehat{\mathrm{HSIC}}_b$ proposed by [59]. We will use the a bound from Hoeffding [70] which applies to U-statistics of HSIC.

Theorem 40 (HSIC is Concentrated) *Assume k, l are bounded almost everywhere by 1, and are non-negative. Then for $m > 1$ and all $\delta > 0$, with probability at least $1 - \delta$ for all $\Pr_{\mathbf{xy}}$*

$$\left|\widehat{\mathrm{HSIC}}_u - \mathrm{HSIC}\right| \leq 4\sqrt{2\log(2/\delta)/m}. \tag{4.45}$$

Proof [Sketch] By virtue of (4.41) we see immediately that $\widehat{\mathrm{HSIC}}_u$ is a U-statistic of order 4, where each term is contained in $[-2, 2]$. Applying Hoeffding's bound for U-statistics we set

$$2\exp\left(-\frac{2\epsilon^2 m/4}{(b-a)^2}\right) = \delta \tag{4.46}$$

and obtain $\epsilon = 4\sqrt{2\log(2/\delta)/m}$. If k and l were just bounded by 1 in terms of absolute value the bound of Theorem 40 would be worse by a factor of 2. ■

The concentration result means that if we estimate HSIC from training data we can guarantee with high probability its deviation from the population counterpart is bounded. Furthermore, the deviation drops at rate $O(m^{1/2})$ as the sample size increase. This is desirable for learning, since we want to generalize the properties we obtain from the training data to future data. This concentration result provides us confidence on how close our empirical estimate is to the true quantity we try to estimate.

4.6 Hypothesis Test using HSIC

We now describe tests of statistical (in)dependence for two random variables, based on the test statistics for HSIC. We begin with an introduction to the terminology of statistical hypothesis testing [28, Chapter 8], as it applies to independence testing. Given iid. sample Z defined earlier, the statistical test, $\mathcal{T}(Z) : (\mathcal{X} \times \mathcal{Y})^m \mapsto \{0, 1\}$ is used to distinguish between the null hypothesis $H_0 : \Pr_{\mathbf{xy}} = \Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$ and the alternative hypothesis $H_1 : \Pr_{\mathbf{xy}} \neq \Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$. This is achieved by comparing the test statistic, in our case $\widehat{\mathrm{HSIC}}_u$ or $\widehat{\mathrm{HSIC}}_b$, with a particular threshold: if the threshold is exceeded, then the test rejects the null hypothesis (bearing in mind that a zero population HSIC indicates $\Pr_{\mathbf{xy}} = \Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$). The acceptance region of the test is thus defined as any real number below the threshold. Since the test is based on a finite sample, it is possible that an incorrect answer will be returned: the Type I error is defined as the probability of rejecting H_0 based on the observed sample, despite $\mathbf{x}$ and $\mathbf{y}$ being independent. Conversely, the Type II error is the probability of accepting $\Pr_{\mathbf{xy}} = \Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$ when the underlying variables are dependent. The level α of a test is an upper bound on the Type I error, and is a design parameter of the test, used to set the test threshold. A consistent test achieves a level α, and a Type II error of zero, in the large sample limit.

How, then, do we set the threshold of the test given α? The approach we adopt here is to derive the asymptotic distribution of the empirical estimate of HSIC under H_1 and H_0 respec-

tively. We then use the $1-\alpha$ quantile of this distribution as the test threshold.[2] Our presentation in this section is therefore divided into two main parts. First, we obtain the distribution of $\widehat{\mathrm{HSIC}}$ under both H_1 and H_0; the former distribution is also needed to ensure consistency of the test. We shall see, however, that the distribution under H_0 has a complex form, and cannot be evaluated directly. Thus, in the second part of this section, we describe ways to accurately approximate the $1-\alpha$ quantile of this distribution.

4.6.1 Asymptotic Normality under H_1

Theorem 40 gives *worst case* bounds on the deviation between HSIC and $\widehat{\mathrm{HSIC}}_u$. However, in many cases more accurate bounds for the *typical* case are needed. In particular, we would like to know the limiting distribution of $\widehat{\mathrm{HSIC}}_u$ for large sample sizes. We now show that, in fact, $\widehat{\mathrm{HSIC}}_u$ is asymptotically normal and we derive its variance. For the biased estimator $\widehat{\mathrm{HSIC}}_b$, since the bias drops at rate $O(m^{-1})$, we actually have exactly the same asymptotic distribution for $\widehat{\mathrm{HSIC}}_u$ and $\widehat{\mathrm{HSIC}}_b$. Therefore, in this section we will present the result for $\widehat{\mathrm{HSIC}}_u$ only.

Theorem 41 (Asymptotic Normality) *If $\mathbb{E}[h^2] < \infty$, and two random variables $\mathbf{x}$ and $\mathbf{y}$ are not independent, then their $\widehat{\mathrm{HSIC}}_u$ converges in distribution (a.s. $m \to \infty$) to a Gaussian random variable with mean* HSIC *and estimated variance*

$$\sigma^2_{\widehat{\mathrm{HSIC}}_u} = \frac{16}{m}\left(R - \widehat{\mathrm{HSIC}}_u^2\right), \tag{4.47}$$

where

$$R = \frac{1}{m}\sum_{i=1}^{m}\Big((m-1)_3^{-1}\sum_{(j,q,r)\in \mathbf{i}_3^m\setminus\{i\}} h(i,j,q,r)\Big)^2, \tag{4.48}$$

and $\mathbf{i}_n^m \setminus \{i\}$ denotes the set of all n-tuples drawn without replacement from $\{1,\ldots,m\}\setminus\{i\}$.

Proof [Sketch] This follows directly from [118, Theorem B, p. 193], which shows asymptotic normality of U-statistics. ■

Unfortunately (4.47) is expensive to compute by means of an explicit summation: even computing the kernel h of the U-statistic itself is a nontrivial task. For practical purposes we need an expression which can exploit fast matrix operations. As we shall see, $\sigma^2_{\widehat{\mathrm{HSIC}}_u}$ can be computed in $O(m^2)$, given the matrices $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$. To do so, we first form a vector $\mathbf{h}$ with its ith entry corresponding to $\sum_{(j,q,r)\in\mathbf{i}_3^m\setminus\{i\}} h(i,j,q,r)$. Collecting terms in (4.42) related to matrices $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{L}}$, $\mathbf{h}$ can be written as

$$\begin{aligned}\mathbf{h} =& (m-2)^2(\tilde{\mathbf{K}}\circ\tilde{\mathbf{L}})\mathbf{1} + (m-2)\Big((\operatorname{tr}\tilde{\mathbf{K}}\tilde{\mathbf{L}})\mathbf{1} - \tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1} - \tilde{\mathbf{L}}\tilde{\mathbf{K}}\mathbf{1}\Big) - m(\tilde{\mathbf{K}}\mathbf{1})\circ(\tilde{\mathbf{L}}\mathbf{1}) \\ &+ (\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1})\tilde{\mathbf{K}}\mathbf{1} + (\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1})\tilde{\mathbf{L}}\mathbf{1} - (\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1})\mathbf{1},\end{aligned} \tag{4.49}$$

where $\circ$ denotes elementwise matrix multiplication. Then R in (4.47) can be computed as $R = (4m)^{-1}(m-1)_3^{-2}\mathbf{h}^\top\mathbf{h}$. Combining this with the the unbiased estimator in (4.33) leads to the matrix computation of $\sigma^2_{\widehat{\mathrm{HSIC}}_u}$.

[2] An alternative would be to use a large deviation bound, as provided for instance by [59] based on Hoeffding's inequality. It has been reported in [58], however, that such bounds are generally too loose for hypothesis testing.

Variance of $\widehat{\mathrm{HSIC}}_u$ To compute the variance of $\widehat{\mathrm{HSIC}}_u$ we also need to deal with $(\tilde{\mathbf{K}} \circ \tilde{\mathbf{L}})\mathbf{1}$. For the latter, no immediate linear algebra expansion is available. However, we may use of the following decomposition. Assume that $\mathbf{a}$ and $\mathbf{b}$ are vectors in $\mathbb{R}^m$. In this case

$$((\mathbf{a}\mathbf{a}^\top) \circ (\mathbf{b}\mathbf{b}^\top))\mathbf{1} = (\mathbf{a} \circ \mathbf{b})(\mathbf{a} \circ \mathbf{b})^\top \mathbf{1}, \tag{4.50}$$

which can be computed in $O(m)$ time. Hence we may compute

$$((\mathbf{A}\mathbf{A}^\top) \circ (\mathbf{B}\mathbf{B}^\top))\mathbf{1} = \sum_{i=1}^{d_f} \sum_{j=1}^{d_g} ((\mathbf{A}_{\star i} \circ \mathbf{B}_{\star j})(\mathbf{A}_{\star i} \circ \mathbf{B}_{\star j})^\top)\mathbf{1}, \tag{4.51}$$

which can be carried out in $O(md_fd_g)$ time. To take care of the diagonal corrections note that $(\mathbf{A}\mathbf{A}^\top - \mathbf{D}_\mathbf{K}) \circ \mathbf{D}_\mathbf{L} = \mathbf{0}$. The same holds for $\mathbf{B}$ and $\mathbf{D}_\mathbf{K}$. The remaining term $\mathbf{D}_\mathbf{K}\mathbf{D}_\mathbf{L}\mathbf{1}$ is also computable in $O(m)$ time.

4.6.2 Asymptotic Distribution under H_0

Under H_0, however, the variance of $\widehat{\mathrm{HSIC}}_u$ is degenerate. It means that $\widehat{\mathrm{HSIC}}_u$ can no longer be approximated as a Gaussian distribution, and its variance $\sigma^2_{\widehat{\mathrm{HSIC}}_u}$ as computed from 4.47 becomes zero. In this case, the asymptotic distribution is an infinite sum of χ^2 distributions.

Theorem 42 *Under H_0, the U-statistic $\widehat{\mathrm{HSIC}}_u$ is degenerate, meaning $\mathbb{E}_i\left[h\left(i,j,q,r\right)\right] = 0$. In this case, $\widehat{\mathrm{HSIC}}_u$ converges in distribution according to [118, Section 5.5.2]*

$$\widehat{\mathrm{HSIC}}_b \longrightarrow \frac{1}{m}\sum_{l=1}^{\infty} \lambda_l(\chi_l^2 - 1) \quad \textit{a.s.} \quad m \longrightarrow \infty, \tag{4.52}$$

where χ_l^2 are iid. $\chi^2(1)$ random variables, and λ_l are the solutions to the eigenvalue problem

$$\int h\left(i,j,q,r\right)\psi_l(\mathbf{z}_j)\mathrm{d}\Pr_{\mathbf{z}_i\mathbf{z}_q\mathbf{z}_r} = \lambda_l\psi_l(\mathbf{z}_j), \tag{4.53}$$

where the integral is over the distribution of variables $\mathbf{z}_i$, $\mathbf{z}_q$, and $\mathbf{z}_r$.

Proof This follows from the discussion of [118, Section 5.5.2]. In this case, the asymptotic distribution for the corresponding V-statistic $\widehat{\mathrm{HSIC}}_b$ is also an infinite sum of χ^2 distributions. The difference is that the counterpart of (4.52) for the V-statistic is not centered, ie.

$$\widehat{\mathrm{HSIC}}_b \longrightarrow \frac{1}{m}\sum_{l=1}^{\infty} \lambda_l\chi_l^2 \quad \text{a.s.} \quad m \longrightarrow \infty, \tag{4.54}$$

■

A hypothesis test using $\widehat{\mathrm{HSIC}}_b$ can be derived from Theorem 42 above by computing the $(1-\alpha)$th quantile of the distribution (4.52), where consistency of the test (that is, the convergence to zero Type II error for $m \to \infty$) is guaranteed by the decay as $O(m^{-1})$. The distribution under H_0 is complicated. The question then becomes how to accurately approximate its quantiles.

One approach, taken by [45], is to use a Monte Carlo resampling technique: the ordering of the Y sample is permuted repeatedly while that of X is kept fixed, and the $1-\alpha$ quantile is obtained from the resulting distribution of $\widehat{\mathrm{HSIC}}_b$ values. This can be very expensive, however. A second approach, suggested in [77, p. 34], is to approximate the null distribution as a two-parameter Gamma distribution [76, p. 343, p. 359]: this is one of the more straightforward approximations of an infinite sum of $\chi^2(1)$ variables (see [76, Chapter 18.8] for further ways to approximate such distributions; in particular, we wish to avoid using moments of order greater than two, since these can become expensive to compute). Specifically, we make the approximation

$$\widehat{\mathrm{HSIC}}_b \sim \frac{x^{\alpha-1}e^{-x/\beta}}{m\beta^{\alpha}\Gamma(\alpha)}, \tag{4.55}$$

where

$$\alpha = \frac{(\mathbb{E}[\widehat{\mathrm{HSIC}}_b])^2}{\mathbb{V}[\widehat{\mathrm{HSIC}}_b]}, \quad \beta = \frac{\mathbb{V}[\widehat{\mathrm{HSIC}}_b]}{\mathbb{E}[\widehat{\mathrm{HSIC}}_b]}. \tag{4.56}$$

An illustration of the cumulative distribution function (CDF) obtained via the Gamma approximation is given in Figure 4.2, along with an empirical CDF obtained by repeated draws of HSIC_b. We note the Gamma approximation is quite accurate, especially in areas of high probability (which we use to compute the test quantile). The accuracy of this approximation will be further evaluated experimentally in Section 4.7.

To obtain the Gamma distribution from our observations, we need empirical estimates for $\mathbb{E}[\widehat{\mathrm{HSIC}}_b]$ and $\mathbb{V}[\widehat{\mathrm{HSIC}}_b]$ under the null hypothesis. Expressions for these quantities are given in [77, pp. 26-27], however these are in terms of the joint and marginal characteristic functions, and do not always apply in our more general kernel setting (see also [80, p. 313]). In the following two theorems, we provide much simpler expressions for both quantities, in terms of norms of mean elements $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$, and the variance operators

$$\mathcal{C}_{\mathbf{xx}} := \mathbb{E}_{\mathbf{x}}[(k(\mathbf{x},\cdot)-\mu_{\mathbf{x}}) \otimes (k(\mathbf{x},\cdot)-\mu_{\mathbf{x}})] \tag{4.57}$$

and $\mathcal{C}_{\mathbf{yy}}$ similarly, in feature space. The main advantage of our new expressions is that they are computed entirely in terms of kernels, which makes possible the application of the test to any domains on which kernels can be defined, and not only $\mathbb{R}^d$. Furthermore, we have the following two theorems

Theorem 43 *Under H_0,*

$$\begin{aligned}\mathbb{E}[\widehat{\mathrm{HSIC}}_b] = \frac{1}{m}(&\mathbb{E}_{\mathbf{xy}}\left[k(\mathbf{x},\mathbf{x})l(\mathbf{y},\mathbf{y})\right] + \|\mu_{\mathbf{x}}\|^2_{\mathcal{F}}\|\mu_{\mathbf{y}}\|^2_{\mathcal{G}} \\ &- \|\mu_{\mathbf{x}}\|^2_{\mathcal{F}}\mathbb{E}_{\mathbf{y}}\left[l(\mathbf{y},\mathbf{y})\right] - \|\mu_y\|^2_{\mathcal{G}}\mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x},\mathbf{x})\right]) + O(m^{-2}).\end{aligned} \tag{4.58}$$

An empirical estimate of this statistic is obtained by replacing the norms above with $\widehat{\|\mu_{\mathbf{x}}\|^2_{\mathcal{F}}} = (m)_2^{-1}\sum_{(i,j)\in \mathbf{i}^m_2} k(\mathbf{x}_i,\mathbf{x}_j)$.

Theorem 44 *Under H_0,*

$$\mathbb{V}[\widehat{\mathrm{HSIC}}_b] = \frac{2(m-4)(m-5)}{(m)_4}\|\mathcal{C}_{\mathbf{xx}}\|^2_{\mathrm{HS}}\|\mathcal{C}_{\mathbf{yy}}\|^2_{\mathrm{HS}} + O(m^{-3}). \tag{4.59}$$

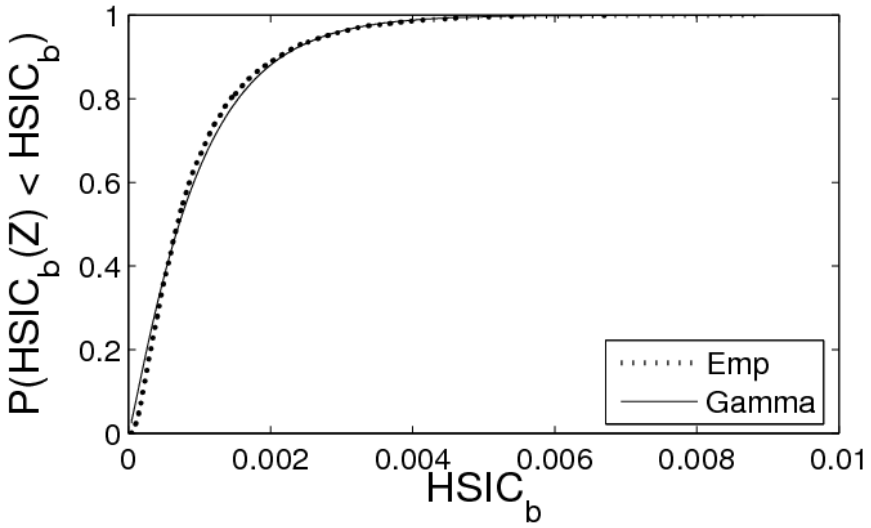


Figure 4.2: The cumulative distribution function of HSIC_b (Emp) under H_0 for $m = 200$, obtained empirically using 5000 independent draws of $\widehat{\mathrm{HSIC}}_b$. The two-parameter Gamma distribution (Gamma) is fit using $\alpha = 1.17$ and $\beta = 8.3 \times 10^{-4}$ in (4.55), with mean and variance computed via Theorems 43 and 44.

Denote by $\circ$ the element wise matrix product, $\mathbf{A}^{\cdot 2}$ the element wise matrix power, and $\mathbf{B} = ((\mathbf{HKH}) \circ (\mathbf{HLH}))^{\cdot 2}$. An empirical estimate with negligible bias may be found by replacing the product of covariance operator norms with $\mathbf{1}^\top (\mathbf{B} - \mathrm{diag}(\mathbf{B}))\, \mathbf{1}$: this is slightly more efficient than taking the product of the empirical operator norms (although the scaling with m is unchanged). We remark that all the above computation can be carried out in $O(m^2)$ time. It is also possible to further improve the test speed by approximating the Gram matrices using incomplete Cholesky decomposition.

4.7 Experiments

General tests of statistical independence are most useful for data having complex interactions that simple correlation does not detect. We investigate two cases where this situation arises: first, we test vectors in $\mathbb{R}^d$ which have a dependence relation but no correlation, as occurs in independent subspace analysis; and second, we study the statistical dependence between a text and its translation.

4.7.1 Independence of Subspaces

One area where independence tests have been applied is in determining the convergence of algorithms for independent component analysis (ICA), which involves separating random variables that have been linearly mixed, using only their mutual independence. ICA generally entails optimization over a non-convex function (including when HSIC is itself the optimization criterion [59]), and is susceptible to local minima, hence the need for these tests (in fact, for classical approaches to ICA, the *global* minimum of the optimization might not correspond to independence for certain source distributions). Contingency table-based tests have been applied [84] in this context, while the test of [77] has been used in [80] for verifying ICA outcomes when the data are stationary random processes (through using a subset of samples

with a sufficiently large delay between them). Contingency tests may be less useful in the case of independent subspace analysis (ISA, see e.g. [135] and its bibliography), where higher dimensional independent random vectors are to be separated. Thus, characteristic function-based tests [45, 77] and kernel independence measures may work better for this problem.

In our experiments, we tested the independence of random vectors, as a way of verifying the solutions of independent subspace analysis. We assume for ease of presentation that our subspaces have respective dimension $d_{\mathbf{x}} = d_{\mathbf{y}} = d$, but this is not required. The data are constructed as follows. First, we generate m samples of two univariate random variables, each drawn at random from the ICA benchmark densities in [61, Table 3]: these include super-Gaussian, sub-Gaussian, multimodal, and unimodal distributions. Second, we mix these random variables using a rotation matrix parameterized by an angle θ, varying from 0 to $\pi/4$ (a zero angle means the data are independent, while dependence becomes easier to detect as the angle increases to $\pi/4$: see the two plots in Figure 4.3, top left). Third, we add $d-1$ dimensional Gaussian noise of zero mean and unit standard deviation to each of the mixtures. Finally, we multiply the two resulting variables by a d-dimensional orthogonal matrix, such that the resulting vectors are dependent across all observed dimensions. We emphasize that classical approaches (such as Spearman's ρ or Kendall's τ) are completely unable to find this dependence, since the variables are uncorrelated; nor can we recover the subspace in which the variables are dependent using PCA, since this subspace has the same second order properties as the noise. We investigated sample sizes $m = 128, 512, 1024, 2048$, and $d = 1, 2, 4$.

We compared two different methods for computing the $1-\alpha$ quantile of the HSIC_b null distribution: repeated random permutation of the Y sample ordering as in [45] (*HSICp*), where we used 200 permutations; and Gamma approximation (*HSICg*), based on (4.55). We used a Gaussian kernel, with kernel size set to the median distance between points in input space. We also compared with the power-divergence based contingency table test of [109] (*PD*), which consisted in partitioning the space, counting the number of samples falling in each partition, and comparing this with the number of samples that would be expected under the null hypothesis (the test we used, described in [84], is more refined than this short description would suggest). Rather than a uniform space partitioning, we divided our space into roughly equiprobable bins as in [84], using a Gessaman partition for higher dimensions [33, Figure 21.4].[3] All remaining parameters were set according to [84]. Results are plotted in Figure 4.3. The y-intercept on these plots corresponds to the acceptance rate of H_0 at independence, or $1 - (\mathrm{Type\ I\ error})$, and should be close to the design parameter of $1 - \alpha = 0.95$. Elsewhere, the plots indicate acceptance of H_0 where the underlying variables are dependent, i.e. the Type II error.

As expected, we observe that dependence becomes easier to detect as θ increases from 0 to $\pi/4$, when m increases, and when d decreases. The *PD* approach performs poorly at $m = 128$, but approaches the performance of HSIC-based tests for increasing m (although it remains slightly worse at $m = 512$ and $d = 1$). *PD* also scales very badly with d, and never rejects the null hypothesis when $d = 4$, even for $m = 2048$. Although HSIC-based tests are unreliable for small θ, they generally do well as θ approaches $\pi/4$ (besides $m = 128,\ d = 2$). We also emphasise that *HSICp* and *HSICg* perform identically, although *HSICp* is far more costly (by a factor of around 100, given the number of permutations used).

[3] Ku and Fine did not specify a space partitioning strategy for higher dimensions, since they dealt only with univariate random variables.

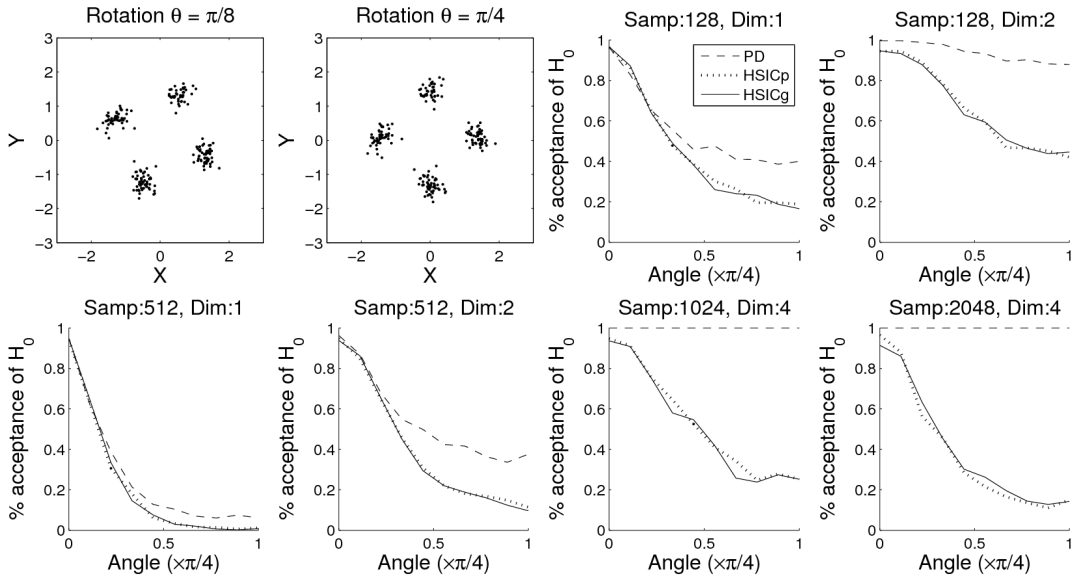


Figure 4.3: Top left plots: Example dataset for $d = 1$, $m = 200$, and rotation angles $\theta = \pi/8$ (left) and $\theta = \pi/4$ (right). In this case, both sources are mixtures of two Gaussians (source *(g)* in [61, Table 3]). We remark that the random variables appear "more dependent" as the angle θ increases, although their correlation is always zero. **Remaining plots:** Rate of acceptance of $\mathcal{H}_0$ for the *PD*, *HSICp*, and *HSICg* tests. "Samp" is the number m of samples, and "dim" is the dimension d of $\mathbf{x}$ and $\mathbf{y}$.

4.7.2 Dependence between Text

In this section, we demonstrate independence testing on text. Our data are taken from the Canadian Hansard corpus (http://www.isi.edu/natural-language/download/hansard/). These consist of the official records of the 36th Canadian parliament, in English and French. We used debate transcripts on the three topics of Agriculture, Fisheries, and Immigration, due to the relatively large volume of data in these categories. Our goal was to test whether there exists a statistical dependence between English text and its French translation. Our dependent data consisted of a set of paragraph-long (5 line) English extracts and their French translations. For our independent data, the English paragraphs were matched to random French paragraphs on the same topic: for instance, an English paragraph on fisheries would always be matched with a French paragraph on fisheries. This was designed to prevent a simple vocabulary check from being used to tell when text was mismatched. We also ignored lines shorter than five words long, since these were not always part of the text (e.g. identification of the person speaking).

We used the k-spectrum kernel of [88], computed according to the method of [134]. We set $k = 10$ for both languages, where this was chosen by cross validating on an SVM classifier for Fisheries vs National Defense, separately for each language (performance was not especially sensitive to choice of k; $k = 5$ also worked well). We compared this kernel with a simple kernel between bags of words [26, pp. 186–189]. Results are in Table 4.3.

Our results demonstrate the excellent performance of the *HSICp* test on this task: even for small sample sizes, *HSICp* with a spectral kernel always achieves zero Type II error, and a Type I error close to the design value (0.95). We further observe for $m = 10$ that *HSICp* with the spectral kernel always has better Type II error than the bag-of words kernel. This suggests that a kernel with a more sophisticated encoding of text structure induces a more sensitive test,

Table 4.3: Independence tests for cross-language dependence detection. Topics are in the first column, where the total number of 5-line extracts for each dataset is in parentheses. BOW(10) denotes a bag of words kernel and $m = 10$ sample size, Spec(50) is a k-spectrum kernel with $m = 50$. The first entry in each cell is the null acceptance rate of the test under H_0 (i.e. $1 - (\text{Type I error})$; should be near 0.95); the second entry is the null acceptance rate under H_1 (the Type II error, small is better). Each entry is an average over 300 repetitions.

Topic	BOW(10)		Spec(10)		BOW(50)		Spec(50)	
	HSICg	*HSICp*	*HSICg*	*HSICp*	*HSICg*	*HSICp*	*HSICg*	*HSICp*
Agriculture (555)	1.00, 0.99	0.94, 0.18	1.00, 1.00	0.95, 0.00	1.00, 0.00	0.93, 0.00	1.00, 0.00	0.95, 0.00
Fisheries (408)	1.00, 1.00	0.94, 0.20	1.00, 1.00	0.94, 0.00	1.00, 0.00	0.93, 0.00	1.00, 0.00	0.95, 0.00
Immigration (289)	1.00, 1.00	0.96, 0.09	1.00, 1.00	0.91, 0.00	0.99, 0.00	0.94, 0.00	1.00, 0.00	0.95, 0.00

although for larger sample sizes, the advantage vanishes. The *HSICg* test does less well on this data, always accepting H_0 for $m = 10$, and returning a Type I error of zero, rather than the design value of 5%, when $m = 50$. It appears that this is due to a very low variance estimate returned by the Theorem 44 expression, which could be caused by the high diagonal dominance of kernels on strings. Thus, while the test threshold for *HSICg* at $m = 50$ still fell between the dependent and independent values of $\widehat{\text{HSIC}}_b$, this was not the result of an accurate modelling of the null distribution. We would therefore recommend the permutation approach for this problem. Finally, we also tried testing with 2-line extracts and 10-line extracts, which yielded similar results.

4.8 Summary

We have introduced a measure of dependence based on Hilbert space embedding of distributions. This new measure computes the Hilbert space distance between the embedded joint distribution $\Pr_{\mathbf{xy}}$ and the product of the marginals $\Pr_{\mathbf{x}} \Pr_{\mathbf{y}}$. For iid. data, we recovered Hilbert-Schmidt norm of the cross covariance operator (HSIC) as a special case. We also derived unbiased estimator and concentration results for HSIC.

HSIC has the flexibility of choosing kernels, and different kernels can be applied to the two domains being tested. Therefore, we can use HSIC for observations from structured domains. For instance, we can use it to detect dependence between texts and their translations. Furthermore, we can also use HSIC to detect dependence between data of completely different types, such as between images and captions.The choice of kernels also allow us to incorporate prior knowledge into the dependence estimation process. A good choice of kernels on the domains being tested can result in very sensitive tests at small sample sizes.

More interestingly, we can also use HSIC for various learning problems. As we discussed in Chapter 3, we can use dependence for feature selection, clustering and dimensionality reduction. In the next three chapters, we will use these several learning problems as examples to demonstrate how we can perform learning via dependence. In particular, when express these learning problems using HSIC, we not only design new algorithms, but also recover many existing algorithms as special cases.

CHAPTER 5

Feature Selection via Dependence Estimaton

In this chapter, we will use dependence as measured by HSIC for feature selection. This problem can be viewed as choosing a subset of features that are most relevant to the labels. We show that selecting features via HSIC subsumes many existing feature selectors as special cases. We also design an algorithm that combines HSIC and a backward elimination strategy for feature selection. We demonstrate its good performance in various real world datasets.

5.1 Introduction

In data analysis we are typically given a set of observations $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subseteq \mathcal{X}$ which can be used for a number of learning tasks, such as novelty detection, low-dimensional representation, or a range of supervised problems. In the latter case we also have a set of labels $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \subseteq \mathcal{Y}$ at our disposal. Supervised tasks include ranking, classification, regression, or sequence annotation. While not always true in practice, we assume in the following, that the data X and Y have been generated identically independently distributed (iid.) from some underlying distribution $\Pr_{\mathbf{xy}}$.

We often want to reduce the dimension of the data (the number of features) before the actual learning [64]; a larger number of features can be associated with higher data collection cost, more difficulty in model interpretation, higher computational cost for the classifier, and *sometimes* decreased generalization ability. In other words, often there exist ulterior motives than finding a well performing estimator which make feature selection a necessity for reasons of speed or deployment cost. It is therefore important to select an informative feature subset.

The problem of supervised feature selection can be cast as a combinatorial optimization problem. We have a full set of features, denoted by $\mathcal{S}$ (each element in $\mathcal{S}$ corresponds to one dimension of the data). It is our aim to select a subset $\mathcal{T} \subseteq \mathcal{S}$ such that this subset retains the relevant information contained in X. Given m observations, suppose the relevance of a feature subset (to the outcome) is quantified by $\mathcal{Q}_m(\mathcal{T})$, and is computed by restricting the data to the dimensions in $\mathcal{T}$. Feature selection can then be formulated as

$$\mathcal{T}_0 = \arg\max_{\mathcal{T} \subseteq \mathcal{S}} \mathcal{Q}_m(\mathcal{T}) \quad \text{subject to } |\mathcal{T}| \leq t, \tag{5.1}$$

where $|\cdot|$ computes the cardinality of a set and t is an upper bound on the number of selected features. Two important aspects of problem (5.1) are the choice of the criterion $\mathcal{Q}_m(\mathcal{T})$ and the selection algorithm.

5.1.1 Criteria for Feature Selection

A number of quality functionals $\mathcal{Q}_m(\mathcal{T})$ are potential candidates to use for feature selection. For instance we could use a estimator of mutual-information or a Hilbert Space based estimator. In any case, the choice of $\mathcal{Q}_m(\mathcal{T})$ should respect the underlying tasks: supervised learning estimates functional dependence f from training data and guarantees f predicts well on test data. Therefore, good criteria should satisfy two conditions:

I: $\mathcal{Q}_m(\mathcal{T})$ is capable of detecting desired (linear or nonlinear) functional dependence between the data and the labels.

II: $\mathcal{Q}_m(\mathcal{T})$ is concentrated with respect to the underlying measure. This guarantees with high probability that detected functional dependence is preserved in test data.

While many criteria have been explored few take these two conditions explicitly into account. Examples include the leave-one-out error bound of SVM [154] and the mutual information [156]. Although the latter has good theoretical justification, it requires density estimation, which is problematic for high dimensional and continuous variables. We sidestep these problems by employing a mutual-information *like* quantity — the Hilbert-Schmidt Independence Criterion (HSIC) we introduced in Chapter 4. HSIC uses kernels to measure the dependence between the feature and the labels. It detects nonlinear as linear dependence when universal kernels are used. Furthermore, empirical estimate of HSIC has good uniform convergence guarantees. Therefore HSIC satisfies conditions I and II, required for a good feature selection criterion.

5.1.2 Feature Selection Algorithms

Finding a global optimum for (5.1) is NP-hard in general [153], unless the criterion is easily decomposable or has properties which make optimization easier, e.g. submodularity [63]. Many algorithms transform (5.1) into a continuous problem by introducing weights on the dimensions [154, 22, 153, 98]. These methods perform well for linearly separable problems. For nonlinear problems, however, the optimization usually becomes non-convex and a local optimum does not necessarily provide good features. Greedy approaches, forward selection and backward elimination, are often used to tackle problem (5.1) directly. Forward selection tries to increase $\mathcal{Q}_m(\mathcal{T})$ as much as possible for each inclusion of features, and backward elimination tries to achieve this for each deletion of features [65]. Although forward selection is computationally more efficient, backward elimination provides better features in general since the features are assessed within the context of all others. See Section 5.5 for experimental details.

In principle, the Hilbert-Schmidt independence criterion can be employed for feature selection using either a weighting scheme, forward selection or a backward selection strategy, or even a mix of several strategies. While the main focus of this paper is on the backward elimination strategy, we also discuss the other selection strategies. As we shall see, several specific choices of a kernel function will lead to well known feature selection and feature rating methods.

Note that backward elimination using HSIC (BAHSIC) is a filter method for feature selection. It selects features independent of a particular classifier. Such decoupling not only facilitates subsequent feature interpretation but also speeds up the computation over wrapper and embedded methods.

We will see that BAHSIC is directly applicable to binary, multiclass, and regression problems. Most other feature selection methods are only formulated either for binary classification

or regression. Multi-class extension of these methods is usually accomplished using a one-versus-the-rest strategy. Still fewer methods handle classification and regression cases at the same time. BAHSIC, on the other hand, accommodates all these cases *and* unsupervised feature selection in a principled way: by choosing different kernels, BAHSIC not only subsumes many existing methods as special cases, but also allows us to define new feature selectors. Such versatility of BAHSIC originates from the generality of HSIC.

5.2 Feature Selection via HSIC

We will use HSIC as the selection *criterion*, and we now describe *algorithms* that conduct feature selection on the basis of this dependence measure. Denote by $\mathcal{S}$ the full set of features, $\mathcal{T}$ a subset of features ($\mathcal{T} \subseteq \mathcal{S}$). We want to find $\mathcal{T}$ such that the dependence between features in $\mathcal{T}$ and the labels is maximized. Moreover, we may choose between different feature selection strategies, that is, whether we would like to build up a catalog of features in an incremental fashion (forward selection) or whether we would like to remove irrelevant features from a catalog (backward selection). For certain kernels, such as a linear kernel, both selection methods are equivalent: the objective function decomposes into individual coordinates, and thus feature selection can be done without recursion in one go. Although forward selection is computationally more efficient, backward elimination in general yields better features (especially for nonlinear features), since the quality of the features is assessed within the context of all other features [64].

5.2.1 Backward Elimination Using HSIC (BAHSIC)

BAHSIC works by generating a list $\mathcal{S}^\dagger$ which contains the features in increasing degree of relevance. At each step $\mathcal{S}^\dagger$ is appended by a feature from $\mathcal{S}$ which is not contained in $\mathcal{S}^\dagger$ yet by selecting the features which are least dependent on the reference set (i.e. Y or the full set X).

Once we perform this operation, the feature selection problem in (5.1) can be solved by simply taking the last t elements from $\mathcal{S}^\dagger$. Our algorithm produces $\mathcal{S}^\dagger$ recursively, eliminating the least relevant features from $\mathcal{S}$ and adding them to the end of $\mathcal{S}^\dagger$ at each iteration. Note that for feature selection, the kernel matrix for the labels remain unchanged. For convenience, we denote HSIC as $\mathrm{HSIC}(\sigma, \mathcal{S})$, where $\mathcal{S}$ are the features used in computing the data kernel matrix $\mathbf{K}$, and σ is the parameter for the data kernel (for instance, this might be the size of a Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma \|\mathbf{x} - \mathbf{x}'\|^2)$). The algorithm is presented in Algorithm 1.

Algorithm 1 BAHSIC

Input: The full set of features $\mathcal{S}$
Output: An ordered set of features $\mathcal{S}^\dagger$

1: $\mathcal{S}^\dagger \leftarrow \varnothing$
2: **repeat**
3: $\quad\sigma \leftarrow \Xi$
4: $\quad\mathcal{I} \leftarrow \arg\max_{\mathcal{I}} \sum_{j \in \mathcal{I}} \mathrm{HSIC}(\sigma, \mathcal{S} \setminus \{j\}),\ \ \mathcal{I} \subset \mathcal{S}$
5: $\quad\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{I}$
6: $\quad\mathcal{S}^\dagger \leftarrow \mathcal{S}^\dagger \cup \mathcal{I}$
7: **until** $\mathcal{S} = \varnothing$

Step 3 of the algorithm denotes a policy for adapting the kernel parameters. Depending on

the availability of prior knowledge and the type of preprocessing, we explored three types of policies

1. If we have prior knowledge about the nature of the nonlinearity in the data, we can use a fixed kernel parameter throughout the iterations. For instance, we can use a polynomial kernel of fixed degree, e.g. $(\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^2$, to select the features for the XOR dataset showed in Figure 5.1(a).

2. If we have no prior knowledge, we can optimize HSIC over a set of kernel parameters. In this case, the policy corresponds to $\arg\max_{\sigma \in \Theta} \mathrm{HSIC}(\sigma, \mathcal{S})$, where Θ is a set of parameters that ensure the kernel is bounded. For instance, σ can be the scale parameter of a Gaussian kernel, $k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma \left\|\mathbf{x} - \mathbf{x}'\right\|^2)$. Optimizing over the scaling parameter allows us to adapt to the scale of the nonlinearity present in the (feature-reduced) data.

3. Adapting kernel parameters via optimization is computational intensive. Alternatively we can use a policy that produces approximate parameters in each iteration. For instance, if we assume features to be independent of each other and normalize each feature separately to zero mean and unit variance, we know that the expected value of the distance between data points, $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{x}'}\left[(\mathbf{x} - \mathbf{x}')^2\right]$, is $2d$ (d is the dimension of the data). When using a Gaussian kernel, we can then use a policy that assigns σ to $1/(2d)$ as the dimension of the data is reduced.

Step 4 of the algorithm is concerned with the selection of a set $\mathcal{I}$ of features to eliminate. While one could choose a single element of $\mathcal{S}$, this would be inefficient when there are a large number of irrelevant features. On the other hand, removing too many features at once risks the loss of relevant features. In our experiments, we found a good compromise between speed and feature quality is to remove 10% of the current features at each iteration.

In BAHSIC, the kernel matrix $\mathbf{L}$ for the labels is fixed through the whole process. It can be precomputed and stored for speedup if needed. Therefore, the major computation comes from repeated calculation of the kernel matrix $\mathbf{K}$ for the dimension-reduced data. Assume that computing an entry in $\mathbf{K}$ requires constant time irrespective of the dimension of the data, then the ith iteration of BAHSIC takes $O(\beta^{i-1} d m^2)$ time (d is the total number of features, and $1 - \beta$ is the portion of removed features in each iteration; therefore $\beta^{i-1} d$ features remain after $i - 1$ iterations, and we have m^2 elements in the kernel matrix in total). If we want to reduce the number of features to t we need at most $\tau = \log_\beta(t/d)$ iterations. This brings the total time complexity to $O\left(\frac{1-\beta^\tau}{1-\beta} d m^2\right) = O\left(\frac{d-t}{1-\beta} m^2\right)$ operations. When using incomplete Cholesky factorization we may reduce computational complexity somewhat further to $O\left(\frac{d-t}{1-\beta} m (d_f^2 + d_g^2)\right)$ time. This saving is significant as long as $d_f d_g < m$, which may happen, for instance whenever $\mathbf{Y}$ is a binary label matrix. In this case $d_g = 1$, hence incomplete factorizations may yield significant computational gains.

5.2.2 Forward Selection Using HSIC (FOHSIC)

FOHSIC uses the converse approach to backward selection: it builds a list of features in *decreasing* degree of relevance. This is achieved by adding one feature at a time to the set of features $\mathcal{S}^\dagger$ obtained so far using HSIC as a criterion for the quality of the so-added features. For faster selection of features, we can choose a group of features (for instance, a fixed proportion γ) at step 4 and add them in one shot at step 6. The adaptation of kernel parameters in step

3 follows the same policies as those for BAHSIC. The feature selection problem in (5.1) can be solved by simply taking the *first* t elements from $\mathcal{S}^\dagger$.

Algorithm 2 FOHSIC

Input: The full set of features $\mathcal{S}$, desired number of features t
Output: An ordered set of features $\mathcal{S}^\dagger$

1: $\mathcal{S}^\dagger \leftarrow \varnothing$
2: **repeat**
3: $\quad\sigma \leftarrow \Xi$
4: $\quad\mathcal{I} \leftarrow \arg\max_{\mathcal{I}} \sum_{j\in\mathcal{I}} \mathrm{HSIC}(\sigma, \mathcal{S}^\dagger \cup \{j\}),\ \mathcal{I} \subset \mathcal{S}$
5: $\quad\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{I}$
6: $\quad\mathcal{S}^\dagger \leftarrow \mathcal{S}^\dagger \cup \mathcal{I}$
7: **until** $|\mathcal{S}^\dagger| = t$

Under the same assumption as BAHSIC, the ith iteration of FOHSIC takes $O((1-\gamma)^{i-1}dm^2)$ time. The total number of iterations τ to obtain t features is $t = [1-(1-\gamma)^\tau]d$, that is $\tau = \frac{\log(d-t)-\log d}{\log(1-\gamma)}$ iterations. Performing τ steps will therefore take $\sum_{i=0}^{\tau-1} d(1-\gamma)^i = d(1-(1-\gamma)^\tau)/\gamma = t/\gamma$ operations. This means that FOHSIC takes $O(tm^2/\gamma)$ time to extract t features.

5.2.3 Feature Weighting Using HSIC

Besides backward elimination algorithm, feature selection using HSIC can also proceed by converting problem (5.1) into a continuous optimization problem. By adding a penalty on the number of nonzero terms, such as a relaxed L_0 "norm" of a weight vector over the features we are able to solve the problem with continuous optimization methods. Unfortunately, this approach does not perform as well as the the backward elimination procedure proposed in the main text. For completeness and since related methods are somewhat popular in the literature, the approach is described below.

We introduce a weighting $\mathbf{w} \in \mathbb{R}^n$ on the dimensions of the data: $\mathbf{x} \longmapsto \mathbf{w} \circ \mathbf{x}$, where $\circ$ denotes element-wise product. Thus feature selection using HSIC becomes an optimization problem with respect to $\mathbf{w}$ (for convenience we write HSIC as a function of $\mathbf{w}$, $\mathrm{HSIC}(\mathbf{w})$). To obtain a sparse solution of the selected features, the zero "norm" $\|\mathbf{w}\|_0$ is also incorporated into our objective function (clearly $\|.\|_0$ is not a proper norm). $\|\mathbf{w}\|_0$ computes the number of non-zero entries in $\mathbf{w}$ and the sparsity is achieved by imposing heavier penalty on solutions with large number of non-zero entries. In summary, feature selection using HSIC can be formulated as

$$\mathbf{w} = \arg\max_{\mathbf{w}} \ \mathrm{HSIC}(\mathbf{w}) - \lambda \|\mathbf{w}\|_0 , \tag{5.2}$$

where $\mathbf{w} \in [0,\infty)^n$. The zero "norm" is not a continuous function. However, it can be approximated well by a concave function [50] ($\alpha = 5$ works well in practice):

$$\|\mathbf{w}\|_0 \approx \mathbf{1}^\top(\mathbf{1} - \exp(-\alpha\mathbf{w})). \tag{5.3}$$

While the optimization problem in (5.2) is non-convex, we may use relatively more efficient optimization procedures for the concave approximation of the L_0 norm. For instance, we may use the convex-concave procedure (CCCP) of [155]. For a Gaussian kernel HSIC can be

decomposed into the sum of a convex and a concave function:

$$\mathrm{HSIC}(\mathbf{w}) - \lambda\|\mathbf{w}\|_0 \approx \mathrm{tr}(\mathbf{K}(\mathbf{I} - m^{-1}\mathbf{1}\mathbf{1}^\top)\mathbf{L}(\mathbf{I} - m^{-1}\mathbf{1}\mathbf{1}^\top)) - \lambda\mathbf{1}^\top(\mathbf{1} - e^{-\alpha\mathbf{w}}). \quad (5.4)$$

Depending on the choice of $\mathbf{L}$ we need to assign all terms involving $\exp$ with positive coefficients into the convex and all terms involving negative coefficients to the concave function.

5.3 Variants of BAHSIC

So far we discussed a set of algorithms to select features *once* we decided to choose a certain family of kernels k, l to measure dependence between two sets of observations. We now proceed to discussing a number of design choices for k and l. This will happen in two parts: in the current section we discuss generic choices of kernels on data and labels. Various combinations of such kernels will then lead to new algorithms that aim to discover different types of dependence between features and labels (or between a full and a restricted dataset whenever we are interested in unsupervised feature selection). After that (in section 5.4) we will study specific choices of kernels which correspond to existing feature selection methods.

5.3.1 Kernels on Data

There exists a great number of kernels on data. Different kernels induce distinctive similarity measure on the data and will correspond to a range of different assumptions on the type of dependence between random variables $\mathbf{x}$ and $\mathbf{y}$. For instance

Linear kernel The simplest choice for k is to take a linear kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$. This means that we are just using the underlying Euclidean space to define the similarity measure. Whenever the dimensionality d of $\mathbf{x}$ is very high, this may allow for more complexity in the function class than what we could measure and assess otherwise. An additional advantage of this setting is that the kernel decomposes into the sum of products between individual coordinates. This means that any expression of the type $\mathrm{tr}\,(\mathbf{KM})$ can be maximized with respect to the subset of available features via

$$\mathrm{tr}\,(\mathbf{KM}) = \mathrm{tr}\left(\mathbf{X}\mathbf{X}^\top\mathbf{M}\right) = \sum_{j=1}^{d} \mathbf{X}_{\star j}^\top \mathbf{M} \mathbf{X}_{\star j}. \quad (5.5)$$

This means that the optimality criterion decomposes into a sum over the scores of individual coordinates. Hence maximization with respect to a subset of size t is trivial, since it just involves finding the t largest contributors. In this case both FOHSIC and BAHSIC generate the *optimal* feature selection with respect to the criterion applied.

Polynomial kernel Clearly in some cases the use of linear features can be quite limiting. It is possible, though, to use higher order correlations between data for the purpose of feature selection. This is achieved by using a polynomial kernel $k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + a)^b$ $(a \geq 0, b \in \mathbb{N})$. This kernel incorporates all polynomial interactions up to degree b (provided that $a > 0$). For instance, if we wanted to take only mean and variance into account, we would only need to consider $b = 2$ and $a = 1$. Placing a higher emphasis on means is achieved by increasing the constant offset a.

Radial Basis Function kernel Note that polynomial kernels only map data into a *finite* dimensional space: while potentially huge, the dimensionality of polynomials of bounded degree is finite, hence criteria arising from such kernels will not provide us with guarantees for detecting dependence of any kind. On the other hand, many radial basis function kernels, such as the Gaussian RBF kernel map $\mathbf{x}$ into an *infinite* dimensional space. One may show that these kernels are in fact universal in the sense of [131]. That is, we use kernels of the form

$$k(\mathbf{x}, \mathbf{x}') = \kappa(\|\mathbf{x} - \mathbf{x}'\|), \tag{5.6}$$

where $\kappa(\cdot) = \exp(-\cdot)$ and $\kappa(\cdot) = \exp(-\cdot^2)$ to obtain Laplace and Gaussian kernels respectively. Since the spectrum of the corresponding matrices is rapidly decaying it is easy to compute incomplete Cholesky factorizations of the kernel matrix efficiently.

String and Graph kernel One of the key advantages of our approach is that it is not limited to vectorial data. For instance, we can perform feature selection on strings or graphs. For string kernels we have

$$k(\mathbf{x}, \mathbf{x}') = \sum_{a \sqsubseteq \mathbf{x}} w_a \#_a(\mathbf{x}) \#_a(\mathbf{x}'), \tag{5.7}$$

where $a \sqsubseteq \mathbf{x}$ is a substring of $\mathbf{x}$ [144, 89]. Similar decompositions can be made for graphs, where kernels on random walks and paths can be defined. As before, we could use BAHSIC to remove or FOHSIC to generate a list of features such that only relevant ones remain. That said, given that such kernels are additive in their features, we can use the same argument as made above for linear kernels to determine meaningful features in one go.

5.3.2 Kernels on Labels

The kernels on the data described our inherent assumptions on which properties of $\mathbf{x}$ (e.g. linear, polynomial, or nonparametric) are relevant for estimation. We now describe the complementary part, namely a set of possible choices for kernels on labels. Note that these kernels can be just as general as those defined on the data. This means that we may apply our algorithms to classification, regression, ranking, etc. in the same fashion. This is a significant difference to previous approaches which only apply to specialized settings such as binary classification. For completeness we begin with the latter.

Binary Classification The simplest kernel we may choose is

$$l(y, y') = yy', \tag{5.8}$$

where $y, y' \in \{\pm 1\}$. In this case the label kernel matrix $\mathbf{L} = \mathbf{y}\mathbf{y}^\top$ has rank 1 and it is simply the outer product of the vector of labels. Note that we could transform l by adding a positive constant c to obtain $l(y, y') = yy' + c$ which yields $l(y, y') = 2\delta_{y,y'}$ for $c = 1$. This transformation, however, is immaterial: once $\mathbf{K}$ has been centered it is orthogonal to constant matrices.

A second transformation, however, leads to nontrivial changes: we may change the relative weights of positive and negative classes. This is achieved by transforming $y \to c_y y$. For

instance, we may pick $c_+ = \frac{1}{m_+}$ and $c_- = \frac{1}{m_-}$. That is, we choose

$$\mathbf{y} = \left(\frac{1}{m_+} \mathbf{1}_{m_+}^\top, \frac{1}{m_-} \mathbf{1}_{m_-}^\top \right)^\top, \tag{5.9}$$

which leads to $l(y, y') = \frac{yy'}{m_y m_{y'}}$

That is, we give different weight to positive and negative class according to their sample size. As we shall see in the next section, this corresponds to making the feature selection independent of the class size and it will lead to criteria derived from the estimator of Maximum Mean Discrepancy [58].

Multiclass Classification Here we have a somewhat larger choice of options to contend with. Clearly the simplest kernel would be

$$l(y, y') = c_y \delta_{y,y'}, \tag{5.10}$$

where $c_y > 0$ and $y, y' \in \{1, \ldots, n\}$ for n classes. For $c_y = \frac{1}{m_y}$ we obtain a per-class normalization. Clearly, for n classes, the kernel matrix $\mathbf{L}$ can be represented by the outer product of a rank-n matrix, where each row is given by $c_y \mathbf{e}_y^\top$, where $\mathbf{e}_y$ denotes the y-th unit vector in $\mathbb{R}^n$. Alternatively, we may adjust the inner product between classes to obtain $l(y, y') = \langle \psi(y), \psi(y') \rangle$ where

$$\psi(y) = \frac{m}{m_y(m - m_y)} \mathbf{e}_y - \mathbf{z} \quad \text{and} \quad \mathbf{z} = (\frac{1}{m - m_1}, \ldots, \frac{1}{m - m_n})^\top. \tag{5.11}$$

This corresponds to assigning a "one versus the rest" feature to each class and taking the inner product between them. As before in the binary case, note that we may drop $\mathbf{z}$ from the expansion, since constant offsets do not change the relative values of HSIC for feature selection. In this case we recover (5.10) with $c_y = \frac{m^2}{m_y^2 (m - m_y)^2}$.

Regression This is one of the situations where the advantages of using HSIC are clearly apparent: we are able to adjust our method to such situations simply by choosing appropriate kernels. Clearly, we could just use a linear kernel $l(y, y') = yy'$ which would select simple correlations between data and labels. Another choice is to use an RBF kernel on the labels, such as

$$l(y, y') = \exp\left(-\bar{\sigma} \left\| y - y' \right\|^2 \right). \tag{5.12}$$

This will ensure that we capture arbitrary nonlinear dependence between $\mathbf{x}$ and y. The price is that in this case $\mathbf{L}$ will have full rank, hence computation of BAHSIC and FOHSIC are correspondingly more expensive.

5.4 Connections to Other Approaches

We now show that several feature selection criteria are special cases of BAHSIC by choosing appropriate preprocessing of data and kernels. We will directly relate these criteria to the biased estimator $\widehat{\mathrm{HSIC}}_b$ in (4.32). Given the fact that $\widehat{\mathrm{HSIC}}_b$ converges to $\widehat{\mathrm{HSIC}}_u$ with rate

$O(m^{-1})$ it follows that the criteria are well related. Additionally we can infer from this that by using $\widehat{\mathrm{HSIC}}_u$ these other criteria could also be improved by correcting their bias. In summary BAHSIC is capable of finding and exploiting dependence of a much more general nature (for instance, dependence between data and labels with graph and string values).

Pearson Correlation Pearson's correlation is commonly used in microarray analysis [141, 44]. It is defined as

$$R_j := \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\mathbf{X}_{ij}-\mu_j}{s_j}\right)\left(\frac{y_i-\mu_y}{s_y}\right). \tag{5.13}$$

This means that all features are individually centered by μ_j and scaled by their coordinate-wise variance s_j as a preprocessing step. Performing those operations before applying a linear kernel yields the equivalent $\widehat{\mathrm{HSIC}}_u$ formulation:

$$\mathrm{tr}\left(\mathbf{KHLH}\right) = \mathrm{tr}\left(\mathbf{X}\mathbf{X}^\top\mathbf{H}\mathbf{y}\mathbf{y}^\top\mathbf{H}\right) = \left\|\mathbf{X}^\top\mathbf{H}\mathbf{y}\right\|^2 \tag{5.14}$$

$$= \sum_{j=1}^{d}\left(\sum_{i=1}^{m}\left(\frac{\mathbf{X}_{ij}-\mu_j}{s_j}\right)\left(\frac{y_i-\mu_y}{s_y}\right)\right)^2 = \sum_{j=1}^{d}R_j^2. \tag{5.15}$$

Hence $\widehat{\mathrm{HSIC}}_b$ computes the sum of the squares of the Pearson Correlation (pc) coefficients. Since the terms are additive, feature selection is straightforward by picking the list of best performing features.

Mean Difference and its Variants The difference between the means of the positive and negative classes at the jth feature, $(\mu_{j+}-\mu_{j-})$, is useful for scoring individual features. With different normalization of the data and the labels, many variants can be derived. In our experiments we compare a number of these variants. For example, the centroid (lin) [15], t-statistic (t), signal-to-noise ratio (snr), moderated t-score (m-t) and B-statistics (lods) [126] all belong to this family. In the following we make those connections more explicit.

Centroid [15] use $v_j := \lambda\mu_{j+} - (1-\lambda)\mu_{j-}$ for $\lambda \in (0,1)$ as the score for feature j.[1] Features are subsequently selected according to the absolute value $|v_j|$. In experiments the authors typically choose $\lambda = \frac{1}{2}$. In this case, we can achieve the same goal by choosing $\mathbf{L}_{ii'} = \frac{y_i y_{i'}}{m_{y_i} m_{y_{i'}}}$ $(y_i, y_{i'} \in \{\pm 1\})$, in which case $\mathbf{HLH} = \mathbf{L}$, since the label kernel matrix is already centered. When we use a linear kernel on the data, ie. $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$, we have

$$\mathrm{tr}\left(\mathbf{KHLH}\right) = \sum_{i,i'=1}^{m}\frac{y_i y_{i'}}{m_{y_i} m_{y_{i'}}}\mathbf{x}_i^\top\mathbf{x}_{i'} \tag{5.16}$$

$$= \sum_{j=1}^{d}\left(\sum_{i,i'=1}^{m}\frac{y_i y_{i'}\mathbf{X}_{ij}\mathbf{X}_{i'j}}{m_{y_i} m_{y_{i'}}}\right) = \sum_{j=1}^{d}(\mu_{j+}-\mu_{j-})^2.$$

This proves that the centroid feature selector can be viewed as a special case of BAHSIC in the case of $\lambda = \frac{1}{2}$. From our analysis we see that other values of λ amount to

[1] The parametrization in [15] is different but it can be shown to be equivalent.

effectively rescaling the patterns $\mathbf{x}_i$ *differently* for different classes, which may lead to undesirable features being selected.

t-Statistic The normalization for the jth feature is computed as

$$\bar{s}_j = \left[\frac{s_{j+}^2}{m_+} + \frac{s_{j-}^2}{m_-} \right]^{\frac{1}{2}}. \tag{5.17}$$

In this case we define the t-statistic for the jth feature via $t_j = (\mu_{j+} - \mu_{j-})/\bar{s}_j$.

Compared to the Pearson correlation, the key difference is that now we normalize each feature not by the overall sample standard deviation but rather by a value which takes each of the two classes separately into account.

Signal to noise ratio is yet another criterion to use in feature selection. The key idea is to normalize each feature by $\bar{s}_j = s_{j+} + s_{j-}$ instead. Subsequently the $(\mu_{j+} - \mu_{j-})/\bar{s}_j$ are used to score features.

Moderated t-score is similar to t-statistic and is used for microarray analysis [126]. Its normalization for the jth feature is derived via a Bayes approach as

$$\tilde{s}_j = \frac{m\bar{s}_j^2 + m_0\bar{s}_0^2}{m + m_0}, \tag{5.18}$$

where $\bar{s}_j$ is from (5.17), and $\bar{s}_0$ and m_0 are hyperparameters for the prior distribution on $\bar{s}_j$ (all $\bar{s}_j$ are assumed to be iid). $\bar{s}_0$ and m_0 are estimated using information from all feature dimensions. This effectively borrows information from the ensemble of features to aid with the scoring of an individual feature. More specifically, $\bar{s}_0$ and m_0 can be computed as [126]

$$m_0 = 2\Gamma'^{-1}\left(\frac{1}{d}\sum_{j=1}^{d}(z_j - \bar{z})^2 - \Gamma'\left(\frac{m}{2}\right) \right), \tag{5.19}$$

$$\bar{s}_0^2 = \exp\left(\bar{z} - \Gamma\left(\frac{m}{2}\right) + \Gamma\left(\frac{m_0}{2}\right) - \ln\left(\frac{m_0}{m}\right) \right), \tag{5.20}$$

where $\Gamma(\cdot)$ is the gamma function, $'$ denotes derivative, $z_j = \ln(\bar{s}_j^2)$ and $\bar{z} = \frac{1}{d}\sum_{j=1}^{d} z_j$.

B-statistic is the logarithm of the posterior odds (lods) that a feature is differentially expressed. [90, 126] show that, for large number of features, B-statistic is given by

$$B_j = a + b\tilde{t}_j^2, \tag{5.21}$$

where both a and b are constant ($b > 0$), and $\tilde{t}_j$ is the moderated-t statistic for the jth feature. Here we see that B_j is monotonic increasing in $\tilde{t}_j$, and thus results in the same gene ranking as the moderated-t statistic.

The reason why these connections work is that the signal-to-noise ratio, moderated t-statistic, and B-statistic are three variants of the t-test. They differ only in their respective denominators, and are thus special cases of HSIC_b if we preprocess the data accordingly.

Maximum Mean Discrepancy For binary classification, an alternative criterion for selecting features is to check whether the distributions $\Pr(\mathbf{x}|y=1)$ and $\Pr(\mathbf{x}|y=-1)$ differ and subsequently pick those coordinates of the data which primarily contribute to the difference between the two distributions.

More specifically, we could use Maximum Mean Discrepancy (MMD) [19], which is a generalization of mean difference for Reproducing Kernel Hilbert Spaces, given by

$$\mathrm{MMD} = \|\mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x},\cdot)|y=1\right] - \mathbb{E}_{\mathbf{x}}\left[k(\mathbf{x},\cdot)|y=-1\right]\|^2_{\mathcal{H}}\,. \tag{5.22}$$

A biased estimator of the above quantity can be obtained simply by replacing expectations by averages over a finite sample. We relate a biased estimator of MMD to $\widehat{\mathrm{HSIC}}_b$ again by setting $\frac{1}{m_+}$ as the labels for positive samples and $-\frac{1}{m_-}$ for negative samples. If we apply a linear kernel on labels, $\mathbf{L}$ is automatically centered, i.e. $\mathbf{L1} = \mathbf{0}$ and $\mathbf{HLH} = \mathbf{L}$. This yields

$$\begin{aligned}
\mathrm{tr}\left(\mathbf{KHLH}\right) &= \mathrm{tr}\left(\mathbf{KL}\right) \\
&= \frac{1}{m_+^2}\sum_{i,j}^{m_+} k(\mathbf{x}_i,\mathbf{x}_j) + \frac{1}{m_-^2}\sum_{i,j}^{m_-} k(\mathbf{x}_i,\mathbf{x}_j) - \frac{2}{m_+ m_-}\sum_{i}^{m_+}\sum_{j}^{m_-} k(\mathbf{x}_i,\mathbf{x}_j) \\
&= \left\| \frac{1}{m_+}\sum_{i}^{m_+} k(\mathbf{x}_i,\cdot) - \frac{1}{m_-}\sum_{j}^{m_-} k(\mathbf{x}_j,\cdot) \right\|^2_{\mathcal{H}}\,.
\end{aligned} \tag{5.23}$$

The quantity in the last line is an estimator of MMD with bias $O(m^{-1})$ 4. This implies that $\widehat{\mathrm{HSIC}}_b$ and the biased estimator of MMD are identical up to a constant factor. Since the bias of $\widehat{\mathrm{HSIC}}_b$ is also $O(m^{-1})$, this effectively show that scaled MMD and $\widehat{\mathrm{HSIC}}_u$ converges to each other with rate $O(m^{-1})$.

Kernel Target Alignment Alternatively, one could use Kernel Target Alignment (KTA) [31] to test directly whether there exists any correlation between data and labels. KTA has been used for feature selection in this context. Formally it is defined as $\mathrm{tr}(\mathbf{KL})/\|\mathbf{K}\|_{\mathrm{Frob}}\|\mathbf{L}\|_{\mathrm{Frob}}$, that is, the cosine between the kernel matrix and the label matrix.

The nonlinear dependence on $\mathbf{K}$ makes it somewhat hard to optimize for. Indeed, for computational convenience the normalization is often omitted in practice [100], which leaves us with $\mathrm{tr}(\mathbf{KL})$, the corresponding estimator of MMD. Note the key difference, though, that normalization of $\mathbf{L}$ according to label size does not occur. Nor does KTA take centering into account. Whenever the sample sizes for both classes are approximately matched, such lack of normalization is negligible and we see that both criteria effectively check a similar criterion.

Hence in some cases in binary classification, selecting features that maximizes HSIC also maximizes MMD and KTA. Note that in general (multiclass, regression, or generic binary classification) this connection does not hold. Moreover, the use of HSIC offers uniform convergence bounds on the tails of the distribution of the estimators.

Shrunken Centroid The shrunken centroid (pam) method [136, 137] performs feature ranking using the differences from the class centroids to the centroid of all the data, ie.

$$\left(\mu_{j+} - \mu_j\right)^2 + \left(\mu_{j-} - \mu_j\right)^2, \tag{5.24}$$

as a criterion to determine the relevance of a given feature. It also scores each feature separately.

To show that this criterion is related to HSIC we need to devise an appropriate map for the labels y. Consider the feature map $\psi(y)$ with $\psi(1) = (\frac{1}{m_+}, 0)^\top$ and $\psi(-1) = (0, \frac{1}{m_-})^\top$. Clearly, when applying $\mathbf{H}$ to $\mathbf{Y}$ we obtain the following centered effective feature maps

$$\bar{\psi}(1) = (\frac{1}{m_+} - \frac{1}{m}, -\frac{1}{m}) \text{ and } \bar{\psi}(-1) = (-\frac{1}{m}, \frac{1}{m_-} - \frac{1}{m}). \tag{5.25}$$

Consequently we may express $\mathrm{tr}(\mathbf{KHLH})$ via

$$\begin{aligned}
\mathrm{tr}\left(\mathbf{KHLH}\right) &= \left\| \frac{1}{m_+} \sum_{i=1}^{m_+} \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i \right\|^2 + \left\| \frac{1}{m_-} \sum_{i=1}^{m_-} \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i \right\|^2 \\
&= \sum_{j=1}^{d} \left(\left(\frac{1}{m_+} \sum_{i=1}^{m_+} \mathbf{X}_{ij} - \frac{1}{m} \sum_{i=1}^{m} \mathbf{X}_{ij} \right)^2 + \left(\frac{1}{m_-} \sum_{i=1}^{m_-} \mathbf{X}_{ij} - \frac{1}{m} \sum_{i=1}^{m} \mathbf{X}_{ij} \right)^2 \right) \\
&= \sum_{j=1}^{d} \left((\mu_{j+} - \mu_j)^2 + (\mu_{j-} - \mu_j)^2 \right).
\end{aligned} \tag{5.26}$$

This is the information used by the shrunken centroid method, hence we see that it can be seen to be a special case of HSIC when using a linear kernel on the data and a specific feature map on the labels. Note that we could assign different weights to the two classes, which would lead to a weighted linear combination of distances from the centroid. Finally, it is straightforward how this definition can be extended to multiclass settings, simply by considering the map $\psi : y \to \frac{1}{m_y}\mathbf{e}_y$.

Ridge Regression BAHSIC can also be used to select features for regression problems, except that in this case the labels are continuous variables. We could, in principle, use an RBF kernel or similar on the labels to address the feature selection issue. What we show now is that even for a simple linear kernel, interesting results can be obtained. More to the point, we show that feature selection using ridge regression can also be seen to arise as a special case of HSIC feature selection. We assume here that $\mathbf{y}$ is centered.

In ridge regression [67], we estimate the outputs $\mathbf{y}$ using the design matrix $\mathbf{V}$ and a parameter vector $\mathbf{w}$ by minimizing the following regularized risk functional

$$J = \|\mathbf{y} - \mathbf{V}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2. \tag{5.27}$$

Here the second term is known as the regularizer. If we choose $\mathbf{V} = \mathbf{X}$ we obtain the family of *linear* models. In the general (nonlinear) case $\mathbf{V}$ may be an arbitrary matrix, where each row consists of a set of basis functions, e.g. a feature map $\phi(\mathbf{x})$. One might conclude that small values of J correspond to good sets of features, since there a $\mathbf{w}$ with small norm would still lead to a small approximation error. It turns out that J is minimized for $\mathbf{w} = (\mathbf{V}^\top\mathbf{V} + \lambda\mathbf{I})^{-1}\mathbf{y}$. Hence the minimum is given by

$$\begin{aligned}
J^* &= \mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{V}(\mathbf{V}^\top\mathbf{V} + \lambda\mathbf{I})^{-1}\mathbf{V}^\top\mathbf{y} \\
&= \text{constant} - \mathrm{tr}\left(\mathbf{V}(\mathbf{V}^\top\mathbf{V} + \lambda\mathbf{I})^{-1}\mathbf{V}^\top\right)\mathbf{y}\mathbf{y}^\top.
\end{aligned} \tag{5.28}$$

Whenever we are only given $\mathbf{K} = \mathbf{V}^\top \mathbf{V}$ we have the following equality

$$J^* = \text{constant} - \operatorname{tr}\left(\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1}\right) \mathbf{y}\mathbf{y}^\top. \tag{5.29}$$

This means that the matrices

$$\bar{\mathbf{K}} := \mathbf{V}(\mathbf{V}^\top \mathbf{V} + \lambda \mathbf{I})^{-1} \mathbf{V}^\top \text{ and } \bar{\mathbf{K}} := \mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1} \tag{5.30}$$

are equivalent kernel matrices to be used in BAHSIC. Note that instead of using $\mathbf{y}\mathbf{y}^\top$ as a kernel on the labels $\mathbf{L}$ we could use a nonlinear kernel *in conjunction* with the matrices arrived at from feature selection by ridge regression. It also generalizes the setting of [67] to situations other than regression.

Quadratic Mutual Information [138] introduces the quadratic mutual information for feature selection. That is, he uses the L_2 distance between the joint and the marginal distributions on $\mathbf{x}$ and $\mathbf{y}$ as a criterion for how dependent the two distributions are:

$$QI(\mathbf{x}, \mathbf{y}) = \int_{\mathcal{X} \times \mathcal{Y}} \left(\mathrm{d}\Pr_{\mathbf{xy}} - \mathrm{d}\Pr_{\mathbf{x}} \mathrm{d}\Pr_{\mathbf{y}}\right)^2 \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \tag{5.31}$$

In general, (5.31) is not efficiently computable. That said, when using a Parzen windows estimate of the joint and the marginals, it is possible to evaluate $QI(\mathbf{x}, \mathbf{y})$ explicitly. Since we only have a finite number of observations, one uses the estimates

$$\hat{p}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} \kappa_{\mathbf{x}}(\mathbf{x}_i - \mathbf{x}), \tag{5.32a}$$

$$\hat{p}(\mathbf{y}) = \frac{1}{m} \sum_{i=1}^{m} \kappa_{\mathbf{y}}(\mathbf{y}_i - \mathbf{y}), \tag{5.32b}$$

$$\hat{p}(\mathbf{x}, \mathbf{y}) = \frac{1}{m} \sum_{i=1}^{m} \kappa_{\mathbf{x}}(\mathbf{x}_i - \mathbf{x}) \kappa_{\mathbf{y}}(\mathbf{y}_i - \mathbf{y}). \tag{5.32c}$$

Here $\kappa_{\mathbf{x}}$ and $\kappa_{\mathbf{y}}$ are appropriate kernels of the Parzen windows density estimator. Denote by

$$\mathbf{K}_{ij} = \int_{\mathcal{X}} \kappa_{\mathbf{x}}(\mathbf{x}_i - \mathbf{x}) \kappa_{\mathbf{x}}(\mathbf{x}_j - \mathbf{x}) \mathrm{d}\mathbf{x} \quad \text{and} \tag{5.33}$$

$$\mathbf{L}_{ij} = \int_{\mathcal{Y}} \kappa_{\mathbf{y}}(\mathbf{y}_i - \mathbf{y}) \kappa_{\mathbf{y}}(\mathbf{y}_j - \mathbf{y}) \mathrm{d}\mathbf{y} \tag{5.34}$$

inner products between Parzen windows kernels, and compile these entries into matrices $\mathbf{K}$ and $\mathbf{L}$. Then we have

$$\begin{aligned} \|\hat{p}(\mathbf{x}, \mathbf{y}) - \hat{p}(\mathbf{x})\hat{p}(\mathbf{y})\|^2 &= \frac{1}{m^2} \left[\operatorname{tr}(\mathbf{KL}) - 2\mathbf{1}^\top \mathbf{KL1} + \mathbf{1}^\top \mathbf{K11}^\top \mathbf{L1}\right] \\ &= \frac{1}{m^2} \operatorname{tr}(\mathbf{KHLH}). \end{aligned} \tag{5.35}$$

In other words, we obtain the same criterion as what can be derived from a biased estimator of HSIC. The key difference, though, is that this analogy only works whenever $\kappa_{\mathbf{x}}$ and $\kappa_{\mathbf{y}}$ can be seen to be arising from an inner product between Parzen windows kernel estimates. However,

for graphs, trees or strings, the generalization of a density estimator can be difficult, which poses a serious limitation. Moreover, since we are using a plug-in estimate of the densities, we inherit an innate slow-down of convergence due to the convergence of the density estimators. This issue is discussed in detail in [7].

Recursive Feature Elimination with Support Vectors Another popular feature selection algorithm is to use Support Vector Machines and to determine the relevance of features by the size of the induced margin as a solution of the dual optimization problem [65]. While the connection to BAHSIC is somewhat more tenuous in this context, it is still possible to recast this algorithm in our framework. Before we do so, we describe the basic idea of the method, using ν-SVM instead of plain C-SVMs: for ν-SVM without a constant offset b we have the following dual optimization problem [114].

$$\underset{\alpha}{\text{minimize}} \frac{1}{2}\alpha^\top(\mathbf{K} \circ \mathbf{L})\alpha \text{ subject to } \alpha^\top \mathbf{1} = \nu m \text{ and } \alpha_i \in [0, 1]. \tag{5.36}$$

This problem is first solved with respect to α for the full set of features. Features are then selected from (5.36) by removing coordinates such that the objective function decreases least (if at all). For computational convenience, α is not recomputed for a number of feature removals, since solving a quadratic program repeatedly is computationally expensive.

We now show that this procedure can be viewed as a special case of BAHSIC, where now the class of kernels, parameterized by σ is the one of *conformal* kernels. Given a base kernel $k(\mathbf{x}, \mathbf{x}')$ [5] propose the following kernel:

$$\bar{k}(\mathbf{x}, \mathbf{x}') = \alpha(\mathbf{x})\alpha(\mathbf{x}')k(\mathbf{x}, \mathbf{x}') \quad \text{where} \quad \alpha(\mathbf{x}) \geq 0. \tag{5.37}$$

It is easy to see that

$$\alpha^\top(\mathbf{K} \circ \mathbf{L})\alpha = \mathbf{y}^\top (\operatorname{diag} \alpha)\, \mathbf{K}\, (\operatorname{diag} \alpha)\, \mathbf{y} = \mathbf{y}^\top \bar{\mathbf{K}} \mathbf{y}, \tag{5.38}$$

where $\bar{\mathbf{K}}$ is the kernel matrix arising from the conformal kernel $\bar{k}(\mathbf{x}, \mathbf{x}')$. Hence for fixed α the objective function is given by a quantity which can be interpreted as a biased version of HSIC. Re-optimization with respect to α is consistent with the kernel adjustment step in Algorithm 1. The only difference being that here the kernel parameters are given by α rather than a kernel width σ. That said, it is also clear from the optimization problem that this style of feature selection may not be as desirable, since the choice of kernel parameters emphasizes only points close to the decision boundary.

5.5 Experiments on Benchmark Data

We analyze BAHSIC and related algorithms in an extensive set of experiments. The current section contains results on synthetic and real benchmark data, that is, data from Statlib, the UCI repository, and data from the NIPS feature selection challenge. Sections 5.6 and 5.7 then discusses applications to biological data, namely brain signal analysis and feature selection for microarrays.

Since the number of possible choices for feature selection within the BAHSIC family is huge, it is clearly impossible to investigate and compare all of them to all possible other feature selectors. In the present section we pick the following three feature selectors as representative

examples. A wider range of kernels and choices is investigated in Section 5.6 and 5.7 in the context of bioinformatics applications.

In this section, we presents three concrete examples of BAHSIC which are used for our later experiments. We apply a Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma\|\mathbf{x}-\mathbf{x}'\|^2)$ on data, while varying the kernels on labels. These BAHSIC variants are dedicated respectively to the following setttings:

Binary classification (BIN) Use the feature map in (5.9) and apply a linear kernel.

Multiclass classification (MUL) Use the feature map in (5.10) and apply a linear kernel.

Regression problem (REG) Use the kernel in (5.12), i.e. a Gaussian RBF kernel on $\mathbf{Y}$.

For the above variants a further speedup of BAHSIC is possible by updating entries in the data kernel matrix incrementally. We use the fact that distance computation of a RBF kernel decomposes into individual coordinates, i.e. we use that $\|\mathbf{x}_i - \mathbf{x}_{i'}\|^2 = \sum_{j=1}^{d} \|\mathbf{X}_{ij} - \mathbf{X}_{i'j}\|^2$. Hence $\|\mathbf{x}_i - \mathbf{x}_{i'}\|^2$ needs to be computed only once, and subsequent updates are effected by subtracting $\|\mathbf{X}_{ij} - \mathbf{X}_{i'j}\|^2$.

We will use BIN, MUL and REG as the particular instances of BAHSIC in our experiments. We will refer to them commonly as BAHSIC since the exact meaning will be clear depending on the datasets encountered. Furthermore, we also instantiate FOHSIC using the same kernels as BIN, MUL and REG, and we adopt the same convention when we refer to it in our experiments.

5.5.1 Artificial Data

We constructed 3 artificial datasets, as illustrated in Figure 5.1, to illustrate the difference between BAHSIC variants with linear and nonlinear kernels. Each dataset has 22 dimensions — only the first two dimensions are related to the prediction task and the rest are just Gaussian noise. These datasets are (*i*) **Binary XOR data**: samples belonging to the same class have multimodal distributions; (*ii*) **Multiclass data**: there are 4 classes but 3 of them are collinear; (*iii*) **Nonlinear regression data**: labels are related to the first two dimension of the data by $y = \mathbf{x}(1)\exp(-\mathbf{x}(1)^2 - \mathbf{x}(2)^2) + \epsilon$, where ϵ denotes additive Gaussian noise. We compare BAHSIC to FOHSIC, Pearson's correlation, mutual information [156], and RELIEF (RELIEF works only for binary problems). We aim to show that when nonlinear dependencies exist in the data, BAHSIC with nonlinear kernels is very competent in finding them.

We instantiate the artificial datasets over a range of sample sizes (from 40 to 400), and plot the median rank, produced by various methods, for the first two dimensions of the data. All numbers in Figure 5.1 are averaged over 10 runs. In all cases, BAHSIC shows good performance. More specifically, we observe:

Binary XOR Both BAHSIC and RELIEF correctly select the first two dimensions of the data even for small sample sizes; while FOHSIC, Pearson's correlation, and mutual information fail. This is because the latter three evaluate the goodness of each feature independently. Hence they are unable to capture nonlinear interaction between features.

Multiclass Data BAHSIC, FOHSIC and mutual information select the correct features irrespective of the size of the sample. Pearson's correlation only works for large sample size. The collinearity of 3 classes provides linear correlation between the data and the labels, but due to the interference of the fourth class such correlation is picked up by Pearson's correlation only for a large sample size.

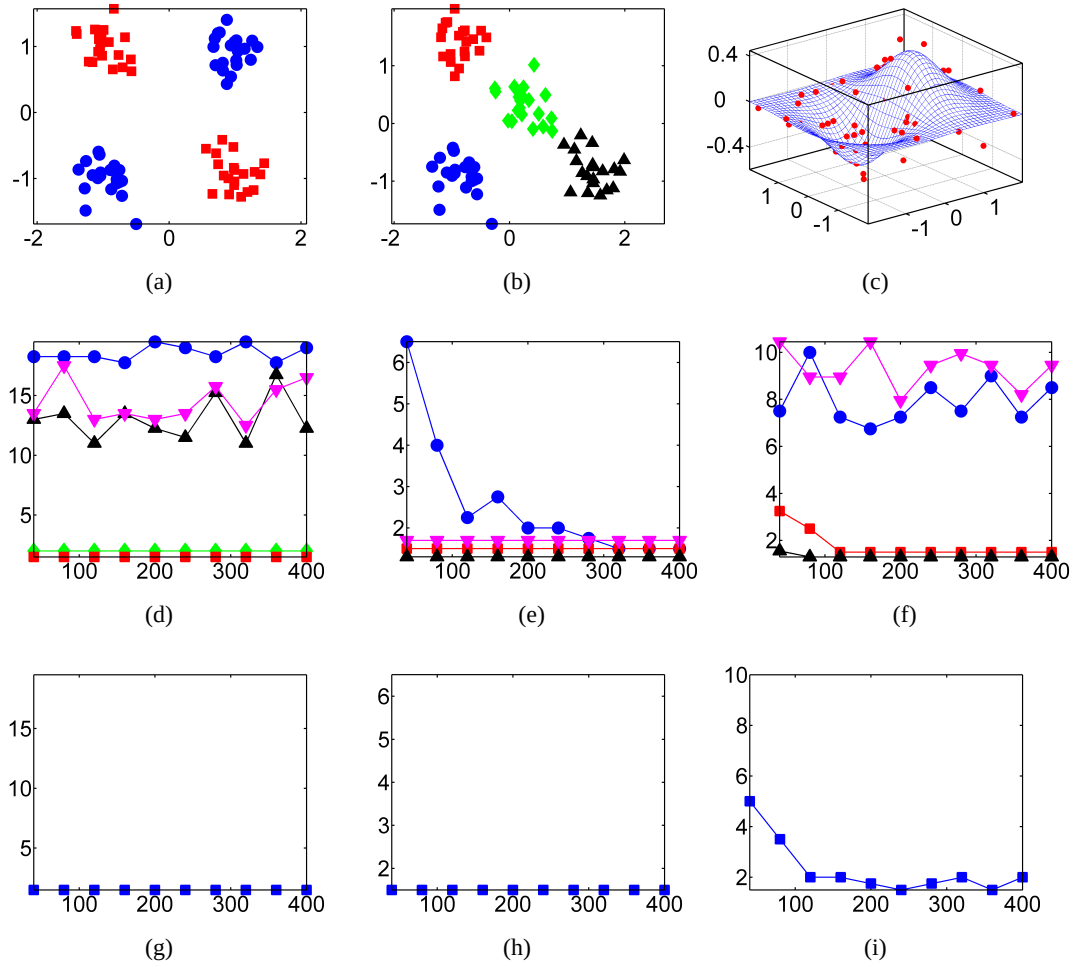


Figure 5.1: Artificial datasets and the performance of different methods when varying the number of observations. **The first row** contains plots for the first 2 dimension of the (a) binary (b) multiclass and (c) regression data. Different classes are encoded with different colors. **The second row** plots the median rank (y-axis) of the two relevant features as a function of sample size (x-axis) for the corresponding datasets in the first row. **The third row** plots median rank (y-axis) of the two relevant features produced in the first iteration of BAHSIC as a function of the sample size. (Blue circle: Pearson's correlation; Green triangle: RELIEF; Magenta downward triangle: mutual information; Black triangle: FOHSIC; Red square: BAHSIC. Note that RELIEF only works for binary classification.)

Nonlinear Regression Data The performance of Pearson's correlation and mutual information is slightly better than random. BAHSIC and FOHSIC quickly converge to the correct answer as the sample size increases.

In fact, we observe that as the sample size increases, BAHSIC is able to rank the relevant features (the first two dimensions) correctly almost in the first iteration. In the third row of Figure 5.1, we show the median rank of the relevant features produced in the first iteration as a function of the sample size. It is clear from the pictures that BAHSIC effectively selects features in a single iteration when the sample size is large enough. For the regression case, we also see that BAHSIC with several iterations, indicated by the red square in Figure 5.1(f), slightly improves the correct ranking over BAHSIC with a single iteration, given by the blue

square in Figure 5.1(i).

While this does not prove BAHSIC with nonlinear kernels is always better than that with a linear kernel, it illustrates the competence of BAHSIC in detecting nonlinear features. This is useful in a real-world situations. The second advantage of BAHSIC is that it is readily applicable to both classification and regression problems, by simply choosing a different kernel on the labels.

5.5.2 Public Benchmark Data

Algorithms In this experiment, we show that the performance of BAHSIC can be comparable to other state-of-the-art feature selectors, namely SVM Recursive Feature Elimination (RFE) [65], RELIEF [81], L_0-norm SVM (L_0) [153], and R2W2 [154]. We used the implementation of these algorithms as given in the Spider machine learning toolbox, since those were the only publicly available implementations.[2] Furthermore, we also include filter methods, namely FOHSIC, Pearson's correlation (PC), and mutual information (MI), in our comparisons.

Datasets We used various real world datasets taken from the UCI repository,[3] the Statlib repository,[4] the LibSVM website,[5] and the NIPS feature selection challenge[6] for comparison. Due to scalability issues in Spider, we produced a balanced random sample of size less than 2000 for datasets with more than 2000 samples.

Experimental Protocol We report the performance of an SVM using a Gaussian kernel on a feature subset of size 5 and 10-fold cross-validation. These 5 features were selected per fold using different methods. Since we are comparing the selected features, we used the same SVM for all methods: a Gaussian kernel with σ set as the median distance between points in the sample [115] and regularization parameter $C = 100$. On classification datasets, we measured the performance using the error rate, and on regression datasets we used the percentage of variance *not*-explained (also known as $1 - r^2$). The results for binary datasets are summarized in the first part of Table 5.1. Those for multiclass and regression datasets are reported respectively in the second and the third parts of Table 5.1.

To provide a concise summary of the performance of various methods on binary datasets, we measured how the methods compare with the best performing one in each dataset in Table 5.1. We recorded the best absolute performance of *all* feature selectors as the baseline, and computed the distance of each algorithm to the best possible result. In this context it makes sense to penalize catastrophic failures more than small deviations. In other words, we would like to have a method which is almost always very close to the best performing one. Taking the ℓ_2 distance achieves this effect, by penalizing larger differences more heavily. It is also our goal to choose an algorithm that performs homogeneously well across all datasets. The ℓ_2 distance scores are listed for the binary datasets in Table 5.1. In general, the smaller the ℓ_2 distance, the better the method. In this respect, BAHSIC and FOHSIC have the best performance. We did not produce the ℓ_2 distance for multiclass and regression datasets, since the limited number of such datasets did not allow us to draw statistically significant conclusions.

[2] http://www.kyb.tuebingen.mpg.de/bs/people/spider
[3] http://www.ics.uci.edu/ mlearn/MLSummary.html
[4] http://lib.stat.cmu.edu/datasets/
[5] http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[6] http://clopinet.com/isabelle/Projects/NIPS2003/

Besides using 5 features, we also plot the performance of the learners as a function of the number of selected features for 9 datasets (Covertype, Ionosphere, Sonar, Satimage, Segment, Vehicle, Housing, Bodyfat and Abalone) in Figure 5.2. Generally speaking, the smaller the plotted number the better the performance of the corresponding learner. For multiclass and regression datasets, it is clear that the curves for BAHSIC very often lie along the lower bound of all methods. For binary classification, however, SVM-RFE as a member of our framework performs the best in general. The advantage of BAHSIC becomes apparent when a small percentage of features is selected. For instance, BAHSIC is the best when only 5 features are selected from data set 1 and 2. Note that in these cases, the performance produced by BAHSIC is very close to that using all features. In a sense, BAHSIC is able to shortlist the most informative features.

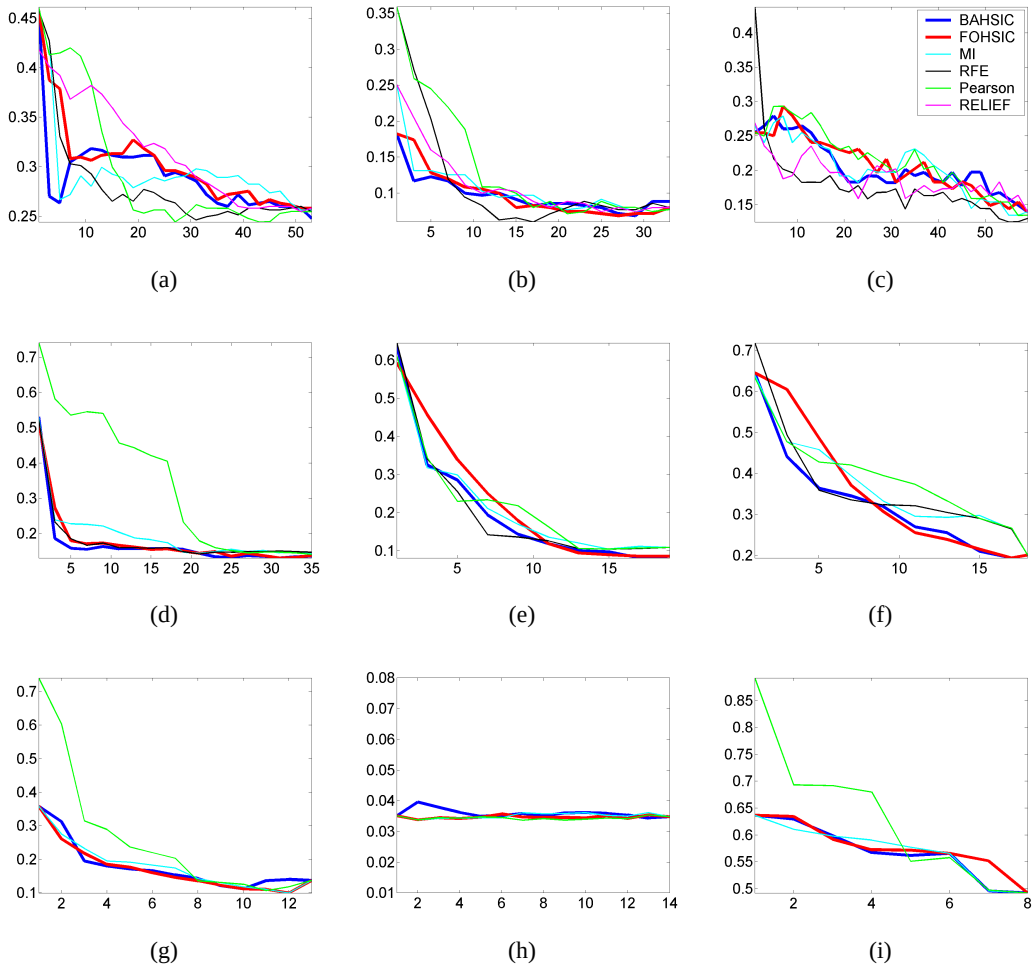


Figure 5.2: The performance of a classifier or a regressor (vertical axes) as a function of the number of selected features (horizontal axes). Note that the maximum of the horizontal axes are equal to the total number of features in each data set. (a-c) Balanced error rate by a SVM classifier on the binary data sets Covertype (1), Ionosphere (2) and Sonar (3) respectively; (d-f) balanced error rate by a one-versus-the-rest SVM classfier on multiclass data sets Satimage (22), Segment (23) and Vehicle (24) respectively; (g-i) percentage of variance *not*-explained by a SVR regressor on regression data set Housing (25), Body fat (26) and Abalone (27) respectively.

Table 5.1: Classification error (%) or percentage of variance *not*-explained (%). The best result, and those results not significantly worse than it, are highlighted in bold (one-sided Welch t-test with 95% confidence level). 100.0±0.0*: program is not finished in a week or crashed. -: not applicable.

Data	BAHSIC	FOHSIC	PC	MI	RFE	RELIEF	L_0	R2W2
covertype	**26.3±1.5**	37.9±1.7	40.3±1.3	**26.7±1.1**	33.0±1.9	42.7±0.7	43.4±0.7	44.2±1.7
ionosphere	**12.3±1.7**	**12.8±1.6**	**12.3±1.5**	**13.1±1.7**	20.2±2.2	**11.7±2.0**	35.9±0.4	13.7±2.7
sonar	27.9±3.1	25.0±2.3	25.5±2.4	26.9±1.9	**21.6±3.4**	24.0±2.4	36.5±3.3	32.3±1.8
heart	**14.8±2.4**	**14.4±2.4**	16.7±2.4	**15.2±2.5**	21.9±3.0	21.9±3.4	30.7±2.8	19.3±2.6
breastcancer	3.8±0.4	3.8±0.4	4.0±0.4	3.5±0.5	**3.4±0.6**	**3.1±0.3**	32.7±2.3	**3.4±0.4**
australian	**14.3±1.3**	**14.3±1.3**	**14.5±1.3**	**14.5±1.3**	**14.8±1.2**	**14.5±1.3**	35.9±1.0	**14.5±1.3**
splice	22.6±1.1	22.6±1.1	22.8±0.9	21.9±1.0	**20.7±1.0**	22.3±1.0	45.2±1.2	24.0±1.0
svmguide3	**20.8±0.6**	**20.9±0.6**	21.2±0.6	**20.4±0.7**	21.0±0.7	21.6±0.4	23.3±0.3	23.9±0.2
adult	24.8±0.2	24.4±0.6	**18.3±1.1**	21.6±1.1	21.3±0.9	24.4±0.2	24.7±0.1	100.0±0.0*
cleveland	**19.0±2.1**	**20.5±1.9**	21.9±1.7	**19.5±2.2**	20.9±2.1	22.4±2.5	25.2±0.6	21.5±1.3
derm	**0.3±0.3**	**0.3±0.3**	**0.3±0.3**	**0.3±0.3**	**0.3±0.3**	**0.3±0.3**	24.3±2.6	**0.3±0.3**
hepatitis	**13.8±3.5**	**15.0±2.5**	**15.0±4.1**	**15.0±4.1**	**15.0±2.5**	17.5±2.0	16.3±1.9	17.5±2.0
musk	29.9±2.5	29.6±1.8	**26.9±2.0**	31.9±2.0	34.7±2.5	**27.7±1.6**	42.6±2.2	36.4±2.4
optdigits	**0.5±0.2**	**0.5±0.2**	**0.5±0.2**	3.4±0.6	3.0±1.6	0.9±0.3	12.5±1.7	0.8±0.3
specft	**20.0±2.8**	**20.0±2.8**	**18.8±3.4**	**18.8±3.4**	37.5±6.7	26.3±3.5	36.3±4.4	31.3±3.4
wdbc	**5.3±0.6**	**5.3±0.6**	**5.3±0.7**	6.7±0.5	7.7±1.8	7.2±1.0	16.7±2.7	6.8±1.2
wine	**1.7±1.1**	**1.7±1.1**	**1.7±1.1**	**1.7±1.1**	3.4±1.4	4.2±1.9	25.1±7.2	**1.7±1.1**
german	29.2±1.9	29.2±1.8	26.2±1.5	26.2±1.7	27.2±2.4	33.2±1.1	32.0±0.0	**24.8±1.4**
gisette	**12.4±1.0**	**13.0±0.9**	16.0±0.7	50.0±0.0	42.8±1.3	16.7±0.6	42.7±0.7	100.0±0.0*
arcene	**22.0±5.1**	**19.0±3.1**	31.0±3.5	45.0±2.7	34.0±4.5	30.0±3.9	46.0±6.2	32.0±5.5
madelon	**37.9±0.8**	**38.0±0.7**	38.4±0.6	51.6±1.0	41.5±0.8	38.6±0.7	51.3±1.1	100.0±0.0*
ℓ_2	**11.2**	14.8	19.7	48.6	42.2	25.9	85.0	138.3
satimage	**15.8±1.0**	17.9±0.8	52.6±1.7	22.7±0.9	18.7±1.3	-	22.1±1.8	-
segment	28.6±1.3	33.9±0.9	**22.9±0.5**	27.1±1.3	24.5±0.8	-	68.7±7.1	-
vehicle	**36.4±1.5**	48.7±2.2	42.8±1.4	45.8±2.5	**35.7±1.3**	-	40.7±1.4	-
svmguide2	**22.8±2.7**	**22.2±2.8**	26.4±2.5	27.4±1.6	35.6±1.3	-	34.5±1.7	-
vowel	**44.7±2.0**	**44.7±2.0**	48.1±2.0	**45.4±2.2**	51.9±2.0	-	85.6±1.0	-
usps	**43.4±1.3**	**43.4±1.3**	73.7±2.2	67.8±1.8	55.8±2.6	-	67.0±2.2	-
housing	**18.5±2.6**	**18.9±3.6**	25.3±2.5	**18.9±2.7**	-	-	-	-
bodyfat	**3.5±2.5**	**3.5±2.5**	**3.4±2.5**	**3.4±2.5**	-	-	-	-
abalone	**55.1±2.7**	**55.9±2.9**	**54.2±3.3**	56.5±2.6	-	-	-	-

5.6 Analysis of Brain Computer Interface Data

In this experiment, we show that BAHSIC selects features that are meaningful in practice. Here we use it to select frequency bands for a brain-computer interface (BCI) dataset from the Berlin BCI group [39]. The data contains EEG signals (118 channels, sampled at 100 Hz) from five healthy subjects ('aa', 'al', 'av', 'aw' and 'ay') recorded during two types of motor imaginations. The task is to classify the imagination for individual trials.

Our experiment proceeds in 3 steps: (*i*) A Fast Fourier transformation (FFT) is performed on each channel and the power spectrum is computed. (*ii*) The power spectra from all channels

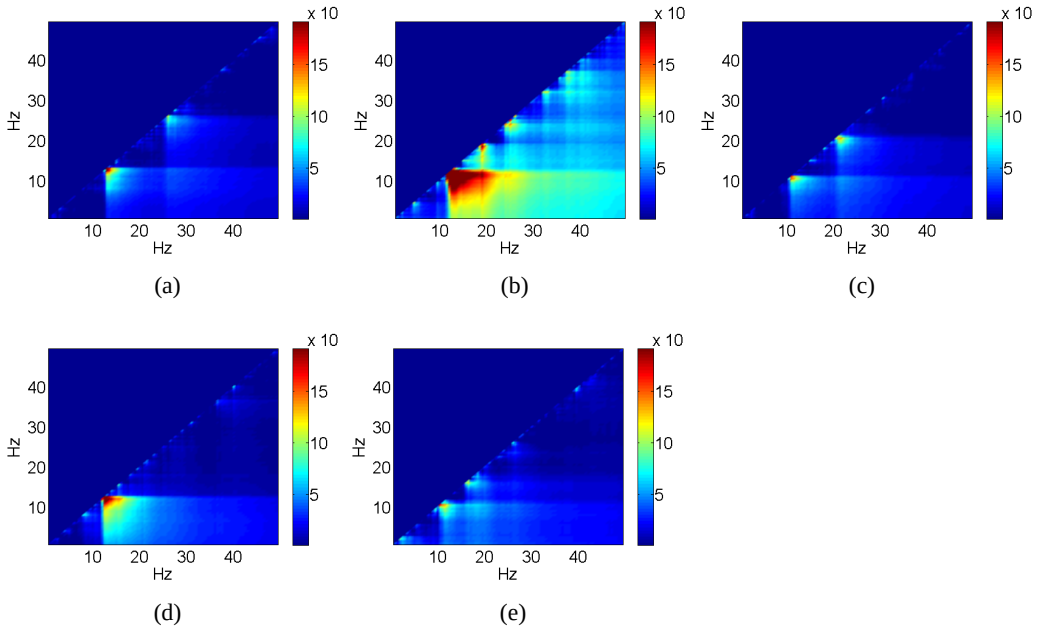


Figure 5.3: HSIC, encoded by the colour value for different frequency bands. The x-axis corresponds to the upper cutoff and the y-axis denotes the lower cutoff (clearly no signal can be found where the lower bound exceeds the upper bound). Red corresponds to strong dependence, whereas blue indicates that no dependence was found. The figures are for subject (a) 'aa', (b) 'al', (c) 'av', (d) 'aw' and (e) 'ay'.

are averaged to obtain a single spectrum for each trial. (*iii*) BAHSIC is used to select the top 5 discriminative frequency components based on the power spectrum. The 5 selected frequencies and their 4 nearest neighbors are used to reconstruct the temporal signals (with all other Fourier coefficients eliminated). The result is then passed to a normal CSP method [39] for feature extraction and then classified using a linear SVM.

Automatic filtering using BAHSIC is then compared to other filtering approaches: normal CSP method with manual filtering (8-40 Hz), the CSSP method [87] and the CSSSP method [40]. All results presented in Table 5.2 are obtained using 50×2-fold cross-validation. Our method is very competitive and obtains the first and second place for 4 of the 5 subjects. While the CSSP and the CSSSP methods are *specialized* embedded methods (w.r.t. the CSP method) for frequency selection on BCI data, our method is entirely generic. BAHSIC decouples feature selection from CSP, while proving competitive.

In Figure 5.3, we use HSIC to visualize the responsiveness of different frequency bands to motor imagination. The horizontal and the vertical axes in each subfigure represent the lower and upper bounds for a frequency band, respectively. HSIC is computed for each of these bands. [40] report that the μ rhythm (approx. 12 Hz) of EEG is most responsive to motor imagination, and that the β rhythm (approx. 22 Hz) is also responsive. We expect that HSIC will create a strong peak at the μ rhythm and a weaker peak at the β rhythm, and the absence of other responsive frequency components will create block patterns. Both predictions are confirmed in Figure 5.3. Furthermore, the large area of the red region for subject 'al' indicates good responsiveness of his μ rhythm. This also corresponds well with the lowest classification error obtained for him in Table 5.2.

Table 5.2: Classification errors (%) on BCI data after selecting a frequency range.

Method	aa	al	av	aw	ay
CSP(8-40Hz)	17.5±2.5	3.1±1.2	32.1±2.5	7.3±2.7	**6.0±1.6**
CSSP	14.9±2.9	2.4±1.3	33.0±2.7	**5.4±1.9**	6.2±1.5
CSSSP	**12.2±2.1**	2.2±0.9	31.8±2.8	6.3±1.8	12.7±2.0
BAHSIC	13.7±4.3	**1.9±1.3**	**30.5±3.3**	6.1±3.8	9.0±6.0

5.7 Analysis of Microarray Data

The fact that BAHSIC may be instantiated in numerous ways may create problems for application, that is, it is not immediately clear which criteria we might want to choose. Here we provide guidelines for choosing a specific member of the BAHSIC family by using gene selection as an illustration.

Datasets While some past work focused on analysis of a *specific* single microarray dataset we decided to perform a large scale comparison of a range of techniques on many datasets. We believe that this leads to a more accurate description of the performance of feature selectors. We ran our experiments on 28 datasets, of which 15 are two-class datasets and 13 are multiclass datasets. These datasets are assigned a reference number for convenience. Two-class datasets have a reference number less than or equal to 15, and multiclass datasets have reference numbers of 16 and above. Only one dataset, yeast, has feature dimension less than 1000 (79 features). All other datasets have dimensions ranging from approximately 2000 to 25000. The number of samples varies between approximately 50 and 300 samples. A summary of the datasets and their sources is as follows:

- The six datasets studied in [44]. Three deal with breast cancer [141, 140, 146] (numbered 1, 2 and 3), two with lung cancer [18, 16] (4, 5), and one with hepatocellular carcinoma [74] (6). The B cell lymphoma dataset [111] is not used because none of the tested methods produce classification errors lower than 40%.
- The six datasets studied in [147]. Two deal with prostate cancer [34, 151] (7, 8), two with breast cancer [62, 152] (9, 10), and two with leukaemia [24, 139] (16, 17).
- Five commonly used bioinformatics benchmark datasets on colon cancer [2] (11), ovarian cancer [17] (12), leukaemia [57](13), lymphoma [1](18), and yeast [23](19).
- Nine datasets from the NCBI GEO database. The GDS IDs and reference numbers for this paper are GDS1962 (20), GDS330 (21), GDS531 (14), GDS589 (22), GDS968 (23), GDS1021 (24), GDS1027 (25), GDS1244 (26), GDS1319 (27), GDS1454 (28), and GDS1490 (15), respectively.

Classification Error and Robustness of Genes We used stratified 10-fold cross-validation and SVMs to evaluate the predictive performance of the top 10 features selected by various members of BAHSIC. For two-class datasets, a nonlinear SVM with a Gaussian RBF kernel, $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$, was used. The regularization constant C and the kernel width σ were tuned on a grid of $\{0.1, 1, 10, 10^2, 10^3\} \times \{1, 10, 10^2, 10^3\}$. Classification performance is measured as the fraction of misclassified samples. For multiclass datasets, all procedures

are the same except that we used the SVM in a one-versus-the-rest fashion. A new BAHSIC member are also included in the comparison, with kernels $(\|\mathbf{x} - \mathbf{x}'\| + \epsilon)^{-1}$ (dis; ϵ is a small positive number to avoid singularity) on the data.

The classification results for binary and multiclass datasets are reported in Table 5.3 and Table 5.4, respectively. In addition to error rate we also report the overlap between the top 10 gene lists created in each fold. The multiclass results are presented separately since some older members of the BAHSIC family, and some competitors, are not naturally extensible to multiclass datasets. From the experiments we make the following observations:

When comparing the overall performance of various gene selection algorithms, it is of primary interest to choose a method which works well *everywhere*, rather than one which sometimes works well and sometimes performs catastrophically. It turns out that the linear kernel (lin) outperforms all other methods in this regard, both for binary and multiclass problems.

To show this, we measure how various methods compare with the best performing one in each dataset in Tables 5.3 and 5.4. The deviation between algorithms is taken as the square of the difference in performance. This measure is chosen because gene expression data is relative expensive to obtain, and we want an algorithm to select the best genes from them. If an algorithm selects genes that are far inferior to the best possible among all algorithms (catastrophic case), we downgrade the algorithm more heavily. Squaring the performance difference achieves exactly this effect, by penalizing larger differences more heavily. In other words, we want to choose an algorithm that performs homogeneously well in all datasets. To provide a concise summary, we add these deviations over the datasets and take the square root as the measure of goodness. These scores (called ℓ_2 distance) are listed in Tables 5.3 and 5.4. In general, the smaller the ℓ_2 distance, the better the method. It can been seen that the linear kernel has the smallest ℓ_2 distance on both the binary and multiclass datasets.

Subtype Discrimination using Nonlinear Kernels We now investigate why it is that nonlinear kernels (RBF and dis) provide better genes for classification in three datasets from Table 5.4 (datasets 18 [1], 27 (GDS1319), and 28 (GDS1454)). These datasets all represent multiclass problems, where at least two of the classes are subtypes with respect to the same supertype.[7] Ideally, the selected genes should contain information discriminating the classes. To visualise this information, we plot in Figure 5.4 the expression value of the top-ranked gene against that of a second gene ranked in the top 10. This second gene is chosen so that it has minimal correlation with the first gene. We use colors and shapes to distinguish data from different classes (datasets 18 and 28 each contain 3 classes, therefore we use 3 different colour and shape combinations for them; dataset 27 has 4 classes, so we use 4 such combinations).

We found that genes selected using nonlinear kernels provide better separation between the two classes that correspond to the same supertype (red dots and green diamonds), while the genes selected with the linear kernel do not separate these subtypes well. In the case of dataset 27, the increased discrimination between red and green comes at the cost of a greater number of errors in another class (black triangle), however these mistakes are less severe than the errors made between the two subtypes by the linear kernel. This eventually leads to better classification performance for the nonlinear kernels (see Table 5.4).

The principal characteristic of the datasets is that the blue square class is clearly separated

[7] For dataset 18, the 3 subtypes are diffuse large B-cell lymphoma and leukemia, follicular lymphoma, and chronic lymphocytic leukemia; For dataset 27, the 4 subtypes are various C blastomere mutant embryos: wild type, pie-1, pie-1+pal-1, and mex-3+skn-1; For dataset 28, the 3 subtypes are normal cell, IgV unmutated B-cell, and IgV mutated B-cell.

from the rest, while the difference between the two subtypes (red dots and green diamonds) is less clear. The first gene provides information that distinguishes the blue square class, however it provides almost no information about the separation between the two subtypes. The linear kernel does not search for information complementary to the first gene, whereas nonlinear kernels are able to incorporate complementary information. In fact, the second gene that distinguishes the two subtypes (red dots and green diamonds) does not separate all classes. From this gene alone, the blue square class is heavily mixed with other classes. However, combining the two genes together results in better separation between all classes.

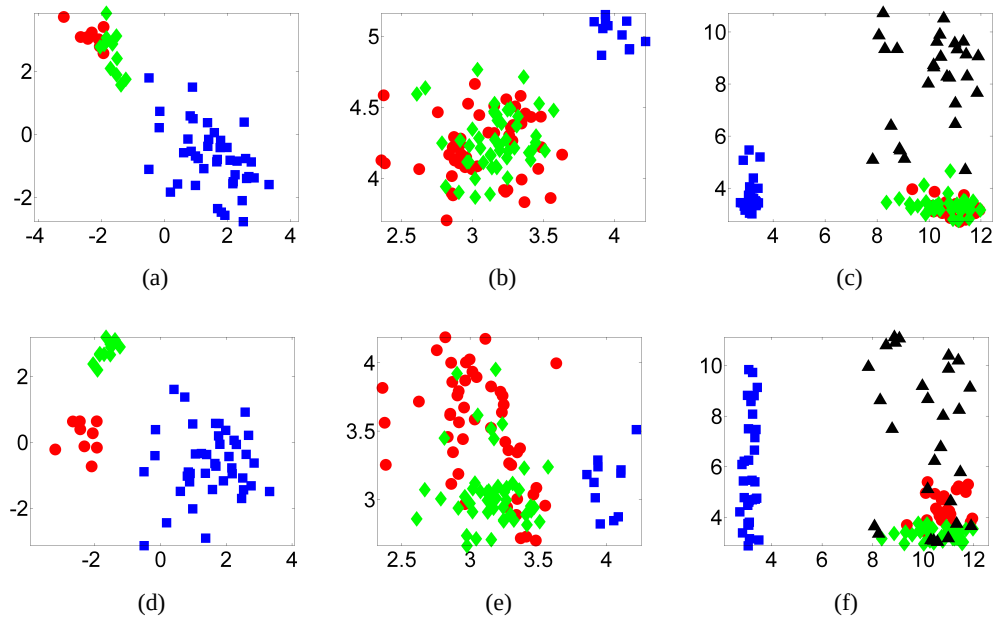


Figure 5.4: Nonlinear kernels (MUL and dis) select genes that discriminate subtypes (red dots and green diamonds) where the linear kernel fails. The two genes in the first row are representative of those selected by the linear kernel, while those in the second row are produced with a nonlinear kernel for the corresponding datasets. Different colors and shapes represent data from different classes. (a,d) dataset 18; (b,e) dataset 28; and (e,f) dataset 27.

Rules of Thumb and Implication to Gene Activity To conclude these experiments, considering the fact that the linear kernel performed best in our feature selection evaluation, yet also taking into account the existence of nonlinear interaction between genes (as demonstrated in section 5.7), we propose the following two rules of thumb for gene selection:

1. Always apply a linear kernel for general purpose gene selection.
2. Apply a Gaussian kernel if nonlinear effects are present, such as multimodality or complementary effects of different genes.

This result should come as no surprise, due to the high dimensionality of microarray datasets, but we corroborate our claims by means of an extensive experimental evaluation. These experiments also imply a desirable property of gene activity as a whole: it correlates well with the observed outcomes. Multimodal and highly nonlinear situations exist, where a nonlinear feature selector is needed (as can be seen in the outcomes on datasets 18, 27 and 28), yet they occur relatively rarely in practice.

Table 5.3: Two-class datasets: classification error (%) and number of common genes (overlap) for 10-fold cross-validation using the top 10 selected features. Each *row* shows the results for a dataset, and each *column* is a method. Each entry in the table contains two numbers separated by "|": the first number is the classification error and the second number is the number of overlaps. For classification error, the best result, and those results not significantly worse than it, are highlighted in bold (one-sided Welch t-test with 95% confidence level; a table containing the standard errors is provided in the *supplementary material*). For the overlap, largest overlaps for each dataset are highlighted (no significance test is performed). The second last *row* summarizes the number of times a method was the best. The last *row* contains the ℓ_2 distance of the error vectors between a method and the best performing method on each dataset. We use the following abbreviations: pc - Pearson's correlation, snr - signal-to-noise ratio, pam - shrunken centroid, t - t-statistics, m-t - moderated t-statistics, lods - B-statistics, lin - centroid, dis - $(\|\mathbf{x}-\mathbf{x}'\|+\epsilon)^{-1}$, rfe - svm recursive feature elimination)

Dataset	pc	snr	pam	t	m-t	lods	lin	RBF	dis	rfe
1	12.7|3	11.4|3	11.4|**4**	12.9|3	12.9|**4**	12.9|**4**	15.5|3	19.1|1	13.9|2	14.3|0
2	33.2|1	33.9|2	33.9|1	**29.5**|1	29.5|1	**27.8**|1	32.9|2	31.5|**3**	32.8|2	34.2|0
3	37.4|0	37.4|0	37.4|0	**34.6**|**6**	**34.6**|**6**	**34.6**|**6**	37.4|1	37.4|0	37.4|0	37.4|0
4	41.6|0	**38.8**|0	41.6|0	40.7|**1**	**40.7**|0	**37.8**|0	41.6|0	41.6|0	**39.7**|0	41.6|0
5	**27.8**|0	**26.7**|0	**27.8**|0	**26.7**|**2**	**26.7**|**2**	**26.7**|**2**	**27.8**|0	**27.8**|0	**27.6**|0	**27.8**|0
6	30.0|2	**25.0**|0	31.7|0	**25.0**|**5**	**25.0**|**5**	**25.0**|**5**	30.0|0	31.7|0	30.0|1	30.0|0
7	**2.0**|**6**	**2.0**|5	**2.0**|5	28.7|4	26.3|4	26.3|4	**2.0**|3	**2.0**|4	30.0|0	**2.0**|0
8	3.3|3	**0.0**|4	**0.0**|4	**0.0**|4	3.3|**6**	3.3|**6**	3.3|2	3.3|1	6.7|2	**0.0**|0
9	**10.0**|**6**	**10.0**|**6**	**8.7**|4	34.0|5	37.7|**6**	37.7|**6**	12.0|3	**10.0**|5	12.0|1	**10.0**|0
10	**16.0**|2	**18.0**|2	**14.0**|2	**14.0**|8	22.0|**9**	22.0|**9**	**16.0**|2	**16.0**|0	**18.0**|0	32.5|0
11	12.9|5	12.9|5	12.9|5	19.5|0	22.1|0	33.6|0	**11.2**|4	**9.5**|**6**	16.0|4	19.0|0
12	30.3|2	36.0|2	31.3|2	26.7|**3**	35.7|0	35.7|0	**18.7**|1	35.0|0	33.0|1	29.7|0
13	8.4|5	11.1|0	7.0|5	22.1|3	27.9|**6**	15.4|1	7.0|2	9.6|0	11.1|0	**4.3**|1
14	20.8|1	20.8|1	20.2|0	20.8|**3**	20.8|**3**	20.8|**3**	20.8|0	20.2|0	**19.7**|0	20.8|0
15	**0.0**|7	0.7|1	**0.0**|5	4.0|1	0.7|**8**	0.7|**8**	**0.0**|3	**0.0**|2	2.0|2	**0.0**|1
best	5|2	7|1	6|1	6|6	4|10	5|9	6|0	6|2	4|0	6|0
ℓ_2	16.9	20.9	17.3	43.5	50.5	50.3	**13.2**	22.9	35.4	26.3

Table 5.4: Multiclass datasets: in this case *columns* are the datasets, and *rows* are the methods. The remaining conventions follow Table 5.3.

Data	16	17	18	19	20	21	22	23	24	25	26	27	28	best	ℓ_2
lin	36.7|1	**0.0**|3	5.0|**3**	10.5|6	35.0|3	**37.5**|**6**	**18.6**|1	**40.3**|**3**	**28.1**|**3**	**26.6**|**6**	**5.6**|**6**	27.9|7	45.1|1	7|6	**32.4**
RBF	**33.3**|**3**	5.1|4	**1.7**|**3**	**7.2**|**9**	33.3|0	40.0|1	22.1|0	72.5|0	39.5|0	**24.7**|4	**5.6**|**6**	22.1|**10**	**21.5**|3	6|5	37.9
dis	**29.7**|2	28.8|**5**	6.7|0	**8.2**|**9**	**29.4**|**7**	**38.3**|4	43.4|**4**	66.1|0	40.8|0	38.9|4	7.6|1	**8.2**|8	31.6|3	5|4	51.0

5.8 Summary

This chapter provides a *unifying* framework for a range of feature selection methods. This allows us to extend the tail bounds and asymptotic analysis for HSIC to these feature selectors. Moreover, we are able to design new feature selectors which work well in practice by means of the Hilbert-Schmidt Independence Criterion (HSIC).

The idea behind the resulting algorithm, BAHSIC, is to choose the feature subset that maximizes the dependence between the data and labels. The good convergence property of the empirical HSIC estimate provide a strong theoretical justification for using HSIC in this context. Although BAHSIC is a filter method, it still demonstrates good performance compared with more specialized methods in both artificial and real world data. It is also very competitive in terms of runtime performance.[8]

[8] Code is available as part of the Elefant package at http://elefant.developer.nicta.com.au.

CHAPTER 6

Clustering via Dependence Estimation

In this chapter, we will use dependence as measured by HSIC for clustering. We show that clustering is a process of maximizing the dependence between the data and the generated labels. Furthermore, clustering via HSIC (CLUHSIC) is equivalent to the 0-Extension problem in theoretical computer science, and thereby it also provides a statistical interpretation for the 0-Extension problem. We will design heuristics to solve CLUHSIC and also study its stability under perturbations.

6.1 Introduction

Given a set of observations $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \in \mathcal{X}$, the goal of clustering is to associate with each observation a label chosen from a smaller set $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \in \mathcal{Y}$, such that the $\mathbf{x}_i$ grouped together share some underlying property. This is useful both in terms of gaining a better understanding of the data, and in obtaining a succinct representation.

A popular clustering algorithm is k-means [91], which can be kernelized straightforwardly [116, 55]. Points are grouped in k clusters so as to minimize intra-cluster variance, and the resulting k cluster centroids are used to represent their respective clusters. This problem is NP-hard; however a simple heuristic often produces acceptable results: random initialization followed by iterative update of cluster centroids and the data partition [91]. Recently, [158, 36] proposed to use the principal components of the data as a more efficient initialization for k-means. Besides the geometric view of clustering, spectral methods have gained considerable popularity over the past decade, e.g. [121, 101]. These methods treat each datum as a node in a nearest neighbor graph and exploit the piecewise constant eigenvectors of a graph Laplacian for the initial partition. Usually, thresholding or a second k-means is performed to obtain a final cluster assignment. A third view of clustering arises from statistical considerations: given X, we want to find Y such that the statistical dependence between X and Y is maximized. In the case of "hard" clustering, where each observation is assigned to exactly one cluster, mutual information $I(X, Y)$ has been used as the objective [122, Section 4.2]. It is difficult to compute mutual information in high dimensions, however, since sophisticated bias correction methods must be employed [99].

A natural question is how these different approaches are related. Early work by [158, 36] show the connection between the geometric view and spectral view of k-means. In this paper, we will start from a statistical dependence view of clustering, and lead naturally to the unification of the three views at a statistical inference level. Instead of $I(X, Y)$, however, we use the Hilbert-Schmidt Independence Criterion (HSIC) [59] as our measure of statistical dependence. HSIC has several advantages: first, it does not require density estimation, and has good uniform convergence guarantees; second, it has very little bias, even in high dimensions;

and third, a number of algorithms can be viewed as maximizing HSIC subject to constraints on Y, for particular Hilbert spaces on the inputs and labels. Thus, we propose a family of feature extraction and clustering methods, of which k-means and spectral clustering are special cases. Another distinctive feature of our framework is that rich choices of kernels can also be applied to the label spaces. This provides us with new clustering algorithms that generate cluster labels with structures.

Given the many possible clustering algorithms that arise from particular kernel choices, we face the difficulty of choosing which kernel gives the most useful partition of our data. We find the kernels that provide the most reasonable clustering in our examples are also the most stable under perturbation. We measure the stability using a novel bound on the change in clustering performance under perturbation. Unlike previous such bounds by [101, 78, 94], ours holds even under large perturbations, and is distribution independent. We also use this bound to determine the accuracy required for low rank approximations to the input Gram matrix, while still maintaining good clustering performance.

6.2 Clustering via HSIC

Using HSIC as our dependence measure, we now devise an optimization problem for clustering via

$$\begin{aligned} Y^* = \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \ & \operatorname{tr}\left(\mathbf{HKHL}(Y)\right) \\ \text{subject to constraints on } & Y, \end{aligned} \tag{6.1}$$

where $\mathbf{L}(Y)$ represents the kernel matrix of the generated labels. The constraints on Y serve two purposes: to ensure dependence is comparable as Y changes; and to endow Y with particular structures. For example, we can use a vector of real numbers, $\mathbf{y}_i$, as the label (from which we form the label feature matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)^\top$). If we further require that the columns of $\mathbf{Y}$ be orthogonal with unit norm, we can recover kernel PCA by simply applying a linear kernel on $\mathbf{Y}$ [116]. However, the focus of this chapter is to investigate the case of clustering, where the label space $\mathcal{Y} = \{y_1, \ldots, y_c\}$ contains only $c \ll m$ distinctive elements.

In this case, we effectively have the liberty of choosing a symmetric positive semidefinite matrix $\mathbf{A}$ of size $c \times c$ defining the similarity between elements in $\mathcal{Y}$. Then the kernel matrix of the labels can be parameterized using a partition matrix, Π, of size $m \times c$,

$$\mathbf{L} = \Pi \mathbf{A} \Pi^\top, \ (\Pi \mathbf{1} = \mathbf{1}, \ \Pi_{ij} \in \{0, 1\}). \tag{6.2}$$

Each row of the partition matrix, Π, contains all zeros but a single entry of 1, and $\mathbf{1}$ denotes a vector of all ones. Π effectively constrains us to assign each data point to a particular label by putting 1 in an appropriate column. In this case, (6.1) becomes

$$\begin{aligned} \Pi^* = \underset{\Pi}{\operatorname{argmax}} \ & \operatorname{tr}\left(\mathbf{HKH}\Pi\mathbf{A}\Pi^\top\right) \\ \text{subject to } & \Pi\mathbf{1} = \mathbf{1}, \ \Pi_{ij} \in \{0, 1\}. \end{aligned} \tag{6.3}$$

Using properties of the trace,

$$\operatorname{tr}\left((\Pi^\top \mathbf{HKH}\Pi)\mathbf{A}\right) = \mathbf{1}^\top \left((\Pi^\top \mathbf{HKH}\Pi) \circ \mathbf{A}\right) \mathbf{1}, \tag{6.4}$$

where $\circ$ denotes elementwise matrix multiplication. This means that in (6.3) we try to partition the kernel matrix such that the sum of kernel entries in the corresponding blocks best align with the given structure in $\mathbf{A}$. Very often we will use the normalized partition matrix $\mathbf{P}$ instead of Π. These are related by $\mathbf{P} = \Pi\mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix with entries $\mathbf{D}_{ii} = (\mathbf{1}^\top \Pi_{\star i})^{-1/2}$ (We use subscript $\star i$ to represent a column; similarly, $i\star$ represents a row. We will denote the number of inputs assigned to cluster i as $m_i = |\Pi_{\star i}|$ and $j \in \Pi_{\star i}$ denotes the indices of the inputs that are in cluster i). In this case, the sum of kernel entries in each block is normalized before the alignment. By choosing different $\mathbf{A}$s, we will obtain a family of different clustering methods as shown in the next section.

6.3 CLUHSIC Family

Our formulation in (6.3) is very general: we can obtain a family of clustering algorithms by combining a kernel for the input space and another one for the labels. While different kernels for the input space have been explored extensively [eg. 116, 55], almost no studies have explicitly investigated kernels on the label space. In this section, we will focus on the rich choice of kernels for the labels, and the associated clustering algorithms. Although these can in principle be very general, we will present certain label spaces that are of particular interest for clustering.

Plain (kernel) k-means We set $\mathbf{A}$ to a diagonal matrix [158, 36, 157] (3 cluster example)

$$\mathbf{A} = \operatorname{diag}\left(\left[\tfrac{1}{m_1} \quad \tfrac{1}{m_2} \quad \tfrac{1}{m_3}\right]\right), \tag{6.5}$$

where $\operatorname{diag}(\mathbf{a})$ is a diagonal matrix with non-zero entries $\mathbf{a}$. Alternatively, we can use the normalized partition matrix $\mathbf{P}$ and set $\mathbf{A}$ as the identity matrix. To show its equivalence to k-means clustering, we start from the objective of k-means: partition the data into k clusters such that the overall within-cluster variance is minimized, ie.

$$\underset{\Pi}{\text{minimize}} \sum_{i=1}^{c} \sum_{j \in \Pi_{\star i}} \|k(\mathbf{x}_j, \cdot) - \mu_i\|_{\mathcal{F}}^2 \tag{6.6}$$

$$\text{subject to } \mu_i = \frac{1}{m_i} \sum_{j \in \Pi_{\star i}} k(\mathbf{x}_j, \cdot),\ \Pi\mathbf{1} = \mathbf{1},\ \Pi_{ij} \in \{0, 1\}. \tag{6.7}$$

Expanding the objective and taking into account the constraints we have

$$\begin{aligned}
&\sum_{i=1}^{c} \sum_{j \in \Pi_{\star i}} \left(k(\mathbf{x}_j, \mathbf{x}_j) - 2\left\langle k(\mathbf{x}_j, \cdot), \mu_i \right\rangle_{\mathcal{F}} + \|\mu_i\|_{\mathcal{F}}^2\right) \\
&= \sum_{j=1}^{m} k(\mathbf{x}_j, \mathbf{x}_j) - 2\sum_{i=1}^{c} \sum_{j \in \Pi_{\star i}} \left\langle k(\mathbf{x}_j, \cdot), \mu_i \right\rangle_{\mathcal{F}} + \sum_{i=1}^{c} m_i \|\mu_i\|_{\mathcal{F}}^2 \\
&= \text{constant} - \sum_{i=1}^{c} m_i \|\mu_i\|_{\mathcal{F}}^2 = \text{constant} - \sum_{i=1}^{c} \frac{1}{m_i} \sum_{j,s \in \Pi_{\star i}} k(\mathbf{x}_j, \mathbf{x}_s) \\
&= \text{constant} - \operatorname{tr}\left(\mathbf{K}\Pi\mathbf{A}\Pi^\top\right).
\end{aligned} \tag{6.8}$$

Therefore, minimizing the overall within-cluster variance is equivalent to maximizing the statistical dependence, provided that the inputs are already centered in the feature space (ie. $\mathbf{K} = \mathbf{HKH}$). Note that the label kernel $\mathbf{A}$ for k-means requires that the labels be mutually orthogonal and it assumes no relation between clusters. Hence each cluster is assigned to an arbitrary column in Π. This can be misleading when hierarchical or other structures in the inputs exist. Especially when using k-means as a quantization scheme, we are losing the relation between the inputs after the quantization.

Weighted k-means We can generalize k-means algorithm by introducing a weight w_i for each input $\mathbf{x}_i$. The weight can be interpreted as the relative importance of each input. In this case, we set $\mathbf{A}$ to a diagonal matrix (3 cluster example)

$$\mathbf{A} = \operatorname{diag}\left(\begin{bmatrix} \frac{1}{\sum_{i\in\Pi_{\star 1}} w_i} & \frac{1}{\sum_{i\in\Pi_{\star 2}} w_i} & \frac{1}{\sum_{i\in\Pi_{\star 3}} w_i} \end{bmatrix}\right), \tag{6.9}$$

To show this, we first compare it to the objective of weighted k-means

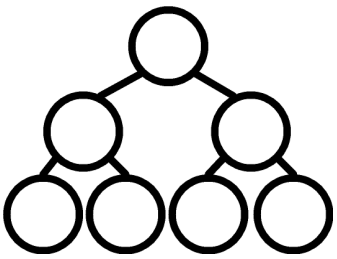
$$\begin{aligned} \underset{\Pi}{\text{minimize}} \sum_{i=1}^{c} \sum_{j\in\Pi_{\star i}} \|w_j k(\mathbf{x}_j,\cdot) - \mu_i\|_{\mathcal{F}}^2 \\ \text{s.t. } \mu_i = \frac{\sum_{j\in\Pi_{\star i}} w_j k(\mathbf{x}_j,\cdot)}{\sum_{j\in\Pi_{\star i}} w_j},\ \Pi\mathbf{1} = \mathbf{1},\ \Pi_{ij} \in \{0,1\}. \end{aligned} \tag{6.10}$$

Expanding similar to the case of k-mean we have

$$\text{constant} - \sum_{i=1}^{c} \frac{\sum_{j,s\in\Pi_{\star i}} w_j w_s k(\mathbf{x}_j,\mathbf{x}_s)}{\sum_{j\in\Pi_{\star i}} w_j} = \text{constant} - \operatorname{tr}\left(\tilde{\mathbf{K}}\Pi\mathbf{A}\Pi^\top\right), \tag{6.11}$$

where the entries in $\tilde{\mathbf{K}}$ is related to those in $\mathbf{K}$ via $\tilde{k}(\mathbf{x}_i,\mathbf{x}_s) = w_i w_s k(\mathbf{x}_i,\mathbf{x}_s)$ [35]. This is equivalent to use a conformal kernel on the inputs [6], while use the inverse of the sum of weights in a cluster as the label similarity. Note that weighted k-means also ignore relations between clusters.

Hierarchical clustering We can easily specify a hierarchy in the cluster labels by making some off-diagonal entries of $\mathbf{A}$ nonzero. For example, we may have a two-level clustering problem: first divide the inputs in 4 clusters, and then further group them into 2 super-clusters (each containing 2 sub-clusters). This requirement can be tackled using $\mathbf{A}$ in equation (6.12). Unlike k-means, the assignment of clusters to columns of Π is no longer arbitrary. Cluster membership will be arranged according to the hierarchical structure specified in $\mathbf{A}$. Therefore, once the inputs are clustered, we can recognize the relations between the clusters by observing the labels.

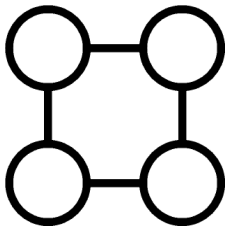


$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}. \tag{6.12}$$

Cluster data in a chain In some cases, the inputs may reside on a manifold, and we want to cluster them by grouping adjacent data points. The clusters are related: they may form a chain or a grid. The choice of $\mathbf{A}$ in equation (6.13) encodes this in the labels (a chain of 3 clusters). This label kernel enforces that adjacent clusters be similar. Therefore, adjacent columns of Π represent adjacent clusters.

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}. \tag{6.13}$$

Cluster data in a ring Besides a chain or a grid structure on the clusters, the inputs may reside on a closed manifold. For example, a ring structured cluster relationship can be induced by using the $\mathbf{A}$ in equation (6.14) (an example with 4 clusters). The two corner entries in the anti-diagonal of $\mathbf{A}$ close the chain structure of (6.13) into a ring. Explicitly enforcing a ring structure in the clusters is useful since this may help avoid ambiguity during the clustering (see experiments). Finally, we note that additional structure might be discovered in the data using more complex label kernels than those covered here, for instance deeper tree hierarchies.



$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}. \tag{6.14}$$

Note that the applicable label kernels are not limited to the ones we mentioned above. Basically, we can design our label kernels such that the resulting clusterings conform to the desired structure. For the examples we mentioned above, we design the kernel matrices for the labels based on graph adjacency matrices. Basically, we first express the relations between the labels as a graph; then we augment the diagonal of the adjacency matrix until it is positive definite. It turns out that our generalization of clustering via HSIC is closely related to the 0-Extension problem in theoretical computer science. Thus our formulation also provides statistical interpretation for the 0-Extension problem.

6.4 Relation to 0-Extension Problem

0-Extension takes as input an undirected graph G with the set of nodes $V(G)$ and the set of edges $E(G)$. A weight function $k(\mathbf{x}_i, \mathbf{x}_j)$ is defined over each edge $\{\mathbf{x}_i, \mathbf{x}_j\} \in E(G)$.[1] Given a set of labels $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_c\}$ and a metric d between its elements, the goal of 0-Extension is to assign each node to a label, such that the following cost is minimized

$$\sum_{\{\mathbf{x}_i, \mathbf{x}_j\} \in E(G)} k(\mathbf{x}_i, \mathbf{x}_j) d(\mathbf{y}_i, \mathbf{y}_j). \tag{6.15}$$

Here we use $\mathbf{y}_i \in \mathcal{Y}$ to denote the label assigned to node $\mathbf{x}_i$. The objective means that we are partitioning the graph into c pieces, where the cost of sending endpoint $\mathbf{x}_i$ and $\mathbf{x}_j$ of an edge to

[1] Note that we denote a node of the graph G by $\mathbf{x}_i$, and we deliberately use $k(\mathbf{x}_i, \mathbf{x}_j)$ as the edge weight. In another word, the edge weight will correspond to the similarity between two nodes $\mathbf{x}_i \in V(G)$ and $\mathbf{x}_j \in V(G)$.

different parts depends on the parts to which $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned. Intuitively, if two nodes are very similar, then assigning them to different labels incurs a large cost. Multiway cut [32] is a special case of 0-Extension: multiway cut uses a simple uniform metric as d, ie. it optimizes over the objective

$$\sum_{\{\mathbf{x}_i,\mathbf{x}_j\}\in E(G)} k(\mathbf{x}_i,\mathbf{x}_j)(1-\delta_{\mathbf{y}_i\mathbf{y}_j}). \tag{6.16}$$

0-Extension takes its name from the fact that we wish to extend the metric d on the labels to a semi-metric on all nodes $\mathbf{x}_i \in V(G)$ subject to the restriction that every nodes within a partition must be at distance 0 from each other.

6.4.1 0-Extension as CLUHSIC

We can show that 0-Extension is equivalent to clustering via HSIC. In this case, the graph G is a complete graph and each input is treated as a node in G. Then the edge weights correspond to the entries in the centered input kernel matrix. Let $d(\mathbf{y}_i,\mathbf{y}_j) = l(\mathbf{y}_i,\mathbf{y}_i) - 2l(\mathbf{y}_i,\mathbf{y}_j) + l(\mathbf{y}_i,\mathbf{y}_j)$, we have

$$\begin{aligned}\sum_{\{\mathbf{x}_i,\mathbf{x}_j\}\in E(G)} k(\mathbf{x}_i,\mathbf{x}_j)d(\mathbf{y}_i,\mathbf{y}_j) &= \sum_{\{\mathbf{x}_i,\mathbf{x}_j\}\in E(G)} (\mathbf{HKH})_{ij}\left(l(\mathbf{y}_i,\mathbf{y}_i) - 2l(\mathbf{y}_i,\mathbf{y}_j) + l(\mathbf{y}_j,\mathbf{y}_j)\right)\\ &= \sum_{\{\mathbf{x}_i,\mathbf{x}_j\}\in E(G)} (\mathbf{HKH})_{ij}\left(\mathbf{A}_{ii} - 2\mathbf{A}_{ij} + \mathbf{A}_{jj}\right)\\ &= -\operatorname{tr}\left(\mathbf{HKH}\Pi\mathbf{A}\Pi^\top\right). \end{aligned} \tag{6.17}$$

where we express the assignment using the partition matrix Π and use the following properties of a centered kernel matrix, ie.

$$\sum_{\mathbf{x}_j\in V(G)} (\mathbf{HKH})_{ij}\,\mathbf{A}_{ii} = 0 \text{ and} \tag{6.18}$$

$$\sum_{\mathbf{x}_i\in V(G)} (\mathbf{HKH})_{ij}\,\mathbf{A}_{jj} = 0. \tag{6.19}$$

The connection between CLUHSIC and 0-Extension suggests that we can use algorithms for 0-Extension to solve our clustering problem here. Unfortunately, 0-Extension is an NP-hard problem [79, 9], and the state-of-the-art approximate algorithm is based on a relaxation using Earthmover distance. Furthermore, [79] has proved that this approximation can be computed in polynomial-time and guaranteed a performance ratio of $O(\sqrt{\operatorname{diam}(d)})$, where $\operatorname{diam}(d)$ is the ratio of the largest to smallest nonzero distances in the label metric. In the next two sections, we will introduce Earthmover distance and the corresponding relaxation method for 0-Extension.

6.4.2 Earthmover Distance

Earthmover distance (EMD) is a distance measure between sets of elements or distributions which naturally extends the notion of (ground) distance between single elements in the set. Intuitively, given two distributions, one can be seen as piles of earth properly spread in space, and the other as a collection of holes in the same space. Then the EMD measures the amount of work needed to fill the holes with the earth. We assume for simplicity that the amount of earth

is the same as the capacity of the holes, and a unit of work corresponds to transporting a unit of earth by a unit of ground distance. Earthmover distance has wide applications, for instance in image retrieval [112].

Earthmover distance can be computed via linear programming. Let $\mathcal{I}$ be the set of piles of earth, $\mathcal{J}$ the set of holes, and c_{ij} the cost to ship a unit of earth from pile $i \in \mathcal{I}$ to hole $j \in \mathcal{J}$. We want to find a set of flows of earth f_{ij} that minimize the overall cost

$$\text{EMD}(\mathcal{I}, \mathcal{J}) = \underset{f_{ij}}{\text{minimize}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} f_{ij} \tag{6.20}$$

$$\text{subject to} \sum_{i \in \mathcal{I}} f_{ij} = q_j, \ \forall j \in \mathcal{J} \tag{6.21}$$

$$\sum_{j \in \mathcal{J}} f_{ij} = p_i, \ \forall i \in \mathcal{I} \tag{6.22}$$

$$f_{ij} \geq 0, \ \forall i \in \mathcal{I}, \ \forall j \in \mathcal{J}, \tag{6.23}$$

where p_i is the amount of earth in pile i and q_j is the capacity of hole j. Constraint (6.21) forces the holes to fill up all their capacities and constraint (6.22) limits the supply from a pile to its total amount. Constraint (6.23) enforces positive flow of earth. A feasibility condition of the linear programming is that the total amount of earth is the same as the total capacity of the holes

$$\sum_{j \in \mathcal{J}} q_j \ = \ \sum_{i \in \mathcal{I}} p_i. \tag{6.24}$$

Then EMD is equal to the overall cost produced by the optimal flow of earth. If the ground distance is a metric then the EMD is also a metric.

6.4.3 CLUHSIC using Relaxed 0-Extension

0-Extension can be relaxed using Earthmover distance. The basic idea is that rather than assigning each node to a single label we instead associate a node with a probability of its assignment to different labels. We then replace the metric on the assigned labels by the Earthmover distance between their assignment probabilities. Thereby we optimize over the positive real numbers of the probabilities instead of a hard assignment. Once we obtain a solution of the relaxed problem (the soft assignment), we then apply a randomized algorithm to round it to the hard assignment. In this section, we will mainly talk about the earthmover relaxation to 0-Extension; a randomized algorithm for rounding the solution can be found in [9].

Let Pr_i denote the probability of label assignment associated with node $\mathbf{x}_i$. Then the Earthmover distance relaxation to 0-Extension can be formulated as

$$\underset{\text{Pr}}{\text{minimize}} \sum_{\{\mathbf{x}_i, \mathbf{x}_j\} \in E(G)} k(\mathbf{x}_i, \mathbf{x}_j) \text{EMD}(\text{Pr}_i, \text{Pr}_j). \tag{6.25}$$

Given c different labels, this minimization problem involves $c^2 \left|E(G)\right| + c \left|V(G)\right|$ number of variables. The first part of variables ($c^2 \left|E(G)\right|$) originates from the flow computation for each pair of nodes in an edge; the second part ($c \left|V(G)\right|$) originates from specifying the probability distribution Pr_i for each node $\mathbf{x}_i$. For CLUHSIC, the associated graph for the 0-Extension problem is complete, which means that given m inputs, the number of variables will

Algorithm 3 Iterative CLUHSIC

Input: The kernel matrices $\mathbf{K}$ and $\mathbf{A}$, the number of data m, and the number of cluster c.
Output: The cluster assignment, Π^*.

1: $\Pi \leftarrow \Pi_0$
2: **repeat**
3: $\quad \Pi_0 \leftarrow \Pi$
4: $\quad$ **for all** $i \in \{1 \dots m\}$ **do**
5: $\quad\quad j_0 \leftarrow \operatorname{argmax}_j \operatorname{tr}(\mathbf{K}\Pi_{i\star}^{j}\mathbf{A}\Pi_{i\star}^{j\top}),\ j \in \{1 \dots c\}$
$\quad\quad\quad$ subject to $\Pi\mathbf{1} = \mathbf{1},\ \Pi_{ij} \in \{0, 1\}$
6: $\quad\quad \Pi \leftarrow \Pi_{i\star}^{j_0}$
7: $\quad$ **end for**
8: **until** $\Pi_0 = \Pi$

be $\frac{c^2 m(m-1)}{2} + cm$. For a small-sized problem of $m = 100$ and $c = 2$, we will then have more than $10,000$ variables. For large scale problem, this approach may not be practical since the linear programming becomes very large. Therefore, in the next section, we will design other heuristic to solve CLUHSIC.

6.5 Algorithms for CLUHSIC

In this section, we present several heuristics for clustering using HSIC. We expect them to be more efficient than solving the relaxed 0-Extension problem.

6.5.1 Iterative CLUHSIC

An iterative procedure for CLUHSIC is presented in Algorithm 3. It first randomly initializes the partition matrix Π (step 1). Then it iterates through each data point, finding the cluster assignment of each point that maximizes the dependence (step 5). The algorithm terminates when changes of assignment of any single data point can no longer increase the dependence (step 8). We denote $\Pi_{i\star}^{j}$ as the partition matrix derived from Π by reassigning data point i to cluster j.

This greedy procedure is guaranteed to terminate, since in each step we increase the objective and there is only finite number of different cluster assignments. We observe experimentally that the outer loop repeats around 20 times. Further speedup is also possible by taking into account the fact that adjacent computations of $\operatorname{tr}(\mathbf{K}\Pi_{i\star}^{j}\mathbf{A}\Pi_{i\star}^{j\top})$ share many common structures.

6.5.2 Spectral Method for CLUHSIC

In some cases, methods more sophisticated can be used to solve CLUHSIC (step 1 in Algorithm 3). [158] showed that spectral relaxation can be effectively used to initialize k-means clustering. Viewed in our framework, the connection between k-means and its spectral relaxation is much simpler conceptually: both are dependence maximization processes, and they differ only in their label spaces.

Let $\mathcal{D}(Y^*) = \operatorname{tr}(\mathbf{HKHL}(Y^*))$ be the maximal dependence captured by a label space $\mathcal{Y}$. A larger label space (for the same input kernel) may capture more information about the data:

given two label spaces $\mathcal{Y}_1$ and $\mathcal{Y}_2$, we have

$$\mathcal{Y}_1 \subseteq \mathcal{Y}_2 \Rightarrow \mathcal{D}(Y_1^*) \leq \mathcal{D}(Y_2^*), \tag{6.26}$$

where $Y_1^* \in \mathcal{Y}_1$ and $Y_2^* \in \mathcal{Y}_2$ are the maximizers of problem (6.1) for $\mathcal{Y}_1$ and $\mathcal{Y}_2$, respectively. This relation suggests a general strategy for problem (6.1): apply a cascade of relaxations on the label spaces $\mathcal{Y} \subseteq \mathcal{Y}_1 \subseteq \ldots \subseteq \mathcal{Y}_n$, and use the more relaxed solutions to initialize the less relaxed ones. Note that using a graph kernel for the inputs establishes the link between k-means and spectral clustering [121, 101, 95].

This strategy has been exploited in the spectral relaxation of k-means [158, 36]: first, we drop all constrains on the normalized partition matrix $\mathbf{P}$ except $\mathbf{P}^\top\mathbf{P} = \mathbf{I}$. We can then efficiently obtain an initialization for k-means via principal component analysis and a pivoted QR decomposition [158, 36]. In this case, the relaxation takes the following trace maximization form

$$\begin{aligned} &\underset{\mathbf{P}}{\text{maximize}} \ \operatorname{tr}\left(\mathbf{P}^\top\mathbf{H}\mathbf{K}\mathbf{H}\mathbf{P}\right) \\ &\text{subject to } \mathbf{P}^\top\mathbf{P} = \mathbf{I} \end{aligned} \tag{6.27}$$

which can be solved via eigen-decomposition due to the following theorem by Ky Fan [72].

Theorem 45 (Ky Fan) *Let* $\mathbf{HKH}$ *be symmetric matrix with eigenvalues*

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m, \tag{6.28}$$

and the corresponding eigenvectors $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m)$*. Then the optimal solution for the trace maximization problem in equation (6.27) is given by* $\mathbf{P}^*\mathbf{Q}$ *where* $\mathbf{P}^* = (\mathbf{u}_1, \ldots, \mathbf{u}_c)$ *and* $\mathbf{Q}$ *an arbitrary orthogonal matrix. The maximum is equal to* $\sum_{i=1}^{c} \lambda_i$*.*

In a way, we obtain a dimension reduced representation $\mathbf{P}^*$ of the inputs, and we want to derive a cluster assignment from this new representation. The basic idea is to rotate $\mathbf{P}^*$ such that the cluster assignment becomes apparent. Pivoted QR decomposition is suitable for this task if we make the following assumption: the variance in the new presentation is mainly caused by between-cluster variation; and we hope that by picking the pivots for the QR decomposition we are picking typical points from each cluster. Suppose we have properly arranged the inputs such that the c pivots have been placed in the first c rows, then we will have the decomposition

$$\mathbf{P}^* = \mathbf{R}\mathbf{Q} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{pmatrix} \mathbf{Q} \tag{6.29}$$

where $\mathbf{R}_1$ is a $c \times c$ lower triangular matrix and $\mathbf{Q}$ is a $c \times c$ rotation matrix. Therefore, $\mathbf{R}$ is an equivalent solution to $\mathbf{P}^*$. If we further transform $\mathbf{R}$ such that

$$\hat{\mathbf{R}} = \begin{pmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{pmatrix} \mathbf{R}_1^{-1} = \begin{pmatrix} \mathbf{I} \\ \mathbf{R}_2\mathbf{R}_1^{-1} \end{pmatrix}, \tag{6.30}$$

then the cluster assignments of the pivots become apparent: each pivot is unambiguously assigned to a separate column due to the form of $\mathbf{I}$. The assignments of other inputs can be determined by the column indices of the largest element in the corresponding rows. Note that, however, this heuristic based on QR decomposition is applicable only to k-means clustering. For other cases where we have more complicated kernels on the labels (ie. $\mathbf{A} \neq \mathbf{I}$), we need

more generic initialization schemes. The nonnegative matrix factorization technique described in the next section provides an alternative.

6.5.3 Nonnegative Matrix Factorization

Approximate solution to CLUHSIC can also be obtained via nonnegative matrix factorization technique (NMF). In this case, however, we need to modify the kernel matrix $\mathbf{K}$ such that all entries are positive. We can easily do this by adding a constant offset to the kernel matrix. In another word, the kernel matrix is no longer centered. Although this modification does not corresponds exactly to maximizing the dependence, it provides good results in some cases in our later experiments. Furthermore, note that NMF only works when the kernel on the labels $\mathbf{A}$ is also nonnegative. For simplicity, we will use $\mathbf{K}$ to denote the modified kernel matrix, and then the NMF formulation of CLUHSIC takes the following form

$$\underset{\mathbf{P}}{\text{minimize}} \; -\operatorname{tr}\left(\mathbf{KPAP}^\top\right) \tag{6.31}$$

$$\text{subject to } \mathbf{P}^\top\mathbf{P} = \mathbf{I},\ \mathbf{P} \geq \mathbf{0}. \tag{6.32}$$

The Lagrangian of the problem is

$$\mathcal{L} = -\operatorname{tr}\left(\mathbf{KPAP}^\top\right) + \operatorname{tr}\left(\lambda\left(\mathbf{P}^\top\mathbf{P} - \mathbf{I}\right)\right). \tag{6.33}$$

Taking the derivative of the Lagrangian and setting it to zero, we have at optimality

$$-\mathbf{KPA} + \mathbf{P}\lambda = 0 \text{ ie.}$$

$$\lambda = \mathbf{P}^\top\mathbf{KPA}. \tag{6.34}$$

To minimize the Lagrangian in equation (6.33), $\mathbf{P}$ is computed iteratively using a multiplicative update rule. Initialize all entries of $\mathbf{P}$ to positive numbers, and the update can be carried out using

$$\mathbf{P}_{ij}^{(t+1)} \longleftarrow \mathbf{P}_{ij}^{(t)}\sqrt{\frac{(\mathbf{KPA})_{ij}}{(\mathbf{PP}^\top\mathbf{KPA})_{ij}}}. \tag{6.35}$$

The correctness and convergence of this iterative update can be shown via an auxiliary function approach [85, 38, 37]. Basically, at each iteration, we upper bound the objective by a convex function; and then we move $\mathbf{P}$ to the minimizer of this convex function. We alternate between the steps of convex upper bounding and finding the minimizer till we hit a local optimum of the original function. More formally, an auxiliary function of $f(\mathbf{P})$ is denoted as $g(\mathbf{P}, \mathbf{P}')$. At iteration t, we expect $g\left(\mathbf{P}, \mathbf{P}^{(t)}\right)$ to satisfy the following properties

$$g\left(\mathbf{P}, \mathbf{P}^{(t)}\right) \geq f(\mathbf{P}) \text{ and} \tag{6.36}$$

$$g\left(\mathbf{P}^{(t)}, \mathbf{P}^{(t)}\right) = f\left(\mathbf{P}^{(t)}\right). \tag{6.37}$$

If we update $\mathbf{P}$ using the following rule

$$\mathbf{P}^{(t+1)} = \underset{\mathbf{P}}{\operatorname{argmin}}\ g\left(\mathbf{P}, \mathbf{P}^{(t)}\right), \tag{6.38}$$

then by construction we have

$$f\left(\mathbf{P}^{(t)}\right) = g\left(\mathbf{P}^{(t)}, \mathbf{P}^{(t)}\right) \geq g\left(\mathbf{P}^{(t+1)}, \mathbf{P}^{(t)}\right) \geq f\left(\mathbf{P}^{(t+1)}\right). \tag{6.39}$$

Thus $f\left(\mathbf{P}^{(t)}\right)$ is monotonically decreasing by construction. Then the key is to find an appropriate auxiliary function $g\left(\mathbf{P}, \mathbf{P}'\right)$.

For NMF, we commonly use the following convex upper bounds and concave lower bounds for $\mathrm{tr}\left(\mathbf{KPAP}^\top\right)$ and $\mathrm{tr}\left(\lambda\mathbf{P}^\top\mathbf{P}\right)$ respectively [37]

$$\sum_{ijkl} \mathbf{K}_{ik}\mathbf{A}_{lj}\mathbf{P}'_{kl}\mathbf{P}'_{ij}\left(1 + \log\left(\frac{\mathbf{P}_{kl}\mathbf{P}_{ij}}{\mathbf{P}'_{kl}\mathbf{P}'_{ij}}\right)\right) \leq \mathrm{tr}\left(\mathbf{KPAP}^\top\right) \leq \sum_{ij} \frac{\left(\mathbf{KP}'\mathbf{A}\right)_{ij}\mathbf{P}^2_{ij}}{\mathbf{P}'_{ij}}, \tag{6.40}$$

$$\sum_{ijkl} \lambda_{lj}\mathbf{P}'_{kl}\mathbf{P}'_{ij}\left(1 + \log\left(\frac{\mathbf{P}_{kl}\mathbf{P}_{ij}}{\mathbf{P}'_{kl}\mathbf{P}'_{ij}}\right)\right) \leq \mathrm{tr}\left(\lambda\mathbf{P}^\top\mathbf{P}\right) \leq \sum_{ij} \frac{\left(\mathbf{P}'\lambda\right)_{ij}\mathbf{P}^2_{ij}}{\mathbf{P}'_{ij}}. \tag{6.41}$$

In our case, we take the lower bound of $\mathrm{tr}\left(\mathbf{KPAP}^\top\right)$ and the upper bound of $\mathrm{tr}\left(\lambda\mathbf{P}^\top\mathbf{P}\right)$ to form our auxiliary function $g(\mathbf{P}, \mathbf{P}')$, ie.

$$g(\mathbf{P}, \mathbf{P}') = \sum_{ijkl} \mathbf{K}_{ik}\mathbf{A}_{lj}\mathbf{P}'_{kl}\mathbf{P}'_{ij}\left(1 + \log\left(\frac{\mathbf{P}_{kl}\mathbf{P}_{ij}}{\mathbf{P}'_{kl}\mathbf{P}'_{ij}}\right)\right) + \sum_{ij} \frac{\left(\mathbf{P}'\lambda\right)_{ij}\mathbf{P}^2_{ij}}{\mathbf{P}'_{ij}}. \tag{6.42}$$

We can obtain the minimizer of $g(\mathbf{P}, \mathbf{P}^{(t)})$ at iteration t by simply setting its derivative to 0, ie.

$$\frac{\partial g(\mathbf{P}, \mathbf{P}^{(t)})}{\partial \mathbf{P}_{ij}} = -2\left(\mathbf{KP}^{(t)}\mathbf{A}\right)_{ij}\frac{\mathbf{P}^{(t)}_{ij}}{\mathbf{P}_{ij}} + 2\left(\mathbf{P}^{(t)}\lambda\right)_{ij}\frac{\mathbf{P}_{ij}}{\mathbf{P}^{(t)}_{ij}} = 0. \tag{6.43}$$

Rearranging the terms and plug in the value of λ as computed from equation (6.34), we get the update rule for $\mathbf{P}$ in equation (6.35).

6.6 Learning the Kernel on the Labels

When we already have a label assignment $\mathbf{P}$ based on k-means, we may want to refine our knowledge about the similarity between the labels. That is we want to learn the kernel on the labels. This can be achieved by the following semi-definite programming [21]

$$\begin{aligned} &\underset{\mathbf{A}}{\text{maximize}}\ \mathrm{tr}\left(\mathbf{P}^\top\mathbf{HKHPA}\right) \\ &\text{subject to}\ \|\mathbf{A}\|_F^2 = 1,\ \mathbf{A} \succeq 0 \end{aligned} \tag{6.44}$$

Given that the objective is linear in $\mathbf{A}$, we can turn the constraint $\|\mathbf{A}\|_F^2 = 1$ into inequality, ie. $\langle\mathbf{A}, \mathbf{A}\rangle_F \leq 1$ because of the maximization. Using the Schur complement lemma,[2] this

[2] Let $\mathbf{A} \succ 0$ and consider

$$\mathbf{X} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix}.$$

Then $\mathbf{X} \succeq 0 \Leftrightarrow \mathbf{S} = \mathbf{C} - \mathbf{B}^\top\mathbf{A}^{-1}\mathbf{B} \succeq 0$. $\mathbf{S}$ is called Schur complement of $\mathbf{X}$.

constraint can be expressed as

$$\begin{pmatrix} \mathbf{I} & \operatorname{vec}(\mathbf{A}) \\ \operatorname{vec}(\mathbf{A})^\top & 1 \end{pmatrix} \succeq 0. \tag{6.45}$$

Stacking the constraints into linear matrix inequality, we have the following standard semidefinite programming problem

$$\begin{aligned} &\underset{\mathbf{A}}{\text{maximize}} \operatorname{vec}\left(\mathbf{P}^\top \mathbf{K} \mathbf{P}\right)^\top \operatorname{vec}(\mathbf{A}) \\ &\text{subject to} \begin{pmatrix} \mathbf{I} & \operatorname{vec}(\mathbf{A}) & \mathbf{0} \\ \operatorname{vec}(\mathbf{A})^\top & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A} \end{pmatrix} \succeq 0. \end{aligned} \tag{6.46}$$

Solving for $\mathbf{A}$ and we will learn a kernel between the labels.

6.7 Stability under Perturbation

Given the wide variety of data kernels we have proposed for HSIC-based clustering, we require a means of choosing the kernel that best reveals meaningful data structure. In this section we provide a perturbation bound for the clustering, which may be used in choosing the kernel: kernels that provide the most reasonable clustering are also the most stable under perturbation.

Let Δ be the perturbation such that the perturbed (centered) kernel matrix $\tilde{\mathbf{K}}$ is related to the original (centered) kernel matrix $\mathbf{K}$ by $\mathbf{K} = \tilde{\mathbf{K}} + \Delta$. Two clusterings obtained respectively before and after the perturbation are denoted $\mathbf{P}$ and $\tilde{\mathbf{P}}$ (both are normalized partition matrices, and can be treated as the labels). Let c be the number of clusters, and $\mathbf{L}$ and $\mathcal{D}$ be the label kernel matrix and the dependence estimated by $\mathbf{P}$, respectively. Denote $\mathcal{D}^\circ$ as the dependence estimated by the first $c-1$ principal eigenvectors $\mathbf{U}$ of $\mathbf{K}$. Similarly, define $\tilde{\mathbf{L}}$, $\tilde{\mathcal{D}}$, $\tilde{\mathcal{D}}^\circ$, and $\tilde{\mathbf{U}}$ for $\tilde{\mathbf{K}}$.

First, we note that the dependence $\operatorname{tr}(\mathbf{HLH}\tilde{\mathbf{L}})$ constitutes a measure of similarity between two clusterings. In fact, a variant called χ^2 functional has already been employed for comparing clusterings [93]. A distance, ϵ, between the clusterings can be obtained as

$$\begin{aligned} \epsilon &= \operatorname{tr}(\mathbf{HLHL}) + \operatorname{tr}(\mathbf{H}\tilde{\mathbf{L}}\mathbf{H}\tilde{\mathbf{L}}) - 2\operatorname{tr}(\mathbf{HLH}\tilde{\mathbf{L}}) \\ &= \|\mathbf{HLH} - \mathbf{H}\tilde{\mathbf{L}}\mathbf{H}\|_{\text{Frob}}^2 = \|\mathbf{L} - \tilde{\mathbf{L}}\|_{\text{Frob}}^2. \end{aligned} \tag{6.47}$$

where $\|\cdot\|_{\text{Frob}}$ denotes matrix Frobenius norm, and the centering matrix $\mathbf{H}$ does not affect the distance. We will relate ϵ to two factors: (*i*) the estimated dependence between the inputs and the labels; (*ii*) the size of the perturbation.

Let λ_{c-1} and λ_c be the $c-1$ and c-th principal eigenvalues of $\mathbf{K}$ (and define $\tilde{\lambda}_{c-1}$ and $\tilde{\lambda}_c$ accordingly for $\tilde{\mathbf{K}}$). Further define $\eta = \operatorname{tr}(\Delta\tilde{\mathbf{U}}\tilde{\mathbf{U}}^\top)$, which measures the size of the perturbation. The following theorem quantifies the relationship between ϵ and its determining factors.

Theorem 46 *The distance, ϵ, between the two clusterings, $\mathbf{P}$ and $\tilde{\mathbf{P}}$, is bounded by:*

$$\epsilon \leq 2\left(\sqrt{\delta} + \sqrt{\tilde{\delta}} + \sqrt{\gamma}\right)^2, \tag{6.48}$$

where δ, $\tilde{\delta}$ and γ are defined respectively as

$$\frac{\mathcal{D}^\circ - \mathcal{D}}{\lambda_{c-1} - \lambda_c}, \frac{\tilde{\mathcal{D}}^\circ - \tilde{\mathcal{D}}}{\tilde{\lambda}_{c-1} - \tilde{\lambda}_c}, \frac{\mathcal{D}^\circ - \tilde{\mathcal{D}}^\circ - \eta}{\lambda_{c-1} - \lambda_c}. \tag{6.49}$$

Proof [**Theorem 46**] Let $\mathbf{U}$ and $\tilde{\mathbf{U}}$ be the first $c-1$ principal eigenvectors of $\mathbf{K}$ and $\tilde{\mathbf{K}}$ respectively, When using them as labels, they form kernel matrices $\mathbf{L}^\circ$ and $\tilde{\mathbf{L}}^\circ$ respectively. By triangle inequality, we have

$$\begin{aligned} \|\mathbf{L} - \tilde{\mathbf{L}}\|_{\text{Frob}}^2 &= \|\mathbf{L} - \mathbf{L}^\circ + \mathbf{L}^\circ - \tilde{\mathbf{L}}^\circ + \tilde{\mathbf{L}}^\circ - \tilde{\mathbf{L}}\|_{\text{Frob}}^2 \\ &\leq \left(\|\mathbf{L} - \mathbf{L}^\circ\|_{\text{Frob}} + \|\mathbf{L}^\circ - \tilde{\mathbf{L}}^\circ\|_{\text{Frob}} + \|\tilde{\mathbf{L}}^\circ - \tilde{\mathbf{L}}\|_{\text{Frob}}\right)^2. \end{aligned} \tag{6.50}$$

We then bound the three terms respectively.

First, we note $\mathbf{P} = \mathbf{U}\mathbf{R} + \mathbf{U}_\perp\mathbf{E}$ where $(\mathbf{U}, \mathbf{U}_\perp) \in \mathbb{R}^{m\times m}$ forms a set of orthonormal bases, and $\mathbf{R} \in \mathbb{R}^{(c-1)\times(c-1)}$, $\mathbf{E} \in \mathbb{R}^{(m-c+1)\times(c-1)}$ are matrices of coefficients. Then we have

$$\begin{aligned} \|\mathbf{L} - \mathbf{L}^\circ\|_{\text{Frob}}^2 &= \|\mathbf{L}\|_{\text{Frob}}^2 + \|\mathbf{L}^\circ\|_{\text{Frob}}^2 - 2\langle \mathbf{L}, \mathbf{L}^\circ\rangle_{\text{Frob}} \\ &= 2(c-1) - 2\|\mathbf{R}\|_{\text{Frob}}^2 = 2\|\mathbf{E}\|_{\text{Frob}}^2. \end{aligned} \tag{6.51}$$

In the second step we use $\|\mathbf{L}\|_{\text{Frob}}^2 = \|\mathbf{L}^\circ\|_{\text{Frob}}^2 = c-1$, and in the third step we exploit $\|\mathbf{R}\|_{\text{Frob}}^2 + \|\mathbf{E}\|_{\text{Frob}}^2 = c-1$. Taking the square root, we have $\|\mathbf{L} - \mathbf{L}^\circ\|_{\text{Frob}} = \sqrt{2}\|\mathbf{E}\|_{\text{Frob}}$.

Similarly, let $\tilde{\mathbf{U}} = \mathbf{U}\mathbf{R}^\circ + \mathbf{U}_\perp\mathbf{E}^\circ$ and we have $\|\mathbf{L}^\circ - \tilde{\mathbf{L}}^\circ\|_{\text{Frob}} = \sqrt{2}\|\mathbf{E}^\circ\|_{\text{Frob}}$. Let $\tilde{\mathbf{P}} = \tilde{\mathbf{U}}\tilde{\mathbf{R}} + \tilde{\mathbf{U}}_\perp\tilde{\mathbf{E}}$, we then have $\|\tilde{\mathbf{L}} - \tilde{\mathbf{L}}^\circ\|_{\text{Frob}} = \sqrt{2}\|\tilde{\mathbf{E}}\|_{\text{Frob}}$.

Now we relate $\|\mathbf{E}\|_{\text{Frob}}$, $\|\tilde{\mathbf{E}}\|_{\text{Frob}}$ and $\|\mathbf{E}^\circ\|_{\text{Frob}}$ to δ, $\tilde{\delta}$ and γ. First, $\|\mathbf{E}\|_{\text{Frob}} \leq \sqrt{\delta}$ and $\|\tilde{\mathbf{E}}\|_{\text{Frob}} \leq \sqrt{\tilde{\delta}}$ [94]. Second, let $\mathbf{K} = \tilde{\mathbf{K}} + \Delta$, we have

$$\begin{aligned} \mathcal{D}^\circ - \tilde{\mathcal{D}}^\circ &= \text{tr}(\mathbf{K}\mathbf{U}\mathbf{U}^\top) - \text{tr}(\tilde{\mathbf{K}}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^\top) \\ &= \text{tr}(\mathbf{K}\mathbf{U}\mathbf{U}^\top) - \text{tr}(\mathbf{K}\tilde{\mathbf{U}}\tilde{\mathbf{U}}^\top) \\ &\quad + \text{tr}(\Delta\tilde{\mathbf{U}}\tilde{\mathbf{U}}^\top) \\ &\geq (\lambda_{c-1} - \lambda_c)\|\mathbf{E}^\circ\|_{\text{Frob}}^2 + \eta. \end{aligned} \tag{6.52}$$

which leads to $\|\mathbf{E}^\circ\|_{\text{Frob}} \leq \sqrt{\gamma}$. Combining the intermediate results, we have the theorem. ■

We see that ϵ is mainly influenced by the ratio of the dependence gap (e.g. $\mathcal{D}^\circ - \mathcal{D}$) to the eigenvalue gap (e.g. $\lambda_{c-1} - \lambda_c$). If changes in dependence due to perturbation and discretization are small compared to eigenvalue gaps, it is likely that the clustering will remain unchanged. Our error bound is distribution free, and does not require the perturbation to be small. This is in contrast to the results by [101, 78, 94]. Hence, our result may have wider applicability.

One case of particular interest is where the perturbation is caused by our performing a low rank approximation of the kernel matrix, $\mathbf{K} \approx \mathbf{B}\mathbf{B}^\top = \tilde{\mathbf{K}}$ (e.g. incomplete Cholesky factorization [46]). In this case, the eigenvalues of $\mathbf{K}$ always dominate the corresponding ones in $\tilde{\mathbf{K}}$. Denoting as $\xi = \text{tr}(\Delta)$ the approximation error, we see that $\xi \geq \mathcal{D}^\circ - \tilde{\mathcal{D}}^\circ - \eta$. Instead of using a fixed error tolerance for the decomposition, this inequality suggests a data-dependent way of setting it: with a tolerance of $\lambda_{c-1} - \lambda_c$, almost no clustering error will be incurred due to the low rank approximation. We will also demonstrate this in our experiments.

6.8 Experiments

Our experiments address four main questions: (*i*) how to choose a kernel for a particular clustering problem; (*ii*) how to use the perturbation bound for kernel matrix approximation; (*iii*) how to cluster inputs with rich structured labels and (*iv*) how to learn the kernel for the labels.

6.8.1 Kernel Choice and Stability

In our first series of experiments, we evaluate kernels with respect to their ability to solve certain clustering problems, and their stability after data perturbation.

Choice of Kernels We investigated three artificial datasets (Collinear, Ring, and XOR; see Figure 6.1), to show how the choice of kernel affects spectral initialization and the correct clustering.

For the Collinear dataset, both the linear and RBF kernels are capable of capturing information about the spherical shape of the clusters. $\mathbf{K}$ computed with a linear kernel has only one piecewise constant eigenvector due to the collinearity of the cluster centers. By contrast, a Gaussian kernel produces two such informative eigenvectors and allows a spectral initialization of k-means. This is because $\mathbf{K}$ produced by the Gaussian kernel has full rank [115] and hence avoids the collinear degeneracy.

For the Ring dataset, a Gaussian kernel is not able to produce the correct clustering. This is reflected in its eigenvectors. Although some authors [eg 55], obtain correct clustering using an RBF kernel, this is actually a local minimum. The eigenvectors of a graph kernel provide a very good indication of the clusters, and lead to correct clustering.

For the XOR dataset, although a graph kernel provides piecewise constant eigenvectors, the piecewise structure is not consistent with the cluster structure. Hence the resulting clustering is incorrect. The polynomial kernel of degree 2 gives consistent information for spectral initialization and correct clustering.

These examples provide us with two rules for choosing kernels. First, a kernel should introduce the correct notion of similarity. Second, it should be powerful enough (e.g. a Gaussian kernel vs. a linear one) to avoid the degenerate case for spectral clustering.

Stability of Kernels In this section, we perturb the datasets from the previous section, and plot the scaling of γ computed from different kernels as a function of the amount of perturbation (Figure 6.2). The scaling gives a very good indication of the fitness of a kernel for a particular dataset. This also provides a method for choosing an appropriate kernel.

For the Collinear dataset, both the Gaussian kernel and the graph kernel provide similar stability. This is consistent with the fact that both can detect spherical clusters. For the Ring dataset, however, the graph kernel is clearly better than the other two. This is because the notion of similarity in this dataset is well reflected by graph connectivity. For the XOR dataset, the polynomial kernel becomes the most stable one, since it is the only one that defines a correct similarity measure on the data.

6.8.2 Kernel Matrix Approximation

In our second set of experiments, we focus on speeding up clustering by kernel matrix approximation. Incomplete Cholesky decomposition is used to approximate the kernel matrix, i.e.

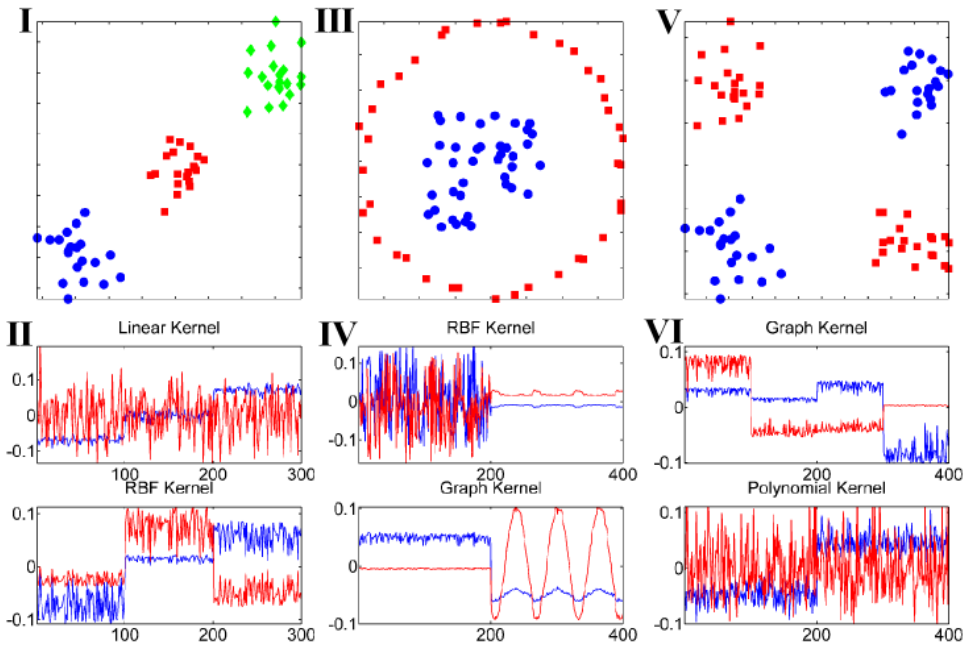


Figure 6.1: Three artificial datasets and the first 2 principal eigenvectors computed from various kernels matrices. **Left column, top to bottom:** I) Collinear dataset, III) Ring dataset, and V) XOR dataset (data from the same cluster scatter diagonally around the origin). Data points with identical colors and shapes belong to the same class. **Right column:** eigenvectors computed for the corresponding datasets on the left. The first principal eigenvector is colored blue and the second in red. For Colinear, results are for linear and RBF kernels (II). For Ring, results are for RBF and graph kernels (IV). For XOR, results are for graph and polynomial kernels (VI).

Table 6.1: Clustering error and speed before and after performing incomplete Cholesky decomposition. err1, t1: clustering error and time using the full kernel matrix; err2, t2: clustering error and time using the incomplete Cholesky factor. col#: number of columns in the incomplete Cholesky factor. (m, d): sample size and dimension. c: number of clusters.

	Breastcancer	Iris	Wine	Soybean	Vehicle	Glass	Segment	USPS	Vowel
(m,d)	(669,10)	(150,4)	(178,13)	(47,21)	(846,18)	(214,9)	(2310,18)	(2007,256)	(990,10)
c	2	3	3	4	4	6	7	10	11
err1	3.7	16.0	4.5	0.0	65.4	51.4	36.0	47.0	68.9
err2	3.7	18.0	5.1	0.0	65.4	51.4	36.0	46.9	68.9
t1	18.9	0.2	0.4	0.0	38.0	0.6	647.0	537.9	57.1
t2	0.0	0.0	0.0	0.0	0.3	0.0	11.4	157.3	2.1
col#	35	12	61	32	137	93	228	1518	309

$\mathbf{K} \approx \mathbf{B}\mathbf{B}^\top$ ($\mathbf{B} \in \mathbb{R}^{m \times d}$ where $d \ll m$). Here we use the perturbation bound to set the approximation tolerance of the incomplete Cholesky decomposition. More precisely, we further decompose the kernel matrix if the approximation error ξ is larger than the eigengap $\lambda_{c-1} - \lambda_c$.

We carried out experiments in 9 datasets taken from the UCI and Statlib repositories, and report results in Table 6.1. The number of classes in the datasets varies from 2 to 11. We are interested in the number of data points clustered differently from the true labels, err1 and err2, computed respectively before and after the incomplete Cholesky decomposition. We also report the time (in seconds), t1, taken to compute the eigenvectors of the full kernel matrix, and

the time, t2, to compute the singular vectors of the incomplete Cholesky approximation. The time is recorded for matlab using the routines eig and svd, respectively.

We find that by guiding the incomplete Cholesky decomposition using the eigengap and performing singular value decomposition to obtain cluster initialization, almost no *clustering errors* result [or misclassification errors in 94]. At the same time, the computation time is greatly reduced for several datasets, up to an order of magnitude.

6.8.3 Clustering with Rich Label Kernels

In third set of experiments, we demonstrate clustering with rich label kernels, for both hierarchical and ring structures. The experiments were all conducted on image data, with pixel vectors normalized in each dimension to zero mean and unit variance. We used a Gaussian kernel $\exp(-\sigma\|\mathbf{x}-\mathbf{x}'\|^2)$ with $\sigma = d^{-1}$ (d:dimension of the data) for the kernel matrix $\mathbf{K}$. Note that we only show that results produced by our iterative clustering algorithms. We also tried the NMF approach in both datasets. However, it only produced correct result for the face dataset and could not cluster the teapot dataset correctly.

Hierarchical Facial Expression Clustering Computer recognition of facial expressions is an important component in automated understanding of human communication [104]. We describe a hierarchical clustering of face images that takes into account both the identity of the individual and the emotion being expressed. We collected 185 images (size 217×308) of 3 types of facial expressions (NE: neutral, HA: happy, SO: shock) from 3 subjects (CH, LE, AR), in alternating order, with around 20 repetitions each. We registered the images by aligning the eyes, and adjusting the average pixel intensities to be the same. We plotted the data points in Figure 6.3(a), using the entries in the top 3 eigenvectors of $\mathbf{K}$ as the coordinates. Typical images from each clusters are plotted beside the clusters. Besides clustering the images into individuals and different expressions (9 clusters), we also require that images belong to the same person should also be grouped together (3 groups). This hierarchy is shown in Figure 6.3(b), as successfully obtained using $\mathbf{P}$ and $\mathbf{A}$ in (6.12). This can be checked by examining the columns of the resulting partition matrix: CLUHSIC clusters each person into adjacent columns according to $\mathbf{A}$. Although plain k-means is also able to cluster the images correctly into 9 cluster, the hierarchy is lost: k-means place the cluster assignments for each person into arbitrary columns.

Clustering Teapots in a Ring We used the 400 consecutive teapot images (of size 76×101 with RGB color) from [149]. The images were taken successively as the teapot was rotated 360°. We plot the data points as dots in Figure 6.4, using their corresponding entries in the top 3 eigenvectors of $\mathbf{K}$ as the coordinates. To make the clustering problem more difficult, we perturb the images at the points where they are close (the perturbed images have indices in two ranges: from 90 to 105, and from 290 to 305), so as to bring these clusters closer together (Table 6.2). We wish to divide the data into 10 clusters, such that images in the same cluster come from similar viewing angles, and images in adjacent clusters come from consecutive angles. We present the different clusters produced by plain k-means and CLUHSIC in Figure 6.4, where images in the same cluster are dots of identical shape and color. k-means incorrectly groups images of opposite angle into the same cluster (yellow square dots). CLUHSIC avoids this error by coding a ring structure into $\mathbf{A}$ as in (6.14). In addition, the labels generated by CLUHSIC

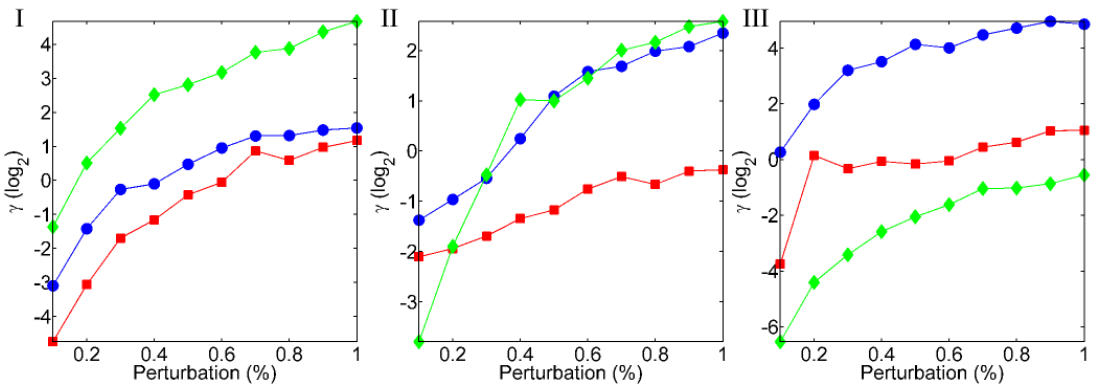


Figure 6.2: γ as a function of the amount of perturbation applied to the artificial datasets in Figure 6.1 using three kernels: RBF kernel (blue circle), graph kernel (red square) and polynomial kernel (green diamond). Results are for Colinear (I), Ring (II), and XOR (III) datasets.

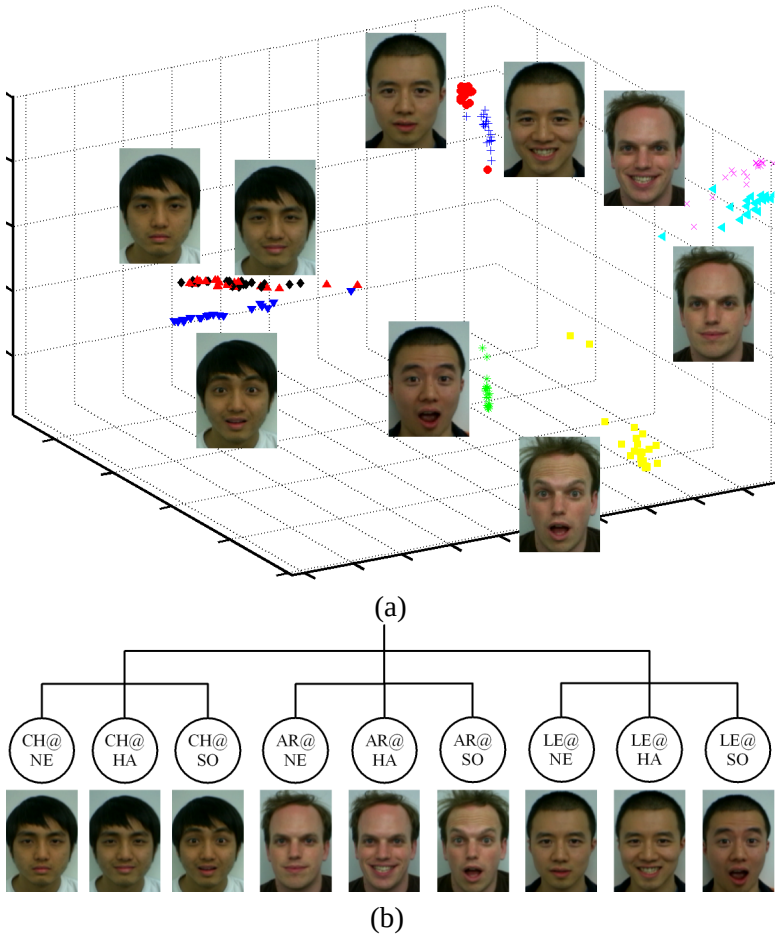
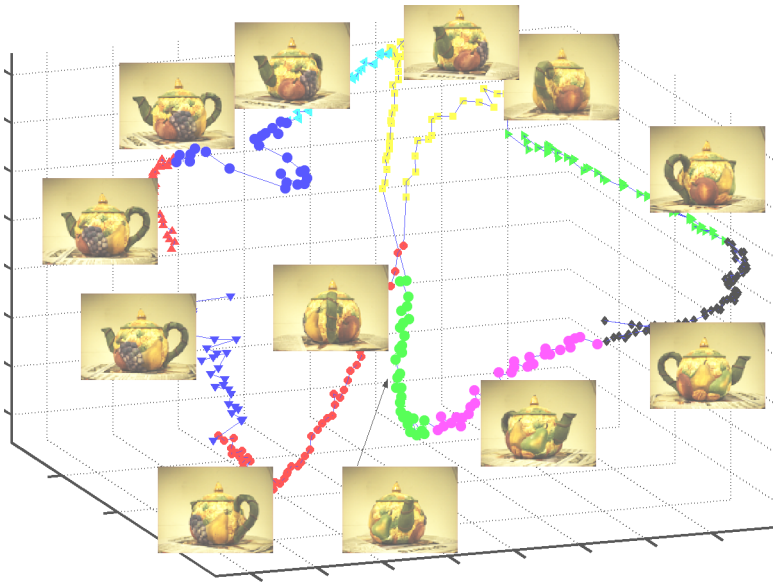
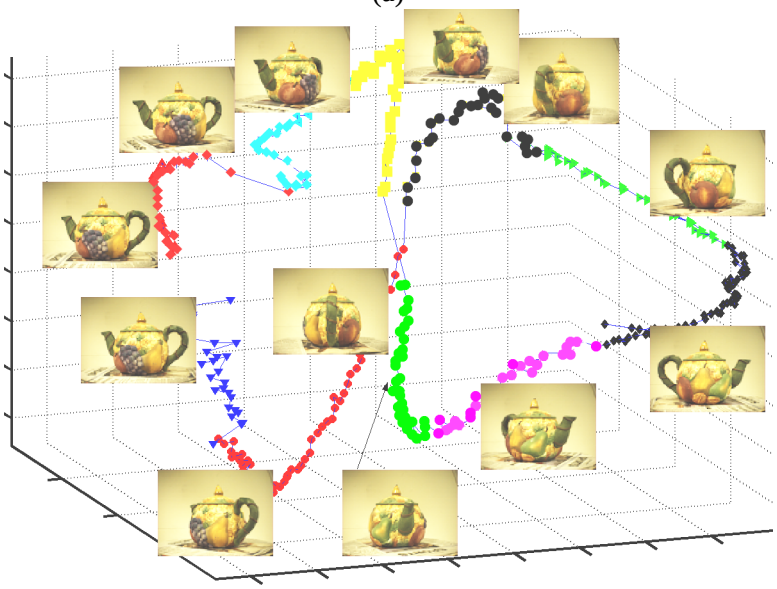


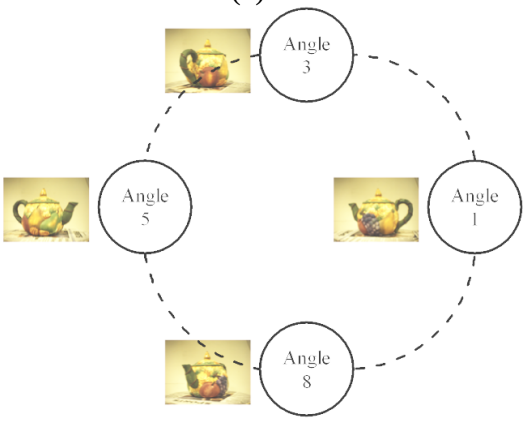
Figure 6.3: (a) Facial expression images embedded into 3 dimensional space; different marker shape/colour combinations represent the true identity/expression clusters. (b) The two-level hierarchy recovered from the data.



(a)



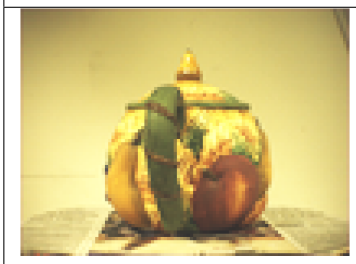
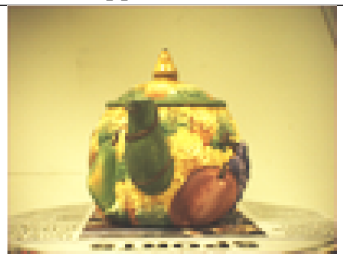
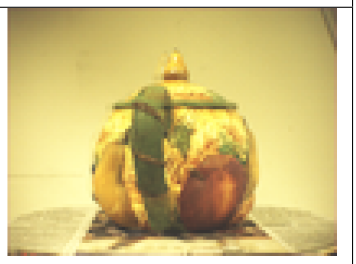
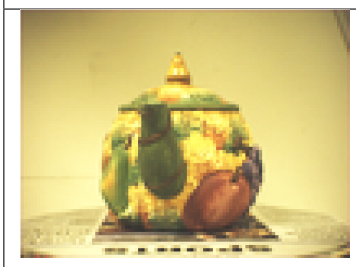
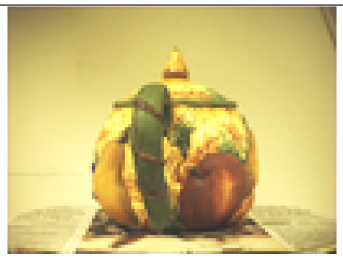
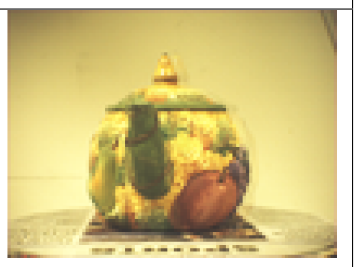
(b)



(c)

Figure 6.4: Teapot images, 360° rotation and clustered (a) by k-means and (b) by CLUHSIC with the label kernel. (c) The schematic diagram showing that the teapot images live in a manifold with a ring structure.

Table 6.2: Perturbing the images from opposite views such that they confuses k-means clustering. The perturbation is carried out pixel by pixel. The perturbed view is computed from the pixel value of the main view, a_m, and that of the opposite view, a_o, by $0.75 \times a_m + 0.25 \times a_o$. Notice the blurred edge in the perturbed view.

Main View	Opposite View	Perturbed View
		
		

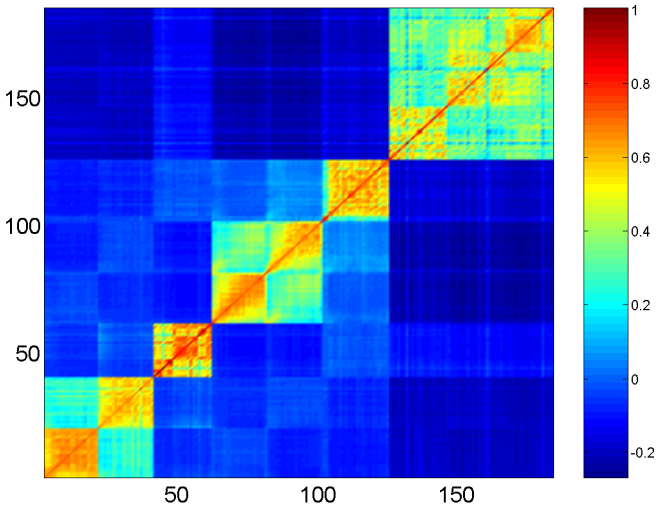
preserve the ordering of the images (adjacent clusters represent consecutive angles, and the clusters form a ring), while k-means does not.

6.8.4 Learning the Kernel on the Labels

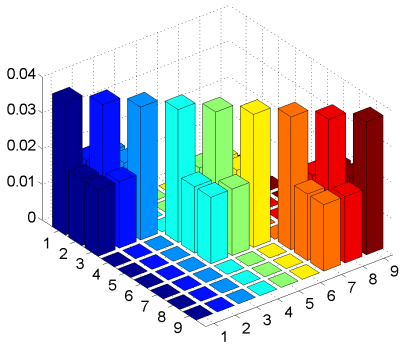
In this section we illustrate how to refine the kernel on the labels when we already have a cluster assignment. In this experiment, we use the facial image dataset as an example. The kernel matrix for the inputs is plotted using color map in Figure 6.5(a). Note that we have arranged the inputs such that images are grouped first according to subjects and then according to different expressions. This allows us to see the clear block structure in the kernel matrix. Then we use the semidefinite programming in (6.46) to learn the kernel matrix on the labels. The learned result is shown in Figure 6.5(c). Comparing it to the kernel matrix $\mathbf{A}$ we manually designed in equation (6.12) (plotted in Figure 6.5(b)), we see that the learned matrix bears a close resemblance to the manual design. Furthermore, they also closely reflect the structure we see from the kernel matrix for the inputs.

6.9 Summary

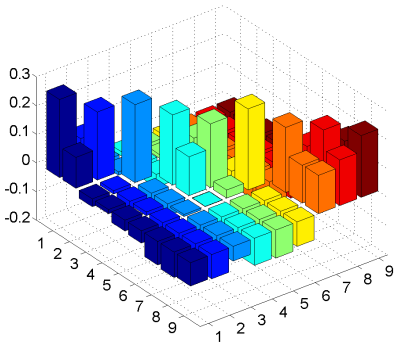
In this chapter, we view clustering as a process of inferring the labels from the data, and employ the Hilbert-Schmidt Independence Criterion (HSIC) as the underlying inference rule. Clustering via HSIC allows us to unify various views of clustering, and to generalize to clustering in hierarchies, chains, rings, and other complex structures. Furthermore, we provide a perturbation bound on the stability of the clustering, which has practical implications for kernel selection. We expect that our framework will provide useful guidance to the practice of clustering.



(a)



(b)



(c)

Figure 6.5: Learning the kernel for the labels. (a) The kernel matrix for the data. (b) Manually designed kernel matrix for the labels. (c) Learned kernel matrix for the labels.

CHAPTER 7

Dimensionality Reduction via Dependence Estimation

In this chapter, we will present an approach for dimensionality reduction that takes side information into account. It uses the HSIC as the objective for optimization, and tries to preserve the distance between neighbors in the reduced representation. This new approach has the popular maximum variance unfolding (MVU) as a special case, and thereby also provides MVU with statistical justifications. We will also show the good performance of our method on real world datasets.

7.1 Introduction

In recent years maximum variance unfolding (MVU), introduced by Weinberger et al. [150], has gained popularity as a method for dimensionality reduction. This method is based on a simple heuristic: maximizing the overall variance of the embedding while preserving the local distances between neighboring observations. Sun et al. [133] show that there is a dual connection between MVU and the goal of finding a fast mixing Markov chain. This connection is intriguing. However, it offers limited insight as to why MVU can be used for data representation.

This chapter provides a statistical interpretation of MVU. We show that the algorithm attempts to extract features from the data which simultaneously preserve the identity of individual observations *and* their local distance structure. Our reasoning relies on a dependence measure between sets of observations, the Hilbert-Schmidt Independence Criterion (HSIC) [59].

Relaxing the requirement of retaining maximal information about individual observations, we are able to obtain "colored" MVU. Unlike traditional MVU which takes only *one* source of information into account, "colored" MVU allows us to integrate *two* sources of information. That is, we are able to find an embedding that leverages between two goals:

- preserve local distances according to the *first* source of information (the data);
- and maximally align with the *second* sources of information (side information).

Note that not all features inherent in the data are interesting for an ulterior objective. For instance, if we want to retain a reduced representation of the data for later classification, then only those discriminative features will be relevant. "Colored" MVU achieves the goal of elucidating primarily relevant features by aligning the embedding to the objective provided in the side information. Some examples illustrate this situation in more details:

- Given a-bag-of-pixels representation of images (the data), such as USPS digits, find an embedding which reflects the categories of the images (side information).

- Given a vector space representation of texts on the web (the data), such as newsgroups, find an embedding which reflects a hierarchy of the topics (side information).
- Given a TF/IDF representation of documents (the data), such as NIPS papers, find an embedding which reflects co-authorship between the documents (side information).

There is a strong motivation for *not* simply merging the two sources of information into a single distance metric: Firstly, the data and the side information may be heterogenous. It is unclear how to combine them into a single distance metric; Secondly, the side information may appear in the form of similarity rather than distance. For instance, co-authorship relations is a similarity between documents (if two papers share more authors, they tends to be more similar), but it does not induce a distance between the documents (if two papers share no authors, we cannot assert they are far apart). Thirdly, at test time (i.e. when inserting a new observation into an existing embedding) only one source of information might be available, i.e. the side information is missing.

7.2 Maximum Variance Unfolding

We begin by giving a brief overview of MVU and its projection variants, as proposed in [150]. Given a set of m observations $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\} \subseteq \mathcal{Z}$ and a distance metric $d : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$ find an inner product matrix (kernel matrix) $\mathbf{K} \in \mathbb{R}^{m \times m}$ with $\mathbf{K} \succeq 0$ such that

1. The distances are preserved, i.e. $\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = d_{ij}^2$ for all (i, j) pairs which are sufficiently close to each other, such as the n nearest neighbors of each observation. We denote this set by $\mathcal{N}$. We will also use $\mathcal{N}$ to denote the graph formed by having these (i, j) pairs as edges.
2. The embedded data is centered, i.e. $\mathbf{K}\mathbf{1} = \mathbf{0}$ (where $\mathbf{1} = (1, \ldots, 1)^\top$ and $\mathbf{0} = (0, \ldots, 0)^\top$).
3. The trace of $\mathbf{K}$ is maximized (the maximum variance unfolding part).

Several variants of this algorithm, including a large scale variant [148] have been proposed. By and large the optimization problem looks as follows

$$\underset{\mathbf{K} \succeq 0}{\text{maximize}} \ \operatorname{tr}(\mathbf{K}) \tag{7.1}$$

$$\text{subject to } \mathbf{K}\mathbf{1} = \mathbf{0} \text{ and } \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = d_{ij}^2 \text{ for all } (i, j) \in \mathcal{N}. \tag{7.2}$$

Numerous variants of (7.1) exist, e.g. where the distances are only allow to shrink, where slack variables are added to the objective function to allow approximate distance preserving, or where one uses low-rank expansions of $\mathbf{K}$ to cope with the computational complexity of semidefinite programming.

A major drawback with MVU is that its results necessarily come as somewhat of a surprise. That is, it is never clear before invoking MVU what specific interesting results it might produce. While in hindsight it is easy to find an insightful interpretation of the outcome, it is not a-priori clear which aspect of the data the representation might emphasize. A second drawback is that while in general generating brilliant results, its statistical origins are somewhat more obscure. We aim to address these problems by means of the Hilbert-Schmidt Independence Criterion.

7.3 Colored Maximum Variance Unfolding (MUHSIC)

HSIC has been used to measure independence between random variables [59], to select features (Chapter 5) or to cluster data (Chapter 6). Here we use it in a different way:

> We try to construct a kernel matrix $\mathbf{K}$ for the dimension-reduced data X which preserves the local distance structure of the original data Z, such that X is maximally dependent on the side information Y as seen from its kernel matrix $\mathbf{L}$.

HSIC has several advantages in this context. First, it satisfies concentration of measure conditions [59]: for random draws of observation from $\Pr_{\mathbf{xy}}$, HSIC provides values which are very similar. This is desirable, as we want our metric embedding to be robust to small changes. Second, HSIC is easy to compute, since only the kernel matrices are required and no density estimation is needed. The freedom of choosing a kernel for $\mathbf{L}$ allows us to incorporate prior knowledge into the dependence estimation process. The consequence is that we are able to incorporate various side information by simply choosing an appropriate kernels for Y.

We state the algorithmic modification first and subsequently we explain why this is reasonable: the key idea is to replace $\operatorname{tr}(\mathbf{K})$ in (7.1) by $\operatorname{tr}(\mathbf{KL})$, where $\mathbf{L}$ is the covariance matrix of the domain (side information) with respect to which we would like to extract features. For instance, in the case of NIPS papers which happen to have author information, $\mathbf{L}$ would be the kernel matrix arising from coauthorship and $d(\mathbf{z}, \mathbf{z}')$ would be the Euclidean distance between the vector space representations of the documents. Key to our reasoning is the following lemma:

Lemma 47 *Denote by $\mathbf{L}$ a positive semidefinite matrix in $\mathbb{R}^{m\times m}$ and let $\mathbf{H} \in \mathbb{R}^{m\times m}$ be defined as $\mathbf{H}_{ij} = \delta_{ij} - m^{-1}$. Then the following two optimization problems are equivalent*

$$\text{1.} \quad \underset{\mathbf{K}}{\text{maximize}} \ \operatorname{tr}(\mathbf{HKHL}) \tag{7.3a}$$
$$\text{subject to } \mathbf{K} \succeq 0 \text{ and constraints on } \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}.$$
$$\text{2.} \quad \underset{\mathbf{K}}{\text{maximize}} \ \operatorname{tr}(\mathbf{KL}) \tag{7.3b}$$
$$\text{subject to } \mathbf{K} \succeq 0 \text{ and constraints on } \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} \text{ and } \mathbf{K1} = \mathbf{0}.$$

Any solution of (7.3b) solves (7.3a) and any solution of (7.3a) solves (7.3b) after centering $\mathbf{HKH}$.

Proof Denote by $\mathbf{K}_a$ and $\mathbf{K}_b$ the solutions of (7.3a) and (7.3b) respectively. $\mathbf{K}_b$ is feasible for (7.3a) and $\operatorname{tr}(\mathbf{K}_b\mathbf{L}) = \operatorname{tr}(\mathbf{HK}_b\mathbf{HL})$. This implies that

$$\operatorname{tr}(\mathbf{HK}_a\mathbf{HL}) \geq \operatorname{tr}(\mathbf{HK}_b\mathbf{HL}). \tag{7.4}$$

Vice versa $\mathbf{HK}_a\mathbf{H}$ is feasible for (7.3b). Moreover

$$\operatorname{tr}(\mathbf{HK}_a\mathbf{HL}) \leq \operatorname{tr}(\mathbf{K}_b\mathbf{L}) \tag{7.5}$$

by requirement on the optimality of $\mathbf{K}_b$. Combining both inequalities shows that $\operatorname{tr}(\mathbf{HK}_a\mathbf{HL}) = \operatorname{tr}(\mathbf{K}_b\mathbf{L})$, hence both solutions are equivalent. ■

This means that the centering imposed in MVU via constraints is equivalent to the centering in HSIC by means of the dependence measure $\operatorname{tr}(\mathbf{HKHL})$ itself. In other words, MVU equivalently maximizes $\operatorname{tr}(\mathbf{HKHI})$, i.e. the dependence between $\mathbf{K}$ and the identity matrix $\mathbf{I}$ which

corresponds to retain maximal diversity between observations via $\mathbf{L}_{ij} = \delta_{ij}$. This suggests the following colored version of MVU

$$\begin{aligned} &\underset{\mathbf{K}}{\text{maximize}} \ \operatorname{tr}(\mathbf{HKHL}) \\ &\text{subject to } \mathbf{K} \succeq 0 \text{ and } \mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} = d_{ij}^2 \text{ for all } (i,j) \in \mathcal{N}. \end{aligned} \tag{7.6}$$

Using (7.6) we see that we are now extracting a Euclidean embedding which maximally depends on the coloring matrix $\mathbf{L}$ (of the side information) while preserving local distance structure. A second advantage of (7.6) is that whenever we restrict $\mathbf{K}$ further, e.g. by only allowing for $\mathbf{K}$ be part of a linear subspace formed by the principal vectors in some space, (7.6) remains feasible, whereas the (constrained) MVU formulation may become infeasible (i.e. $\mathbf{K1} = 0$ may not be satisfied). In the following sections, we will also refer to colored maximum variance unfolding as MUHSIC, meaning maximum unfolding via HSIC.

7.4 MUHSIC in the Dual

In this section, we derive the dual problem for MUHSIC in (7.6) to show further insight on why it is able to reduce the dimension of the data.

7.4.1 Dual Problem

Our approach makes use of the results from [133]. First we define auxiliary matrices $\mathbf{E}^{ij} \in \mathbb{R}^{m \times m}$ for each edge $(i,j) \in \mathcal{N}$, such that it has only four nonzero entries $\mathbf{E}_{ii}^{ij} = \mathbf{E}_{jj}^{ij} = 1$ and $\mathbf{E}_{ij}^{ij} = \mathbf{E}_{ji}^{ij} = -1$. Then the distance preserving constraint can be written as $\operatorname{tr}\left(\mathbf{KE}^{ij}\right) = d_{ij}^2$. Thus we have the following Lagrangian

$$\begin{aligned} \mathcal{L}(\mathbf{K}, \mathbf{Z}, w) &= \operatorname{tr}(\mathbf{KHLH}) + \operatorname{tr}(\mathbf{KZ}) - \sum_{(i,j)\in\mathcal{N}} w_{ij}\left(\operatorname{tr}\left(\mathbf{KE}^{ij}\right) - d_{ij}^2\right) \\ &= \operatorname{tr}\left(\mathbf{K}\left(\mathbf{HLH} + \mathbf{Z} - \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij}\right)\right) + \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2, \end{aligned} \tag{7.7}$$

where $\mathbf{Z} \succeq 0$. The dual objective is

$$\begin{aligned} g(\mathbf{Z}, w) &= \sup_{\mathbf{K}} \ \mathcal{L}(\mathbf{K}, \mathbf{Z}, w) \\ &= \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2 + \sup_{\mathbf{K}} \ \operatorname{tr}\left(\mathbf{K}\left(\mathbf{HLH} + \mathbf{Z} - \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij}\right)\right), \end{aligned} \tag{7.8}$$

which is easily determined analytically, since a linear function on $\mathbf{K}$ is bounded above only when it is identically zero. Thus, $g(\mathbf{Z}, w) = \infty$ except when $\mathbf{HLH} + \mathbf{Z} - \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij} = 0$, in which case it is simply $\sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2$, ie.

$$g(\mathbf{Z}, w) = \begin{cases} \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2 & \mathbf{HLH} + \mathbf{Z} - \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij} = \mathbf{0}, \\ \infty & \text{otherwise.} \end{cases} \tag{7.9}$$

Using $\mathbf{Z} \succeq 0$, we can derive the following equivalence

$$\mathbf{HLH} + \mathbf{Z} - \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij} = \mathbf{0} \;\Leftrightarrow\; \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij} - \mathbf{HLH} \succeq \mathbf{0}. \tag{7.10}$$

Thus, we have the dual problem of (7.6)

$$\begin{aligned} &\underset{w}{\text{minimize}} \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2 \\ &\text{subject to } \mathbf{G}(w) \succeq \mathbf{HLH} \text{ and } \mathbf{G}(w) = \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij}. \end{aligned} \tag{7.11}$$

Note that $\mathbf{G}(w)$ amounts to the graph Laplacian of a weighted graph with weights given by w. The dual constraint $\mathbf{G}(w) \succeq \mathbf{HLH}$ effectively requires that the eigen-spectrum of the graph Laplacian is bounded below by that of $\mathbf{HLH}$.

When we try to retain maximal diversity between observations by setting $\mathbf{L} = \mathbf{I}$, the inequality constraint in (7.11) becomes $\mathbf{G}(w) \succeq \mathbf{H}$ ($\mathbf{HIH} = \mathbf{H}$). Since $\mathbf{H}$ has all eigenvalues equal to 1 expect a single zero eigenvalue, the inequality is equivalent to $\lambda(\mathbf{G})_{m-1} \geq 1$ where $\lambda(\mathbf{G})_{m-1}$ is the second smallest eigenvalue of $\mathbf{G}$. Furthermore, both the objective $\sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2$ and the constraint are homogenous in w.[1] Thus, we can just as well maximize $\lambda(\mathbf{G})_{m-1}$ subject to constraint on $\sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2$. This leads to an alternative formulation of the dual problem when $\mathbf{L} = \mathbf{I}$,

$$\begin{aligned} &\underset{w}{\text{maximize}}\; \lambda(\mathbf{G})_{m-1} \\ &\text{subject to } \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2 = c \text{ and } \mathbf{G}(w) = \sum_{(i,j)\in\mathcal{N}} w_{ij}\mathbf{E}^{ij}. \end{aligned} \tag{7.12}$$

where $c > 0$ is any constant. The formulation in (7.12) is closely related to the problem of finding the fastest mixing Markov process on a graph [133]. However, for general $\mathbf{L} \succeq 0$, this physical interpretation no longer holds.

7.4.2 Duality and Optimality Conditions

The following duality results hold for the primal and dual pair in (7.6) and (7.11).

Weak duality For any primal feasible $\mathbf{K}$ and any dual feasible w, we have

$$\operatorname{tr}(\mathbf{KHLH}) \leq \sum_{(i,j)\in\mathcal{N}} w_{ij} d_{ij}^2. \tag{7.13}$$

Thus any dual feasible w gives an upper bound on the optimal value of primal problem. This can be easily seen from the Lagrangian (let $\mathbf{K}^*$ be the primal optimal solution, and $\mathbf{Z}^*, w^*$ be

[1] A function $f(x)$ is called homogenous in its parameter x if $f(\alpha x) = \alpha f(x)$ for $\alpha \in \mathbb{R}$.

the dual optimal solution)

$$\begin{aligned}\sum_{(i,j)\in\mathcal{N}} w_{ij}d_{ij}^2 &= \sup_{\mathbf{K}} \mathcal{L}(\mathbf{K},\mathbf{Z},w) \\ &\geq \operatorname{tr}(\mathbf{K}^*\mathbf{HLH}) + \operatorname{tr}(\mathbf{K}^*\mathbf{Z}) - \sum_{(i,j)\in\mathcal{N}} w_{ij}\left(\operatorname{tr}\left(\mathbf{K}^*\mathbf{E}^{ij}\right) - d_{ij}^2\right) \\ &\geq \operatorname{tr}(\mathbf{K}^*\mathbf{HLH}),\end{aligned} \tag{7.14}$$

where the second inequality holds because for any primal feasible we have $\operatorname{tr}(\mathbf{KZ}) \geq 0$ and $\operatorname{tr}\left(\mathbf{KE}^{ij}\right) - d_{ij}^2 = 0$. The gap between the primal and the dual objective, ie. $\operatorname{tr}(\mathbf{KHLH}) - \sum_{(i,j)\in\mathcal{N}} w_{ij}d_{ij}^2$, is called duality gap. Weak duality means that if this gap is zero for a particular $\mathbf{K}$ and w pair, then $\mathbf{K}$ is the optimal for the primal, and w is the optimal for the dual. In other words, zero duality gap guarantees optimality.

Strong Duality Strong duality asserts that there exists a primal-dual feasible pair $\mathbf{K}^*$ and w^* with zeros duality gap, ie.

$$\operatorname{tr}(\mathbf{K}^*\mathbf{HLH}) = \sum_{(i,j)\in\mathcal{N}} w_{ij}^* d_{ij}^2. \tag{7.15}$$

Strong duality can be checked via the Slater's condition for constraint qualification [21]. Basically, for the primal problem in (7.6), the objective is concave in $\mathbf{K}$ and the inequality constraint is affine in $\mathbf{K}$, Slater's condition implies that the strong duality holds.

Optimal Conditions Let $\mathbf{K}^*$, w^* and $\mathbf{Z}^*$ be the primal and dual optimal points with zero duality gap, they satisfy the following Karush-Kuhn-Tucker (KKT) conditions

- Primal feasibility

$$\mathbf{K}^* \succeq 0, \quad \operatorname{tr}\left(\mathbf{K}^*\mathbf{E}^{ij}\right) - d_{ij}^2, \quad \forall (i,j) \in \mathcal{N}. \tag{7.16}$$

- Dual feasibility

$$\mathbf{Z}^* \succeq 0, \quad \sum_{(i,j)\in\mathcal{N}} w_{ij}^* \mathbf{E}^{ij} - \mathbf{HLH} \succeq 0. \tag{7.17}$$

- Complementary slackness

$$\operatorname{tr}(\mathbf{K}^*\mathbf{Z}^*) = 0. \tag{7.18}$$

- Vanishing gradient

$$\mathbf{HLH} + \mathbf{Z}^* - \sum_{(i,j)\in\mathcal{N}} w_{ij}^* \mathbf{E}^{ij} = 0. \tag{7.19}$$

7.4.3 Ability for Dimensionality Reduction

We are interested in the properties of the solution $\mathbf{K}^*$ of the primal problem, in particular the number of nonzero eigenvalues. Recall that at optimality the complementary slackness

condition in (7.18) implies $\mathrm{tr}(\mathbf{K}^*\mathbf{Z}^*) = 0$ (equivalently $\mathrm{tr}(\mathbf{Z}^*\mathbf{K}^*) = 0$). Since both $\mathbf{K}^* \succeq \mathbf{0}$ and $\mathbf{Z}^* \succeq \mathbf{0}$, this condition means $\mathbf{Z}^*\mathbf{K}^* = \mathbf{0}$, ie. the row space of $\mathbf{K}^*$ lies in the null space of $\mathbf{Z}^*$. This also means that the rank of $\mathbf{K}$ is upper bounded by the dimension of the null space of $\mathbf{Z}^*$. From the vanishing gradient condition in (7.19), we know that $\mathbf{Z}^* = \sum_{(i,j)\in\mathcal{N}} w_{ij}^* \mathbf{E}^{ij} - \mathbf{HLH}$, where $\mathbf{G}(w^*) = \sum_{(i,j)\in\mathcal{N}} w_{ij}^* \mathbf{E}^{ij}$ is a graph Laplacian of a weighted graph with edge weights w_{ij}^*. Then we only need to study the number of vanishing eigenvalues of $\mathbf{G}(w^*) - \mathbf{HLH}$.

First, we know $\mathbf{G}(w^*) \succeq \mathbf{0}$ by design, and the multiplicity of vanishing eigenvalues of $\mathbf{G}(w^*)$ is equal to the number of connected components in the corresponding graph. *If the graph is connected*, only one eigenvalue of $\mathbf{G}(w^*)$ vanishes. Hence, the only other zero eigenvalues of $\mathbf{G}(w^*) - \mathbf{HLH}$ would correspond to those eigenvectors lying in the image of $\mathbf{HLH}$. If $\mathbf{L}$ arises from a label kernel matrix, e.g. for an n-class classification problem, then we will only have up to n vanishing eigenvalues in $\mathbf{G}(w^*) - \mathbf{HLH}$. This translates into only up to n nonvanishing eigenvalues in $\mathbf{K}$.

Contrast this observation with plain MVU. In this case $\mathbf{L} = \mathbf{I}$, that is, only one eigenvalue of $\mathbf{HLH}$ vanishes. Hence it is likely that $\mathbf{G}(w^*) - \mathbf{HLH}$ will have many vanishing eigenvalues which translates into many nonzero eigenvalues of $\mathbf{K}$. This is corroborated by experiments (Section 7.6).

7.5 Efficient Implementations

In practice, instead of requiring the distances to remain unchanged in the embedding we only require them to be preserved approximately [148]. We do so by penalizing the slackness between the original distance and the embedding distance, i.e.

$$\begin{aligned} &\underset{\mathbf{K}}{\text{maximize}} \ \mathrm{tr}\left(\mathbf{KHLH}\right) \ - \ \nu \sum_{(i,j)\in\mathcal{N}} \left(\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij} - d_{ij}^2\right)^2 \\ &\text{subject to } \mathbf{K} \succeq 0. \end{aligned} \tag{7.20}$$

Here ν controls the tradeoff between dependence maximization and distance preservation.

7.5.1 Reduced Semidefinite Programming

The semidefinite program usually has a time complexity up to $O(m^6)$. This renders direct implementation of optimization (7.21) infeasible for anything but toy problems. To reduce the computation, we approximate $\mathbf{K}$ using an orthonormal set of vectors $\mathbf{V}$ (of size $m \times n$) and a smaller positive definite matrix $\mathbf{A}$ (of size $n \times n$), i.e. $\mathbf{K} = \mathbf{VAV}^\top$. Conveniently we choose the number of dimensions n to be much smaller than m ($n \ll m$) such that the resulting semidefinite program with respect to $\mathbf{A}$ becomes tractable (clearly this is an approximation).

To obtain the matrix $\mathbf{V}$ we employ a regularization scheme as proposed in [148]. First, we construct a nearest neighbor graph according to $\mathcal{N}$ (we will also refer to this graph and its adjacency matrix as $\mathcal{N}$). Then we form $\mathbf{V}$ by stacking together the bottom n eigenvectors of the graph Laplacian of the neighborhood graph via $\mathcal{N}$. The key idea is that neighbors in the original space remain neighbors in the embedding space. As we require them to have similar locations, the bottom eigenvectors of the graph Laplacian provide a set of good bases for functions smoothly varying across the graph. However, it remains an open question whether this graph Laplacian based regularization is the most natural form of regularization. For instance,

with class labels as the side information, the objective $\mathrm{tr}\,\mathbf{KHLH}$ encourages distant inputs of the same class to be collapsed (since entries in $\mathbf{L}$ are the same for inputs of the same class and we try to make $\mathbf{K}$ similar to $\mathbf{L}$). This seems contradictory to the local distance preserving constraints which prevent similar inputs of the same class from being collapsed. Intuitively, it would seem reasonable to apply the local distance preserving constraints only to inputs of the same class. These issues are left for future investigation and in this thesis we will focus on the graph Laplacian based regularization as appeared in [148].

With these simplifications, we obtain the following optimization problem

$$\begin{aligned} \underset{\mathbf{A}}{\text{maximize}}\ & \mathrm{tr}\left(\mathbf{AV}^\top\mathbf{HLHV}\right) \\ & -\nu\sum_{(i,j)\in\mathcal{N}}\left((\mathbf{VAV}^\top)_{ii}+(\mathbf{VAV}^\top)_{jj}-2(\mathbf{VAV}^\top)_{ij}-d_{ij}^2\right)^2 \\ \text{subject to}\ & \mathbf{A}\succeq 0. \end{aligned} \tag{7.21}$$

Note that the objective in (7.22) is quadratic in the entries of $\mathbf{A}$. Let $\mathbf{a}=\mathrm{vec}(\mathbf{A})$, and we can turn the objective into the following equivalent form (up to an additive constant)

$$\mathbf{b}^\top\mathbf{a}-\mathbf{a}^\top\mathbf{Ba}, \tag{7.22}$$

where

$$\mathbf{b} = \mathrm{vec}(\mathbf{V}^\top\mathbf{HLHV})+2\nu\sum_{(i,j)\in\mathcal{N}}d_{ij}^2\mathbf{b}^{ij}, \tag{7.23}$$

$$\mathbf{B} = \nu\sum_{(i,j)\in\mathcal{N}}\mathbf{b}^{ij}\mathbf{b}^{ij\top}, \tag{7.24}$$

$$\mathbf{b}^{ij} = \mathrm{vec}\left(\mathbf{V}_{i\star}\mathbf{V}_{i\star}^\top+\mathbf{V}_{j\star}\mathbf{V}_{j\star}^\top-\mathbf{V}_{i\star}\mathbf{V}_{j\star}^\top-\mathbf{V}_{j\star}\mathbf{V}_{i\star}^\top\right). \tag{7.25}$$

Furthermore, we introduce a new variable α as a convex upper bound of $\mathbf{a}^\top\mathbf{Ba}$, ie. $\alpha\geq\mathbf{a}^\top\mathbf{Ba}$. Using the Schur complement lemma,[2] we can express the optimization problem in (7.22) into the following standard form of semi-definite programming

$$\begin{aligned} \underset{\mathbf{a},\alpha}{\text{maximize}}\ & \mathbf{b}^\top\mathbf{a}-\alpha \\ \text{subject to}\ & \begin{pmatrix}\mathbf{I} & \mathbf{B}^{1/2}\mathbf{a} & \mathbf{0}\\ \left(\mathbf{B}^{1/2}\mathbf{a}\right)^\top & \alpha & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{A}\end{pmatrix}\succeq\mathbf{0}. \end{aligned} \tag{7.26}$$

7.5.2 Refinement via Gradient Descent

Subsequent to the semidefinite program we perform local refinement of the embedding via gradient descent. Here the objective is reformulated using an $m\times n$ dimensional vector $\mathbf{X}$, i.e. $\mathbf{K}=\mathbf{XX}^\top$. The initial value $\mathbf{X}_0$ is obtained using the n leading eigenvectors of the solution

[2] $\alpha\geq\mathbf{a}^\top\mathbf{Ba}=\alpha-\left(\mathbf{B}^{1/2}\mathbf{a}\right)^\top\mathbf{I}^{-1}\mathbf{B}^{1/2}\mathbf{a}\geq 0\Leftrightarrow\begin{pmatrix}\mathbf{I} & \mathbf{B}^{1/2}\mathbf{a}\\ \left(\mathbf{B}^{1/2}\mathbf{a}\right)^\top & \alpha\end{pmatrix}\succeq\mathbf{0}.$

of (7.21). Suppose $\mathbf{x} = \mathrm{vec}(\mathbf{X})$, the objective and the gradient have the following form

$$\mathbf{x}^\top \left(\mathbf{I} \otimes \mathbf{HLH}\right) \mathbf{x} - \nu \sum_{(i,j)\in\mathcal{N}} \left(\mathbf{x}^\top \tilde{\mathbb{I}}^{ii}\mathbf{x} + \mathbf{x}^\top \tilde{\mathbb{I}}^{jj}\mathbf{x} - 2\mathbf{x}^\top \tilde{\mathbb{I}}^{ij}\mathbf{x} - d_{ij}^2\right)^2, \tag{7.27}$$

$$\left(\mathbf{I} \otimes \mathbf{HLH}\right) \mathbf{x} - 4\nu \sum_{(i,j)\in\mathcal{N}} \left(\mathbf{x}^\top \tilde{\mathbb{I}}^{ii}\mathbf{x} + \mathbf{x}^\top \tilde{\mathbb{I}}^{jj}\mathbf{x} - 2\mathbf{x}^\top \tilde{\mathbb{I}}^{ij}\mathbf{x} - d_{ij}^2\right) \left(\tilde{\mathbb{I}}^{ii} - \tilde{\mathbb{I}}^{ij}\right) \mathbf{x}, \tag{7.28}$$

where $\tilde{\mathbb{I}}^{ij} = \mathbf{I}_{n\times n} \otimes \mathbb{I}^{ij}$ and $\mathbb{I}^{ij}$ is an $m \times m$ indicator matrix with all zeros but a single entry of 1 in position ij.

7.6 Experiments

Ultimately the justification for an algorithm is practical applicability. We demonstrate this based on three datasets: embedding of digits of the USPS database, the Newsgroups 20 dataset containing Usenet articles in text form, and a collection of NIPS papers from 1987 to 1999. We compare our results to the visualizations produced by MVU [150] and PCA. We will highlight places where maximal unfolding using HSIC (MUHSIC or "colored" MVU) produces more meaningful results by incorporating the side information. [3]

For images we use the Euclidean distance between pixel values as the base metric. Clearly more appropriate distances exist, such as the Earth Mover distance [112]. However for purpose of illustrating our results we used a distance which both PCA, MVU and MUHSIC are able to exploit.

For text documents, we perform four standard preprocessing steps: (*i*) the words are stemmed using the Porter stemmer [105]; (*ii*) we filter out common but meaningless stopwords; (*iii*) we delete words that appear in less than 3 documents; (*iv*) we represent each document as a vector using the usual TF/IDF (term frequency / inverse document frequency) weighting scheme. As before, the Euclidean distance on those vectors is used to find the nearest neighbors.

As in [148] we construct the nearest neighbor graph by considering the 1% nearest neighbors of each point. Subsequently the adjacency matrix of this graph is symmetrized. The regularization parameter ν as given in (7.21) is set to 1 as a default. Moreover, as in [148] we choose 10 dimensions ($n = 10$) to decompose the embedding matrix $\mathbf{K}$. Final visualization is carried out using 2 dimensions. This makes our results very comparable to previous work.

7.6.1 Visualization of Three Large Datasets

USPS Digits This dataset consists of images of hand written digits of a resolution of 16×16 pixels. We normalized the data to the range $[-1, 1]$ and used the test set containing 2007 observations. Since it is a digit recognition task, we have $\mathcal{Y} \in [0, \ldots, 9]$. $\mathcal{Y}$ is used to construct the matrix $\mathbf{L}$ by applying the kernel $k(y, y') = \delta_{y,y'}$. This kernel further promotes embedding where images from the same class are grouped tighter. Figure 7.1 shows the results produced by MUHSIC, MVU and PCA.

The overall properties the embeddings are similar across the three methods ('2' on the left, '1' on the right, '7' on top, and '8' at the bottom). Arguably MUHSIC produces a clearer visualization. For instance, images of '5' are clustered tighter in this case than the other two

[3] The datasets are available at http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html for USPS, http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html for the 20 Newsgroups, and http://nips.djvuzone.org/ for NIPS respectively.

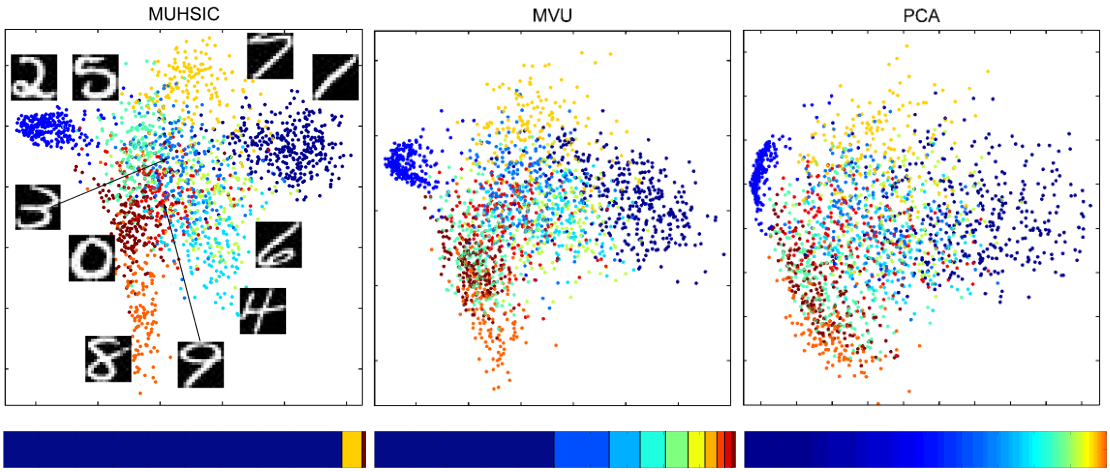


Figure 7.1: Embedding of 2007 USPS digits produced by MUHSIC, MVU and PCA respectively. Colors of the dots are used to denote digits from different classes. The color bar below each figure shows the eigenspectrum of the learned kernel matrix $\mathbf{K}$.

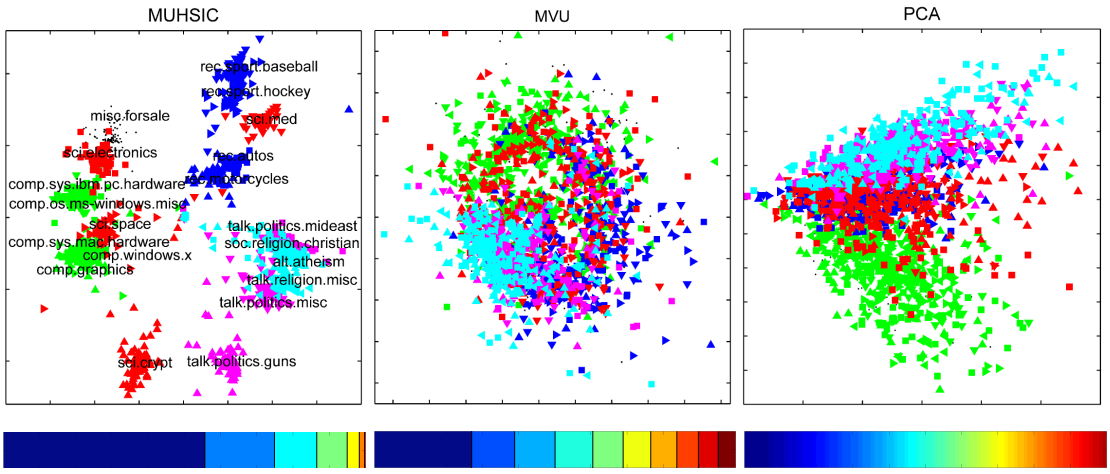


Figure 7.2: Embedding of 2000 newsgroup articles produced by MUHSIC, MVU and PCA respectively. Colors and shapes of the dots are used to denote articles from different newsgroups. The color bar below each figure shows the eigenspectrum of the learned kernel matrix $\mathbf{K}$.

methods. Furthermore, MUHSIC also results in much better separation between images from different classes. For instance, the overlap between '4' and '6' produce by MVU and PCA are largely reduced by MUHSIC. Similar results also hold for '0' and '5.'

Figure 7.1 also shows the eigenspectrum of $\mathbf{K}$ produced by different methods. The eigenvalues are sorted in descending order and normalized by the trace of $\mathbf{K}$. Each patch in the color bar represents an eigenvalue. We see that MUHSIC results in 3 significant eigenvalues, MVU results in 10, while PCA produces a grading of many eigenvalues (as can be seen by an almost continuously changing spectrum in the spectral diagram). This confirms our reasoning of Section 7.4 that the spectrum generated by MUHSIC is likely to be considerably sparser than that of MVU.

Newsgroups This dataset consists of Usenet articles collected from 20 different newsgroups. We use a subset of 2000 documents for our experiments (100 articles from each newsgroup). We remove the headers from the articles before the preprocessing while keeping the subject line. There is a clear hierarchy in the newsgroups. For instance, 5 topics are related to computer science, 3 are related to religion, and 4 are related to recreation. We will use these different topics as side information and apply a delta kernel $k(y, y') = \delta_{y,y'}$ on them. Similar to USPS digits we want to preserve the identity of individual newsgroups. While we did not encode hierarchical information for MVU we recover a meaningful hierarchy among topics, as can be seen in Figure 7.2.

A distinctive feature of the visualizations is that MUHSIC groups articles from individual topics more tightly than MVU and PCA. Furthermore, the semantic information is also well preserved by MUHSIC. For instance, on the left side of the embedding, all computer science topics are placed adjacent to each other; *comp.sys.ibm.pc.hardware* and *comp.os.ms-windows.misc* are adjacent and well separated from *comp.sys.mac.hardware* and *comp.windows.x* and *comp.graphics*. The latter is meaningful since Apple computers are more popular in graphics (so are X windows based systems for scientific visualization). Likewise we see that on the top we find all recreational topics (with rec.sport.baseball and rec.sport.hockey clearly distinguished from the rec.autos and rec.motorcycles groups). A similar adjacency between *talk.politics.mideast* and *soc.religion.christian* is quite interesting. The layout suggests that the content of *talk.politics.guns* and of *sci.crypt* is quite different from other Usenet discussions.

NIPS Papers We used the 1735 regular NIPS papers from 1987 to 1999. They are scanned from the proceedings and transformed into text files via OCR. The table of contents (TOC) is also available. We parse the TOC and construct a coauthor network from it. Our goal is to embed the papers by taking the coauthor information into account. As kernel $l(y, y')$ we simply use the number of authors shared by two papers. To illustrate this we highlighted some known researchers. Furthermore, we also annotated some papers to show the semantics revealed by the embedding. Figure 7.3 shows the results produced by MUHSIC, MVU and PCA.

All three methods correctly represent the two major topics of NIPS papers: artificial systems, i.e. machine learning (they are positioned on the left side of the visualization) and natural systems, i.e. computational neuroscience (which lie on the right). This is be confirmed by examining the highlighted researchers. For instance, the papers by *Smola*, *Schölkopf* and *Jordan* are embedded on the left, whereas the many papers by *Sejnowski*, *Dayan* and *Bialek* can be found on the right.

Unique to the visualization of MUHSIC is that there is a clear grouping of the papers by researchers. For instance, papers on reinforcement learning (*Barto*, *Singh* and *Sutton*) are on the upper left corner; papers by *Hinton* (computational cognitive science) are near the lower left corner; and papers by *Sejnowski* and *Dayan* (computational neuroscientists) are clustered to the right side and adjacent to each other. Interestingly, papers by *Jordan* (currently best-known for his work in graphical models) are grouped close to the papers on reinforcement learning. This is because *Singh* used to be a postdoc of *Jordan*. Another interesting trend is that papers on new fields of research are embedded on the edges. For instance, papers on reinforcement learning (*Barto*, *Singh* and *Sutton*), are along the left edge. This is consistent with the fact that they presented some interesting new results during this period (recall that the time period of the dataset is 1987 to 1999).

Note that while MUHSIC groups papers according to authors, thereby preserving the macroscopic coauthor structure of the data it also reveals the microscopic semantics between the pa-

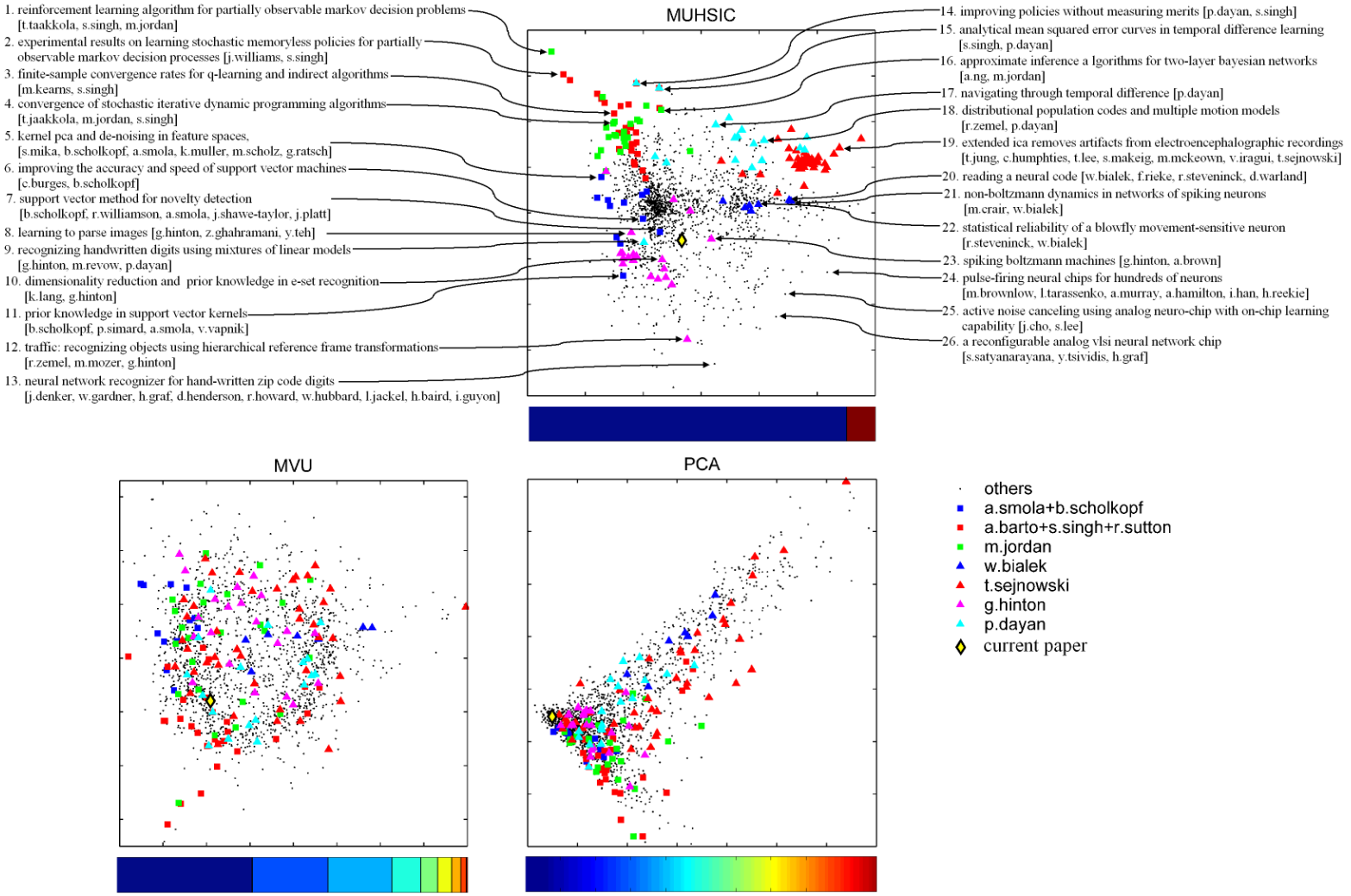


Figure 7.3: Embedding of 1735 NIPS papers produced by MUHSIC, MVU and PCA. Papers by some representative (combinations of) researchers are highlighted as indicated by the legend. The color bar below each figure shows the eigenspectrum of the learned kernel matrix $\mathbf{K}$. The yellow triangle in the graph denotes the embedding of a new paper [128] as submitted to NIPS.

pers. For instance, the 4 papers (numbered from 6 to 9 in Figure 7.3) by *Smola*, *Scholköpf*, *Hinton* and *Dayan* are very close to each other. Although their titles do not convey strong similarity information, these papers all used handwritten digits for the experiments. A second example are papers by *Dayan*. Although most of his papers are on the neuroscience side, two of his papers (numbered 14 and 15) on reinforcement learning can be found on the machine learning side. A third example are papers by *Bialek* and *Hinton* on spiking neurons (numbered 20, 21 and 23). Although *Hinton's* papers are mainly on the left, his paper on spiking Boltzmann machines is closer to *Bialek's* two papers on spiking neurons.

7.6.2 Influence of the Adjacency Matrices

An observation of our experiments is that MUHSIC considerably improves the embedding of the newsgroups and NIPS papers datasets, but its improvement over MVU on USPS digits dataset seems to be minor. Except for clearer separation between classes and an embedding with lower dimension, the overall visualization remains very similar to that by MVU and PCA. To investigate this, we plotted the adjacency matrix of the corresponding nearest neighbor graph in Figure 7.4.

We find that the nearest neighbor graphs for newsgroups and NIPS papers datasets are noisier with no clear block-diagonal structure, whereas the USPS digits dataset has an almost block-diagonal form. Also note that for newsgroups and NIPS papers dataset, several data points almost have all other data points as nearest neighbors. Second, while we have ordered data points from the same class in contiguous places for both USPS digits and newsgroups datasets, only the adjacency matrix of USPS digits dataset show clear correspondence with the class labels, that is, only the USPS digits dataset exhibits a clear block structure. In this case clearly the additional labels do not convey much additional information over the similarity matrix between observations and it is not too surprising that in this case MUSIC generates results not much different from MVU. This suggests that MUHSIC may provide improvement in cases where:

1. The nearest neighbor information is inexact.

2. The side information provides complementary information.

It also indicates that wherever the dominant features of the data are present in the nearest neighbor graph MVU will be able to recover them. However, in general, it is not clear that the desired properties are necessarily those which are dominant. For instance a linguist might care more about the date, length and vocabulary diversity of the documents rather than their topics (or vice versa).

7.6.3 Influence of the Local Refinement Step

As pointed out by [148] it is preferable to perform gradient descent on the embedding after solving the low dimensional approximation of the overall optimization problem. The latter allows for visually more appealing low dimensional representations. In this sense, our implementation (which is based on that by [148]) shares the same properties. Figures 7.5 and 7.6 visualize this fact quite clearly.

The experiments show that while both MVU and MUHSIC strive to generate a low dimensional embedding which preserves local distance information, explicit side information used for MUHSIC ensures that after the initial guess of the subspace which is used to keep optimization

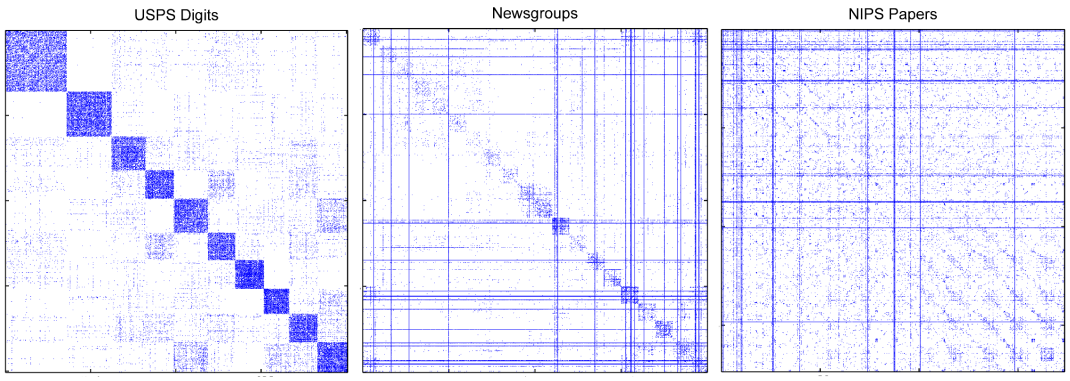


Figure 7.4: Adjacency matrices of the nearest neighbor graphs for the three datasets. We used the Euclidean distance between the vector space representations of the USPS digits, and the TF.IDF representations of the Newsgroups and NIPS papers datasets. 1% of the data were chosen as nearest neighbors and the graphs were symmetrized subsequently.

tractable, the algorithm finds the more representative subspace suitable for visualization. Note that while the full semidefinite program and the low dimensional semidefinite program are both convex, a *dimensionality-constrained* version of the full problem is clearly not, hence the need for a local refinement step. (We will refer to the initial solutions of the unfolding algorithms before the gradient refinement as MUHSIC$^-$ and MVU$^-$ respectively.)

7.6.4 Further Comparison to Other Methods

[56] propose Neighborhood Component Analysis (NCA) for extracting low dimensional representations of data which is optimized for classification. More specifically they minimize a smoothed-out variant of the k-nearest neighbor classification error (the exact k-nearest neighbor error is a piecewise constant function and hence intractable for optimization). That is, the optimization is carried out *directly* on the features of the data. This is advantageous insofar as it generates a direct projection of the data onto a lower-dimensional space which is thought to be representative for the problem. Such a representation can be very convenient at test time.

At the same time, this setting has several drawbacks: firstly, it is restricted to Euclidean spaces underpinning the space of observations, which makes nontrivial Banach space distances, such as [112] inapplicable. Secondly, it being a nonconvex method, optimization may become stuck in local optima, thereby rendering the generation of a specific representation somewhat irreproducible. The main drawback, however, is that computational cost increases with the dimensionality of the data. In practice this means that we were unable to apply NCA to datasets other than the USPS digits dataset, since the dimensionality of the features and the sample size were just too high.[4]

Two other algorithms used for comparison purposes were Relevant Component Analysis (RCA) by [13] and Linear Discriminant Analysis (LDA) by [47]. Examples of their performance on the USPS dataset are given in Figure 7.7. Arguably NCA performs best on this dataset.

To obtain a more quantifiable measure of the performance of low dimensional representations of the data we computed the nearest neighbor scores produced by various embedding

[4]The algorithm kindly provided by [56] did not run successfully on the newsgroups dataset.

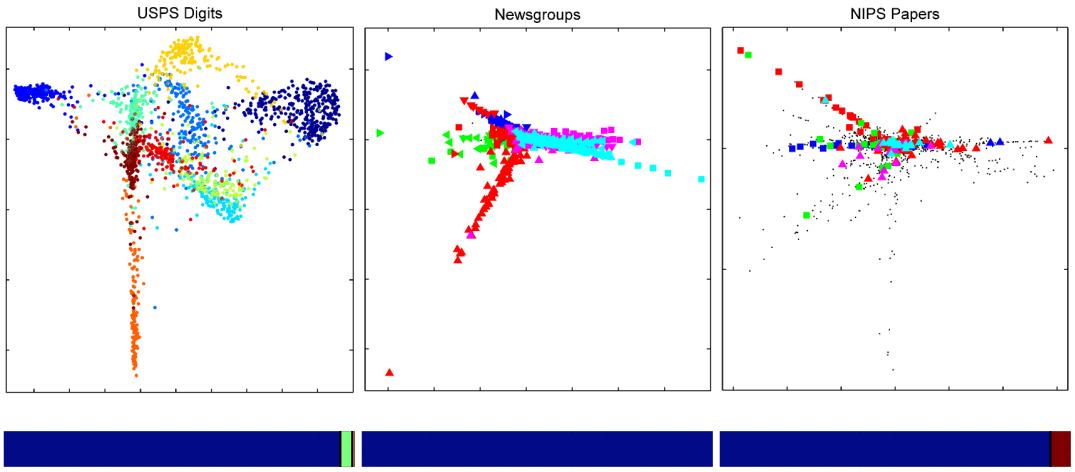


Figure 7.5: Embedding of the three datasets produced by MUHSIC *without* the refinement via gradient descent (MUHSIC$^-$). Colors of the dots are used to denote digits from different classes. The color bar below each figure shows the eigenspectrum of the learned kernel matrix $\mathbf{K}$.

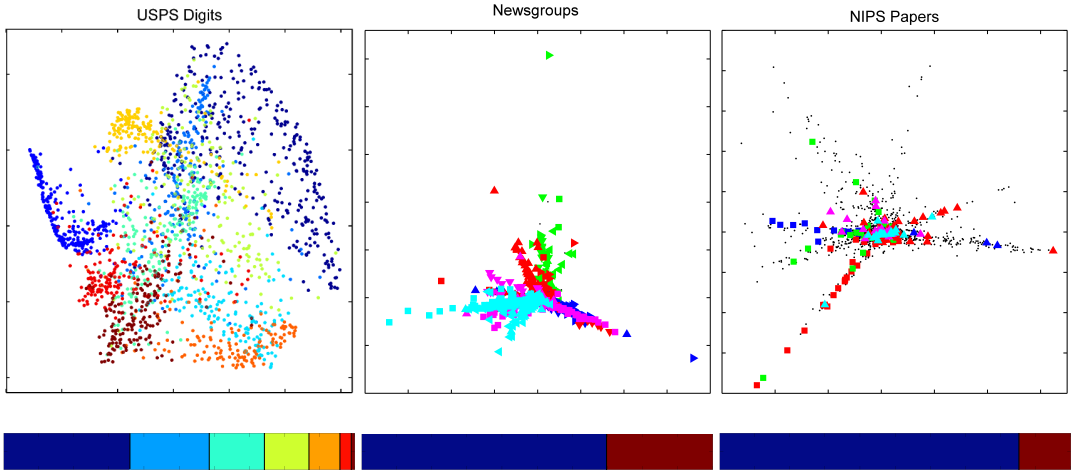


Figure 7.6: Embedding of the three datasets produced by MVU *without* the refinement via gradient descent (MVU$^-$). Colors of the dots are used to denote digits from different classes. The color bar below each figure shows the eigenspectrum of the learned kernel matrix $\mathbf{K}$.

algorithms (the nearest neighbor score computes the percentage of the data points in the embedding that has data point from the same class as the nearest neighbor). The performance is given in Table 7.1 below. We can see that (not very surprisingly) MUHSIC outperforms MVU. More surprising is that it also outperforms RCA and NCA in most problems.

We visualize the embeddings generated by the different methods on an additional datasets (DNA dataset) in Figures 7.8. These are further examples where MUHSIC manages to separate different classes particularly well.

7.6.5 Embedding a New Observation

We also embedded the main text of a 2007 paper [128] into the visualization of the NIPS papers to illustrate how to insert a new data point at test time. Basically, we represent the new paper as a TF.IDF vector and then place it in the location of its nearest neighbor among the NIPS

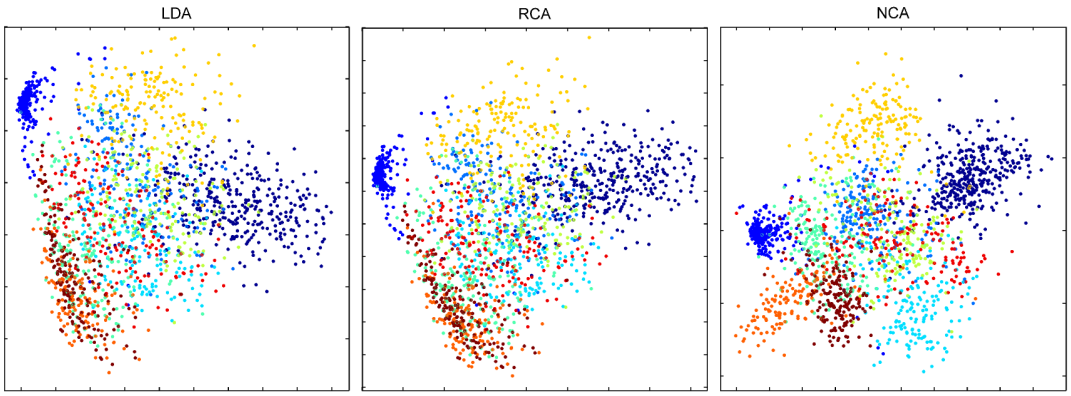


Figure 7.7: Embedding of 2007 USPS digits produced by LDA, RCA and NCA methods. The same color scheme is used as that for MUHSIC. These methods directly learn a 2 dimensional projection, so no eigenspectrum of $\mathbf{K}$ is shown.

Table 7.1: Nearest neighbor scores in % for various multiclass datasets produced by various methods. The sizes of the datasets are listed as triples: (size of dataset, number of dimensions, number of classes). We typically used k =1% of the data points as nearest neighbor for MVU and MUHSIC. In the case that the resulting nearest neighbor graph is not connected, we increase the neighbor size to 2%. Furthermore, we typically used the top n =10 eigenvectors of the graph Laplacian as the bases for optimizing MVU and MUHSIC. In the case that the dimension of the data is small ($\leq$ 100), we decrease the number of bases used to 5.

Dataset	Size	k (%)	n	PCA	LDA	RCA	NCA	MVU	MVU$^-$	MUHSIC	MUHSIC$^-$
USPS	(2007, 256, 10)	1	10	43.9	50.2	50.0	66.2	49.8	59.4	59.4	**71.2**
Wine	(178, 13, 3)	2	5	96.6	**97.2**	**97.2**	**97.2**	95.5	93.8	93.8	94.4
Satimage	(1331, 36, 6)	1	5	75.7	77.3	77.1	77.1	78.4	79.0	**79.1**	78.4
Segment	(2310, 19, 7)	2	5	77.9	82.6	83.3	83.3	82.6	**87.8**	84.5	87.1
Vehicle	(846, 10, 11)	1	5	51.9	45.7	46.2	46.2	42.7	50.1	**57.7**	54.0
DNA	(2000, 180, 3)	1	10	70.5	88.9	88.9	92.4	54.4	60.8	**95.6**	63.9
Vowel	(528, 10, 11)	2	5	52.8	67.1	65.3	65.3	**72.5**	66.5	70.1	44.9
SVMguide2	(391, 20, 3)	1	5	56.0	70.1	67.5	67.5	61.4	62.9	**79.0**	60.1

papers. Its embedding is represented as the yellow triangle in Figure 7.3. We note that this paper used images of handwritten digits for experiments. The location of its embedding is very close to other papers (numbered from 6 to 9) which also used similar dataset for experiments.

To do this, we first represent the new paper using the TF.IDF vector with the model we obtained from the NIPS papers dataset (We produced the TF vector for the main text, and then computed the TF.IDF presentation using the vocabulary and IDF obtained from the NIPS paper dataset). Since the nearest neighbor graph of the NIPS paper dataset is noisy (some data points have almost all other papers as neighbors), we excluded those papers which have more than 3% of the papers as neighbors (Note when we build the nearest neighbor graph, we only require 1% of the data points)

7.7 Summary

In summary, MUHSIC does what it sets out to do, namely provide an embedding of the data which preserves side information possibly available at training time. This way we have a

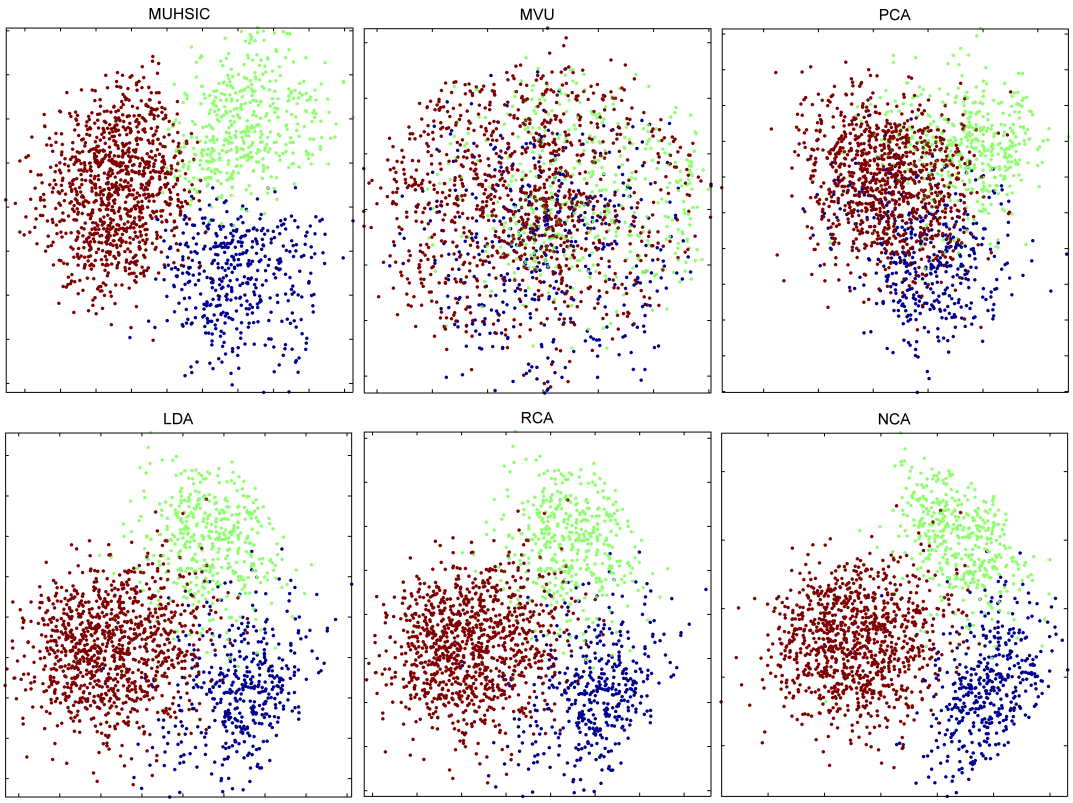


Figure 7.8: The embeddings of the DNA dataset produced by various methods.

means of controlling *which* representation of the data we obtain rather than having to rely on our luck that the representation found by MVU just happens to match what we want to obtain. It makes feature extraction robust to spurious interactions between observations and noise. A fortuitous side-effect is that if the matrix containing side information is of low rank, the spectrum obtained by MUHSIC is of lower rank than that obtained by MVU, too. Finally, we showed that MVU and MUHSIC can be formulated as feature extraction for obtaining maximally dependent features. This provides a statistical footing for the (brilliant) heuristic of maximizing the trace of a covariance matrix [150].

CHAPTER 8

Conclusion

In this thesis, we propose a framework for learning based on Hilbert space embedding of distributions. By embedding the distributions into Hilbert spaces via the mean map, we are able to compare distributions by simply computing their distances in Hilbert spaces. Our analysis shows that learning via empirical mean map provides a good sample convergence to its population counterpart. This new distance between distributions offers us a principled drop-in replacement for other quantities such as Pearson's correlation and mutual information, and allows us to address a range of learning problems such as:

- Two sample tests;
- Covariate shift correction;
- Kernels on sets;
- Density estimation;
- Dependence measure;
- Feature selection;
- Clustering;
- Kernelized sorting;
- Dimensionality reduction.

In this thesis, we focused on the design of a new measure of statistical dependence, and a common framework for learning algorithms based on this dependence measure. In many cases, this leads to new algorithms with superior performance, and it also provides insight to many existing algorithms:

- Hilbert-Schmidt Independence Criterion (HSIC). We derived HSIC via the new measure of distance between distributions applied to a joint distribution and its product of marginals. This definition leads to a simple expression for HSIC which consists only of expectations over kernel terms. We utilized the theory of U-statistics to derive a novel unbiased estimator for HSIC, and studied its asymptotic behavior. Compared to other dependence measures, HSIC has the advantage that it requires no density estimation, and it has a simple and unbiased estimator. Besides, HSIC is also a versatile measure for dependence. By using kernels, HSIC can measure dependence between heterogeneous data types, for example dependence between English and French documents.

- Backward elimination for feature selection via HSIC (BAHSIC). We showed that the BAHSIC family of feature selection algorithms subsumes a whole battery of feature selectors as special cases. Many of these feature selectors are widely used in the bioinformatics community, such as Pearson's correlation coefficient, signal-to-noise ratio, Shrunken Centroid and ridge regression. Such a unifying framework sheds light on their theoretical connections as well as their differences. This explains why different selectors prefer different features: in essence they preprocess data differently and use different kernels in BAHSIC; while linear kernel is visible only to linear dependence, other kernels focus on nonlinear dependence of different types. This also provides us a theoretical basis to choose the best feature selector for a particular task.

- Clustering via HSIC (CLUHSIC). We viewed clustering as maximizing dependence between data and class labels. In particular, class labels are generated to data objects, such that the dependence between the data and their labels is maximized. This is a novel view of clustering which allows us to apply a kernel matrix on the labels as well. The kernel on the labels creates a rich framework for expressing inter-dependencies between the clusters which allows CLUHSIC to utilize structure information in the data and clustering data into clusters with chain, ring or hierarchical relations. Furthermore, when we impose no structure on the cluster labels, we can recover traditional k-means clustering as a special case. In a way, CLUHSIC builds a connection between the geometric, statistical and information theoretic views of clustering.

- Maximal unfolding for dimensionality reduction via HSIC (MUHSIC). We viewed dimensionality reduction as maximizing dependence between reduced representation and side information subject to local distance constraints. A distinctive feature of this new method is that side information can be readily incorporated into the dimensionality reduction process via HSIC. This leads improved representation of the data in term of both the separability of the underlying clusters and the interpretability of representation. We further derived the convex dual of MUHSIC and showed that using HSIC to incorporate side information leads to more aggressive reduction in dimensionality than methods ignoring such information.

As of future work, what follows logically is to explore other learning problems using this distribution embedding approach. For instance, one promising direction is to further explore its connection to graphical models and exponential families. In the case where the distribution $\Pr_{\mathbf{x}}$ satisfies the conditional independence relations specified by a undirected graphical model, the sufficient statistics of an exponential family model decompose along the maximal cliques of the conditional independence graph. Then a mean map using a universal kernel that decomposes in a similar way can map distribution $\Pr_{\mathbf{x}}$ into a unique element in RKHS. Then all subsequent operations on structured domains can simply be dealt with by manipulating elements in RKHS.

Interesting projects spinning off from this theory include, but are not limited to, measuring statistical dependence for structured/non-i.i.d. data and learning a graphical model from the data. The first application is highly relevant to time series, string and sequence analysis. In this case, the dependence measure no longer has a simple estimator as Δ, since the kernels here need to take structural information into account (for instance, first order Markov condition). Four important questions to be answered are: how to design the cliques and the kernels, what is the estimator, whether there is any theoretic guarantee for the estimator, and how to perform learning for structured data via dependence estimation. The second application is related to real world problems such as discovering biological pathways. In this case, one key question

is how to efficiently optimize over the decomposition of the cliques such that it is maximally relevant to an external goal. Here incorporating prior knowledge is very important: it not only limits the search space via hard constraints, but also helps choose appropriate kernels.

Given the above successes and potentials of the embedding approach for distributions, we believe that it will also be useful in many other scenarios beyond what is studied in this thesis. Furthermore, by treating various algorithms under a unifying framework and elucidating their connections, we also expect our work to benefit practitioners in their specific applications.

Bibliography

[1] Alizadeh, A., Eisen, M., Davis, R., et al., "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling," *Nature*, Vol. 403, 2000, pp. 503–511. 5.7, 5.7

[2] Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., and Levine, A., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proc. Natl. Acad. Sci.*, Vol. 96, 1999, pp. 6745–6750. 5.7

[3] Altun, Y. and Smola, A., "Unifying divergence minimization and statistical inference via convex duality," *Proc. Annual Conf. Computational Learning Theory*, edited by H. Simon and G. Lugosi, LNCS, Springer, 2006, pp. 139–153. 3.2.1, 3.2.1, 3.3.4

[4] Amari, S. and Nagaoka, H., *Methods of Information Geometry*, Oxford University Press, 1993. 1, 3.1

[5] Amari, S. and Wu, S., "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, Vol. 12, No. 6, 1999, pp. 783–789. 5.4

[6] Amari, S. and Wu, S., "An information-geometrical method for improving performance of support vector machine classifiers," *Proceedings of ICANN'99*, edited by D. Willshaw and A. Murray, Vol. 1, IEE Press, 1999, pp. 85–90. 6.3

[7] Anderson, N., Hall, P., and Titterington, D., "Two-sample test statistics for measuring discrepancies between two multivariate probability density functions using kernel-based density estimates," *Journal of Multivariate Analysis*, Vol. 50, 1994, pp. 41–54. 3.3.1, 5.4

[8] Anemuller, J., Duann, J.-R., Sejnowski, T. J., and Makeig, S., "Spatio-temporal dynamics in fmri recordings revealed with complex independent component analysis," *Neurocomputing*, Vol. 69, 2006, pp. 1502–1512. 4.3

[9] Archer, A., Fakcharoenphol, J., Harrelson, C., Krauthgamer, R., Talwar, K., and Tardos, E., "Approximate classification via earthmover metrics," *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004. 6.4.1, 6.4.3

[10] Arcones, M. and Giné, E., "On the bootstrap of u and v statistics," *The Annals of Statistics*, Vol. 20, No. 2, 1992, pp. 655–674. 3.3.1

[11] Bach, F. R. and Jordan, M. I., "Kernel independent component analysis," *J. Mach. Learn. Res.*, Vol. 3, 2002, pp. 1–48. 3.4, 4.1, 4.2.3

[12] Baker, C., "Joint measures and cross-covariance operators," *Transactions of the American Mathematical Society*, Vol. 186, 1973, pp. 273–289. 4.4

[13] Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D., "Learning distance functions using equivalence relation," *Proc. Intl. Conf. Machine Learning*, 2003. 7.6.4

[14] Bartlett, P. L. and Mendelson, S., "Rademacher and Gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, Vol. 3, 2002, pp. 463–482. 2.4.2, 2.4.2, 2.4.2, 3.2.1

[15] Bedo, J., Sanderson, C., and Kowalczyk, A., "An efficient alternative to svm based recursive feature elimination with applications in natural language processing and bioinformatics," *Artificial Intelligence*, 2006. 5.4, 1

[16] Beer, D. G., Kardia, S. L., Huang, S. L., et al., "Gene-expression profiles predict survival of patients with lung adenocarcinoma," *Nat. Med.*, Vol. 8, 2002, pp. 816–824. 5.7

[17] Berchuck, A., Iversen, E., and et al., J. L., "Patterns of gene expression that characterize long-term survival in advanced stage serous ovarian cancers," *Clin. Cancer Res.*, Vol. 11, 2005, pp. 3686–3696. 5.7

[18] Bhattacharjee, A., Richards, W. G., Staunton, W. G., et al., "Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses," *Proc. Natl. Acad. Sci.*, Vol. 98, 2001, pp. 13790–13795. 5.7

[19] Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J., "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics (ISMB)*, Vol. 22, No. 14, 2006, pp. e49–e57. 2.2.4, 2.2.4, 3.3.1, 5.4

[20] Bottou, L. and Vapnik, V. N., "Local learning algorithms," *Neural Computation*, Vol. 4, No. 6, 1992, pp. 888–900. 3.3.2

[21] Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, Cambridge, England, 2004. 2.1.1, 6.6, 7.4.2

[22] Bradley, P. S. and Mangasarian, O. L., "Feature selection via concave minimization and support vector machines," *Proc. Intl. Conf. Machine Learning*, edited by J. Shavlik, Morgan Kaufmann Publishers, San Francisco, California, 1998, pp. 82–90, ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z. 5.1.2

[23] Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares, M., and Haussler, D., "Knowledge-based analysis of microarray gene expression data by using support vector machines," *Proc. Natl. Acad. Sci.*, Vol. 97, 2000, pp. 262–267. 5.7

[24] Bullinger, L., Dohner, K., Bair, E., Frohling, S., Schlenk, R. F., Tibshirani, R., Dohner, H., and Pollack, J. R., "Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia," *New England Journal of Medicine*, Vol. 350, No. 16, Apr 2004, pp. 1605–1616. 5.7

[25] Burges, C. J. C. and Vapnik, V., "A new method for constructing artificial neural networks," Interim technical report, ONR contract N00014-94-c-0186, AT&T Bell Laboratories, 1995. 3.4.1, 4.3

[26] Calvino, I., *If on a winter's night a traveler*, Harvest Books, Florida, 1982. 4.7.2

[27] Canu, S. and Smola, A., "Kernel methods and the exponential family," *Neurocomputing*, Vol. 69, No. 7-9, 2006, pp. 714–720. 2.3.2

[28] Casella, G. and Berger, R., *Statistical Inference*, Duxbury, Pacific Grove, CA, 2nd ed., 2002. 4.6

[29] Comon, P., "Independent component analysis, a new concept?" *Signal Processing*, Vol. 36, 1994, pp. 287–314. 3.4

[30] Cover, T. M. and Thomas, J. A., *Elements of Information Theory*, John Wiley and Sons, New York, 1991. 1, 3.1, 3.2.2, 4.1, 4.2.2

[31] Cristianini, N., Kandola, J., Elisseeff, A., and Shawe-Taylor, J., "On optimizing kernel alignment," Tech. rep., UC Davis Department of Statistics, 2003. 5.4

[32] Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M., "The complexity of multiterminal cuts," *SIAM Journal on Computing*, Vol. 23, 1994, pp. 864–894. 6.4

[33] Devroye, L., Györfi, L., and Lugosi, G., *A Probabilistic Theory of Pattern Recognition*, Vol. 31 of *Applications of mathematics*, Springer, New York, 1996. 4.7.1

[34] Dhanasekaran, S. M., Barrette, T. R., Ghosh, D., Shah, R., Varambally, S., Kurachi, K., Pienta, K. J., Rubin, M. A., and Chinnaiyan, A. M., "Delineation of prognostic biomarkers in prostate cancer," *Nature*, Vol. 412, No. 6849, Aug 2001, pp. 822–826. 5.7

[35] Dhillon, I. S., Guan, Y., and Kulis, B., "Kernel kmeans, spectral clustering and normalized cuts," *Conference on Knowledge Discovery and Data Mining*, 2004. 6.3

[36] Ding, C. and He, X., "K-means clustering via principal component analysis," *Proc. Intl. Conf. Machine Learning*, 2004. 6.1, 6.3, 6.5.2

[37] Ding, C., Li, T., and Jordan, M., "Convex and semi-nonnegative matrix factorizations," *LBNL Tech Report 60428*, 2007. 6.5.3, 6.5.3

[38] Ding, C., Li, T., Peng, W., and Park, H., "Orthogonal nonnegative matrix tri-factorizations for clustering," *Proceedings International Conference on Knowledge Discovery and Data Mining*, 2006. 6.5.3

[39] Dornhege, G., Blankertz, B., Curio, G., and Müller, K., "Boosting bit rates in non-invasive EEG single-trial classifications by feature combination and multi-class paradigms," *IEEE Trans. Biomed. Eng.*, Vol. 51, 2004, pp. 993–1002. 5.6

[40] Dornhege, G., Blankertz, B., Krauledat, M., Losch, F., Curio, G., and Müller, K., "Optimizing spatio-temporal filters for improving BCI," *Advances in Neural Information Processing Systems 18*, 2006. 5.6

[41] Dudík, M., Schapire, R., and Phillips, S., "Correcting sample selection bias in maximum entropy density estimation," *Advances in Neural Information Processing Systems 17*, 2005. 3.3.4

[42] Dudík, M. and Schapire, R. E., "Maximum entropy distribution estimation with generalized regularization," *Proc. Annual Conf. Computational Learning Theory*, edited by G. Lugosi and H. U. Simon, Springer Verlag, June 2006. 3.3.4

[43] Dudley, R. M., *Real analysis and probability*, Cambridge University Press, Cambridge, UK, 2002. 3.2.1

[44] Ein-Dor, L., Zuk, O., and Domany, E., "Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer," *Proc. Natl. Acad. Sci. USA*, Vol. 103, No. 15, Apr 2006, pp. 5923–5928. 5.4, 5.7

[45] Feuerverger, A., "A consistent test for bivariate dependence," *International Statistical Review*, Vol. 61, No. 3, 1993, pp. 419–433. 4.3, 4.6.2, 4.7.1

[46] Fine, S. and Scheinberg, K., "Efficient SVM training using low-rank kernel representations," *Journal of Machine Learning Research*, Vol. 2, Dec 2001, pp. 243–264. 4.5.3, 6.7

[47] Fisher, R. A., "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, Vol. 7, 1936, pp. 179–188. 7.6.4

[48] Fukumizu, K., Bach, F., and Gretton, A., "Statistical consistency of kernel canonical correlation analysis," *J. Mach. Learn. Res.*, Vol. 8, 2007, pp. 361–383. 4.2.3

[49] Fukumizu, K., Bach, F. R., and Jordan, M. I., "Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces," *J. Mach. Learn. Res.*, Vol. 5, 2004, pp. 73–99. 3.2.1, 4.4

[50] Fung, G., Mangasarian, O. L., and Smola, A. J., "Minimal kernel classifiers," *Journal of Machine Learning Research*, Vol. 3, 2002, pp. 303–321. 5.2.3

[51] Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Series of Books in Mathematical Sciences, W. H. Freeman, 1979. 2.2.4

[52] Gärtner, T., Flach, P., and Wrobel, S., "On graph kernels: Hardness results and efficient alternatives," *Proc. Annual Conf. Computational Learning Theory*, edited by B. Schölkopf and M. K. Warmuth, Springer, 2003, pp. 129–143. 2.2.4

[53] Gärtner, T., Flach, P. A., Kowalczyk, A., and Smola, A. J., "Multi-instance kernels," *Proc. Intl. Conf. Machine Learning*, 2002. 3.3.3

[54] Gibbs, A. and Su, F., "On choosing and bounding probability metrics," *International Statistical Review*, Vol. 70, 2002, pp. 419. 3.2.2

[55] Girolami, M., "Mercer kernel based clustering in feature space," *IEEE Transactions on Neural Networks*, Vol. 4, No. 13, 2001, pp. 780–784. 6.1, 6.3, 6.8.1

[56] Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R., "Neighbourhood component analysis," *Advances in Neural Information Processing Systems 17*, 2004. 7.6.4, 4

[57] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S., "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, Vol. 286, No. 5439, Oct 1999, pp. 531–537. 5.7

[58] Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A., "A kernel method for the two-sample-problem," *Advances in Neural Information Processing Systems 19*, MIT Press, Cambridge, MA, 2007. 1, 2.5, 3.2.1, 3.2.1, 3.2.1, 3.3.1, 3.3.1, 3.3.1, 3.3.1, 2, 5.3.2

[59] Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B., "Measuring statistical dependence with Hilbert-Schmidt norms," *Proc. Intl. Conf. on Algorithmic Learning Theory*, 2005, pp. 63–78. 4, 3.4, 3.4.1, 4.1, 4.3, 4.4, 4.4, 4.5.1, 35, 4.5.3, 4.5.4, 2, 4.7.1, 6.1, 7.1, 7.3

[60] Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schölkopf, B., and Smola, A., "A kernel statistical test of independence," *Neural Information Processing Systems*, 2007. 2.5

[61] Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B., "Kernel methods for measuring independence," *J. Mach. Learn. Res.*, Vol. 6, 2005, pp. 2075–2129. (document), 3.4, 4.1, 4.2.4, 4.2.4, 4.7.1, 4.3

[62] Gruvberger, S., Ringner, M., Chen, Y., Panavally, S., Saal, L. H., Borg, A., Ferno, M., Peterson, C., and Meltzer, P. S., "Estrogen receptor status in breast cancer is associated with remarkably distinct gene expression patterns," *Cancer Res*, Vol. 61, No. 16, Aug 2001, pp. 5979–5984. 5.7

[63] Guestrin, C., Krause, A., and Singh, A., "Near-optimal sensor placements in gaussian processes," *International Conference on Machine Learning ICML'05*, 2005. 1, 5.1.2

[64] Guyon, I. and Elisseeff, A., "An introduction to variable and feature selection," *Journal of Machine Learning Research*, Vol. 3, March 2003, pp. 1157–1182. 5.1, 5.2

[65] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V., "Gene selection for cancer classification using support vector machines," *Machine Learning*, Vol. 46, 2002, pp. 389–422. 5.1.2, 5.4, 5.5.2

[66] Ham, J., Lee, D., Mika, S., and Schölkopf, B., "A kernel view of the dimensionality reduction of manifolds," *Proceedings of the Twenty-First International Conference on Machine Learning*, ACM Press, New York, NY, USA, 2004, pp. 369–376. 3.4.3

[67] Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning*, Springer, New York, 2001. 5.4, 5.4

[68] Hein, M. and Bousquet, O., "Hilbertian metrics and positive definite kernels on probability measures," *Proc. of AI & Statistics*, edited by Z. Ghahramani and R. Cowell, Vol. 10, 2005. 3.2.3

[69] Hoeffding, W., "A class of statistics with asymptotically normal distribution," *The Annals of Mathematical Statistics*, Vol. 19, No. 3, 1948, pp. 293–325. 4.5.4

[70] Hoeffding, W., "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, Vol. 58, 1963, pp. 13–30. 2.4.1, 15, 2.5, 20, 3.3.1, 4.5.4

[71] Hofmann, T., Schölkopf, B., and Smola, A. J., "Kernel methods in machine learning," Tech. Rep. 156, Max-Planck-Institut für biologische Kybernetik, 2006, To appear in the Annals of Statistics. 2.2.2

[72] Horn, R. A. and Johnson, C. R., *Matrix Analysis*, Cambridge University Press, Cambridge, 1985. 6.5.2

[73] Huang, J., Smola, A., Gretton, A., Borgwardt, K., and Schölkopf, B., "Correcting sample selection bias by unlabeled data," *Advances in Neural Information Processing Systems 19*, MIT Press, Cambridge, MA, 2007. 1, 3.3.2, 3.3.2

[74] Iizuka, N., Oka, M., Yamada-Okabe, H., et al., "Oligonucleotide microarray for prediction of early intrahepatic recurrence of hepatocellular carcinoma after curative resection," *Lancet*, Vol. 361, 2003, pp. 923–929. 5.7

[75] Jebara, T., "Kernelizing sorting, permutation and alignment for minimum volume pca," *Conference on Learning Theory*, 2004. 3.4.2

[76] Johnson, N. L., Kotz, S., and Balakrishnan, N., *Continuous Univariate Distributions. Volume 1 (Second Edition)*, John Wiley and Sons, 1994. 3.3.1, 4.6.2

[77] Kankainen, A., *Consistent Testing of Total Independence Based on the Empirical Characteristic Function*, Ph.D. thesis, University of Jyväskylä, 1995. 4.3, 4.6.2, 4.6.2, 4.7.1

[78] Kannan, R., Vempala, S., and Vetta, A., "On clusterings: good, bad and spectral," *Journal of the ACM*, Vol. 51, 2004, pp. 497–515. 6.1, 6.7

[79] Karloff, H., Khot, S., Mehta, A., and Rabani, Y., "On earthmover distance, metric labeling, and 0-extension," *Proceedings of the 38th annual ACM symposium on Theory of computing*, 2006. 6.4.1

[80] Karvanen, J., "A resampling test for the total independence of stationary time series: Application to the performance evaluation of ICA algorithms," *Neural Processing Letters*, Vol. 22, No. 3, 2005, pp. 311 – 324. 4.6.2, 4.7.1

[81] Kira, K. and Rendell, L., "A practical approach to feature selection," *Proc. 9th Intl. Workshop on Machine Learning*, 1992, pp. 249–256. 5.5.2

[82] Koltchinskii, V., "Rademacher penalties and structural risk minimization," *IEEE Trans. Inform. Theory*, Vol. 47, 2001, pp. 1902–1914. 3.2.1

[83] Krause, A. and Guestrin, C., "Near-optimal nonmyopic value of information in graphical models," *Uncertainty in Artificial Intelligence UAI'05*, 2005. 1, 3.1, 3.4.3

[84] Ku, C. and Fine, T., "Testing for stochastic independence: application to blind source separation," *IEEE Transactions on Signal Processing*, Vol. 53, No. 5, 2005, pp. 1815–1826. 4.7.1

[85] Lee, D. and Seung, H., "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems 13*, edited by T. K. Leen, T. G. Dietterich, and V. Tresp, MIT Press, 2001. 6.5.3

[86] Lee, T.-W., Girolami, M., Bell, A., and Sejnowski, T., "A unifying framework for independent component analysis," *Comput. Math. Appl.*, Vol. 39, 2000, pp. 1–21. 3.4

[87] Lemm, S., Blankertz, B., Curio, G., and Mülller, K.-R., "Spatio-spectral filters for improving the classification of single trial EEG," *IEEE Trans. Biomed. Eng.*, Vol. 52, 2005, pp. 1541–1548. 5.6

[88] Leslie, C., Eskin, E., and Noble, W. S., "The spectrum kernel: A string kernel for SVM protein classification," *Proceedings of the Pacific Symposium on Biocomputing*, World Scientific Publishing, Singapore, 2002, pp. 564–575. 4.7.2

[89] Leslie, C., Eskin, E., Weston, J., and Noble, W. S., "Mismatch string kernels for SVM protein classification," *Advances in Neural Information Processing Systems 15*, edited by S. Becker, S. Thrun, and K. Obermayer, Vol. 15, MIT Press, Cambridge, MA, 2002. 5.3.1

[90] Lönnstedt, I. and Speed, T., "Replicated microarray data," *Statistica Sinica*, Vol. 12, 2002, pp. 31–46. 5.4

[91] MacQueen, J., "Some methods of classification and analysis of multivariate observations," *Proc. 5th Berkeley Symposium on Math., Stat., and Prob.*, edited by L. M. LeCam and J. Neyman, U. California Press, Berkeley, CA, 1967, p. 281. 6.1

[92] McDiarmid, C., "On the method of bounded differences," *Surveys in Combinatorics*, 1969, pp. 148–188, Cambridge University Press. 2.4.1, 14, 3.3.1

[93] Meilă, M., "The local equivalence of two distances between clustering: the misclassification error metric and the chi-squared distance," *Proc. Intl. Conf. Machine Learning*, 2005. 6.7

[94] Meilă, M., "The uniqueness of a good optimal for k-means," *Proc. Intl. Conf. Machine Learning*, 2006. 6.1, 6.7, 6.7, 6.8.2

[95] Meilă, M., "Data centering in feature space," *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2003. 6.5.2

[96] Müller, A., "Integral probability metrics and their generating classes of functions," Vol. 29, No. 2, 1997, pp. 429–443. 3.2.2

[97] Müller, A., "Stochastic orders generated by integrals: a unified study," *Advances in Applied Probability*, Vol. 29, No. 2, 1997, pp. 414–428. 3.2.2

[98] Neal, R. M., "Assessing relevance determination methods using delve," *Neural Networks and Machine Learning*, Springer, 1998, pp. 97–129. 5.1.2

[99] Nemenman, I., Shafee, F., and Bialek, W., "Entropy and inference, revisited," *Neural Information Processing Systems*, Vol. 14, MIT Press, Cambridge, MA, 2002. 1, 3.1, 3.2.2, 4.5.2, 6.1

[100] Neumann, J., Schnörr, C., and Steidl, G., "Combined SVM-based feature selection and classification," *Machine Learning*, Vol. 61, 2005, pp. 129–150. 5.4

[101] Ng, A., Jordan, M., and Weiss, Y., "Spectral clustering: Analysis and an algorithm (with appendix)," *Advances in Neural Information Processing Systems 14*, edited by T. G. Dietterich, S. Becker, and Z. Ghahramani, 2002. 6.1, 6.5.2, 6.7

[102] Owen, A. and Tribble, S., "A quasi-monte carlo metropolis algorithm," *PNAS*, Vol. 102, No. 25, 2005, pp. 8844–8849. 3.3.4

[103] Paninski, L., "Estimation of entropy and mutual information," *Neural Computation*, Vol. 15, 2003, pp. 1191–1254. 3.2.2

[104] Pantic, M. and Rothkrantz, L., "Automatic analysis of facial expressions: The state of the art," *PAMI*, Vol. 22, No. 12, 2000, pp. 1424 – 1445. 6.8.3

[105] Porter, M., "An algorithm for suffix stripping," *Program*, Vol. 14, No. 3, 1980, pp. 130–137. 7.6

[106] Rachev, S. T., *Probability metrics and the stability of stochastic models*, Wiley, Chichester, 1991. 3.2.2

[107] Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006. 3.1

[108] Ravikumar, P. and Lafferty, J., "Variational Chernoff bounds for graphical models," *Uncertainty in Artificial Intelligence UAI04*, 2004. 2.3.3

[109] Read, T. and Cressie, N., *Goodness-Of-Fit Statistics for Discrete Multivariate Analysis*, Springer-Verlag, New York, 1988. 4.7.1

[110] Rényi, A., "On measures of dependence," *Acta Math. Acad. Sci. Hungar.*, Vol. 10, 1959, pp. 441–451. 4.1, 4.2, 4.2.3

[111] Rosenwald, A., Wright, G., Chan, G., et al., "The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma," *N. Engl. J. Med.*, Vol. 346, 2002, pp. 1937–1947. 5.7

[112] Rubner, Y., Tomasi, C., and Guibas, L., "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, Vol. 40, No. 2, 2000, pp. 99–121. 6.4.2, 7.6, 7.6.4

[113] Schölkopf, B., *Support Vector Learning*, R. Oldenbourg Verlag, Munich, 1997, Download: http://www.kernel-machines.org. 4.3

[114] Schölkopf, B., Bartlett, P. L., Smola, A. J., and Williamson, R. C., "Shrinking the tube: a new support vector regression algorithm," *Advances in Neural Information Processing Systems 11*, edited by M. S. Kearns, S. A. Solla, and D. A. Cohn, MIT Press, Cambridge, MA, 1999, pp. 330–336. 5.4

[115] Schölkopf, B. and Smola, A., *Learning with Kernels*, MIT Press, Cambridge, MA, 2002. 2.1, 2.1.1, 3.1, 5.5.2, 6.8.1

[116] Schölkopf, B., Smola, A. J., and Müller, K.-R., "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, Vol. 10, 1998, pp. 1299–1319. 3.4.3, 6.1, 6.2, 6.3

[117] Schölkopf, B., Tsuda, K., and Vert, J.-P., *Kernel Methods in Computational Biology*, MIT Press, Cambridge, MA, 2004. 2.2.2

[118] Serfling, R., *Approximation Theorems of Mathematical Statistics*, Wiley, New York, 1980. 2.5, 2.5, 2.5, 3.3.1, 3.3.1, 4.6.1, 42, 4.6.2, A.2

[119] Shawe-Taylor, J. and Cristianini, N., *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK, 2004. 2.2.2

[120] Shen, H., Jegelka, S., and Gretton, A., "Fast kernel ICA using an approximate Newton method," *AISTATS 11*, 2007. 3.4

[121] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *Proc. Intl. Conf. Computer Vision*, Jun. 1997. 6.1, 6.5.2

[122] Slonim, N., *The Information Bottleneck: Theory and Applications*, Ph.D. thesis, The Hebrew University of Jerusalem, 2002. 3.4.2, 6.1

[123] Slonim, N. and Tishby, N., "Agglomerative information bottleneck," *Advances in Neural Information Processing Systems 12*, edited by S. A. Solla, T. K. Leen, and K.-R. Müller, MIT Press, Cambridge, MA, 2000, pp. 617–623. 1, 3.1

[124] Smola, A. J., Gretton, A., Song, L., and Schölkopf, B., "A Hilbert space embedding for distributions," *Proc. Intl. Conf. Algorithmic Learning Theory*, Vol. 4754 of *LNAI*, Springer-Verlag, 2007, pp. 13–31. 1

[125] Smola, A. J. and Kondor, I. R., "Kernels and regularization on graphs," *Proc. Annual Conf. Computational Learning Theory*, edited by B. Schölkopf and M. K. Warmuth, Lecture Notes in Comput. Sci., Springer-Verlag, Heidelberg, Germany, 2003, pp. 144–158. 3.4.3

[126] Smyth, G., "Linear models and empirical bayes methods for assessing differential expressionin microarray experiments," *Statistical Applications in Genetics and Molecular Biology*, Vol. 3, 2004. 5.4, 5.4, 5.4, 5.4

[127] Song, L., Bedo, J., Borgwardt, K., Gretton, A., and Smola, A., "Gene selection via the BAHSIC family of algorithms," *Bioinformatics (ISMB)*, Vol. 23, No. 13, 2007, pp. i490–i498. 1

[128] Song, L., Smola, A., Borgwardt, K., and Gretton, A., "Colored maximum variance unfolding," *Neural Information Processing Systems*, 2007. (document), 1, 7.3, 7.6.5

[129] Song, L., Smola, A., Gretton, A., and Borgwardt, K., "A dependence maximization view of clustering," *Proc. Intl. Conf. Machine Learning*, Omnipress, 2007, pp. 815–822. 1

[130] Song, L., Smola, A., Gretton, A., Borgwardt, K., and Bedo, J., "Supervised feature selection via dependence estimation," *Proc. Intl. Conf. Machine Learning*, Omnipress, 2007, pp. 823–830. 1

[131] Steinwart, I., "On the influence of the kernel on the consistency of support vector machines," *J. Mach. Learn. Res.*, Vol. 2, 2001, pp. 67–93. 1, 2.2.3, 5, 6, 2.2.4, 2.2.4, 34, 5.3.1

[132] Stögbauer, H., Kraskov, A., Astakhov, S., and Grassberger, P., "Least dependent component analysis based on mutual information," *Phys. Rev. E*, Vol. 70, No. 6, 2004, pp. 066123. 1, 3.1

[133] Sun, J., Boyd, S., Xiao, L., and Diaconis, P., "The fastest mixing markove process on a graph and a connection to a maximum variance unfolding problem," *SIAM Review*, Vol. 48, No. 4, 2006, pp. 681–699. 7.1, 7.4.1, 7.4.1

[134] Teo, C. H. and Vishwanathan, S. V. N., "Fast and space efficient string kernels using suffix arrays," *ICML '06: Proceedings of the 23rd international conference on Machine learning*, ACM Press, New York, NY, USA, 2006, pp. 929–936. 4.7.2

[135] Theis, F., "Towards a general independent subspace analysis," *Neural Information Processing Systems*, 2006. 4.7.1

[136] Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G., "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *National Academy of Sciences*, Vol. 99, 2002, pp. 6567–6572. 5.4

[137] Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G., "Class prediction by nearest shrunken centroids, with applicaitons to dna microarrays," *Stat Sci*, Vol. 18, 2003, pp. 104–117. 5.4

[138] Torkkola, K., "Feature extraction by non-parametric mutual information maximization," *J. Mach. Learn. Res.*, Vol. 3, 2003, pp. 1415–1438. 5.4

[139] Valk, P. J., Verhaak, R. G., Beijen, M. A., Erpelinck, C. A., van Waalwijk van Doorn-Khosrovani, S. B., Boer, J. M., Beverloo, H. B., Moorhouse, M. J., van der Spek, P. J., Lowenberg, B., and Delwel, R., "Prognostically useful gene-expression profiles in acute myeloid leukemia," *New England Journal of Medicine*, Vol. 350, No. 16, Apr 2004, pp. 1617–1628. 5.7

[140] van de Vijver, M. J., He, Y. D., van 't Veer, L. J., et al., "A gene-expression signature as a predictor of survival in breast cancer," *N. Engl. J. Med.*, Vol. 247, 2002, pp. 1999–2009. 5.7

[141] van't Veer, L. J., Dai, H., van de Vijver, M. J., He, Y. D., Hart, A. A. M., et al., "Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, Vol. 415, 2002, pp. 530–536. 5.4, 5.7

[142] Vapnik, V., *Statistical Learning Theory*, John Wiley and Sons, New York, 1998. 3.4.1, 4.3

[143] Vapnik, V. and Chervonenkis, A., "On the uniform convergence of relative frequencies of events to their probabilities," *Theory Probab. Appl.*, Vol. 16, No. 2, 1971, pp. 264–281. 3.2.1

[144] Vishwanathan, S. V. N. and Smola, A. J., "Fast kernels for string and tree matching," *Advances in Neural Information Processing Systems 15*, edited by S. Becker, S. Thrun, and K. Obermayer, MIT Press, Cambridge, MA, 2003, pp. 569–576. 5.3.1

[145] Wainwright, M. J. and Jordan, M. I., "Graphical models, exponential families, and variational inference," Tech. Rep. 649, UC Berkeley, Department of Statistics, September 2003. 2.3.3, 2.3.3, 3.2.3

[146] Wang, Y., Klijn, J. G., Zhang, Y., et al., "Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer," *Lancet*, Vol. 365, 2005, pp. 671–679. 5.7

[147] Warnat, P., Eils, R., and Brors, B., "Cross-platform analysis of cancer microarray data improves gene expression based classification of phenotypes," *BMC Bioinformatics*, Vol. 6, Nov 2005, pp. 265. 5.7

[148] Weinberger, K., Sha, F., Zhu, Q., and Saul, L., "Graph laplacian regularization for large-scale semidefinte programming," *Neural Information Processing Systems*, 2006. 7.2, 7.5, 7.5.1, 7.6, 7.6.3

[149] Weinberger, K. Q. and Saul, L. K., "An introduction to nonlinear dimensionality reduction by maximum variance unfolding," *Proceedings of the National Conference on Artificial Intelligence*, 2006. 1, 3.4.3, 6.8.3

[150] Weinberger, K. Q., Sha, F., and Saul, L. K., "Learning a kernel matrix for nonlinear dimensionality reduction," *Proceedings of the* 21^{st} *International Conference on Machine Learning*, Banff, Canada, 2004. 7.1, 7.2, 7.6, 7.7

[151] Welsh, J. B., Sapinoso, L. M., Su, A. I., Kern, S. G., Wang-Rodriguez, J., Moskaluk, C. A., r. Frierson HF, J., and Hampton, G. M., "Analysis of gene expression identifies candidate markers and pharmacological targets in prostate cancer," *Cancer Res*, Vol. 61, No. 16, Aug 2001, pp. 5974–5978. 5.7

[152] West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Jr, J. O., J.R.Marks, and J.R.Nevins, "Predicting the clinical status of human breast cancer by using gene expression profiles," *PNAS*, Vol. 98, No. 20, 2001. 5.7

[153] Weston, J., Elisseeff, A., Schölkopf, B., and Tipping, M., "Use of zero-norm with linear models and kernel methods," *Journal of Machine Learning Research*, Vol. 3, 2003, pp. 1439–1461. 5.1.2, 5.5.2

[154] Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V., "Feature selection for SVMs," *Advances in Neural Information Processing Systems 13*, 2000, pp. 668–674. 5.1.1, 5.1.2, 5.5.2

[155] Yuille, A. and Rangarajan, A., "The concave-convex procedure," *Neural Computation*, Vol. 15, 2003, pp. 915–936. 5.2.3

[156] Zaffalon, M. and Hutter, M., "Robust feature selection using distributions of mutual information," *Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, edited by A. Darwiche and N. Friedman, Morgan Kaufmann, San Francisco, CA., 2002, pp. 577–584. 1, 5.1.1, 5.5.1

[157] Zass, R. and Shashua, A., "A unifying approach to hard and probablistic clustering," *Proc. Intl. Conf. Computer Vision*, 2005. 6.3

[158] Zha, H., Ding, C., Gu, M., He, X., and Simon, H., "Spectral relaxation for k-means clustering," *Advances in Neural Information Processing Systems 14*, 2001, pp. 1057–1064. 6.1, 6.3, 6.5.2, 6.5.2

[159] Zolotarev, V. M., "Probility metrics," *Theory of Probability and its Applications*, Vol. 28, No. 2, 1984, pp. 278–302. 3.2.2

APPENDIX **A**

Mean and Variance of $\widehat{\mathrm{HSIC}}_b$ under H_0

In this appendix, we prove Theorem 43 and 44. We begin by noting that under the null hypothesis, all the terms common to $\widehat{\mathrm{HSIC}}_u$ in (4.33) and $\widehat{\mathrm{HSIC}}_b$ in (4.32) vanish, since under these conditions $\mathbb{E}_Z(\widehat{\mathrm{HSIC}}_u) = 0$. Therefore, the only part of $\widehat{\mathrm{HSIC}}_b$ that does not vanish is the bias $\widehat{\mathrm{HSIC}}_b - \widehat{\mathrm{HSIC}}_u$. The mean and variance of $\widehat{\mathrm{HSIC}}_b$ under H_0 are exactly those of the bias.

A.1 Mean

We prove Theorem 43 in this section. Let $k_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ and l_{ij} similarly. We split the difference $\widehat{\mathrm{HSIC}}_b - \widehat{\mathrm{HSIC}}_u$ into the difference between three terms

$$\frac{1}{m^2}\sum_{i,j} k_{ij}l_{ij} - \frac{1}{(m)_2}\sum_{(i,j)\in \mathbf{i}_2^m} k_{ij}l_{ij} = \frac{1}{m^2}\sum_{i} k_{ii}l_{ii} - \frac{1}{m(m)_2}\sum_{(i,j)\in \mathbf{i}_2^m} k_{ij}l_{ij} \tag{A.1}$$

$$\begin{aligned}\frac{1}{m^3}\sum_{i,j,q} k_{ij}l_{iq} - \frac{1}{(m)_3}\sum_{(i,j,q)\in \mathbf{i}_3^m} k_{ij}l_{iq} &= \frac{1}{m^3}\sum_{(i,j)\in \mathbf{i}_2^m} (k_{ii}l_{ij} + k_{ij}l_{ii} + k_{ij}l_{ij}) \\ &\quad - \frac{3}{m(m)_3}\sum_{(i,j,q)\in \mathbf{i}_3^m} k_{ij}l_{iq} + O(m^{-2})\end{aligned} \tag{A.2}$$

$$\begin{aligned}\frac{1}{m^4}\sum_{i,j,q,r} k_{ij}l_{qr} - \frac{1}{(m)_4}\sum_{(i,j,q,r)\in \mathbf{i}_4^m} k_{ij}l_{qr} &= \frac{1}{m^4}\sum_{(i,j,q)\in \mathbf{i}_3^m} (k_{ii}l_{jq} + 4k_{ij}l_{iq} + k_{ij}l_{qq}) \\ &\quad - \frac{6}{m(m)_4}\sum_{(i,j,q,r)\in \mathbf{i}_4^m} k_{ij}l_{qr} + O(m^{-2}).\end{aligned} \tag{A.3}$$

Combining terms, we get

$$\begin{aligned}m(\widehat{\mathrm{HSIC}}_b - \widehat{\mathrm{HSIC}}_u) &= \frac{1}{m}\sum_{i} k_{ii}l_{ii} - \frac{2}{m^2}\sum_{(i,j)\in \mathbf{i}_2^m} (k_{ii}l_{ij} + k_{ij}l_{ii}) \\ &\quad + \frac{1}{m^3}\sum_{(i,j,q)\in \mathbf{i}_3^m} (k_{ii}l_{jq} + k_{ij}l_{qq}) \\ &\quad - \frac{3}{(m)_2}\sum_{(i,j)\in \mathbf{i}_2^m} k_{ij}l_{ij} + \frac{10}{(m)_3}\sum_{(i,j,q)\in \mathbf{i}_3^m} k_{ij}l_{iq} \\ &\quad - \frac{6}{(m)_4}\sum_{(i,j,q,r)\in \mathbf{i}_4^m} k_{ij}l_{qr} + O(m^{-2}).\end{aligned} \tag{A.4}$$

We next take the expectation of this expression. Introducing the notation $\mathbb{E}_{\mathbf{y}} l = \mathbb{E}_{\mathbf{y}} [l(\mathbf{y}, \mathbf{y})]$ and $\mathbb{E}_{\mathbf{xyy}'} kl = \mathbb{E}_{\mathbf{xyy}'} [k(\mathbf{x}, \mathbf{x}) l(\mathbf{y}, \mathbf{y}')]$ (with the remaining notation defined by analogy), we obtain under H_0 that

$$\begin{aligned} m\mathbb{E}_Z \left[\widehat{\mathrm{HSIC}}_b\right] &= m\mathbb{E}_Z \left[\widehat{\mathrm{HSIC}}_b - \widehat{\mathrm{HSIC}}_u\right] \\ &= \mathbb{E}_{\mathbf{xy}} kl - 2(\mathbb{E}_{\mathbf{xyy}'} kl + \mathbb{E}_{\mathbf{xx}'\mathbf{y}} kl) + \mathbb{E}_{\mathbf{xy}'\mathbf{y}''} kl + \mathbb{E}_{\mathbf{xx}'\mathbf{y}''} kl \\ &\quad - 3\mathbb{E}_{\mathbf{xx}'\mathbf{yy}'} kl + 10\mathbb{E}_{\mathbf{xx}'\mathbf{yy}''} kl - 6\mathbb{E}_{\mathbf{xx}'} k \mathbb{E}_{\mathbf{yy}'} l \end{aligned} \tag{A.5}$$

At independence, the three terms on the final line merge, and we are left with

$$\widehat{\mathrm{HSIC}}_b = \frac{1}{m}(\mathbb{E}_{\mathbf{xy}} kl + \|\mu_{\mathbf{x}}\|^2_{\mathcal{F}} \|\mu_{\mathbf{y}}\|^2_{\mathcal{G}} - \mathbb{E}_{\mathbf{y}} l \|\mu_{\mathbf{x}}\|^2_{\mathcal{F}} - \mathbb{E}_{\mathbf{x}} k \|\mu_{\mathbf{y}}\|^2_{\mathcal{G}})$$

A.2 Variance

We prove Theorem 44 in this section. We begin by computing the variance of the U-statistic $\widehat{\mathrm{HSIC}}_u$ associated with the V-statistic $\widehat{\mathrm{HSIC}}_b$, i.e.

$$\widehat{\mathrm{HSIC}}_b = \frac{1}{(m)_4} \sum_{(i,j,q,r) \in \mathbf{i}_4^m} h_{ijqr}, \tag{A.6}$$

where h_{ijqr} is defined in (4.42) According to [118, Section 5.2.1], the variance of the U-statistic (A.6) with kernel (4.42) is

$$\mathbb{V}[\widehat{\mathrm{HSIC}}_u] = \binom{m}{4}^{-1} \sum_{c=1}^{4} \binom{4}{c} \binom{m-4}{4-c} \zeta_c, \tag{A.7}$$

where we need consider only

$$\zeta_1 = \mathbb{E}_{jqr} \left[(\mathbb{E}_i h_{ijqr})^2\right], \quad \text{and} \quad \zeta_2 = \mathbb{E}_{ij} \left[(\mathbb{E}_{qr} h_{ijqr})^2\right], \tag{A.8}$$

since the remaining terms decay faster than ζ_2, and can be neglected. Using degeneracy, $\zeta_1 = 0$, we can write the variance as

$$\mathbb{V}[\widehat{\mathrm{HSIC}}_u] = \frac{72(n-4)(n-5)}{n(n-1)(n-2)(n-3)} \zeta_2 + O(m^{-3}). \tag{A.9}$$

Note that direct computation of the variance costs $\mathrm{O}(m^4)$, and is not practical.

We now describe how to compute the variance under H_0 efficiently. We will require additional notation to make the expressions readable. Let $\mathbb{E}_{\mathbf{y}} l_i := \mathbb{E}_{\mathbf{y}} l(\mathbf{y}_i, \mathbf{y})$ and $\mathbb{E}_{\mathbf{yy}'} l := \mathbb{E}_{\mathbf{yy}'} l(\mathbf{y}, \mathbf{y}')$. Using that $\mathbf{x}$ and $\mathbf{y}$ are independent, we get

$$\begin{aligned} 12\mathbb{E}_{qr} h_{ijqr} &= k_{ij} \left(2l_{ij} + 2\mathbb{E}_{\mathbf{yy}'} l - 2\mathbb{E}_{\mathbf{y}} l_i - 2\mathbb{E}_{\mathbf{y}} l_j\right) \\ &\quad - 2\mathbb{E}_{\mathbf{x}} k_i \left(l_{ij} + \mathbb{E}_{\mathbf{yy}'} l - \mathbb{E}_{\mathbf{y}} l_i - \mathbb{E}_{\mathbf{y}} l_j\right) \\ &\quad - 2\mathbb{E}_{\mathbf{x}} k_j \left(l_{ij} + \mathbb{E}_{\mathbf{yy}'} l - \mathbb{E}_{\mathbf{y}} l_i - \mathbb{E}_{\mathbf{y}} l_j\right) \\ &\quad + \mathbb{E}_{\mathbf{xx}'} k \left(2l_{ij} + 2\mathbb{E}_{\mathbf{yy}'} l - 2\mathbb{E}_{\mathbf{y}} l_i - 2\mathbb{E}_{\mathbf{y}} l_j\right), \end{aligned} \tag{A.10}$$

or simplifying further,

$$\mathbb{E}_{qr} h_{ijqr} = \frac{1}{6}\left(k_{ij} + \mathbb{E}_{\mathbf{x}\mathbf{x}'}k - \mathbb{E}_{\mathbf{x}}k_i - \mathbb{E}_{\mathbf{x}}k_j\right)\left(l_{ij} + \mathbb{E}_{\mathbf{y}\mathbf{y}'}l - \mathbb{E}_{\mathbf{y}}l_i - \mathbb{E}_{\mathbf{y}}l_j\right). \tag{A.11}$$

Again under the assumption that $\mathbf{x}$ and $\mathbf{y}$ are independent, the expectation $\mathbb{E}_{ij}\left(\mathbb{E}_{qr}h_{ijqr}\right)^2$ factorizes into a product of two expectations, the first of the form

$$\begin{aligned} &\mathbb{E}_{ij}\left(k_{ij} + \mathbb{E}_{\mathbf{x}\mathbf{x}'}k - \mathbb{E}_{\mathbf{x}}k_i - \mathbb{E}_{\mathbf{x}}k_j\right)^2 \\ &= \mathbb{E}_{ij}\left\langle k(\mathbf{x}_i,\cdot) - \mu_{\mathbf{x}}, k(\mathbf{x}_j,\cdot) - \mu_{\mathbf{x}}\right\rangle_{\mathcal{F}}^2 \\ &= \mathbb{E}_{ij}\left\langle \left(k(\mathbf{x}_i,\cdot) - \mu_{\mathbf{x}}\right) \otimes \left(k(\mathbf{x}_i,\cdot) - \mu_{\mathbf{x}}\right), \left(k(\mathbf{x}_j,\cdot) - \mu_{\mathbf{x}}\right) \otimes \left(k(\mathbf{x}_j,\cdot) - \mu_{\mathbf{x}}\right)\right\rangle_{\mathcal{F}} \\ &= \left\|\mathcal{C}_{\mathbf{x}\mathbf{x}}\right\|_{\mathrm{HS}}^2, \end{aligned} \tag{A.12}$$

and the second $\left\|\mathcal{C}_{\mathbf{y}\mathbf{y}}\right\|_{\mathrm{HS}}^2$ by analogy. Combining these expressions, we obtain

$$\mathbb{V}[\widehat{\mathrm{HSIC}}_u] = \frac{2(m-4)(m-5)}{(m)_4}\left\|\mathcal{C}_{\mathbf{x}\mathbf{x}}\right\|_{\mathrm{HS}}^2\left\|\mathcal{C}_{\mathbf{y}\mathbf{y}}\right\|_{\mathrm{HS}}^2 + O(m^{-3}). \tag{A.13}$$

The variance of $\widehat{\mathrm{HSIC}}_b$ is identical, since the additional terms that arise from the bias vanish faster than the leading terms retained in (A.13). For the empirical estimate it is slightly more efficient not to compute the variance as the product of the empirical HS norms; rather, denoting by $\circ$ the element wise matrix product, by $\mathbf{A}^{\cdot 2}$ the element wise matrix power, and $\mathbf{B} = \left((\mathbf{HKH}) \circ (\mathbf{HLH})\right)^{\cdot 2}$, then our empirical estimate of the product of HS norms can be computed as $\mathbf{1}^\top\left(\mathbf{B} - \mathrm{diag}(\mathbf{B})\right)\mathbf{1}$, where the bias is of the same order as the terms we neglected in (A.13).