Learning Representations via Kernel Dependence Measure

A Dissertation Presented

by

Chieh Wu

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Northeastern University

Boston, Massachusetts

April 2020

To my wife Kathia

Contents

List of Figures

List of Tables

Acknowledgments

This journey has been the greatest adventure of my life. I would not have learned so much without the help of my family, friends, and advisors. I would like to especially acknowledge the support of my wife. While I have been constantly absent from the demand for research, she has alone carried the burden of running a family while sacrificing her own career. This work is only possible because of her.

Additionally, I would like to thank my main advisor Jennifer Dy and my two other co-advisors Stratis Ioannidis and Mario Sznaier. Although this journey has been difficult and soul-crushing at times, you have done the best to walk beside me. I may not have been the most promising traveling companion, yet, you have continued to have faith in me. Lastly, I would like to thank my friends in the lab: Jared, Yale, Khan, and Aria. You have reached out and pulled me up when I could not climb higher. It has been an honor to work with you.

Lastly, we would like to acknowledge support for this project from the NSF grant IIS-1546428.

Abstract of the Dissertation

Learning Representations via Kernel Dependence Measure

by

Chieh Wu

Doctor of Philosophy in Electrical and Computer Engineering

Northeastern University, April 2020

Jennifer Dy, Adviser

In this thesis, we investigate learning feature representations given input data for various downstream applications. We leverage *feature map* learning for two key applications in Machine Learning: clustering, and classification. Specifically, we solve a class of problems which is referred to as the *Interpretable Kernel Dimension Reduction* (IKDR) problems. While traditional Kernel dimensionality reduction (KDR) algorithms first map the data into a high dimensional feature space using kernels prior to a projection onto a low dimensional space. Interpretable KDR (IKDR) is different in that it projects onto a subspace *before* the kernel feature mapping, therefore, the projection matrix can indicate how the original features linearly combine to form the new features. This work demonstrates how IKDR problems can be solved and used for applications in supervised classification, unsupervised clustering, and semi-supervised Alternative Clustering. Specifically, we discovered a common formulation among these problems, and propose the Iterative Spectral Method (ISM) to solve this class of nonlinear optimization problems on a Stiefel manifold. We provide theoretical guarantees for the solution of ISM at convergence where both the 1st and 2nd order optimality conditions are satisfied. We generalize the type of kernels usable by ISM by defining the ISM family of kernels and identified that a conic combination of ISM kernels remains within the family of ISM kernels.

Our work with IKDR and ISM inspired us to design a multi-layered neural network by combining a chain of IKDR layers. This network representation allows us to design a greedy strategy to train a deep network for multi-class classification, where each layer is defined as a composition of a linear projection and a nonlinear mapping. This nonlinear mapping is defined as the feature map of a Gaussian kernel, and the linear projection is learned by maximizing the dependence between

the layer output and the labels, using the Hilbert Schmidt Independence Criterion (HSIC) as the dependence measure. Since each layer is trained greedily in sequence, all learning is local, and neither backpropagation nor even gradient descent is needed. The depth and width of the network are determined via natural guidelines, and the procedure regularizes its weights in the linear layer. As the key theoretical result, the function class represented by the network is proved to be sufficiently rich to learn any dataset labeling using a finite number of layers, in the sense of reaching minimum mean-squared error or cross-entropy, as long as no two data points with different labels coincide. Experiments demonstrate good generalization performance of the greedy approach across multiple benchmarks while showing a significant computational advantage against a multilayer perceptron of the same complexity trained globally by backpropagation.

Chapter 1

Introduction

Machine Learning enables algorithms to learn and identify patterns from data of various forms, e.g., images, text, and numbers. This thesis focuses on two specific types of learning: classification and clustering. In classification, predictive models are trained using examples of the input samples and output labels. By looking at samples from a training set, classifiers learn to make labeling predictions even for samples it has not previously seen. For example, with self-driving cars, an input may be a detection of a person crossing the street and the appropriate labeling would call for the car to slow down and stop. While the person may not be the same person like the ones in training, the goal of a classifier is to generalize the learning to novel datasets previously unseen. Since classification use labels as supervision to guide its learning, it is a type of *supervised* learning. Alternatively, the labels may not be available and learning must be conducted *unsupervised*. In these cases, we often wish to group data into clusters based on some notion of similarity: we refer to this type of algorithm as clustering.

The input samples in a learning problem is characterized by a set of original input features. It can be the raw pixels of an image or features specified by a domain expert, such as clinical variables, deemed important for predicting the output labels. However, not all these original input feature representation may be important for prediction. In this thesis, we learn to identify the mapping that generates alternative feature representations appropriate for various downstream learning tasks.

The most important information for a given dataset often lies in a low dimensional space [1, 2, 3, 4]. The first part of the thesis deals with dimensionality reduction. The goal of dimensionality reduction

algorithms is to find a low-dimensional feature representation for various machine learning tasks. Due to the ability of kernel dependence measures for capturing both linear and nonlinear relationships, they are powerful criteria for nonlinear dimensionality reduction (DR) [5, 6]. The standard approach is to first map the data into a high dimensional feature space prior to a projection onto a low dimensional space [7]. This approach has been preferred because it captures the nonlinear relationship with an established solution, i.e., the most dominant eigenvectors of the kernel matrix. However, since the high dimensional feature space maps the original features nonlinearly, it is no longer interpretable. Alternatively, if the projection onto a subspace precedes the feature mapping, the projection matrix can be obtained to inform how the original features linearly combine to form the new features. Exploiting this insight, many formulations have leveraged kernel alignment or Hilbert Schmidt Independence Criterion (HSIC) [8] to model this approach [5, 6, 9, 10, 11, 12, 13, 14, 15, 16]. Together, we refer to these approaches as *Interpretable Kernel Dimension Reduction* (IKDR). Unfortunately, this formulation can no longer be solved via eigendecomposition, instead, it becomes a highly non-convex manifold optimization that is computationally expensive.

A major contribution of this thesis is that we propose an efficient *Iterative Spectral (eigendecomposition) Method* (ISM), a novel algorithm for solving the non-convex optimization constrained on a Stiefel manifold problem inherent in many IKDR problems. Our algorithm has several highly desirable properties. First, it *significantly outperforms* traditional methods such as dimension growth [12] and gradient descent on a Stiefel manifold [17] in terms of both computation time and quality of the produced alternative clustering. Second, the algorithm relies on an *intuitive use of iterative spectral decompositions*, making it both easy to understand as well as easy to implement, using off-the-shelf libraries. Third, ISM has a natural initialization, constructed through a Taylor approximation of the problem's Lagrangian. Therefore, high quality results can be obtained without random restarts in search of a better initialization. Finally, we provide *theoretical guarantees* on its fixed point. In particular, we establish conditions under which the fixed point of ISM satisfies both the 1st and 2nd order necessary conditions for local optimality.

We initially investigated the theoretical guarantees of this algorithm specific to the Gaussian kernel. We later additionally expand ISM's theoretical guarantees to an entire family of kernels (we call the *ISM family*), thereby empowering ISM to solve any kernel method with a similar objective in this ISM family. In identifying this family, we prove that each kernel within the ISM family has a surrogate Φ matrix and the optimal projection is formed by its most dominant eigenvectors. With this

extension, we establish how a wide range of IKDR applications across different learning paradigms can be solved by ISM.

Empowered by extending ISM's theoretical guarantees to other kernels, we present ISM as a solution to IKDR problems across several learning paradigms, including supervised DR [6, 10, 11], unsupervised DR [5, 12], semi-supervised DR[13, 14], and alternative clustering [9, 15, 18]. Indeed, we demonstrate how many of these applications can be reformulated into an identical optimization objective which ISM solves, implying a significant role for ISM that has been previously unknown.

The first part of this thesis focuses on low-dimensional feature representations with shallow models. In the second part of this thesis, we explore deep feature representations with kernels for clustering and for classification.

For clustering, we learn deep feature representations with HSIC. Specifically, we propose a deep learning adaptation for discovering kernels tailored to identifying clusters. While traditional clustering algorithms cannot cluster highly non-convex clustering structures, we overcome this challenge by finding a mapping where the samples become linearly separable. We further experimentally show that our approach outperforms several state-of-the-art deep clustering methods, as well as traditional approaches such as k-means and spectral clustering over a broad array of real and synthetic datasets.

For classification, we designed a deep neural network by using IKDR formulation to model each layer. This perspective allows us to introduce a new function class for supervised classification, which builds a kernel neural network in a greedy layer-wise fashion: we call this the *Kernel Dependence Network* (*KNet*). Each layer is a composition of a linear subspace projection and an infinite-dimensional feature map. A key advantage is that the network can be trained greedily, layer by layer. *KNet* solves each layer by maximizing the dependence between the layer output and the labels, using the Hilbert Schmidt Independence Criterion (HSIC) as the dependence measure [8]. This can be achieved efficiently, even without Stochastic Gradient Descent (SGD), via the Iterative Spectral Method (ISM) [9]. This local and greedy strategy is used in place of backpropagation (BP) to obviate the sharing of the gradient information throughout the network, thereby avoiding exploding/vanishing gradients. As a consequence of our formulation, we demonstrate how the natural choice of the network width and depth also emerges.

Another major contribution of this thesis is the theoretical characterization of the richness of the function class described by our architecture. We prove a property related to that of *finite sample*

expressivity [19, Section 4] for classical neural nets. Specifically, our network construction can achieve any desirable *training accuracy* given a minimum depth of 2 infinitely wide layers, with the implication of minimizing Mean Squared Error (MSE) and Cross-Entropy (CE) on the training data. Similar to traditional MLPs, the richness of *KNet* promises to fit any training data with an overparameterized network, yet it also experimentally generalizes well on test data. To explain this observation for standard MLPs, the regularizing effects of architecture choice and optimization strategies have been used to explore how MLPs generalize [20, 21, 22]. We demonstrate that our architecture and training procedure perform an implicit regularization, similar to the weight penalization arguments of Poggio et al. [23], and we describe the mechanisms by which this is achieved.

Organization of This Thesis.

The rest of this thesis is organized as follows. We start by providing a review of the necessary background to understand the work in Chapter 2. Instead of having a separate chapter on related work, we integrated the related work section into the introduction of each Chapter. This allows us to introduce the core contributions of each idea while constrasting against past work. We first invented ISM as an efficient solution to alternative clustering; thus, in Chapter 3, we introduce Alternative Clustering via HSIC and discuss how the formulation can be solved efficiently with ISM. Chapter 3 utilizes HSIC as a criterion for clustering but with a shallow model; in Chapter 4, we develop an HSIC clustering utilizing highly flexible deep neural network representations to perform deep kernel clustering (we call KNet Clustering). We realized that ISM is a general algorithm that can be applied to more general settings beyond alternative clustering. In Chapter 5, we provide a unifying formulation that can be used to solve Supervised, Unsupervised and Semi-supervised problems: we refer to this unifying objective as the Interpretable Kernel Dimension Reduction (IKDR). We then show how ISM can be utlized to solved IKDR problems. In Chapter 3, we only provided convergence guarantees of ISM for Gaussian kernels. We further extend the convergence guarantees of ISM to a family of kernels (we call the ISM family of kernels) in Chapter 5. In Chapter 6, we further build on the IKDR formulation to deep representations by showing how one can design a deep neural network by using the IKDR formulation to model each layer and investigate its relationship to modern Multi-layered Perceptrons (MLPs). This perspective allows us to introduce a new function class for supervised classification, which builds a kernel neural network in a greedy layer-wise fashion that can be solved via ISM layer-wise instead of gradient descent backpropagation: we call this the *Kernel Dependence Network* (*KNet*). Finally, we provide the conclusions of this thesis in Chapter 7.

Chapter 2

Background

This chapter is organized to provide the necessary background required to understand our thesis. Since most of our work is related to the Reproducing Kernel Hilbert Space (RKHS), it is necessary to understand its fundamental definition and properties. We further help to understand RKHS by putting it within the context of spaces and why they are necessary.

A space defines a set of elements with a certain structure. From our frame of mind, it is perhaps the easiest to understand the 3D Euclidean space. Since we live this in space, we are familiar with the Cartesian coordinate and understand the concept of *distance*. We often take it for granted that there are distance between element in our world, yet this is not necessarily true for a general sets of objects. For example, if we compare the health of two patients, how do we define a distance based on their health?

The Reproducing Kernel Hilbert Space (RKHS) is an example of a space where certain relationships are assumed. First, it is a space inside other spaces, and therefore, it automatically inherits the structure of those space. The purpose of this chapter is to build up the basic building blocks necessary to understand the structure of RKHS, i.e., we must properly defines a set of concepts. As a preview, Fig. 2.1 summarizes the spaces which RKHS lives within.

Many of our applications require a comparison of our results to the ground truth. We lastly review the concept of the Normalized Mutual Information (NMI); a value between 0 and 1 that compares the accuracy of our result against the true labels.

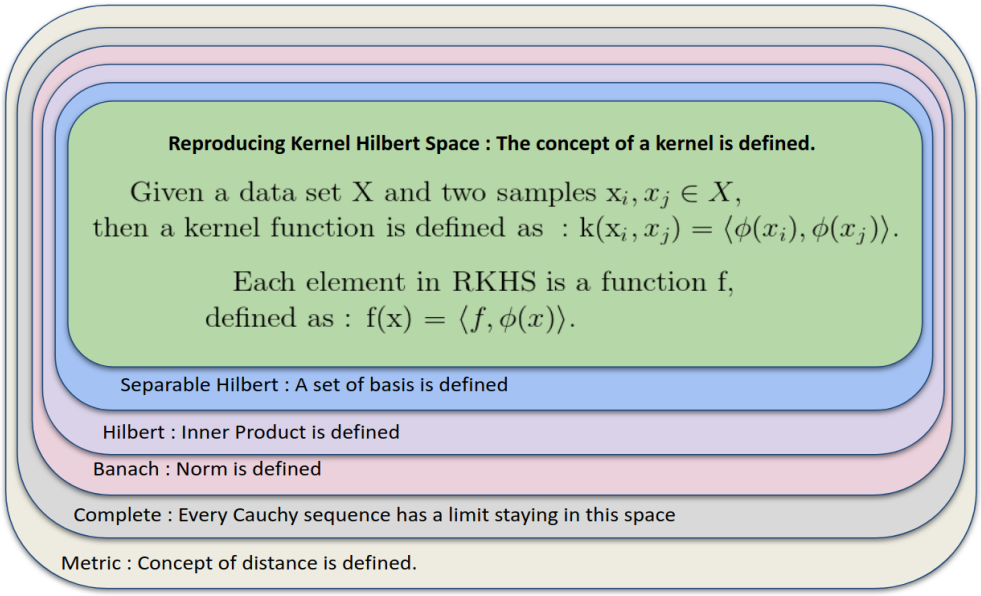

Figure 2.1: A figure of major spaces with their properties.

2.1 The Norm and the Inner Product

The elements of a set are commonly represented as a vector, i.e., a column of numbers. We refer to space where the elements are represented as vectors as the *vector space*. Two fundamental operations are commonly applied to the elements of a vectors space, the norm and the inner product space.

The idea of a norm allows us to define the size of an element. Specifically, it is defined as

Definition 1. *Given a vector space V), $a \in \mathbb{R}$ and $u, v \in V$, a norm on V is a nonnegative-valued function $p : V \to \mathbb{R}$ with the following properties*

1. $p(u + v) \leq p(u) + p(v)$
2. $p(av) = |a|p(v)$
3. $p(v) = 0 \implies v = 0$

By following the most basic criteria of a norm, different types of norm can be defined. By picking a norm for the elements of a space, we turn a generic space into a *norm space*. That is, a space where the concept of a norm is define.

While norms allow us to measure the size of an element in a set, the idea of an *inner product* allows to define the concept of similarity, i.e., a larger inner product usually implies a higher similarity given a constant scale. To be more precise, the inner product is defined as

Definition 2. *Let $\mathcal{H}$ be a vector space over $\mathbb{R}$, a function $\langle ., . \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is said to be an inner product on $\mathcal{H}$ if*

1. $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{H}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{H}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{H}}$
2. $\langle f, g \rangle = \langle g, f \rangle$
3. $\langle f, f \rangle \geq 0 : \langle f, f \rangle = 0 \iff f = 0$

A space where the concept of an inner product is defined is called the *inner produce space*. Within this space, we can then define the concept of a norm using the inner product operation. Namely, given f as an element in the vector space $\mathcal{H}$, norm can be defined as

$$||f||_{\mathcal{H}} = \langle f, f \rangle_{\mathcal{H}}. \tag{2.1}$$

2.2 Convergence, Optimality Conditions and the Cauchy Sequence

The optimal solution of an equation may not have a closed formed solution. Therefore, in practice we designed algorithms that allow us to incrementally get closer to the optimal solution. In other words, we want an algorithm to generate a sequence of values that leads to a good solution. In a mathematical sense, this strategy necessitates two key additional concepts, convergence and optimality. For convergence, it implies that the algorithm generates a sequence of values that leads toward and get stuck on a particular value f^*. However, since getting stuck on a bad solution is not useful, the optimality condition implies that the sequence is not stuck at any f^*, but an f^* that provides a desirable outcome. To be more precise, convergence is defined as the following.

Definition 3. *A sequence $\{S_n\}_{n=1}^{\infty}$ converges to v, if there exists a number l such that for any $\epsilon > 0$, there exists a N such that for all $n > N$*

$$|S_n - v| < \epsilon. \tag{2.2}$$

The concept of convergence is useful for analyzing an algorithm because it guarantees an eventual result when the algorithm reaches f^*. To determine the desirability of f^*, we use the concept of optimality. Namely, if we view our optimizing objective $\mathcal{L}$ as a measurement of desirability, we wish to find f^* where $\mathcal{L}$ is at the highest or the lowest point. For the purpose of this thesis, we formally define the concept of optimality provided by Wright and Nocedal [24].

Theorem 2.1 (Nocedal,Wright, Theorem 12.5 [24])**.** *(2nd Order Necessary Conditions) Consider the optimization problem:* $\min_{W:h(W)=0} f(W)$*, where* $f : \mathbb{R}^{d\times q} \to \mathbb{R}$ *and* $h : \mathcal{R}^{d\times q} \to \mathbb{R}^{q\times q}$ *are twice continuously differentiable. Let* $\mathcal{L}$ *be the Lagrangian and* $h(W)$ *its equality constraint. Then, a local minimum must satisfy the following conditions:*

$$\nabla_W \mathcal{L}(W^*, \Lambda^*) = 0, \tag{2.3a}$$

$$\nabla_\Lambda \mathcal{L}(W^*, \Lambda^*) = 0, \tag{2.3b}$$

$$\begin{aligned} &\mathrm{Tr}(Z^T \nabla^2_{WW} \mathcal{L}(W^*, \Lambda^*) Z) \geq 0 \\ &\quad \text{for all } Z \neq 0, \text{with } \nabla h(W^*)^T Z = 0. \end{aligned} \tag{2.3c}$$

Given Def. 3 and Thm. 2.1, algorithms can now be formally evaluated for its effectiveness. However, in practice, Def. 3 proved to be difficult to satisfy, at least not in the most general case. To overcome this challenge, another concept of convergence emerges: one where the analysis is greatly simplified. This is why Cauchy Sequences are important: it is significantly easier to prove that a sequence is a Cauchy Sequence rather than proving convergence. Formally, Cauchy Sequences are defined as

Definition 4. *A sequence is called a Cauchy sequence if the terms of the sequence eventually all become arbitrarily close to one another. That is, given* $\epsilon > 0$ *there exists* N *such that if* $m, n > N$ *then* $|a_m - a_n| < \epsilon$*.*

However, while the definition of the Cauchy sequences intuitively implies a converging sequence, convergence is actually not automatically guaranteed. Therefore, the conditions of the Cauchy sequence may be easier to prove, but not suitable for all conditions. Yet, a particular tool does not have to solve all problems to be useful. Therefore, as long as Cauchy sequences can be used in the cases we care about, it doesn't matter that it can't be used for all cases. It turns out that the space where a Cauchy sequences always converges is sufficiently large, and therefore, it is currently used for the majority of the cases. That is, in practice, we evaluate an algorithm's ability to generate a

desirable sequence by determining if it is generating a Cauchy sequence with a fixed point at an optimal location. Since the space where Cauchy sequences always converge is so commonly used, it is named the *Complete Space*.

2.3 Banach, Hilbert, and the Reproducing Kernel Hilbert Space

As mentioned by the previous section, spaces are simply sets of elements where a particular structure and relationship is defined. Once these relationships are defined, we can start defining the union and intersection of spaces. For example, we can now define a space that is simultaneously a norm and an inner product space. For example, by defining the norm squared as the inner product of a value with itself, we have just created a space that is simultaneously a norm and an inner product space. Building on this idea, it allows us to define the Banach and Hilbert space.

Definition 5. *A Banach space is a complete norm space.*

Definition 6. *A Hilbert space is a complete inner product space.*

A Reproducing Kernel Hilbert Space (RKHS) is a space that is simultaneously inside a Banach and an Hilbert Space. Therefore, the concept of a norm and inner product is well defined. Specifically, let $u, v \in \mathbb{R}^d$ and let u_i be the $i_t h$ element of the vector, then inner product is defined as

$$\langle u, v \rangle = \sum_{i}^{d} u_i v_i. \tag{2.4}$$

Following the definition of an inner product, a norm of v is simply

$$|v| = \sqrt{\langle v, v \rangle}. \tag{2.5}$$

While the concept of norm and inner product are useful, RKHS is special in its additional definition of a kernel.

Definition 7. *Let $\mathcal{H}$ be a Hilbert space and let $\mathcal{X}$ be an non-empty set, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel if there exists a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that for all $x, x' \in \mathcal{X}$*

$$k(x, x') := \langle \phi(x), \phi(x') \rangle. \tag{2.6}$$

Leveraging the definition of a kernel, we can now properly define a RKHS.

Definition 8. *Given a set $\mathcal{X}$, a subset of a Hilbert Space $\hat{\mathcal{H}} \subset \mathcal{H}$, and a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where its associated feature mapping is $\phi : \mathcal{X} \to \hat{\mathcal{H}}$, then $\hat{\mathcal{H}}$ is a Reproducing Kernel Hilbert Space if k satisfies*

1. $\forall x \in \mathcal{X}, k(., x) \in \hat{\mathcal{H}}$
2. $\forall x \in \mathcal{X}, \forall f \in \hat{\mathcal{H}}, f(x) = \langle f, \phi(x) \rangle$

2.4 RKHS and the Hilbert Schmidt Independence Criterion

The significance of the kernel function is not immediately apparent unless one realizes that the features in RKHS can potentially have infinite dimensions. This implies that when data is projected into RKHS, we are able to examine the same data in an extremely high resolution, i.e., relationships that are not apparent in its original representation emerges with this representation. However, given this level of model flexibility, it also raises feasibility concerns since modern computers cannot possibly handle infinite features. This is where the properties of a kernel function highlights its advantage.

Given a data $X \in \mathbb{R}^{n \times d}$ where n is the number of samples and d the number of features. We let Φ be a function that applies ϕ on each sample inside X. Then we can denote the resulting feature map as $\Phi(X) \in \mathbb{R}^{n \times q}$. Here, q is the dimension of the new feature in RKHS and it can range from 1 to ∞. It would be impossible for a computer to store all the features of this new representation. However, if an algorithm can be reformulated such that every $\Phi(X)$ is paired with $\Phi(X)^T$, i.e., $\Phi(X)\Phi(X)^T$, then we can replace $\Phi(X)\Phi(X)^T$ directly with K. Therefore, instead of working with matrices of dimension $\mathbb{R}^{n \times \infty}$, we can work with K matrices of size $\mathbb{R}^{n \times n}$, a huge reduction of complexity. This trick of performing machine learning operations on the kernel matrix instead of the feature maps is called the kernel trick, and it is reason why we have chosen to work with RKHS. We are able to map data to have infinite features, inheriting its flexibility, without incurring the computational cost.

Once the data is projected into RKHS, given the kernel trick, various machine learning objectives can be used in this space. For our thesis, we notice that given a dataset X and its labels Y, we can discover ϕ by finding the ϕ that maximizing the dependence between $\Phi(X)$ and $\Phi(Y)$. This dependence can be computed in RKHS by finding the norm of the cross covariance between $\Phi(X)$

and $\Phi(Y)$. Specifically, the cross covariance has the equation

$$C_{x,y} = E_{x,y}[(\phi(x) - \bar{\phi(x)}) \otimes (\phi(y) - \bar{\phi(y)})]. \tag{2.7}$$

Note that here, $\bar{\phi(x)}$ denotes the mean $\phi(x)$. Instead of working with the entire matrix, we summarize the matrix by looking at its norm. Together the objective is the Cross Covariance Norm (CCN) where

$$\text{CCN}(X, Y) = ||C_{x,y}||^2_{\text{HS}}. \tag{2.8}$$

The HS in $||.||_{\text{HS}}$ denotes the Hilbert-Schmidt Norm, however, for our purpose, we can simply replace this norm with the Frobenius norm. By following the kernel trick previously mentioned, when this objective is approximated empirically, it can be greatly simplified into an objective with only kernel matrices. This new objective, as proposed by Gretton et al. [8], is refered to as the Hilbert-Schmidt Independence Criterion (HSIC) with the equation

$$\text{HSIC}(X, Y) = (n-1)^{-2}\text{Tr}(K_X H K_Y H). \tag{2.9}$$

By looking at Eq. (2.7), if ϕ is capable of capturing the non-linear information of the data, the dependence between X and Y should remain high after being mapped into RKHS. Therefore, ϕ can be discovered by solving the objective

$$\max_{\phi} \quad \text{Tr}(K_X H K_Y H). \tag{2.10}$$

2.5 Normalized Mutual Information

Mutual information (MI) measures the non-linear statistical dependency between two random variables. Given two random variables X and Y, MI is defined as the KL divergence between their joint distribution against the product of their marginal, i.e.,

$$MI(X, Y) = KL(P(X, Y), P(X)P(Y)) = \int\int p(x, y)\text{log}\frac{p(x, y)}{p(x)p(y)}dxdy. \tag{2.11}$$

MI can also be viewed from the perspective of Entropy. Let Entropy (H) of a random variable X be defined as

$$H(X) = -\int p(x)\text{log}p(x)dx. \tag{2.12}$$

For joint distributions, the Entropy of X and Y is defined as

$$H(X,Y) = -\int\int p(x,y)\text{log}p(x,y)dxdy. \tag{2.13}$$

Then, MI can be reformulated into

$$MI(X,Y) = H(X) + H(Y) - H(X,Y). \tag{2.14}$$

MI can be used to measure the dependency between the predicted labels of our clustering results against the true clustering result. However, while MI is always positive, it does not have a fixed max value. This property makes t difficult to compare the MI between different datasets. To circumvent this challenge, a normalized version of MI is commonly used. Specifically, if we let U and L be two clustering assignments, NMI can be calculated with

$$NMI(L,U) = \frac{I(L,U)}{\sqrt{H(L)H(U)}}. \tag{2.15}$$

Since the Normalized MI is scaled to betwee 0 and 1, with 1 as maximum dependency, it becomes a convenient method to approximate the accuracy of our clustering labels against the ground truth.

Chapter 3

Alternative Clustering

Clustering, i.e., the process of grouping similar objects in a dataset together, is a classic problem. It is extensively used for exploratory data analysis. Traditional clustering algorithms typically identify a single partitioning of a given dataset. However, data is often multi-faceted and can be both interpreted and clustered through multiple viewpoints (or, *views*). For example, the same face data can be clustered based on either identity or based on pose. In real applications, partitions generated by a clustering algorithm may not correspond to the view a user is interested in.

In this chapter, we address the problem of finding an *alternative clustering*, given a dataset and an existing, pre-computed clustering. Ideally, one would like the alternative clustering to be *novel* (i.e., non-redundant) w.r.t. the existing clustering to reveal a new viewpoint to the user. Simultaneously, one would like the result to reveal partitions of high clustering *quality*. Several recent papers propose algorithms for alternative clustering [18, 25, 26, 27, 28, 29]. Among them, Kernel Dimension Alternative Clustering (KDAC) is a flexible approach, shown to have superior performance compared to several competitors [18]. KDAC is as powerful as spectral clustering in discovering arbitrarily-shaped clusters (including ones that are not linearly separable) that are non-redundant w.r.t. an existing clustering. As an additional advantage, KDAC can simultaneously learn the subspace in which the alternative clustering resides.

The flexibility of KDAC comes at a price: the KDAC formulation involves optimizing a non-convex cost function constrained over the space of orthogonal matrices (i.e, the Stiefel manifold). Niu et al. [18] proposed a Dimension Growth (DG) heuristic for solving this optimization problem, which

is nevertheless highly computationally intensive. We elaborate on its complexity in Section 3.1; experimentally, DG is quite slow, with a convergence time of roughly 46 hours on an Intel Xeon CPU, for a 624 sample-sized face data (c.f. Section 3.5). This limits the applicability of KDAC in interactive exploratory data analysis settings, which often require results to be presented to a user within a few seconds. It also limits the scalability of KDAC to large data. Alternately, one can solve the KDAC optimization problem by gradient descent on a Stiefel manifold (SM) [17]. However, given the lack of convexity, both DG or SM are prone to get trapped to local minima. Multiple iterations with random initializations are required to ameliorate the effect of locality. This increases computation time, and also decreases in effectiveness as the dimensionality of the data increases: the increase in dimension rapidly expands the search space and the abundance of local minima. As such, with both DG and SM, the clustering quality is negatively affected by an increase in dimension.

Our Contributions. Motivated by the above issues, we make the following contributions:

- We propose an Iterative Spectral Method (ISM), a *novel algorithm* for solving the non-convex optimization constrained on a Stiefel manifold problem inherent in KDAC. Our algorithm has several highly desirable properties. First, it *significantly outperforms* traditional methods such as DG and SM in terms of both computation time and quality of the produced alternative clustering. Second, the algorithm relies on an *intuitive use of iterative spectral decompositions*, making it both easy to understand as well as easy to implement, using off-the-shelf libraries.
- ISM has a natural initialization, constructed through a Taylor approximation of the problem's Lagrangian. Therefore, high quality results can be obtained without random restarts in search of a better initialization. We show that this initialization is a contribution in its own right, as its use improves performance of competitor algorithms.
- We provide *theoretical guarantees* on its fixed point. In particular, we establish conditions under which the fixed point of ISM satisfies both the 1st and 2nd order necessary conditions for local optimality.
- We extensively evaluate the performance of ISM in solving KDAC with synthetic and real data under various clustering quality and cost measures. Our results show an improvement in execution time by up to a factor of roughly 70 and 10^5, compared to SM and DG, respectively. At the same time, ISM outperforms SM and DG in clustering quality measures along with significantly lower computational cost.

3.1 Kernel Dimension Alternative Clustering (KDAC)

In alternative clustering, a dataset is provided along with existing clustering labels. Given this as input, we seek a *new* clustering that is (a) distinct from the existing clustering, and (b) has high quality with respect to a clustering quality measure. An example illustrating this is shown in Figure 3.1. This dataset comprises 400 points in $\mathcal{R}^4$. Projected to the first two dimensions, the dataset contains two clusters of intertwining parabolas shown as Clustering A. Projected to the last two dimensions, the dataset contains two Gaussian clusters shown as Clustering B. Points clustered together in one view can be in different clusters in the alternative view. In alternative clustering, given (a) the dataset, and (b) one of the two possible clusterings (e.g., Clustering B), we wish to discover the alternative clustering illustrated by the different view.

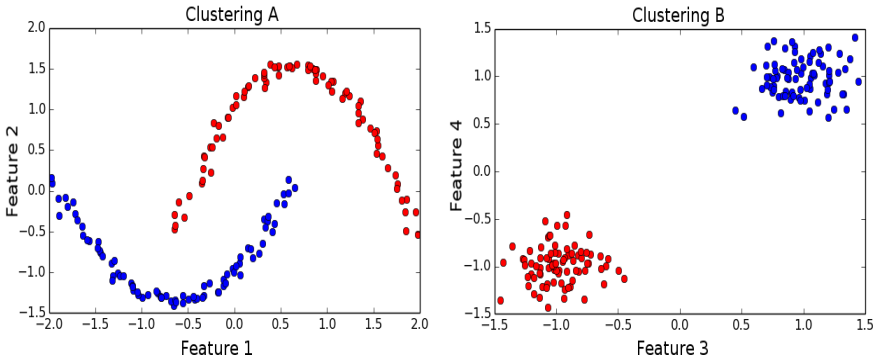

Figure 3.1: Four-dimensional moon dataset. Projection into the first two dimensions reveals different clusters than projection to the latter two dimensions.

Formally, let $X \in \mathcal{R}^{n \times d}$ be a dataset with n samples and d features, along with an existing clustering $Y \in \mathcal{R}^{n \times k}$, where k is the number of clusters. If x_i belongs to cluster j, then $Y_{i,j} = 1$; otherwise, $Y_{i,j} = 0$. We wish to discover an alternative clustering $U \in \mathcal{R}^{n \times k}$ on some lower dimensional subspace of dimension $q \ll d$. Let $W \in \mathcal{R}^{d \times q}$ be a projection matrix such that $XW \in \mathcal{R}^{n \times q}$.

We seek the optimal projection W and clustering U that maximizes the statistical dependence between XW with U, yielding a high clustering quality, while minimizing the dependence between XW and Y, ensuring the novelty of the new clustering. Denoting DM as a Dependence Measure

function, and using λ as a weighing constant, this optimization can be written as:

$$\text{Maximize:} \quad \mathrm{DM}(XW, U) - \lambda\,\mathrm{DM}(XW, Y), \tag{3.1a}$$

$$\text{s.t :} \quad W^T W = I, U^T U = I. \tag{3.1b}$$

As in spectral clustering, the labels of the alternative clustering are retrieved by performing K-means on matrix U, treating its rows as samples. There are many potential choices for DM. The most well-known measures are correlation and mutual information (MI). While correlation performs well in many applications, it lacks the ability to measure non-linear relationships. Although there is clear relationship in Clustering A in Figure 3.1, correlation would mistakenly yield a value of nearly 0. As a dependence measure, MI is superior in that it also measures non-linear relationships. However, due to the probabilistic nature of its formulation, a joint distribution is required. Depending on the distribution, the computation of MI can be prohibitive.

For these reasons, the Hilbert Schmidt Independence Criterion (HSIC) [8] has been proposed for KDAC [18]. Like MI, it captures non-linear relationships. Unlike MI, HSIC does not require estimating a joint distribution, and it relaxes the need to discretize continuous variables. In addition, as shown by Niu et al. [18], HSIC is mathematically equivalent to spectral clustering, further implying that a high HSIC between the data and U yields high clustering quality. A visual comparison of HSIC and correlation can be found in Figure 3.2.

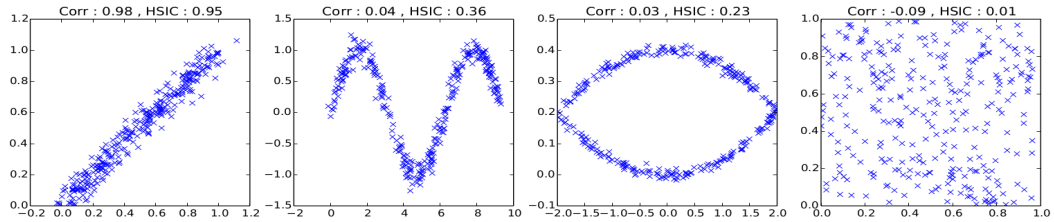

Figure 3.2: The figure demonstrates a visual comparison of HSIC and correlation. It can be seen that HSIC measures non-linear relationships, while correlation does not.

Using HSIC as a dependence measure, the objective of KDAC becomes

$$\text{Maximize:} \quad \mathrm{HSIC}(XW, U) - \lambda\,\mathrm{HSIC}(XW, Y), \tag{3.2a}$$

$$\text{subject to:} \quad W^T W = I, U^T U = I. \tag{3.2b}$$

where $\mathrm{HSIC}(X, Y) \equiv \frac{1}{(n-1)^2}\,\mathrm{Tr}(K_X H K_Y H)$. Here, the variables K_X and K_Y are Gram matrices, and the H matrix is a centering matrix where $H = I - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ with $\mathbf{1}$ the n-sized vector of all

Algorithm 1: KDAC Algorithm

Input : dataset X, original clustering Y

Output : alternative clustering U

Initialize W_0 using W_{init} from (3.12)

Initialize U_0 from original clustering

Initialize D_0 from W and original clustering

while *(U not converged) or (W not converged)* **do**

  Update D

  Update W by solving Equation (3.4)

  Update U by solving Equation (3.3)

Clustering Result ← Apply K-means to U

ones. The elements of K_X and K_Y are calculated by kernel functions $k_X(x_i, x_j)$ and $k_Y(y_i, y_j)$. The kernel functions for Y and U used in KDAC are $K_Y = YY^T$ and $K_U = UU^T$, and the kernel function for XW is the Gaussian $k_{XW}(x_i, x_j) = \exp(-\text{Tr}[(x_i - x_j)^T WW^T(x_i - x_j)]/(2\sigma^2))$. Due to the equivalence of HSIC and spectral clustering, the practice of normalizing the kernel K_{XW} is adopted from spectral clustering by Niu et al. [18]. That is, for K_{XW} the unnormalized Gram matrix, the normalized matrix is defined as $D^{-1/2}K_{XW}D^{-1/2}$ where $D = \text{diag}(\mathbf{1}_n^T K_{XW})$ is a diagonal matrix whose elements are the column-sums of K_{XW}.

KDAC Algorithm. The optimization problem (3.2) is non-convex. The KDAC algorithm solves (3.2) using alternate maximization between the variables U, W and D, updating each while holding the other two fixed. After convergence, motivated by spectral clustering, U is discretized via K-means to provide the alternative clustering. The algorithm proceeds in an iterative fashion, summarized in Algorithm 1. In each iteration, variables D, U, and W are updated as follows:

Updating D: While holding U and W constant, D is computed as $D = \text{diag}(\mathbf{1}_n^T K_{XW})$. Matrix D is subsequently treated as a scaling constant throughout the rest of the iteration.

Updating U: Holding W and D constant and solving for U, (3.2) reduces to :

$$\max_{U:U^TU=I} \text{Tr}(U^T \mathcal{Q} U), \tag{3.3}$$

where $\mathcal{Q} = HD^{-1/2}K_{XW}D^{-1/2}H$. This is precisely spectral clustering [30]: (3.3) can be solved by setting U's columns to the k most dominant eigenvectors of $\mathcal{Q}$, which can be done in $O(n^3)$ time.

Algorithm 2: ISM Algorithm

Input : U,D,X, Y

Output : W^*

Initialize W_0 to the previous value of W in the master loop of KDAC.

while W *not converged* **do**

  $W \leftarrow \text{eig}_{\min}(\Phi(W))$;

Updating W: While holding U and D constant to solve for W, (3.2) reduces to:

$$\text{Minimize:} \quad F(W) = -\sum_{i,j} \gamma_{i,j} e^{-\frac{\text{Tr}[W^T A_{i,j} W]}{2\sigma^2}} \tag{3.4a}$$

$$\text{subject to:} \quad W^T W = I \tag{3.4b}$$

where $\gamma_{i,j}$ are the elements of matrix $\gamma = D^{-1/2} H (UU^T - \lambda YY^T) H D^{-1/2}$, and $A_{i,j} = (x_i - x_j)(x_i - x_j)^T$ (see Appendix A.1 in the supplement for the derivation). This objective, along with a Stiefel Manifold constraint, $W^T W = I$, pose a challenging optimization problem as neither is convex. Niu et al. [18] propose solving (3.4) through an algorithm termed Dimensional Growth (DG). This algorithm solves for W by computing individual columns of W separately through gradient descent (GD). Given a set of computed columns, the next column is computed by GD projected to a subspace orthogonal to the span of computed set. Since DG is based on GD, the computational complexity is dominated by computing the gradient of (3.4a). The latter is given by:

$$\nabla F(W) = \sum_i^n \sum_j^n \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\text{Tr}[W^T A_{i,j} W]}{2\sigma^2}} A_{i,j} W. \tag{3.5}$$

The complexity of DG is $O(t_{DG} n^2 d^2 q)$, where n, d are the dataset size and dimension, respectively, q is the dimension of the subspace of the alternative clustering, and t_{DG} is the number of iterations of gradient descent. The calculation of the gradient contributes the term $O(n^2 d^2 q)$. Although this computation is highly parallelizable, the algorithm still suffers from slow convergence rate. Therefore, t_{DG} often dominates the computation cost.

An alternative approach to optimize (3.4) is through classic methods for performing optimization on the Stiefel Manifold (SM) [17]. The computational complexity of this algorithm is dominated by the computation of the gradient and a matrix inversion with t_{SM} iterations. This yields a complexity of $O(t_{SM} n^2 d^2 + t_{SM} d^3)$ for SM. Finally, as gradient methods applied to a non-convex objective, both SM and DG require multiple executions from random initialization points to find improved local minima. This approach becomes less effective as the dimension d increases.

3.2 An Iterative Spectral Method

The computation of KDAC is dominated by the W updates in Algorithm 1. Instead of using DG or SM to solve the optimization problem for W in KDAC, we propose an Iterative Spectral Method (ISM). Our algorithm is motivated from the following observations. The Lagrangian of (3.4) is:

$$\begin{aligned}\mathcal{L}(W,\Lambda) = &-\textstyle\sum_{i,j}\gamma_{i,j}\exp\left(-\frac{\mathrm{Tr}(W^T A_{i,j} W)}{2\sigma^2}\right)\\ &-\frac{1}{2}\mathrm{Tr}(\Lambda(W^T W - I))\end{aligned} \tag{3.6}$$

Setting $\nabla_W \mathcal{L}(W,\Lambda) = 0$ gives us the equation:

$$\Phi(W)W = W\Lambda, \tag{3.7}$$

where

$$\Phi(W) = \textstyle\sum_{i,j}\frac{\gamma_{i,j}}{\sigma^2}\exp(-\frac{\mathrm{Tr}[W^T A_{i,j} W]}{2\sigma^2})A_{i,j}, \tag{3.8}$$

and Λ is a diagonal matrix. Recall that a feasible W, satisfying (3.4b), is orthonormal. (3.7) is an eigenequation; thus, a stationary point W of the Lagrangian (3.6) comprises of q eigenvectors of $\Phi(W)$ as columns. Motivated by this observation, ISM attempts to find such a W in the following iterative fashion. Let W_0 be an initial matrix. Given W_k at iteration k, the matrix W_{k+1} is computed as:

$$W_{k+1} = \mathrm{eig}_{\min}(\Phi(W_k)), \quad k = 0, 1, 2, \ldots,$$

where the operator $\mathrm{eig}_{\min}(A)$ returns a matrix whose columns are the q eigenvectors corresponding to the smallest eigenvalues of A.

ISM is summarized in Alg. 2. Several important observations are in order. First, the algorithm ensures that W_k, for $k \geq 1$, is feasible, by construction: selected eigenvectors are orthonormal and satisfy (3.4b). Second, it is also easy to see that a fixed point of the algorithm will also be a stationary point of the Lagrangian (3.6) (see also Lemma 3.3). Though it is harder to prove, selecting eigenvectors corresponding to the *smallest* eigenvalues is key: we show that this is precisely the property that relates a fixed point of the algorithm to the local minimum conditions (see Thm. 5.1). Finally, ISM has several computational advantages. For t_{ISM} iterations, the calculation of $\Phi(W)$, and the ensuing eigendecomposition yields a complexity of $O(t_{ISM}(n^2d^2 + d^3))$. Since $q \ll d$, various approximation methods [31][32][33] can be employed to find the few eigenvectors. For example, the Coordinate-wise Power Method[33], approximates the most dominant eigenvalue at

$O(d)$ time, reducing ISM's complexity to $O(t_{ISM}n^2d^2)$. This improvement is further confirmed experimentally (see Figure 3.4). Lastly, t_{ISM} is magnitudes smaller than both t_{DG} and t_{SM}. In general $t_{ISM} < 10$, while $t_{SM} > 50$ and $t_{DG} > 200$.

3.3 Convergence Guarantees

As mentioned above, the selection of the eigenvectors corresponding to the *smallest* eigenvalues of $\Phi(W_k)$ is crucial for the establishment of a stationary point. Namely, we establish the following theorem:

Theorem 3.1. *For large enough σ (satisfying Inequality* (3.10)*), a fixed point W^* of Algorithm 2 satisfies the necessary conditions of a local minimum of* (3.4) *if $\Phi(W^*)$ is full rank.*

Proof. The main body of the proof is organized into a series of lemmas proved in the supplement. Our first auxiliary lemma (from [24]), establishes conditions necessary for a stationary point of the Lagrangian to constitute local minimum.

Lemma 3.1. *[Nocedal,Wright, Theorem 12.5 [24]] (2nd Order Necessary Conditions) Consider the optimization problem: $\min_{W:h(W)=0} f(W)$, where $f : \mathbb{R}^{d\times q} \to \mathbb{R}$ and $h : \mathcal{R}^{d\times q} \to \mathbb{R}^{q\times q}$ are twice continuously differentiable. Let $\mathcal{L}$ be the Lagrangian of this optimization problem. Then, a local minimum must satisfy the following conditions:*

$$\nabla_W \mathcal{L}(W^*, \Lambda^*) = 0, \tag{3.9a}$$

$$\nabla_\Lambda \mathcal{L}(W^*, \Lambda^*) = 0, \tag{3.9b}$$

$$\begin{aligned} &\mathrm{Tr}(Z^T \nabla^2_{WW} \mathcal{L}(W^*, \Lambda^*) Z) \geq 0 \\ &\text{for all } Z \neq 0, \text{with } \nabla h(W^*)^T Z = 0. \end{aligned} \tag{3.9c}$$

Armed with this result, we next characterize the properties of a fixed point of Algorithm 2:

Lemma 3.2. *Let W^* be a fixed point of Algorithm 2. Then it satisfies: $\Phi(W^*)W^* = W^*\Lambda^*$, where $\Lambda^* \in \mathcal{R}^{q\times q}$ is a diagonal matrix containing the q smallest eigenvalues of $\Phi(W^*)$ and ${W^*}^T W^* = I$.*

The proof can be found in App. A.2. Our next result, whose proof is in Appendix A.3, states that a fixed point satisfies the 1st order conditions of Lemma 3.1.

Lemma 3.3. *If W^* is a fixed point and Λ^* is as defined in Lemma 3.2, then W^*, Λ^* satisfy the 1st order conditions (C.45)(3.9b) of Lemma 3.1.*

Our last lemma, whose proof is in Appendix A.4, establishes that a fixed point satisfies the 2nd order conditions of Lemma 3.1, for large enough σ.

Lemma 3.4. *If W^* is a fixed point, Λ^* is as defined in Lemma 3.2, and $\Phi(W^*)$ is full rank, then given a large enough σ (satisfying Inequality* (3.10)*), W^* and Λ^* satisfy the 2nd order condition (C.54) of Lemma 3.1.*

Thm. 5.1 therefore follows. □

Thm. 5.1 is stated in terms of a large enough σ; we can characterize this constraint precisely. In the proof of Lemma 3.4 we establish the following condition on σ:

$$\sigma^2[\min_i(\bar{\Lambda^*}_i) - \max_j(\Lambda^*_j)] \geq \sum_{i,j} \frac{|\gamma_{i,j}|}{\sigma^2} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^*)}{2\sigma^2}} \mathrm{Tr}(A_{i,j}^T A_{ij}). \tag{3.10}$$

Here, Λ^* is the set of q smallest eigenvalues of $\Phi(W)$, and $\bar{\Lambda}^*$ is the set of the remaining eigenvalues. The left-hand side (LHS) of the equation further motivates ISM's choice of eigenvectors corresponding to the q smallest eigenvalues. This selection guarantees that the LHS of the inequality is positive. Therefore, given a large enough σ, Inequality (3.10) and the 2nd order condition is satisfied.

Furthermore, this equation provides a reasonable suggestion for the value of q. Since we wish to maximize the term $(\min_i(\bar{\Lambda^*}_i) - \max_j(\Lambda^*_j))$ to satisfy the inequality, the value q should be set where this gap is maximized. More formally, we will defined

$$\delta_{gap} = \min_i(\bar{\Lambda^*}_i) - \max_j(\Lambda^*_j). \tag{3.11}$$

as the eigengap.

3.4 Spectral Initialization via Taylor Approximation

ISM admits a natural initialization point, constructed via a Taylor approximation of the objective. As we show experimentally in Section 3.5, this initialization is a contribution in its own right: it

improves both clustering quality and convergence time for ISM *as well as* competitor algorithms. To obtain a good initialization, observe that by using the 2nd order Taylor approximation of the objective function (3.4a) at $W = 0$, the Lagrangian can be approximated by

$$\tilde{\mathcal{L}}(W, \Lambda) \approx -\sum_{i,j} \gamma_{i,j} \left(1 - \frac{\operatorname{Tr}(W^T A_{i,j} W)}{2\sigma^2}\right) + \frac{1}{2} \operatorname{Tr}(\Lambda(I - W^T W)).$$

Setting $\nabla_W \tilde{\mathcal{L}}(W, \Lambda) = 0$ reduces the problem into a simple eigendecomposition, namely, the one defined by the system $\left[\sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} A_{i,j}\right] W = W\Lambda$. Hence, the 2nd order Taylor approximation of the original cost objective has a closed form global minimum that can be used as an initialization point, namely:

$$W_{\text{init}} = \operatorname{eig}_{\min}(\textstyle\sum_{i,j} \gamma_{i,j} A_{i,j} / \sigma^2). \tag{3.12}$$

We use this spectral initialization (SI) in the first master iteration of KDAC. In subsequent master iterations, W_0 (the starting point of ISM) is set to be the last value to which ISM converged to previously.

3.5 Alternative Clustering Experimental Results

We experimentally validate the performance of ISM in terms of both speed and clustering quality. The source code is publicly available at github[1]. Because [18] has already performed extensive comparisons of KDAC against other alternative clustering methods, this section will concentrate on comparing ISM to competing models for optimizing KDAC: Dimensional Growth (DG) [18] and gradient descent on the Stiefel Manifold (SM) [17]. SM is a generic approach, while DG is the approach originally proposed to solve KDAC [18].

In addition to introducing a new algorithm for optimizing KDAC, we also proposed an intelligent initialization scheme based on Taylor approximation in Section 3.4, we call Spectral Initialization (SI). We also investigate how SI effects the performance of the various algorithms compared to the standard random initialization (RI).

Datasets. We perform experiments on four synthetic and three real datasets. The synthetic data are displayed in Figure 3.3. Small Gaussian (SG) contains four Gaussian clusters with 40 samples and two

[1] https://github.com/neu-spiral/ISM

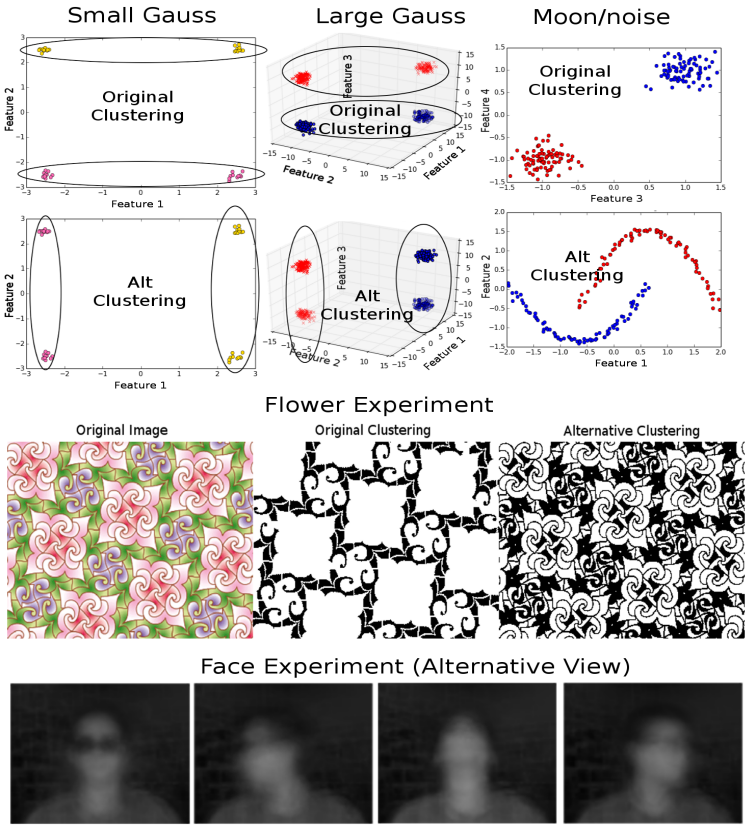

Figure 3.3: Figure includes all original and alternative displayable clusterings. The face data is originally clustered by the identity of the individual. Here the average image of the alternative clustering is displayed. It is observable that the original and alternative clusters are all visually obvious clusters and provide alternative views.

features shown at the top left of the figure. When this data is projected down to feature 2, the original clustering is created. Rotating this projection onto feature 1, the alternative clustering emerges. Large Gaussian (LG) contains four Gaussian clusters with 1000 samples and four dimensions is shown in the top center location. The Gaussian clusters are rotated by 45 degrees to reside in 3D, and the fourth dimension is generated from a uniform noise distribution. This synthetic data is designed to test the response of KDAC within a noisy environment. We generate two additional synthetic datasets shown in top right: Moon and Moon+Noise (MoonN). Both datasets have the first two dimensions as two parabolas and the second two dimensions as Gaussian clusters. The MoonN dataset further includes three noisy dimensions generated from a uniform distribution with 1000 samples. Since the Gaussian clusters have a compact structure and the parabolas have a non-linear

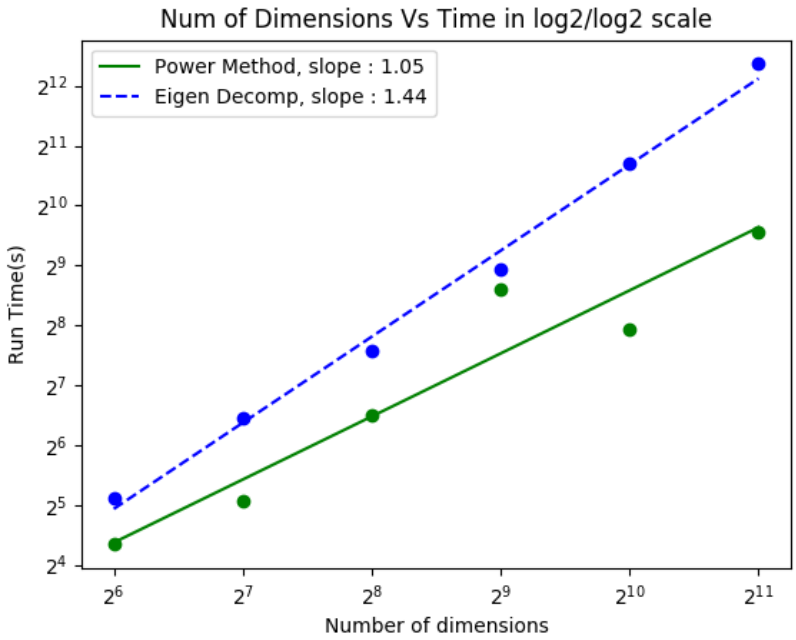

Figure 3.4: Growth in dimension vs time in log/log scale. A slope of 1 is linear growth.

structure, these datasets demonstrate KDAC's ability to handle mixed clustering structures in a uniformly noisy environment. Due to the novelty of alternative clustering, there has been very few public repository data that have at least two alternative labels. The two traditional benchmark datasets are CMU's WebKB dataset [34] and face images from the UCI KDD repository [35]. CMU's WebKB dataset consists of 1041 html pages from 4 universities. One labelling is based on universities, and an alternative labelling based on topic (course, faculty, project, student). After preprocessing by removing rare and stop words, we are left with 500 words. The face dataset consists of 640 images from 20 people in four different poses. Each image has 32x30 pixels. Images are vectorized and PCA is then used to further condense the dataset by keeping 85% of the variance, resulting in a dataset of 624 samples and 20 features. We set the existing clustering based on identity, and seek an alternative clustering based on pose. The last real dataset is the Flower image by Alain Nicolas [36], a 350x256 pixel image. The RGB values of each pixel is taken as a single sample, with repeated samples removed. This results in a dataset of 256 samples and 3 features. Although this dataset does not have labels, the quality of the alternative clustering can be visually observed. With the exception of the high dimensional WebKB data, the remaining datasets are visualized in Figure 3.1 (b).

	σ	λ	q
Gauss A	1	0.04	1
Gauss B 200	5	2	3
Moon 400	0.1	1	3
Moon+N 200	0.2	0.1	6
Flower	2	10	2
Face	3.1	1	17
Web KB	18.7	0.057	4

Table 3.2: Table for the hyperparameters used for each experiment.

Hyperparameter Tuning. ISM has 3 hyperparameters that require tuning, q, λ, and σ. The hyperparameter q with a potential range of (0,d) is initially set to k, the number of clusters. A grid search was then conducted for $\lambda \in (0, 10]$ and $\sigma \in (0, 10]$ to find the highest CQ (λ, σ) pair that satisfies inequality (3.10) at an increment of 0.01. In the event that Inequality (3.10) cannot be satisfied, the highest CQ closest to satisfying (3.10) is used. At this point, since $\Phi(W^*)$ is computed, the q value that maximizes the eigengap as defined in Eq(3.11) can be used if (3.10) is not yet satisfied. Note that competing models do not have a natural way of selecting hyperparameters. Since all competing models optimize the same objective, we report all results using the same hyperparameters. To ensure reproducible experiments, the hyperparameters utilized in each experiment is provided in Table 3.2.

Evaluation Method. To exhaustively evaluate ISM on KDAC, both external and internal measures have been recorded in Table 3.1. More specifically, column 1 and 3 (NMI and Novelty) are external measures because they compare alternative cluster assignments against an externally known ground truth. To make the comparison, Normalized Mutual Information (NMI) as suggested by [37] was used. The NMI is a measure between 0 to 1 with 0 denoting no relationship and 1 as maximum relationship. Column 1 (NMI) is calculated by computing the NMI between the ground truth and the alternative cluster assignments. Column 3 (Novelty) is calculated by computing the NMI between the alternative cluster assignments against the original label. Ideally, we wish for the NMI of the alternative cluster assignments against the ground truth to be 1, and 0 against the original clustering. If we let U and L be two clustering assignments, NMI can be calculated with $NMI(L, U) = \frac{I(L,U)}{\sqrt{H(L)H(U)}}$, where $I(L, U)$ is the mutual information between L and U, and $H(L)$ and $H(U)$ are the entropies of L

and U respectively. Lastly, since no ground truth exists for the Flower dataset, the NMI field is not applicable and indicated with a dash (-).

Columns 2, 4 and 5 (CQ, Cost, Time) are internal measures used to evaluate ISM. They are internal measures since they are computed solely from the data and the algorithm with no external knowledge applied during the comparison. The clustering quality (CQ) is computed with $HSIC(XW, U)$ with U as the clustering solution. This is equivalent to the spectral clustering objective with high values denoting high clustering quality. The cost quality records the objective function of KDAC in Eq. (3.2). The last column, Time, measures the execution time of the entire KDAC algorithm in seconds on an Intel Xeon E7 processor.

In Table 3.1, we report the performance of the various methods: our ISM, SM, and DG, paired with two initialization schemes: our spectral initialization (SI) and random initialization (RI). For random initialization, we repeat these 10 times and report the mean and standard deviation for each measure. The optimal direction of each measure is denoted by the $\uparrow\downarrow$, with $\uparrow$ denoting a preference towards larger values and $\downarrow$ otherwise. For each field, the optimal result is printed in bold font.

Performance Comparison Results. Table 3.1 illustrates that ISM with SI outperforms its competitors in both internal and external quality measure for the first 3 columns. Since it is possible for the objective cost to achieve a low cost with a trivial solution based on different (σ, λ) pairs, the objective should be low, but it is not always indicative of better clustering quality. This is demonstrated in the case of WebKb dataset, where SM+SI achieved a lower cost with a faster convergence. However, upon inspecting the clustering allocation in this case, we observed that it was a trivial classification, with almost all points in the same cluster. Since the demand for a faster KDAC was the original motivator, special attention should be paid to the Time column. The execution time of ISM significantly outperforms the original approach (DG) as well as the standard approach (SM). This speed improvement is especially true in the Faces dataset where ISM improves the speed by 5 folds.

The Effect of Spectral Initialization. By comparing SI against RI, we can isolate the effect of our proposed spectral initialization (SI). Comparing the rows, the spectral initialization improved both time and clustering quality for all methods. From this observation, we conclude that SI contributes to clustering quality by starting each algorithm at a desirable initialization. The convergence improvement came from placing each algorithm closer to the local minimum. Comparing all methods when using SI, we observed that ISM optimization still outperformed other algorithms in terms of

time. From this, we conclude that the proposed SI has a greater impact on the clustering quality while the ISM optimization technique contribute towards faster convergence.

Scalability. Note that KDAC consists of optimizing U and W from Eq. (3.2). Since the computational bottleneck resided in the optimization of W, ISM was designed to speed up this portion. The scalability analysis, therefore, will concentrate on only the optimization of W. ISM has a computational complexity of $O(t_{ISM}(n^2d^2+d^3)$ and the complexity can be divided into the calculation of the derivative and the eigendecomposition of $\Phi(W)$. Although the derivative contributes to $O(n^2d^2)$, it is a highly parallelizable operation that can be rendered trivial through the usage of GPUs. Therefore, ISM's true bottleneck is not the number of samples (n), but the number of dimensions due to the $O(d^3)$ operation of eigendecomposition. As the dimensionality of the data increase, the complexity grows at a cubic rate. Yet, while this growth of dimensionality may exacerbate algorithms such as SM, the negative influence on ISM is limited. This is because the $O(d^3)$ term from SM came from a matrix inversion while ISM uses a spectral method. Since only very few eigenvalues are required, there exists many approximation algorithms to speed up the eigendecomposition [31] [32][33]. To demonstrate this point, Coordinate-wise Power Method (CPM) [33] was implemented and compared to the eigendecomposition operation from Numpy. In this experiment, noisy dimensions of uniform distributions are added to the synthetic dataset of Large Gauss (LG) such that the total dimension increase in multiples of 2. As the dimension of the data grows exponentially, the time of execution is recorded in Figure 3.4 in log scale. Given a log/log scale, a slope of 1 is linear while a slope of 2 is quadratic. From studying the slope of each operation, the eigendecomposition from Numpy grows at a rate between linear and quadratic. However, by utilizing CPM, the growth becomes roughly linear. Therefore, the refinement of utilizing a spectral optimization method is a key reason for speed improvement.

3.6 Alternative Clustering Summary

We have proposed an iterative spectral optimization technique for solving a non-convex optimization problem while being constrained on a Stiefel manifold. This new technique demonstrates speed improvement for any algorithm that could be formulated into Eq (3.4). The ISM algorithm is easy to implement with existing software libraries. Due to the usage of the spectral method, approximation techniques from existing research could be deployed to further speed up the eigendecomposition.

Accompanied with the algorithm are the theoretical guarantees that satisfy the first and second order necessary conditions for a local minimum. Besides these guarantees, we also proposed a natural initialization point based on Taylor approximation of the original cost function. Experiments on synthetic and real data confirmed that our proposed spectral initialization improved the performance of all the optimization algorithms, ISM, SM and DG. Simultaneously, the experiments demonstrate that our proposed ISM algorithm had the best convergence rate compared to competing models with up to 5 folds of speed improvement. Although we focus this paper on an important application of ISM, alternative clustering, the optimization algorithm proposed and guarantees can be extended to other optimization problems involving Gaussian-kernel like objectives constrained over the Stiefel manifold, which is a common formulation for dimensionality reduction with a kernel-based objective. Understanding whether ISM can be applied to such objectives and be leveraged to solve broader classes of problems is a natural future direction for this work.

SG	NMI ↑	CQ ↑	Novelty ↓	Cost ↓	Time ↓
ISM SI	**1**	**2**	**0**	-1.2	**0.02**
ISM RI	0.4±0.49	1.6±0.478	**0.0±0.0**	-1.48±0.381	0.04±0.01
SM SI	**1**	1.99	**0**	**-1.791**	0.404
SM RI	**1±0.0**	1.79±0.398	**0.0±0.0**	-1.59±0.398	1.47±1.521
DG SI	0	1.629	1	-1.316	1.732
DG RI	0.93±0.196	0.99±0.0	0.03±0.104	-0.99±0.0	1.51±0.102
LG					
ISM SI	**1**	**1.9**	**0**	509	0.239
ISM RI	0.8±0.4	**1.9±0.011**	**0.0±0.0**	605±193	**0.18±0.038**
SM SI	**1**	1.9	**0**	509	0.413
SM RI	0.4±0.49	1.9±0.031	0.6±0.49	1095±479	11.21±11.01
DG SI	**1**	**1.9**	**0**	509.975	1595.174
DG RI	0.1±0.32	0.61±0.17	0.9±0.32	**101.5±30.8**	1688±551
Moon					
ISM SI	**1**	**2**	**0**	**149**	**0.575**
ISM RI	0.4±0.4	**2.0±0.0**	0.5±0.5	269.98±43	3.07±1.4
SM SI	**1**	1.998	**0**	**149**	2.653
SM RI	0.27±0.335	**2.0±0.0**	0.5±0.5	277±18	258±374
DG SI	**1**	1.998	**0**	149.697	359.188
DG RI	0.09±0.3	1.27±0.6	0.8±0.4	161.65±102	3212±1368
MoonN					
ISM SI	**1**	**2**	**0**	**15.3**	**0.3**
ISM RI	0.91±0.2	**2.0±0**	0.04±0.1	15.59±0.8	0.55±0.2
SM SI	**1**	**2**	**0**	**15.3**	0.452
SM RI	0.22±0.4	**2.0±0**	0.72±0.4	17.7±1	102±48.5
DG SI	**1**	**2**	**0**	15.4	3836.3
DG RI	0±0.0	1.6±0.04	1.0±0.0	16.2±0.06	3588±304
Flower					
ISM SI	**-**	**0.41**	**0**	20	**0.04**
ISM RI	**-**	**0.41±0.0**	**0.0±0.0**	19.51±0.0	0.1±0.01
SM SI	**-**	**0.41**	**0**	20	0.153
SM RI	**-**	0.3±0.2	0.01±0.012	**0.7±0.5**	0.8±0.6
DG SI	**-**	**0.41**	**0**	20	27.251
DG RI	**-**	0.35±0.02	0.2±0.4	20.6±1	37.0±5
Faces					
ISM SI	**0.57**	**0.61**	0.004	59.6	**1.5**
ISM RI	0.56±0.002	**0.61±0.0**	**0.0±0.0**	59.6±0.118	1.51±0.146
SM SI	0.562	0.608	0.004	59.559	7.96
SM RI	**0.57±0.002**	**0.61±0**	0.004±0	59.7±0.1	109±35.2
DG SI	0.564	0.6	0.004	59.8	100591.071
DG RI	0.458±0.045	0.565±0.018	0.024±0.018	**54.28±3**	166070±4342
WebKb					
ISM SI	**0.37**	0.286	**0**	-0.273	231
ISM RI	0.29±0.07	0.54±0.2	0.01±0.002	-0.52±0.2	137.45±15.7
SM SI	0.048	**3.296**	0.034	**-3.159**	**9.945**
SM RI	0.33±0.06	0.12±0.005	0.008±0.004	-0.11±0.004	13511±13342
DG SI	0.048	1.066	0.034	-1.019	199887.134
DG RI	0.23±0	1.06±0	0.1±0.005	0.616±0.04	727694±41068

Table 3.1: The Normalized Mutual Information, Clustering Quality, and clustering novelty are abbreviated in the first 3 columns as NMI, CQ, and Novelty. The cost of the objective and run time is displayed in the last two columns. For each optimization technique, spectral initialization (SI) and random initialization (RI) are separately tested. With RI, 10 random initial points have been tested with their mean and std displayed.

Chapter 4

Deep Kernel Learning for Clustering

Clustering algorithms group similar samples together based on some predefined notion of similarity. One way of representing this similarity is through kernels. However, the choice for an appropriate kernel is data-dependent; as a result, the kernel design process is frequently an art that requires intimate knowledge of the data. A common alternative is to simply use a general-purpose kernel that performs well under various conditions (e.g., polynomial or Gaussian kernels).

Instead of using a predefined kernel, we wish to leverage HSIC to learn the kernel function itself by demonstrating the relationship between Spectral Clustering and the HSIC objective. It turns out, *feature map* learning is equivalent to learning the appropriate Adjacency matrix for Spectral Clustering. Therefore, learning the *feature map* and the clustering labels together can yield a more flexible model. From this motivation, a deep and a shallow method for clustering can be proposed. This chapter will focus the deep model, leaving the shallow model to the next chapter.

For the deep model, we propose KernelNet Clustering (KNet Clustering), a methodology for learning a kernel and an induced clustering directly from the observed data. In particular, we train a *deep* kernel by combining a neural network representation with a Gaussian kernel. More specifically, given a dataset $\{x_i\}_{i=1}^N$ of N samples in $\mathbb{R}^d$, we learn a kernel $\tilde{k}(\cdot,\cdot)$ of the form:

$$\tilde{k}(x_i, x_j) = e^{-\frac{\|\psi_\theta(x_i)-\psi_\theta(x_j)\|_2^2}{2\sigma^2}} / \sqrt{d_i d_j}, \tag{4.1}$$

where $\psi_\theta(\cdot)$ is an embedding function modeled as a neural network parametrized by θ, and σ, d_i, d_j are normalizing constants. Intuitively, incorporating a neural network (NN) parameterization to a

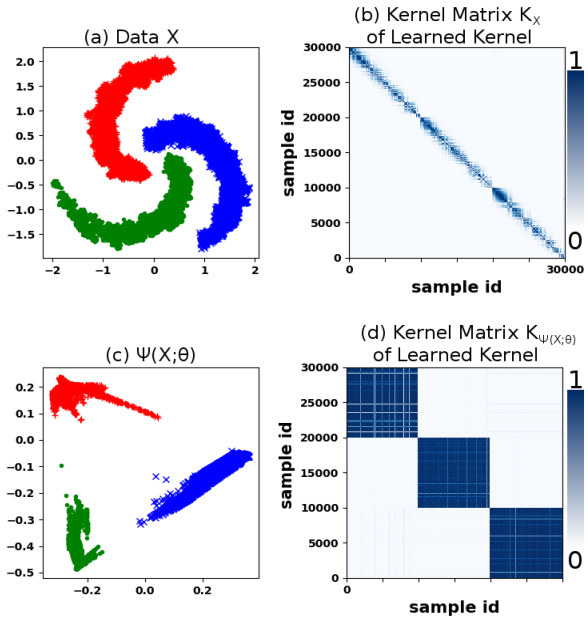

Figure 4.1: Illustration of our learned embedding on the Spiral dataset. The full dataset and its kernel matrix are shown in (a) and (b), respectively. Applying a Gaussian kernel with $\sigma = 0.3$ directly to this dataset leads to a highly uninformative kernel matrix, as shown in (b). Our embedding of the entire dataset and the resulting kernel matrix are shown in (c) and (d), respectively. Our embedding is trained only on 1% of the samples. Yet, it generalizes to the remaining dataset, produces convex clusters, and yields an informative kernel with a clear block diagonal structure.

Gaussian kernel, we are able to learn a flexible deep kernel for clustering, tailored specifically to a given dataset.

We train our deep kernel with a spectral clustering objective based on the Hilbert Schmidt Independence Criterion [8]. This training can be interpreted as learning a non-linear transformation ψ *as well as* its spectral embedding U *simultaneously*. Via an appropriate and intuitive initialization of our training process, we ensure that our clustering method is at least as powerful as spectral clustering. In particular, just as spectral clustering, our learned kernel and the induced clustering work exceptionally well on non-convex clusters. In practice, by training the kernel directly from the data, our proposed method significantly outperforms spectral clustering.

The non-linear transformation ψ learned directly from the dataset allows us to readily handle *out-of-sample* data. Given a new unobserved sample $x_u \in \mathbb{R}^d$, we can easily identify its cluster label by first

computing its image $\psi_\theta(x_u)$, thus embedding it in the same space as the (already clustered) existing dataset. This is in contrast to spectral clustering, that would require a re-execution of the algorithm from scratch on the combined dataset of $N + 1$ samples. The aforementioned properties of our algorithm are illustrated in Fig. 4.1. A dataset of $N = 30,000$ samples in $\mathbb{R}^2$ with non-convex spiral clusters is shown in Fig. 4.1(a). Applying a Gaussian kernel with $\sigma = 0.3$ directly to these samples leads to a highly uninformative kernel matrix, as shown in Fig 4.1(b). We train our embedding $\psi_\theta(\cdot)$ on only 1% of the samples, and apply it to the *entire dataset*; the embedded data, shown in Fig. 4.1(c), consists of nearly-convex, linearly-separable clusters. More importantly, the corresponding learned kernel $\tilde{k}(\cdot, \cdot)$, yields a highly informative kernel matrix that clearly exhibits a block diagonal structure as shown in Fig. 4.1(d). In summary, our major contributions are:

- We propose a novel methodology of discovering a deep kernel tailored for clustering directly from data, using an objective based on the Hilbert-Schmidt Independence Criterion.
- We propose an algorithm for training the kernel by maximizing this objective, as well as for selecting a good parameter initialization. Our algorithm, KNet Clustering, can be perceived as alternating between training the kernel and discovering its spectral embedding.
- We evaluate the performance of KNet Clustering with synthetic and real data compared to multiple state-of-the-art methods for deep clustering. In 5 out 6 datasets, KNet Clustering outperforms state-of-the-art by as much as 57.8%; this discrepancy is more pronounced in datasets with non-convex clusters, which KNet Clustering handles very well.
- Finally, we demonstrate that the algorithm does well in clustering out-of-sample data. This generalization capability means we can significantly accelerate KNet Clustering through subsampling: learning the embedding ψ on only 1%-35% of the data can be used to cluster an entire dataset, leading only to a 0%-3% degradation of clustering performance.

4.1 How HSIC Relates to Clustering

Proposed by Gretton et al. [8], the Hilbert Schmidt Independence Criterion (HSIC) is a statistical dependence measure between two random variables. Like Mutual Information (MI), it measures dependence by comparing the joint distribution of the two random variables with the product of their marginal distributions. However, compared to MI, HSIC is easier to compute empirically, since it does not require a direct estimation of the joint distribution. It is used in many applications due

to this advantage, including dimensionality reduction [12], feature selection [38], and alternative clustering [9], to name a few.

Formally, consider a set of N i.i.d. samples $\{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^c$ are drawn from a joint distribution. Let $X \in \mathcal{R}^{N\times d}$ and $Y \in \mathcal{R}^{N\times c}$ be the corresponding matrices comprising a sample in each row. Let also $k_X : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be any characteristic kernel, in this paper we consider Gaussian kernel $k_X(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$, and $k_Y : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}$ be another characteristic kernel, assumed to be the linear kernel $k_Y(y_i, y_j) = y_i^\top y_j$ here. Define $K_X, K_Y \in \mathbb{R}^{N\times N}$ to be the kernel matrices with entries $K_{X_{i,j}} = k_X(x_i, x_j)$ and $K_{Y_{i,j}} = k_Y(y_i, y_j)$, respectively, and let $\tilde{K}_X \in \mathbb{R}^{N\times N}$ be the normalized Gaussian kernel matrix given by

$$\tilde{K}_X = D^{-1/2} K_X D^{-1/2}, \tag{4.2}$$

where the degree matrix

$$D = \operatorname{diag}(K_X \mathbf{1}_N) \in \mathbb{R}^{N\times N} \tag{4.3}$$

is a normalizing diagonal matrix. Then, the HSIC between X and Y is estimated empirically via:

$$\mathbb{H}(X, Y) = \tfrac{1}{(N-1)^2} \operatorname{Tr}(\tilde{K}_X H K_Y H), \tag{4.4}$$

where intuitively, HSIC empirically measures the dependence between samples of the two random variables. Though HSIC can be more generally defined for any arbitrary characteristic kernels, this particular choice has a direct relationship with (and motivation from) spectral clustering. In particular, given X, consider the optimization:

$$\max_U \quad \mathbb{H}(X, U) \tag{4.5a}$$

$$\text{subject to} \quad U^\top U = I,\ U \in \mathbb{R}^{N\times c}, \tag{4.5b}$$

where $\mathbb{H}$ is given by (4.4). Then, the optimal solution $U_0 \in \mathbb{R}^{N\times c}$ to (4.5) is precisely the spectral embedding of X [12]. Indeed, U_0 comprises the top c eigenvectors of the normalized similarity matrix, given by:

$$\mathcal{L} = H D^{-1/2} K_X D^{-1/2} H. \tag{4.6}$$

This derivation was originally provided by Niu et al. [15]. For completeness, we further prove this derivation in Appendix B.1.

4.2 Clustering Problem Formulation

We are given a dataset of samples grouped in (potentially) non-convex clusters. Our objective is to cluster samples by first embedding them into a space in which the clusters become convex. Given such an embedding, clusters can subsequently be identified via, e.g., k-means. We would like the embedding, modeled as a neural network, to be at least as expressive as spectral clustering: clusters separable by spectral clustering should also become separable via our embedding. In addition, the embedding should generalize to out-of-sample data, thereby enabling us to cluster new samples outside the original dataset.

Learning the Embedding and a Deep Kernel. Formally, we wish to identify c clusters over a dataset $X \in \mathbb{R}^{N\times d}$ of N samples and d features. Let $\psi : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^{d'}$ be an embedding of a sample to $\mathbb{R}^{d'}$, modeled as a DNN parametrized by $\theta \in \mathbb{R}^m$; we denote by $\psi_\theta(x)$ the image of $x \in \mathbb{R}^d$ under parameters θ. We also denote by $\Psi : \mathbb{R}^{N\times d} \times \mathbb{R}^m \to \mathbb{R}^{N\times d'}$ the embedding of the entire dataset induced by ψ, and use $\Psi_\theta(X)$ for the image of X.

Let $U_0 \in \mathbb{R}^{N\times c}$ be the spectral embedding of X, obtained via spectral clustering. We can train ψ to induce similar behavior as U_0 via the following optimization:

$$\max_{\theta\in\mathbb{R}^m} \mathbb{H}(\Psi_\theta(X), U_0), \tag{4.7}$$

where $\mathbb{H}$ is given by Eq. (4.4). Since HSIC is a dependence measure, by training θ so that $\Psi_\theta(X)$ is maximally dependent on U_0, Ψ becomes a surrogate to the spectral embedding, sharing similar properties.

However, the surrogate Ψ learned via (4.7) is restricted by U_0, hence it can only be as discriminative as U_0. To address this issue, we depart from (4.7) by jointly discovering both Ψ as well as a *coupled* spectral embedding U. In particular, we solve the following optimization problem w.r.t. *both* the embedding *and* U:

$$\max_{\theta,U} \quad \mathbb{H}(\Psi_\theta(X), U) - \lambda\|X - f_{\theta,\theta'}(X)\|_2^2 \tag{4.8a}$$

$$\text{subject to:} \quad U^\top U = I, \tag{4.8b}$$

$$\theta, \theta' \in \mathbb{R}^m,\ U \in \mathbb{R}^{N\times c}, \tag{4.8c}$$

where,

$$f_{\theta,\theta'}(X) = \Psi'_{\theta'}(\Psi_\theta(X)) \tag{4.9}$$

is an autoencoder, comprising $\Psi : \mathbb{R}^{n\times d} \times \mathbb{R}^m \to \mathbb{R}^{N\times d'}$ and $\Psi' : \mathbb{R}^{N\times d'} \times \mathbb{R}^m \to \mathbb{R}^{N\times d}$ as an encoder and decoder respectively. This autoencoder objective is theoretically necessary to ensure that the embedding Ψ is injective, as stated in Theorem 1 by Li et al. in [39]. However, as observed in [39], our empirical experiments also suggest that this autoencoder objective is not necessary in practice: the dependence of the local minimum found by gradient descent to the starting point ensures that the trained embedding Ψ is representative of the input X even in the absense of this penalty.

To gain some intuition into how problem (4.8) generalizes (4.7), observe that if the embedding Ψ is fixed to be the identity map (i.e., for $\Psi_\theta(X) \equiv X$) then, by Eq. (4.5), optimizing only for U produces the spectral embedding U_0 . The joint optimization of both Ψ and U allows us to further improve upon U_0, as well as on the coupled Ψ; we demonstrate experimentally that this significantly improves clustering quality.

Kernel Learning. The optimization (4.8) can also be interpreted as an instance of *kernel learning*. Indeed, as discussed in the introduction, by learning ψ, we discover in effect a normalized kernel $\tilde{k}$ of the form

$$\tilde{k}(x_i, x_j) = e^{-\frac{\|\psi_\theta(x_i)-\psi_\theta(x_j)\|_2^2}{2\sigma^2}} / \sqrt{d_i d_j}, \tag{4.10}$$

where d_i, d_j are the corresponding diagonal elements of degree matrix D.

Out-of-Sample Data. The embedding Ψ can readily be applied to clustering out-of-sample data. In particular, having trained Ψ over dataset X, given a new dataset $Y \in \mathbb{R}^{N'\times d}$, we can cluster this new dataset efficiently as follows. First, we use the pre-trained Ψ to map every sample y_i in Y to its image, producing $\Psi_\theta(Y)$: this effectively embeds Y to the same space as $\Psi_\theta(X)$. From this point on, clusters can be recomputed efficiently via, e.g., k-means, or by mapping the images $\psi_\theta(y_i)$ to the closest existing cluster head. In contrast to, e.g., spectral clustering, this avoids recomputing the joint embedding of the entire dataset $(X; Y)$ from scratch.

The ability to handle out-of-sample data can be leveraged to also accelerate training. In particular, given the original dataset X, computation can be sped up by training the embedding Ψ by solving (4.8) *on a small subset of* X. The resulting trained Ψ can be used to embed, and subsequently

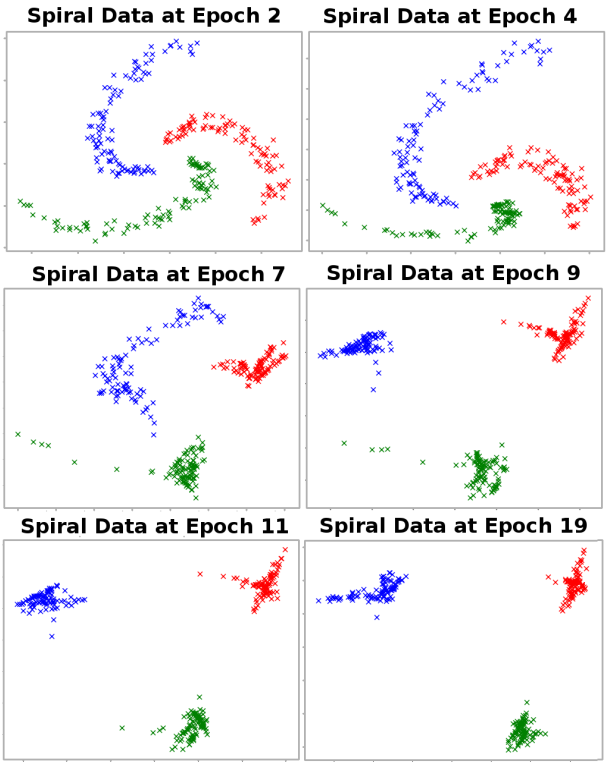

Figure 4.2: Cluster images after several epochs of stochastic gradient descent over (4.8a). The first term of objective (4.8a) pulls cluster images further apart, while making them increasingly convex. Note that this happens in a fully unsupervised fashion.

cluster, the entire dataset. We show experimentally that this approach works very well, leading to a significant acceleration in computations without degrading clustering quality.

Convex Cluster Images. The first term in objective (4.8a) naturally encourages Ψ to form convex clusters. To see this, as derived in Appendix B.2, ignoring the reconstruction error, the objective (4.8a) becomes:

$$\sum_{i,j} \Gamma_{i,j} e^{-\frac{1}{2\sigma^2}||\psi_\theta(\mathbf{x}_i)-\psi_\theta(\mathbf{x}_j)||^2}, \tag{4.11}$$

where $\Gamma_{i,j}$ are the elements of matrix $\Gamma = D^{-1/2}HUU^THD^{-1/2} \in \mathbb{R}^{N\times N}$. The exponential terms in Eq. (4.11) compel samples under which $\Gamma_{i,j} > 0$ to become attracted to each other, while samples for which $\Gamma_{i,j} < 0$ drift farther apart. This is illustrated in Figure 4.2. Linearly separable, increasingly convex cluster images arise over several iterations of solving our algorithm at Eq. (4.8). The algorithm, KNet Clustering, is described in the next section.

4.3 KNet Clustering Algorithm

We solve optimization problem (4.8) by iteratively adapting $\tilde{\theta} = (\theta, \theta')$ and U. In particular, we initialize Ψ to be the identity map, and U to be the spectral embedding U_0, and then alternate between adapting U and $\tilde{\theta}$. We optimize $\tilde{\theta}$ via stochastic gradient ascent (SGA). To optimize U, we adopt two approaches: one based on eigendecomposition, and one based on optimization over the Stiefel manifold. We describe each of these steps in detail below; a summary can be found in Algorithm 3.

Initialization. The non-convexity of (4.8) necessitates a principled approach for selecting good initialization points for U and $\tilde{\theta} = (\theta, \theta')$. We initialize U to U_0, computed via the top-c eigenvectors of the normalized similarity matrix $\mathcal{L}$ of X, given by (4.6). We initialize θ so that Ψ is the identity map; This is accomplished by pre-training θ, θ' via SGD as solutions to:

$$\min_{\theta,\theta'} \|X - \Psi_\theta(X)\|_2^2 + \|X - f_{\theta,\theta'}(X)\|_2^2. \tag{4.12}$$

Note that, in this construction, we use $d' = d$.

Algorithm 3: ISM Algorithm

Input : data $X \in \mathbb{R}^{N \times d}$

Output : Ψ and clustering labels

Initialization : Initialize $\tilde{\theta} = (\theta, \theta')$ via (4.12)

Initialize U with U_0

while K_U *has not converged* **do**

  Update $\tilde{\theta}$ via one epoch of SGA over (4.8a), holding U and D fixed

  Update D via Eq. (4.3)

  KNet$_{\texttt{EIG}}$: Update U via eigendecomposition of Laplacian (4.14), **or**

  KNet$_{\texttt{SMA}}$: Update U via Stiefler Manifold Ascent (4.16)

Run k-means on $\Psi(X; \theta)$

Updating $\tilde{\theta}$. A simple way to update $\tilde{\theta}$ is via gradient ascent, i.e.:

$$\tilde{\theta}_k = \tilde{\theta}_{k-1} + \gamma_k \nabla F(\tilde{\theta}_{k-1}, U_{k-1}), \tag{4.13}$$

for $k \geq 1$, where F is the objective (4.8a). In practice, we wish to apply stochastic gradient ascent over mini-batches; for U fixed, the first term in the objective (4.8a) reduces to (4.11); however, the terms in the sum are coupled via the normalizing degree matrix D, which depends on θ via (4.3). This significantly increases the cost of computing mini-batch gradients. To simplify this computation, instead we hold both U *and* D fixed, and update $\tilde{\theta}$ via one epoch of SGA over (4.8a).

At the conclusion of one epoch, we update the Gaussian kernel K_X and the corresponding degree matrix D via Eq. (4.3). We implemented both this heuristic and regular SGA, and found that it led to a significant speedup without any observable degradation in clustering performance.

Updating U via Eigendecomposition. Our first approach to adapting U relies on the fact that, holding $\tilde{\theta}$ constant, problem (4.8) reduces to the form (4.5). That is, at each iteration, for $\tilde{\theta}$ fixed, the optimal solution

$$U^*(\tilde{\theta}) = \arg\max_{U:U^\top U=I} F(\tilde{\theta}, U)$$

is given by the top c eigenvectors of matrix

$$\mathcal{L}_\theta = HD^{-1/2}K_{\Psi_\theta(X)}D^{-1/2}H. \tag{4.14}$$

Hence, given $\tilde{\theta}$ at an iteration, we update U by returning $U^*(\tilde{\theta})$. Note that when $c \ll N$, there are several algorithms for computing the top eigenvectors efficiently (see, e.g., [31, 40]).

Updating U via Stiefel Manifold Ascent. The manifold in $\mathbb{R}^{N\times c}$ defined by constraint (4.8b), a.k.a. the *Stiefel Manifold*, is not convex; nevertheless, techniques such as those outlined in [17, 41] for optimization over this set are available in the literature. These techniques exploit the fact that descent directions that maintain feasibility can be computed efficiently. In particular, following [17], treating $\tilde{\theta}$ as a constant, and given a feasible $U \in \mathbb{R}^{N\times c}$ and the gradient of the objective $\nabla_U F(U) \in \mathbb{R}^{N\times c}$ w.r.t U, define

$$A(U) = -(\nabla_U F(U)U^T + U\nabla_U F(U)^T) \in \mathbb{R}^{N\times N}. \tag{4.15}$$

Using A and a predefined step length τ, the maximization proceeds iteratively via:

$$U_{k+1} = Q(U_k)U_k, \tag{4.16}$$

where Q is the so-called Cayley transform, defined as

$$Q(U) = (I + \frac{\tau}{2}A(U))^{-1}(I - \frac{\tau}{2}A(U)). \tag{4.17}$$

The Cayley transform satisfies several important properties [17]. First, starting from a feasible point, it maintains feasibility over the Stiefel manifold (4.8b) for all $k \geq 1$. Second, for small enough τ, it is guaranteed to follow an ascent direction; combined with line-search, convergence is guaranteed to a stationary point. Finally, $Q(U_{k+1})$ given by (4.17) can be computed efficiently from $Q(U_k)$, thereby avoiding a full matrix inversion, by using the Sherman-Morrison-Woodbury identity [42]: this results

in a $O(N^2c + c^3)$ complexity for (4.16), which is significantly faster than eigendecomposion when $c \ll N$. In our second approach to updating U, we apply (4.16) rather than eigendecomposition of $\mathcal{L}_\theta$ when adapting U iteratively. Both approaches are summarized in line 7 of Alg. 3; we refer to them as $\text{KNet}_{\texttt{EIG}}$ and $\text{KNet}_{\texttt{SMA}}$.

4.4 Deep Kernel Clustering Experimental Results

Datasets. The first three datasets (**Moon**, **Spiral1**, **Spiral2**) are synthetic and comprise non-convex clusters; they are shown in Figure 4.3. Among the remaining four real-life datasets, the features of **Breast Cancer** [43] are discrete integer values between 0 to 10. The features of the **Wine** dataset [44] consist of a mix of real and integer values. The Reuters dataset (**RCV**) is a collection of news articles labeled by topic. We represent each article via a *tf-idf* vector using the 500 most frequent words and apply PCA to further reduce the dimension to $d = 5$. The **Face** dataset [35] consists of grey scale, 32×30-pixel images of 20 faces in different orientations. We reduce the dimension to $d = 20$ via PCA. As a final preprocessing step, we center and scale all datasets so that their mean is 0 and the standard deviation of each feature is 1.

Data	N	c	d	Type	σ
Moon	1000	2	2	Geometric Shape	0.1701
Spiral1	3000	3	2	Geometric Shape	0.1708
Spiral2	30000	3	2	Geometric Shape	0.1811
Cancer	683	2	9	Medical	3.3194
Wine	178	3	13	Classification	4.939
RCV	10000	4	5	Text	2.364
Face	624	20	27	Image	6.883

Table 4.1: Dataset Summary.

Clustering Algorithms. We evaluate 8 algorithms, including our two versions of KNet Clustering described in Alg. 3. For existing algorithms, we use architecture designs (e.g., depth, width) as recommended by the respective authors during training. We provide details for each algorithm below.

k-means: We use the CUDA implementation[1] by [45].

[1] https://github.com/src-d/kmcuda

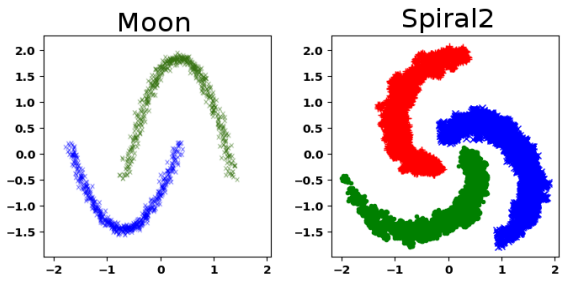

Figure 4.3: Synthetic Datasets. Both dataset contain non-convex clusters. Dataset Spiral2 (depicted) contains $N = 30,000$ samples, while Spiral1 contains a sub-sampled version of $N = 3,000$.

Dataset	AEC	DEC	IMSAT	SN	SC	k-means	$\text{KNet}_{\texttt{EIG}}$	$\text{KNet}_{\texttt{SMA}}$
Moon	56.2 ± 0.0	42.2 ± 0.0	51.3 ± 20.3	$\mathbf{100} \pm 0.0$	72.0 ± 0.0	66.1 ± 0.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
Spiral1	28.3 ± 0.0	32.0 ± 0.01	59.6 ± 7.5	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	42.0 ± 0.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
Cancer	79.9 ± 0.2	79.2 ± 0.0	74.6 ± 2.2	82.9 ± 0.0	69.8 ± 0.0	73.0 ± 0.0	$\mathbf{84.2 \pm 0.4}$	82.5 ± 0.1
Wine	54.6 ± 0.0	80.6 ± 0.0	72.3 ± 11.4	79.7 ± 0.2	88.0 ± 0.0	42.8 ± 0.0	$\mathbf{91.0 \pm 0.8}$	90.0 ± 0.7
RCV	39.3 ± 0.0	51.3 ± 0.0	39.0 ± 5.5	43.5 ± 0.2	46 ± 0.0	$\mathbf{56.0 \pm 0}$	46.3 ± 0.4	46.1 ± 0.2
Face	76.8 ± 0.0	75.8 ± 1.6	83.8 ± 3.5	75.6 ± 0.1	66.0 ± 0.4	91.8 ± 0.0	$\mathbf{93.0 \pm 0.3}$	92.6 ± 0.5

Table 4.2: The **clustering results** measured by NMI as percentages are shown above where the best mean results are highlighted in bold text. Besides the RCV dataset, KNet Clustering generally outperforms competing methods by a significant margin. The improvement is especially large with the Moon and Spiral1 dataset due to KNet Clustering's ability to identify non-convex clusters.

	AEC		DEC		IMSAT		SN	SC	k-means	KN		
Dataset	Prep	RT	Prep	RT	Prep	RT	RT	RT	RT	Prep	$\text{RT}_{\texttt{EIG}}$	$\text{RT}_{\texttt{SMA}}$
Moon	18.23	28.04	331.80	3.52	1.20	34.34	324.00	0.34	0.04	129.00	28.90	18.40
Spiral1	2574.00	7920.00	343.80	121.20	53.06	309.00	444.00	4.20	0.12	300.00	42.00	19.00
Cancer	99.00	280.20	342.00	38.73	1.10	144.00	234.00	0.18	0.03	150.00	19.30	10.30
Wine	33.81	41.69	345.00	38.02	1.10	33.32	330.00	0.03	0.06	462.00	7.20	3.40
RCV	141.60	2536.80	381.00	784.20	10.39	73.20	438.00	83.40	0.35	1080.00	1830.00	1116.00
Face	27.50	189.00	344.40	254.10	1.20	121.80	170.90	0.26	0.15	1320.00	20.90	3.30

Table 4.3: The **preprocessing (Prep) and runtime (RT)** for all benchmark algorithms are displayed in seconds. The table demonstrates that KNet Clustering's speed is comparable to competing methods.

SC: We use the python scikit implementation of the classic spectral clustering algorithm by [46].

AEC: Proposed by [47], this algorithm incorporates a k-means objective in an autoencoder. As suggested by the authors, we use 3 hidden layers of width 1000, 250, and 50, respectively, with an output layer dimension of 10.[2]

DEC: Proposed by [48], this algorithm couples an autoencoder with a soft cluster assignment via a KL-divergence penalty. As recommended, we use 3 hidden layers of width 500, 500, and 2000 with an output layer dimension of 10.[3]

IMSAT: Proposed by [49], this algorithm trains a network adversarially by generating augmented datasets. It uses 2 hidden layers of width 1200 each, the output layer size equals the number of clusters c.[4]

SN: Proposed by [50], SN uses an objective motivated by spectral clustering to map to a target similarity matrix.[5]

KNet$_{\texttt{EIG}}$ and **KNet$_{\texttt{SMA}}$:** These are the two versions of KNet Clustering, as described in Alg. 3, in which U is updated via eigendecomposition and Stiefel Manifold Ascent, respectively. For both versions, the encoder and decoder have 3 layers. For Cancer, Wine, RCV, and Face dataset, we set the width of all hidden layers to d. For Moon and Spiral1, we set the width of all hidden layers to 20. We set the Gaussian kernel σ to be median of the pairwise Euclidean distance between samples in each dataset (see Table **??**).[6]

Evaluation Metrics. We evaluate the clustering quality of each algorithm by comparing the clustering assignment generated to the ground truth assignment via the Normalized Mutual Information (NMI). NMI is a similarity metric lying in $[0, 1]$, with 0 denoting no similarity and 1 as an identical match between the assignments. Originally recommended by [51], this statistic has been widely used for clustering quality validation [9, 12, 27, 52].

Consider two clustering assignments assigning labels in $\{1, \ldots, c\}$ to samples in dataset $\Omega = \{1, \ldots, N\}$. We represent these two assignments through two partitions of Ω, namely $\mathcal{C} = \{C_\ell\}_{\ell=1}^c$, $\mathcal{C}' = \{C'_\ell\}_{\ell=1}^c$, where

$$C_\ell, C'_\ell \subseteq \Omega, \ell = 1, \ldots, c$$

are the sets of samples receiving label ℓ under each assignment. Define the empirical distribution of

[2] https://github.com/developfeng/DeepClustering
[3] https://github.com/XifengGuo/DEC-keras
[4] https://github.com/weihua916/imsat
[5] https://github.com/KlugerLab/SpectralNet
[6] https://github.com/neu-spiral/KernelNet

labels to be:

$$\mathbf{P}(\ell, \ell') = \frac{|C_\ell \cap C'_\ell|}{N} \quad \text{for all } (\ell, \ell') \in \{1, \ldots, c\}^2. \tag{4.18}$$

The NMI is then given by the ratio

$$\frac{I(\mathcal{C}, \mathcal{C}')}{\sqrt{H(\mathcal{C})H(\mathcal{C}')}}$$

where $I(\mathcal{C}, \mathcal{C}')$ is the mutual information and $H(\mathcal{C}), H(\mathcal{C}')$ are the entropies of the marginals of the joint distribution (4.18).

For each algorithm, we also measure the execution time, separating it into preprocessing time (Prep) and runtime (RT); in doing so, we separately evaluate the cost of, e.g., parameter initialization from training.

Experimental Setup. We execute all algorithms over a machine with 16 dual-core CPUs (Intel Xeon® E5-2630 v3 @ 2.40GHz) with 32 GB of RAM with a NVIDIA 1b80 GPU. For methods we can parallelize either over the GPU or over the 16 CPUs (IMSAT,SN,k-means,KNet Clustering), we ran both executions and recorded the fastest time. The code provided for DEC could only be parallelized over the GPU, while methods AEC and SC could only be executed over the CPUs. For each dataset in Table 4.2, we run all algorithms on the full dataset 10 times and report the mean and standard deviation of the NMI of their resulting clustering performance against the ground truth. As SGA is randomized, we repeat experiments 10 times and report NMI averages and standard deviations.

For algorithms that can be executed out-of-sample (AEC, DEC, IMSAT, SN, KNET), we repeat the above experiment by training the embedding only on a random subset of the dataset. Subsequently, we apply the trained embedding to the entire dataset and cluster it via k-means. For comparison, we also find cluster centers via k-means on the subset and new samples to the nearest cluster center. For each dataset, we set the size of the subset (reported in Table 4.4) so that the spectrum of the resulting subset is close, in the ℓ_∞ sense, to the spectrum of X.

Results Selecting λ. As clustering is unsupervised, we cannot rely on ground truth labels to identify the best hyperparameter λ. We, therefore, need an unsupervised method for selecting this value. We find that, in practice, just like [39], selecting $\lambda = 0$ works quite well. Because the problem is not convex, local optima reached by KNet Clustering depend highly on the initialization. Initializing (a) Ψ to approximate the identity map via (4.12), and (b) U to be the spectral embedding U_0 indeed

Dataset	Data %	AEC	DEC	IMSAT	SN	k-means	KNet$_{\texttt{EIG}}$	KNet$_{\texttt{SMA}}$
Moon	25%	51.1	45.6	45.5	**100.0**	21.5	**100.0**	**100.0**
Spiral1	10.0%	32.2	49.5	48.7	**100.0**	56.7	**100.0**	**100.0**
Cancer	30.0%	76.4	76.9	74.9	82.2	Fails	**84.0**	83.6
Wine	75.0%	49.1	81.5	69.4	77.2	25.0	**91.1**	89.3
RCV	6.0%	26.8	43.2	35.2	41.3	**52.5**	45.2	43.1
Face	35.0%	52.5	67.1	77.8	75.5	87.2	**92.7**	91.1

Table 4.4: The **out-of-sample clustering** result measured by NMI as percentages are shown above where the best mean results are highlighted in bold text. All algorithms are trained on a subset of data. We report the results of the total dataset clustered out-of-sample via each algorithm.

	AEC		DEC		IMSAT		SN	k-means	KN		
Data	Prep	RT	Prep	RT	Prep	RT	RT	RT	Prep	RT$_{\texttt{EIG}}$	RT$_{\texttt{SMA}}$
Moon	15.01	22.30	201.00	2.95	1.10	27.63	140.10	0.03s	48.00	2.30	1.10
Spiral1	192.00	498.60	250.20	99.00	47.05	229.80	223.30	0.07	82.00	3.10	1.40
Cancer	55.30	75.00	306.00	30.01	1.65	100.20	38.90	0.03	16.00	4.00	1.80
Wine	25.30	35.31	279.00	32.31	2.40	25.60	20.40	0.03	74.00	1.30	0.09
RCV	63.00	316.20	339.00	672.60	3.21	56.00	40.67	0.03	72.00	5.50	2.10
Face	15.10	171.60	315.00	233.40	1.10	93.60	35.09	0.10	76.80	2.20	0.96

Table 4.5: The **out-of-sample preprocessing (Prep) and runtime (RT)** for all benchmark algorithms are displayed in seconds. The table demonstrates that KNet Clustering's speed is comparable to competing methods.

	KNet$_{\texttt{EIG}}$			KNet$_{\texttt{SMA}}$		
λ	**HSIC**	**AE error**	**NMI**	**HSIC**	**AE error**	**NMI**
10^0	98.11	21.58	0.90	92.01	8.98	0.89
10^{-1}	98.80	72.95	0.88	94.21	72.62	0.87
10^{-2}	112.32	101.45	0.87	101.11	88.39	0.89
0.005	109.01	105.37	0.91	108.22	107.22	0.90
10^{-4}	113.18	124.56	0.91	110.18	126.63	0.91
0	110.89	127.23	0.92	109.36	127.11	0.91

Table 4.6: HSIC, AE reconstruction error, and NMI at convergence of KNet Clustering on the Wine dataset, as a function of λ.

leads to a local maximum that is highly dependent on the input X, eschewing the need for the reconstruction error in the objective (4.8).

Our choie of $\lambda = 0$ is further grounded in experiments shown in Table 4.6 which shows results for the Wine dataset. We found that at smaller values of λ result in improved performance both in terms of HSIC and NMI; tables for additional datasets can be found in Table 4.7 in the supplement. Along with the HSIC we also provide AE reconstruction error at convergence. Beyond the good performance of $\lambda = 0$, the table suggests that an alternative unsupervised method is to select λ so that the ratio of the two terms at convergence is close to one. Of course, this comes at the cost of parameter exploration; in the remainder of this section, we report the performance of KNet Clustering with $\lambda = 0$.

	$\text{KNet}_{\texttt{EIG}}$			$\text{KNet}_{\texttt{SMA}}$		
λ	**HSIC**	**AE error**	**NMI**	**HSIC**	**AE error**	**NMI**
10^0	2.989	0.001	0.446	2.989	0	0.421
10^{-1}	2.989	0.0001	0.412	2.989	0	0.414
10^{-2}	2.989	0.0001	0.421	2.989	0	0.418
10^{-3}	2.989	0.001	0.434	2.989	0	0.419
10^{-4}	2.988	0.001	0.421	2.988	0	0.418
10^{-5}	2.965	0.099	0.557	2.979	0.033	0.558
10^{-6}	1.982	0.652	0.644	2.33	0.583	0.56
10^{-7}	2.124	0.669	0.969	2.037	0.66	0.824
10^{-8}	2.249	0.69	1	2.368	0.67	1
0	2.278	0.715	1	2.121	0.718	1

Table 4.7: HSIC, AE reconstruction error, and NMI at convergence of KNet Clustering on the SPIRAL1 dataset, as a function of λ.

Comparison Against State-of-the-Art. Table 4.2 shows the NMI performance of different algorithms over different datasets. With the exception of RCV dataset, we see that KNet Clustering outperforms every algorithm in Table 4.2. AEC, DEC, and IMSAT perform especially poorly when encountering non-convex clusters as shown in the first two rows of the table. Spectral clustering (SC), and SN that is also based on a spectral-clustering motivated objective, perform equally well as KNet Clustering on discovering non-convex clusters. Nevertheless, KNet Clustering outperforms them for real datasets: e.g., for the Face dataset, $\text{KNet}_{\texttt{EIG}}$ surpasses SN by 28%. Note that, for the RCV dataset, k-means outperformed all methods, though overall performance is quite poor; a reason

for this may be the poor quality of features extracted via TFIDFs and PCA.

KNet Clustering's ability to handle non-convex clusters is evident in the improvement over k-means to KNet Clustering for the first two datasets. The kernel matrix K_X shown in Fig. 4.1(b) illustrates why k-means performs poorly on this dataset. In contrast, the increasingly convex cluster images learned by KNet Clustering, as shown in Fig. 4.1(c), lead to much better separability. This is consistently observed for both the Moon and Spiral1 dataset, for which KNet Clustering achieves NMI; we elaborate on this further in Appendix B.3. demonstrate KNet Clustering's ability to generate convex representations even when the initial representation is non-convex.

We also note that KNet Clustering consistently outperforms spectral clustering. This is worth noting because, as discussed in Sec. 4.2, KNet Clustering's initialization of both Ψ and U are tied to the spectral embedding. Table 4.2 indicates that alternatively learning both the kernel and the corresponding spectral embedding indeed leads to improved clustering performance.

Table 4.3 shows the time performance of each algorithm. In terms of total time, KNet Clustering is faster than AEC and DEC. We also observe that SN is faster than most algorithms in terms of run time. However, SN does require extensive hyperparameter tuning to reach the reported NMI performance (see App. B.4). We note that a significant percentage of the total time for KNet Clustering is spent in the preprocessing step, with $\text{KNet}_{\texttt{SMA}}$ being faster than $\text{KNet}_{\texttt{EIG}}$. This is due to the initialization of Ψ, i.e., training the corresponding autoencoder. Improving this initialization process could dramatically speed up the total runtime. Alternatively, as we discuss in the next section, using only a small subset to train the embedding and clustering out-of-sample can also significantly accelerate the total runtime, without a considerable NMI degradation.

Out-of-Sample Clustering. We report out-of-sample clustering NMI performance in Table 4.4; note that SC cannot be executed out-of-sample. Each algorithm is trained using only a subset of samples, whose size is indicated on the table's first column. Once trained, we report in the table the clustering quality of applying each algorithm to the full set without retraining. We observe that, with the exception of RCV, KNet Clustering clearly outperforms all benchmark algorithms in terms of clustering quality. This implies that KNet Clustering is capable of generalizing the results by using as litle as 6% of the data.

By comparing Table 4.2 against 4.4, we see that AEC, DEC, and IMSAT suffer a significant drop

in performance, while KNet Clustering suffers only a maximum degradation of 3%. Therefore, training on a small subset of the data not only yields high-quality results, the results are almost as good as training on the full set itself. Table 4.5, reporting corresponding times, indicates that this can also lead to a significant acceleration, especially of the preprocessing step. Together, these two observations indicate that KNet Clustering can indeed be applied to clustering of large non-convex datasets by training the embedding on only a small subset of the provided samples.

4.5 Deep Kernel Clustering Summary

KNet Clustering performs unsupervised kernel discovery using only a subset of the data. By discovering a kernel that optimizes the Spectral Clustering objective, it simultaneously discovers an approximation of its embedding through a DNN. Furthermore, experimental results have confirmed that KNet Clustering can be trained using only a subset.

Chapter 5

Solving Interpretable Kernel Dimension Reduction

The most important information for a given dataset often lies in a low dimensional space [1, 2, 3, 4, 10, 12, 16, 38, 53, 54, 55, 56, 57, 58]. Due to the ability of kernel dependence measures for capturing both linear and nonlinear relationships, they are powerful criteria for nonlinear dimensionality reduction (DR) [5, 6]. The standard approach is to first map the data into a high dimensional feature space prior to a projection onto a low dimensional space [7]. This approach has been preferred because it captures the nonlinear relationship with an established solution, i.e., the most dominant eigenvectors of the kernel matrix. However, since the high dimensional feature space maps the original featues nonlinearly, it is no longer interpretable. Alternatively, if the projection onto a subspace precedes the feature mapping, the projection matrix can be obtained to inform how the original features linearly combine to form the new features. Exploiting this insight, many formulations have leveraged kernel alignment or Hilbert Schmidt Independence Criterion (HSIC) [8] to model this approach [5, 6, 9, 10, 11, 12, 13, 14, 15, 16]. Together, we refer to these approaches as Interpretable Kernel Dimension Reduction (IKDR). Unfortunately, this formulation can no longer be solved via eigendecomposition, instead, it becomes a highly non-convex manifold optimization that is computationally expensive.

Numerous approaches have been proposed to solve this complex objective. With its orthogonality constraint, it is a form of optimization on a manifold: i.e., the constraint can be modeled geometrically

as a Stiefel or Grassmann manifold [59, 60, 61]. Earlier work, Boumal and Absil [62] propose to recast a similar problem on the Grassmann manifold and then apply first and second-order Riemannian trust-region methods to solve it. Theis et al. [63] employ a trust-region method for minimizing the cost function on the Stiefel manifold. Wen and Yin [17] later propose to unfold the Stiefel manifold into a flat plane and optimize on the flattened representation. While the manifold approaches perform well under smaller data sizes, they quickly become inefficient when the dimension or sample size increases, which poses a serious challenge to larger modern problems. Besides manifold approaches, Niu et al. [18] propose Dimension Growth (DG) to perform gradient descent via greedy algorithm a column at a time. By keeping the descent direction of the current column orthogonal to all previously discovered columns, DG ensures the constraint compliance.

The approaches discussed thus far have remained inefficient. Recently, Wu et al. [9] proposed the Iterative Spectral Method (ISM) for alternative clustering where their experiments on a dataset of 600 samples showed that it took DG almost 2 days while ISM finished under 2-seconds *with a lower objective cost*. Moreover, ISM retains the ability to use eigendecomposition to solve IKDR. Instead of finding the eigenvectors of kernel matrices, ISM uses a small surrogate matrix Φ to replace the kernel matrix, thereby allowing for a much faster eigendecomposition. Yet, ISM is not without its limitations. Since ISM's theoretical guarantees are specific to Gaussian kernels, repurposing ISM to other kernel methods becomes impractical, i.e., a kernel method of a single kernel significantly limits its flexibility and representational power.

In this chapter, we expand ISM's theoretical guarantees to an entire family of kernels, thereby realizing ISM's potential for a wide range of applications. Within this family, each kernel is associated with a matrix Φ where its most dominant eigenvectors form the solution. Here, Φ matrices replace the concept of kernels to serve as an interchangeable component of any applicable IKDR kernel method. We further extend the family to kernels that are conic combinations of the ISM family of kernels. Here, we prove that any conic combination of kernels within the family also has an associated Φ matrix constructed using the respective conic combination of Φs.

Empowered by extending ISM's theoretical guarantees to other kernels, we present ISM as a solution to IKDR problems across several learning paradigms, including supervised DR [6, 10, 11], unsupervised DR [5, 12], semi-supervised DR[13, 14], and alternative clustering [9, 15, 18]. Indeed, we demonstrate how many of these applications can be reformulated into an identical optimization objective which ISM solves, implying a significant role for ISM that has been previously unknown.

Our Contributions.

- We generalize the theoretical guarantees of ISM to an entire family of kernels and propose the necessary criteria for a kernel to be a member of the family.
- We generalize ISM to conic combinations of kernels from the ISM family.
- We establish that ISM can be used to solve general classes of IKDR learning paradigms.
- We present experimental evidence to highlight the generalization of ISM to a wide range of learning paradigms under a family of kernels and demonstrate its efficiency in terms of speed and better accuracies compared to competing methods.

5.1 A Review for the Iterative Spectral Method (ISM)

Let $X \in \mathbb{R}^{n\times d}$ be a dataset of n samples with d features and let $Y \in \mathbb{R}^{n\times k}$ be the corresponding labels where k denotes the number of classes. Given $\kappa_X(\cdot,\cdot)$ and $\kappa_Y(\cdot,\cdot)$ as two kernel functions that applies respectively to X and Y to construct kernel matrices $K_X \in \mathbb{R}^{n\times n}$ and $K_Y \in \mathbb{R}^{n\times n}$. Also let H be a centering matrix where $H = I - (1/n)\mathbf{1}_n\mathbf{1}_n^T$ with $H \in \mathbb{R}^{n\times n}$, I as the identity matrix and $\mathbf{1}_n$ as a vector of 1s. HSIC measures the nonlinear dependence between X and Y whose empirical estimate is expressed as $\mathcal{H}(X,Y) = \frac{1}{(1-n)^2}\operatorname{Tr}(HK_XHK_Y)$, with $\mathcal{H}(X,Y) = 0$ denoting complete independence and $\mathcal{H}(X,Y) \gg 0$ as high dependence [8].

A general IKDR problem can be posed as discovering a subspace $W \in \mathcal{R}^{d\times q}$ such that $\mathcal{H}(XW,Y)$ is maximized. Since W induces a reduction of dimension, we can assume that $q < d$. To prevent an unbounded solution, the subspace W is constrained such that $W^TW = I$. Since this formulation has a wide range of applications across different learning paradigms, our work investigates the commonality of these problems and discovers that various learning IKDR paradigms can be expressed as the following optimization problem:

$$\max_W \operatorname{Tr}(\Gamma K_{XW}) \quad \text{s.t. } W^TW = I, \tag{5.1}$$

where Γ is a symmetric matrix commonly derived from K_Y. Although this objective is shared among many IKDR problems, the highly non-convex objective continues to pose a serious challenge. Therefore the realization of ISM's ability to solve Eq. (5.1) impacts many applications.

The ISM algorithm solves Eq. (5.1) by setting the q most dominant eigenvectors of a special matrix Φ as its solution W; we define Φ in a later section. We denote these eigenvectors in our context as V_{max} and their eigenvalues as Λ. Since Φ is a function of W, the new W is used to construct the next Φ which we again set its V_{max} as the next W. This process iterates until the change in Λ between each iteration falls below a predefined threshold δ. To initialize the first W, the 2nd order Taylor expansion is used to approximate Φ which yields a matrix Φ_0 that is independent of W. We supply its pseudo-code in Algorithm 4.

Algorithm 4: Generalized ISM Algorithm

Input : Data X, kernel, Subspace Dimension q

Output : Projected subspace W

Initialization : Initialize Φ_0 using Table 5.1.

Set W_0 to V_{max} of Φ_0.

while $||\Lambda_i - \Lambda_{i-1}||_2/||\Lambda_i||_2 < \delta$ **do**

  Compute Φ using Table 5.2

  Set W_k to V_{max} of Φ

Kernel	Approximation of Φs
Linear	$\Phi_0 = X^T \Gamma X$
Squared	$\Phi_0 = X^T \mathcal{L}_\Gamma X$
Polynomial	$\Phi_0 = X^T \Gamma X$
Gaussian	$\Phi_0 = -X^T \mathcal{L}_\Gamma X$
Multiquadratic	$\Phi_0 = X^T \mathcal{L}_\Gamma X$

Table 5.1: Equations for the approximate Φs for the common kernels.

We have discovered that there exists a family of kernels where each kernel possesses its own distinct pair of Φ/Φ_0 matrices. From our proof, we discovered a general formulation of Φ/Φ_0 for any kernel within the family. Moreover, since the only change is the Φ/Φ_0 pair, the ISM algorithm holds by

Kernel	Φ Equations
Linear	$\Phi = X^T \Gamma X$
Squared	$\Phi = X^T \mathcal{L}_\Gamma X$
Polynomial	$\Phi = X^T \Psi X \quad , \quad \Psi = \Gamma \odot K_{XW,p-1}$
Gaussian	$\Phi = -X^T \mathcal{L}_\Psi X$, $\Psi = \Gamma \odot K_{XW}$
Multiquadratic	$\Phi = X^T \mathcal{L}_\Psi X$, $\Psi = \Gamma \odot K_{XW}^{(-1)}$

Table 5.2: Equations for Φs for the common kernels.

simply substituting the appropriate Φ/Φ_0 matrices based on the kernel. We have derived several examples of Φ_0/Φ in Tables 5.1 and 5.2 respectively and supplied the derivation *for each kernel* in Appendices C.2 and C.3.

5.2 Φ for Common Kernels.

After deriving Φ/Φ_0 pairs for the most common kernels, we note several recurrent characteristics. First, Φ scales with the dimension d instead of the size of the data n. Since $n \gg d$ is common across many datasets, the eigendecomposition performed on $\Phi \in \mathcal{R}^{d\times d}$ can be significantly faster while requiring less memory. Second, following Eq. (5.2), they are highly efficient to compute since a vectorized formulation of Φ can be derived for each kernel as shown in Table 5.2; commonly, they reduce to a dot product between a Laplacian matrix $\mathcal{L}$ with the data matrices X. The occurrence of the Laplacian is particularly surprising since nowhere in Eq. (5.1) suggests this relationship. Third, observe from Table 5.2 that Φ can be expressed as $X^T\Omega X$, where Ω is a positive semi-definite (PSD) matrix. Since the formulation of Φ without Ω is the covariance matrix X^TX, Ω scales the covariance matrix by incorporating both the kernel and the label information. In addition, by applying the Cholesky decomposition on Ω to rewrite Φ as $(X^TL)(L^TX)$, L becomes a matrix that adjusts the data itself. Therefore, IKDR can be interpreted as applying PCA on the adjusted data L^TX where the kernel and label information is included.

5.3 Extending the ISM Theoretical Guarantees.

The ISM at its inception only proved that a fixed point W^* of Algorithm 4 is a local maximum of Eq. (5.1) *only if* the Gaussian kernel is used. Our work later extends the theorem to a family of kernels which we refer to as the ISM family. Here, we supply the theoretical foundation for this claim by first providing the following definition.

Definition 9. *Given $\beta = a(x_i, x_j)^TWW^Tb(x_i, x_j)$ with $a(x_i, x_j)$ and $b(x_i, x_j)$ as functions of x_i and x_j, any twice differentiable kernel that can be written in terms of $f(\beta)$ while retaining its symmetric positive semi-definite property is an ISM kernel belonging to the ISM family with an*

associated Φ *matrix defined as*

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}f'(\beta)A_{i,j}. \tag{5.2}$$

where $A_{i,j} = b(x_i, x_j)a(x_i, x_j)^T + a(x_i, x_j)b(x_i, x_j)^T$.

Kernel Name	$f(\beta)$	$a(x_i, x_j)$	$b(x_i, x_j)$
Linear	β	x_i	x_j
Squared	β	$x_i - x_j$	$x_i - x_j$
Polynomial	$(\beta + c)^p$	x_i	x_j
Gaussian	$e^{\frac{-\beta}{2\sigma^2}}$	$x_i - x_j$	$x_i - x_j$
Multiquadratic	$\sqrt{\beta + c^2}$	$x_i - x_j$	$x_i - x_j$

Table 5.3: Converting common kernels to $f(\beta)$.

Since the equation for different kernels varies vastly, it is not clear how they can be reformulated into a single structure that simultaneously satisfies all ISM guarantees. Definition 9 is the key realization that unites a set of kernels into a family. Under this definition, we proved later in Theorem 5.1 that the Gaussian kernel within the original proof of ISM can be replaced by $f(\beta)$. Therefore, the ISM guarantees simultaneously extend to any kernel that satisfies Definition 9. As a result, a general Φ for *any ISM kernel* can be derived as shown in Eq. (5.2). Moreover, note that the family of potential kernels is not limited to a finite set of known kernels, instead, it extends to any conic combinations of ISM kernels. We prove in Appendix C.12 the following proposition.

Proposition 1. *Any conic combination of ISM kernels is still an ISM kernel.*

Properties of Φ**.** Since each ISM kernel is coupled with its own Φ matrix, Φs can conceptually replace kernels. Recall that the $V_{\max}$ of Φ *for any kernel in the ISM family* is the local maximum of Eq. (5.1). This central property is established in the following two theorems.

Theorem 5.1. *Given a full rank* Φ *with an eigengap as defined by Eq. (C.73) in Appendix C.4, a fixed point* W^* *of Algorithm 4 satisfies the 2nd Order Necessary Conditions (Theorem 12.5 [24]) for Eq. (5.1) using any ISM kernel.*

Theorem 5.2. *A sequence of subspaces* $\{W_k W_k^T\}_{k\in\mathbb{N}}$ *generated by Algorithm 4 contains a convergent subsequence.*

Since the entire ISM proof along with its convergence guarantee is required to be revised and generalized under Definition 9, we leave the detail to Appendix C.4 and C.13 while presenting here only the main conclusions. Functionally, our proof is separated into two lemmas to establish Φ as a kernel surrogate. Lemma 1 concludes that given any Φ of an ISM kernel, the gradient of the Lagrangian for Eq. (5.1) is equivalent to $-\Phi W - W\Lambda$. Therefore, when the gradient is set to 0, the eigenvectors of Φ is equivalent to the stationary point of Eq. (5.1). For Lemma 2, given $\bar{\Lambda}$ as the eigenvalues associated with the eigenvectors *not chosen* and $\mathcal{C}$ as constant, it concludes that the 2nd order necessary condition is satisfied when $(\min_i \bar{\Lambda}_i - \max_j \Lambda_j) \geq \mathcal{C}$. This inequality indicates the necessity for the smallest eigenvalue among the un-chosen eigenvectors to be greater than the maximum eigenvalue of the chosen by at least $\mathcal{C}$. Therefore, given the choice of q eigenvectors, the q smallest eigenvalues will maximize the gap. This is equivalent to finding the most dominant eigenvectors of Φ. Putting both lemmas together, we conclude that the most dominant eigenvectors of any Φ within the ISM family is the solution to Eq. (5.1).

Initializing W with Φ_0. After generalizing ISM, different Φs may or may not be a function of W. When Φ is not a function of W, the V_{max} of Φ is immediately the solution. However, if Φ is a function of W, Φ iteratively updates from the previous W. This process is initialized using a Φ_0 that is independent of W. To obtain Φ_0, ISM approximates the Gaussian kernel up to the 2nd order of the Taylor series around $\beta = 0$ and discovers that the approximation of Φ is independent of W. Our work leverages Definition 9 and proves that a common formulation for Φ_0 is possible. We formalize our finding in the following theorem and provided the proof in Appendix C.6.

Theorem 5.3. *For any kernel within the ISM family, a Φ independent of W can be approximated with*

$$\Phi \approx \text{sign}(\nabla_\beta f(0)) \sum_{i,j} \Gamma_{i,j} A_{i,j}. \tag{5.3}$$

5.4 Extending ISM to Conic Combination of Kernels.

The two lemmas of Theorem 5.1 highlights the conceptual convenience of working with Φ in place of kernels. This conceptual replacement extends even to conic combinations of ISM kernels. As a corollary to Theorem 5.1, we discovered that when a kernel is constructed through a conic combination of ISM kernels, it also has an associated Φ matrix. Remarkably, it is equivalent to the

conic combination of Φs from individual kernels using the same coefficients. Formally, we propose the following corollary with its proof in Appendix C.11.

Corollary 1. *The* Φ *matrix associated with a conic combination of kernels is the conic combination of* Φ*s associated with each individual kernel.*

5.5 Interpretable Kernel Dimension Reduction Experimental Results

Datasets. The experiment includes 5 real datasets of commonly encountered data types. Wine [44] consists of continuous data while the Cancer dataset [43] features are discrete. The Face dataset [35] is a standard dataset used for alternative clustering; it includes images of 20 people in various poses. The MNIST [64] dataset includes images of handwritten characters. The Face and the MNIST datasets are chosen to highlight ISM's ability to handle images. The Flower image by Alain Nicolas [36] is another dataset chosen for alternative clustering where we seek alternative ways to perform image segmentation. For more in-depth details on each dataset, see Appendix C.9.

Experimental Setup. We showcase ISM's efficacy on three different learning paradigms, i.e., supervised dimension reduction[11], unsupervised clustering [12], and semi-supervised alternative clustering [14]. As an optimization technique, we compare ISM in Table 5.4 against competing state-of-the-art manifold optimization algorithms: Dimension Growth **(DG)** [18], the Stiefel Manifold approach **(SM)** [17], and the Grassmann Manifold **(GM)** [62, 65]. To emphasize ISM family of kernels, the supervised and unsupervised results using several less conventional kernels are included in Table 5.5. Within this table, we also investigate using conic combination of Φs by combining the Gaussian and the polynomial kernels with center alignment [66]. Since center alignment is specific to supervised cases, this is not repeated for the unsupervised case.

For *supervised* dimension reduction, we perform SVM on XW using 10-fold cross validation. We repeat this process for each fold of cross-validation. From the 10-fold results in Table 5.4, we record the mean and the standard deviation of the run-time, cost, and accuracy. We investigate the scalability in Figure 5.2 by comparing the change in run-time as we increment the sample size. For *unsupervised* dimension reduction, we perform spectral clustering on XW after learning W where we record the run-time, cost, and NMI. For *alternative clustering*, we highlight the ISM family of kernels by reproducing the original ISM results (generated with Gaussian kernel) using the polynomial kernel.

On the Flower image, each sample is a $\mathcal{R}^3$ vector. We supply the original image segmentation result as semi-supervised labels and learn an alternative way to segment the image. The original segmentation and the alternative segmentation are shown in Figure5.1. For the Face dataset, each sample is a vector vectorized from a grayscaled image of individuals. We provide the identity of individuals as the original clustering label and search for an alternative way to cluster the data.

Evaluation Metric. In the supervised case, the test classification accuracy from the 10-fold cross validation is recorded along with the cost and run-time. The time is broken down into days (d), hours (h), minutes (m), and seconds (s). The best results are bold for each experiment. In the unsupervised case, we report the Normalized Mutual Information (NMI) [51] to compare the clustering labels against the ground truth. For a formal definition of NMI, refer to Section.2.5.

Experiment Settings. The median of the pair-wise Euclidean distance is used as σ for all experiments using the Gaussian kernel. Degree of 3 is used for all polynomial kernels. The dimension of subspace q is set to the number of classes/clusters. The convergence threshold δ is set to 0.01. All competing algorithms use their default initialization. All datasets are centered to 0 and scaled to a standard deviation of 1. All sources are written in Python using Numpy and Sklearn [67, 68]. All experiments were conducted on Dual Intel Xeon E5-2680 v2 @ 2.80GHz, with 20 total cores. Due to limited computational resources, each run is limited to 3 days.

Complexity Analysis of Competing Methods. The run-time as a function of linearly increasing sample size is shown for the polynomial kernel in Figure 5.2. Since the complexity analysis for ISM suggests a relationship of $O(n^2)$ with respect to the sample size, $log_2(.)$ is used for the Y-axis. As expected, the ISM's linear run-time growth in Figure 5.2 supports our analysis of $O(n^2)$ relationship. The plot for competing algorithms reported a similar linear relationship with comparable slopes. This indicates that the difference in speed is not a function of the data size, but other factors such as q and t. Using **DG**'s complexity of $O(n^2dq^2t)$ as an example, it normally converges when t is in the ranges of thousands. Since $q = 20$ was used in the figure, the significant speed improvement from ISM can be derived from the q^2t factor since ISM generally converges at t less than 5.

Results. Comparing against other optimization algorithms in Table 5.4, the results confirm ISM as a significantly faster algorithm while consistently achieving a lower cost. This disparity is especially prominent when the data dimension q is higher. We highlight that for the Face dataset on the Gaussian kernel, it took DG 1.92 days, while ISM finished within 0.99 seconds: a 10^5-fold speed difference.

To further confirm these advantages, the same experiment is repeated using the *polynomial kernel* where similar results can be observed. Besides the execution time and cost, the classification accuracy across 5 datasets never falls below 95% in the supervised setting. The same datasets and techniques are repeated in an unsupervised clustering problem. While the clustering quality is comparable across the datasets, ISM clearly produces the lowest cost with the fastest execution time.

Table 5.5 focuses on the generalization of ISM to a family of kernels. Since Table 5.4 already supplied results from the Gaussian and polynomial kernel, we feature 4 more kernels to support the claim. As kernel methods treat kernels as interchangeable components of the algorithm, ISM achieves a similar effect by replacing the Φ matrix. As evidenced from the table, similar accuracy and time can be achieved with this replacement without affecting the rest of the algorithm. In many cases, the multiquadratic kernel outperforms even the Gaussian and the polynomial kernel. In a similar spirit, we repeated the same experiments in the unsupervised case and received further confirmation.

To support Corollary 1, results using a Gaussian + polynomial (G+P) kernel is also supplied in Table 5.5. It is not surprising that a combination of Φs is the best performing kernel. Since the union of the two kernels covers a larger feature space, the expressiveness is also greater. This result supports the claim that a conic combination of Φs can replace the same combination of kernels for Eq. (5.1).

To study the generalized ISM on a (semi-supervised) alternative clustering problem, we use it to recreate the results from the original paper on alternative clustering. We emphasize that our results differ in the choice of using the polynomial kernel instead of the Gaussian. From the Flower experiment, it is visually clear that the original image segmentation of 2 clusters (separated by black and white) is completely different from the alternative segmentation. For the Face data, the original clusters were grouped by the identity of the individuals while the algorithm produced 4 alternative clusters. By averaging the images of each alternative cluster, the new clustering pattern can be visually seen in Figure 5.1; the samples are alternatively clustered by the pose.

5.6 IKDR Summary

We have extended the theoretical guarantees of ISM to a family of kernels beyond the Gaussian kernel via the discovery of the Φ matrix. Our theoretical analysis proves that the family of ISM

Supervised		Gaussian				polynomial			
		ISM	DG	SM	GM	ISM	DG	SM	GM
Wine	Time	**0.02s ± 0.01s**	7.9s ± 2.9s	1.7s ± 0.7s	16.8m ± 3.4s	**0.02s ± 0.0s**	13.2s ± 6.2s	14.77s ± 0.6s	16.82m ± 3.6s
	Cost	**-1311 ± 26**	-1201 ± 25	-1310 ± 26	-1307 ± 25	**-114608 ± 1752**	-112440 ± 1719	-111339 ± 1652	-108892 ± 1590
	Accuracy	**95.0% ± 5%**	93.2% ± 5.5%	**95% ± 4.2%**	**95% ± 6%**	**97.2% ± 3.7%**	93.8% ± 3.9%	96.6% ± 3.7%	96.6% ± 2.7%
Cancer	Time	**0.08s ± 0.0s**	4.5m ± 103s	17s ± 12s	17.8m ± 80s	**0.13s ± 0.0s**	4m ± 1.2m	3.3m ± 3s	17.5m ± 1.1m
	Cost	**-32249 ± 338**	-30302 ± 2297	-31996 ± 499	-30998 ± 560	**-1894 ± 47**	-1882 ± 47	-1737 ± 84	-1690 ± 108
	Accuracy	97.3%± 0.3%	97.3%± 0.3%	97.3%± 0.2%	**97.4%± 0.4%**	**97.4%± 0.3%**	97.3% ± 0.3%	**97.4% ± 0.3%**	97.3% ± 0.3%
Face	Time	**0.99s ± 0.1s**	1.92d ± 11h	10s ± 5s	22.7m ± 18s	**0.7s ± 0.03s**	2.1d ± 13.9h	5.0m ± 5.7s	21.5m ± 9.8s
	Cost	**-3754 ± 31**	-3431 ± 32	-3749 ± 33	-771 ± 28	**-82407 ± 1670**	-78845 ± 1503	-37907 ± 15958	-3257 ± 517
	Accuracy	**100% ± 0%**	**100% ± 0%**	**100% ± 0%**	99.2% ± 0.2%	**100% ± 0%**	**100% ± 0%**	**100% ± 0%**	99.8% ± 0.2%
MNIST	Time	**13.8s ± 2.3s**	¿ 3d	2.5m ± 1.0s	¿ 3d	**12.1s ± 1.4s**	¿ 3d	2.1m ± 3s	¿ 3d
	Cost	**-639 ± 2.3**	N/A	-621 ± 5.1	N/A	**-639 ± 2**	N/A	-620 ± 5.1	N/A
	Accuracy	**99% ± 0%**	N/A	98.5% ± 0.4%	N/A	**99% ± 0%**	N/A	**99% ± 0%**	N/A
Unsupervised									
Wine	Time	**0.01s**	9.9s	0.6s	16.7m	**0.02s**	14.4s	2.9s	33.5m
	Cost	**-27.4**	-25.2	-27.3	-27.3	**-1600**	-1582	-1598	-1496
	NMI	**0.86**	**0.86**	**0.86**	**0.86**	**0.84**	**0.84**	**0.84**	0.83
Cancer	Time	**0.57s**	4.3m	3.9s	44m	**0.5s**	8.0m	8.8m	41m
	Cost	**-243**	-133	-146	-142	**-15804**	-14094	-15749	-11985
	NMI	**0.8**	0.79	**0.8**	0.79	0.79	**0.80**	0.79	**0.80**
Face	Time	**0.3s**	1.3d	5.3s	55.9m	**1.0s**	¿ 3d	22m	1.6d
	Cost	**-169.3**	-167.7	-168.9	-37	**-368**	NA	-348	-321
	NMI	0.94	**0.95**	0.93	0.89	**0.94**	N/A	0.89	0.89
MNIST	Time	**1.8h**	> 3d	1.3d	> 3d	**8.3m**	> 3d	0.9d	> 3d
	Cost	**-2105**	N/A	-2001	N/A	**-51358**	N/A	-51129	N/A
	NMI	**0.47**	N/A	0.46	N/A	**0.32**	N/A	**0.32**	N/A

Table 5.4: Run-time, cost, and objective performance are recorded under supervised/unsupervised objectives. ISM is significantly faster compared to other optimization techniques while achieving lower objective cost.

kernels extend even to conic combinations of ISM kernels. With this extension, ISM becomes an efficient solution for a wide range of supervised, unsupervised and semi-supervised applications. Our experimental results confirm the efficiency of the algorithm while showcasing its wide impact across many domains.

		Supervised				Unsupervised			
		Linear	**Squared**	**Multiquad**	**G+P**		**Linear**	**Squared**	**Multiquad**
Wine	**Time**	**0.003s ± 0s**	0.01s ± 0s	0.02s ± 0.01s	0.007s ± 0s	**Time**	**0.02s**	0.04s	0.06s
	Accuracy	97.2% ± 2.8%	96.6% ± 3.7%	97.2% ± 3.7%	**98.3% ± 2.6%**	**NMI**	0.85	0.85	**0.88**
Cancer	**Time**	**0.02s ± 0.002s**	0.09s ± 0.02s	0.15s ± 0.01s	0.06s ± 0.004s	**Time**	**0.23s**	0.5s	0.56s
	Accuracy	97.2% ± 0.3%	97.3% ± 0.04%	**97.4% ± 0.003%**	**97.4% ± 0.003%**	**NMI**	0.80	0.79	**0.84**
Face	**Time**	**0.2s ± 0.2s**	0.3s ± 0.2s	0.3s ± 0.2s	0.5s ± 0.03s	**Time**	**0.68s**	0.92s	3.7s
	Accuracy	97.3% ± 0.3%	97.1% ± 0.4%	97.3% ± 0.4%	**100% ± 0%**	**NMI**	0.93	**0.95**	0.92
MNIST	**Time**	**6.4s ± 0.4s**	17.4s ± 0.4s	10.6m ± 1.9m	17.6s ± 2.5s	**Time**	**3.1m**	4.7m	52m
	Accuracy	99.1% ± 0.1%	**99.3% ± 0.2%**	99.1% ± 0.1%	**99.3% ± 0.2%**	**NMI**	**0.54**	**0.54**	**0.54**

Table 5.5: Run-time and objective performance are recorded across several kernels within the ISM family. It confirms the usage of Φ or linear combination of Φ in place of kernels.

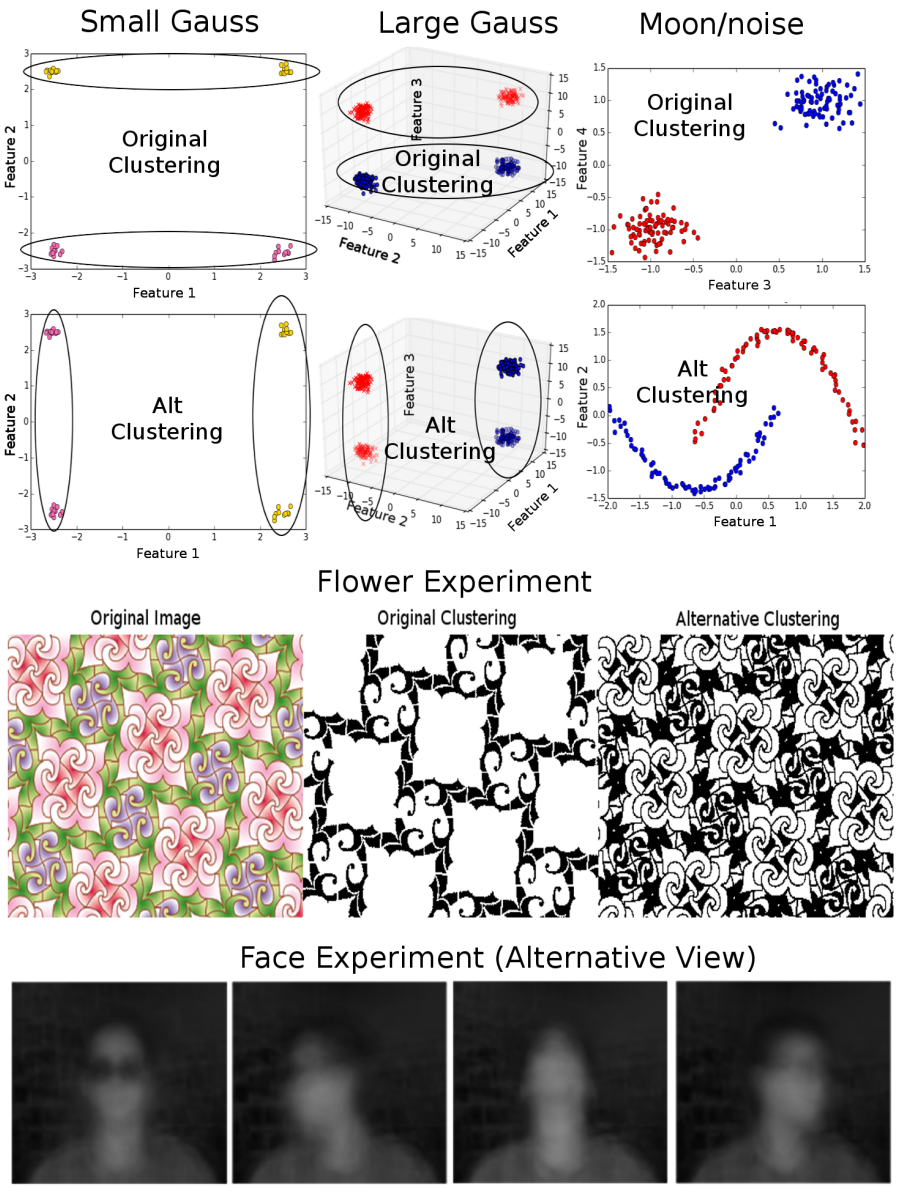

Figure 5.1: Reproducing results from the original ISM paper using polynomial kernels.

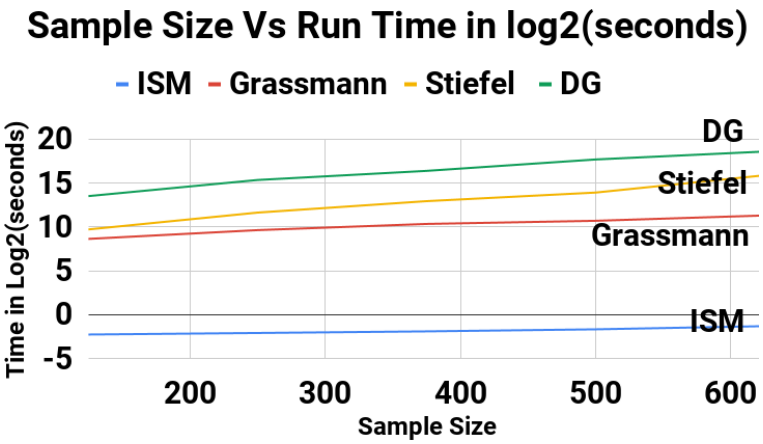

Figure 5.2: Log2 run-time as a function of increasing samples.

Chapter 6

Layer-wise Learning of Kernel Dependence Networks

Since the seminal work by Rumelhart et al. [69], Multilayer Perceptrons (MLPs) have become a popular tool for classification. This success can in part be explained by the expressiveness of the resulting function class [70, 71, 72, 73]; in particular, a two-layer network can approximate any continuous function in a compact domain, to any desired accuracy, albeit with a network size exponential in input dimension. A helpful perspective on networks of large (or, in the limit, even infinite) width can be gained from kernel methods, for which the feature spaces can be infinite-dimensional by construction. For example, the Gaussian process (GP) has been used to understand the limiting behavior of wide networks [74, 75, 76]: in particular, deep GPs give a mechanism for constructing deep networks where each layer has infinitely many features [77, 78, 79, 80]. Following GPs, the Neural Tangent Kernel (NTK) was then proposed to describe the dynamics of the network during training [81, 82], further elucidating the relationship between MLPs and kernels. Moreover, Belkin et al. [83] has shown how MLPs and kernels both yield good generalization results despite overfitting, leading them to link kernel discovery to network training.

In this paper, we introduce a new function class for supervised classification, which builds a kernel neural network in a greedy layer-wise fashion: we call this the Kernel Dependence Network (*KNet*). Each layer is a composition of a linear subspace projection and an infinite-dimensional feature map. A key advantage is that the network can be trained greedily, layer by layer. *KNet* solves each layer

by maximizing the dependence between the layer output and the labels, using the Hilbert Schmidt Independence Criterion (HSIC) as the dependence measure [8]. This can be achieved efficiently, even without Stochastic Gradient Descent (SGD), via the Iterative Spectral Method (ISM) [9, 84]. This local and greedy strategy is used in place of backpropagation (BP) to obviate the sharing of the gradient information throughout the network, thereby avoiding exploding/vanishing gradients. As a consequence of our formulation, we demonstrate how the natural choice of the network width and depth also emerges.

A related work by Ma et al. [85] uses a chain of HSIC dependencies to simulate Information Bottleneck. They employ a standard network structure that is solved by SGD. In contrast, our kernel perspective replaces the conventional activation function with the feature map of a Gaussian kernel (GK), resulting in an infinitely wide network that is solved via the kernel trick. Additionally, our key contribution is the theoretical characterization of the richness of the function class described by our architecture. We prove a property related to that of *finite sample expressivity* [19, Section 4] for classical neural nets. Specifically, our network construction can achieve any desirable *training accuracy* given a minimum depth of 2 infinitely wide layers, with the implication of minimizing Mean Squared Error (MSE) and Cross-Entropy (CE) on the training data.

Similar to traditional MLPs, the richness of *KNet* promises to fit any training data with an overparameterized network, yet it also experimentally generalizes well on test data. To explain this observation for standard MLPs, the regularizing effects of architecture choice and optimization strategies have been used to explore how MLPs generalize [20, 21, 22]. As our second theoretical contribution, we demonstrate that our architecture and training procedure perform an implicit regularization, similar to the weight penalization arguments of Poggio et al. [23], and we describe the mechanisms by which this is achieved. We lastly verify every theoretical guarantee of our model experimentally, where *KNet* achieves comparable generalization results to MLPs of comparable complexity trained by BP.

6.1 Network Model

Let $X \in \mathbb{R}^{n \times d}$ be a dataset of n samples with d features and let $Y \in \mathbb{R}^{n \times \tau}$ be the corresponding one-hot encoded labels with τ number of classes. Let $\odot$ be the element-wise product. The i^{th} sample and label of the dataset is written as x_i and y_i. H is a centering matrix defined as $H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$

where I_n is the identity matrix of size $n \times n$ and $\mathbf{1}_n$ is a vector of 1s also of length n. Given H, we let $\Gamma = HYY^TH$ and let $K_{(\cdot)}$ be the Gaussian kernel (GK) matrix computed using a dataset matrix $(\cdot)$.

We denote the MLP weights as $W_1 \in \mathbb{R}^{d\times q}$ and $W_l \in \mathbb{R}^{m\times q}$ for the 1st layer and the l^{th} layer; assuming $l > 1$. The input and output at the l^{th} layer are $R_{l-1} \in \mathbb{R}^{n\times m}$ and $R_l \in \mathbb{R}^{n\times m}$, i.e., given $\psi : \mathbb{R}^{n\times q} \to \mathbb{R}^{n\times m}$ as the activation function, $R_l = \psi(R_{l-1}W_l)$. For each layer, the i^{th} row of its input R_{l-1} is $r_i \in \mathbb{R}^m$ and it represents the i^{th} input sample. We denote $\mathcal{W}_l$ as a function where $\mathcal{W}_l(R_{l-1}) = R_{l-1}W_l$; consequently, each layer is also a function $\phi_l = \psi \circ \mathcal{W}_l$. By stacking L layers together, the entire network itself becomes a function ϕ where $\phi = \phi_L \circ ... \circ \phi_1$. Given an empirical risk $(\mathcal{H})$ and a loss function $(\mathcal{L})$, our network model assumes an objective of

$$\min_{\phi} \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\phi(x_i), y_i). \tag{6.1}$$

We propose to solve Eq. (6.1) greedily; this is equivalent to solving a sequence of *single-layered* networks where the previous network output becomes the current layer's input. At each layer, we find the W_l that maximizes the dependency between the layer output and the label via HSIC [8]:

$$\max_{W_l} \operatorname{Tr}\left(\Gamma \ \left[\psi(R_{l-1}W_l)\psi^T(R_{l-1}W_l)\right]\right) \quad \text{s. t.}\, W_l^TW_l = I. \tag{6.2}$$

Deviating from the traditional concept of activation functions, we use the GK feature map in their place, simulating an infinitely wide network. Yet, the kernel trick spares us the direct computation of $\psi(R_{l-1}W_l)\psi^T(R_{l-1}W_l)$; we instead compute the GK matrix given $\mathcal{K}(W_l^Tr_i, W_l^Tr_j) = \exp\{-||W_l^Tr_i - W_l^Tr_j||^2/2\sigma^2\}$. We also deviate from a standard MLP by restricting our solution space to a linear subspace. If the solution indeed lives on a linear subspace independent of their scale as suggested by [23, 86, 87], then we can exploit this prior knowledge to narrow the search space during optimization specific to the Stiefel Manifold, i.e., by adding the constraint of $W_l^TW_l = I$. This prior enables us to solve Eq. (6.2) by leveraging the iterative spectral method (ISM) proposed by Wu et al. [9, 84] to simultaneously avoid SGD and identify the network width. Applying ISM to our model, each layer's weight is initialized using the most dominant eigenvectors of

$$\mathcal{Q}_{l^0} = R_{l-1}^T(\Gamma - \text{Diag}(\Gamma 1_n))R_{l-1}, \tag{6.3}$$

where the Diag$(\cdot)$ function places the elements of a vector into the diagonal of a square matrix with zero elements. Once the initial weights W_{l^0} are set, ISM iteratively updates $W_{l^{\text{i}}}$ to $W_{l^{\text{i}+1}}$ by setting

$W_{l^{\mathbf{i}+1}}$ to the most dominant eigenvectors of

$$\mathcal{Q}_{l^i} = R_{l-1}^T(\hat{\Gamma} - \text{Diag}(\hat{\Gamma}1_n))R_{l-1}, \tag{6.4}$$

where $\hat{\Gamma}$ is a function of $W_{l^{\mathbf{i}}}$ computed with $\hat{\Gamma} = \Gamma \odot K_{R_{l-1}W_{l^{\mathbf{i}}}}$. This iterative weight-updating process stops when $\mathcal{Q}_{l^{\mathbf{i}}} \approx \mathcal{Q}_{l^{\mathbf{i}+1}}$, whereupon $\mathcal{Q}_{l^{\mathbf{i}+1}}$ is set to $\mathcal{Q}_l^*$, and its most dominant eigenvectors W_l^* becomes the solution of Eq. (6.2).

ISM solves Eq. (6.2) directly on an infinitely wide network during training, obtaining W_l^*. However, during test, we approximate ψ with Random Fourier Features (RFF) [88], simulating a finite width network passing samples through W_l^* and ψ. Capitalizing on the spectral properties of ISM, the spectrum of $\mathcal{Q}_l^*$ completely determines the the width of the network $W_l^* \in \mathbb{R}^{m \times q}$, i.e., m is equal to the size of the RFF, and q is simply the rank of $\mathcal{Q}_l^*$. Furthermore, since Eq. (6.2) after normalization is upper bounded by 1, we can stop adding new layers when the HSIC value of the current layer approaches this theoretical bound, thereby prescribing a natural depth of the network. The resulting network ϕ after training will map samples of the same class into its own cluster, allowing the test samples to be classified by matching their network outputs to the nearest cluster center. We formally define the hyperparameter settings and provide Algorithm 5 in the Experimental section. The source code is also publicly available at `https://github.com/anonymous`.

6.2 Theoretical Origin of Kernel Dependence Networks

6.2.1 Background and Notations.

Let $\mathcal{S}$ be a set of i, j sample pairs that belong to the same class. Its complement, $\mathcal{S}^c$ contains all sample pairs from different classes. We denote compositions of the first l layers as $\phi_{l^\circ} = \phi_l \circ ... \circ \phi_1$ where $l \leq L$. This notation enables us to connect the data directly to the layer output where $R_l = \phi_{l^\circ}(X)$. Since *KNet* is greedy, it solves MLPs by replacing ϕ in Eq. (6.1) incrementally with a sequence of functions $\{\phi_{l^\circ}\}_{l=1}^L$ where each layer relies on the weights of the previous layer. This implies that we are also solving a sequence of empirical risks $\{\mathcal{H}_l\}_{l=1}^L$, i.e., different versions of Eq. (6.1) given the current ϕ_{l°. We refer to $\{\phi_{l^\circ}\}_{l=1}^L$ and $\{\mathcal{H}_l\}_{l=1}^L$ as the *Kernel Sequence* and the *$\mathcal{H}$-Sequence*.

6.2.2 Classification Strategy.

Classification tasks can be solved using objectives like MSE and CE to match the network output $\phi(X)$ to the label Y. While this approach achieves the desirable outcome, it also constrains the space of potential solutions where $\phi(X)$ must match Y. Yet, if ϕ maps X to the labels $\{0, 1\}$ instead of the true label $\{-1, 1\}$, $\phi(X)$ may not match Y, but the solution is the same. Therefore, enforcing $\phi(X) = Y$ ignores an entire space of solutions that are functionally equivalent. We posit that by relaxing this constraint and accept a larger space of potential global optima, it will be easier during optimization to collide with this space. This intuition motivates us to depart from the tradition of label matching, and instead seek out alternative objectives that focus on solving the underlying prerequisite of classification, i.e., learning a mapping of X where similar and different classes become distinguishable.

Since there are many notions of similarity, it is not always clear which is best for a particular situation. *KNet* overcomes this uncertainty by discovering the optimal similarity measure as a kernel function during training. To understand how, first realize that the i, j_{th} element of Γ, denoted as $\Gamma_{i,j}$, is a positive value for samples in $\mathcal{S}$ and negative for $\mathcal{S}^c$. By defining a kernel function $\mathcal{K}$ as a similarity measure between 2 samples, Eq. (6.2) becomes

$$\max_{W_l} \sum_{i,j \in \mathcal{S}} \Gamma_{i,j} \mathcal{K}_{W_l}(r_i, r_j) - \sum_{i,j \in \mathcal{S}^c} |\Gamma_{i,j}| \mathcal{K}_{W_l}(r_i, r_j). \tag{6.5}$$

Notice that the objective uses the sign of $\Gamma_{i,j}$ as labels to guide the choice of W_l such that it increases $\mathcal{K}_{W_l}(r_i, r_j)$ when r_i, r_j belongs to $\mathcal{S}$ while decreasing $\mathcal{K}_{W_l}(r_i, r_j)$ otherwise. Therefore, by finding a W_l matrix that best parameterizes $\mathcal{K}$, HSIC discovers the optimal pair-wise relationship function $\mathcal{K}_{W_l}$ that separates samples into similar and dissimilar partitions. Given this strategy, we will formally demonstrate how learning $\mathcal{K}$ leads to classification in the following sections.

6.2.3 Optimization Strategy.

MLP is traditionally solved with all the layers jointly via BP. We instead focus on greedily discovering a *Kernel Sequence* that compels the $\mathcal{H}$*-Sequence* to exhibit key behaviors that enable classification. Here, we discuss how these behaviors lead to an optimal solution.

First, the $\mathcal{H}$-*Sequence* must be convergent. We accomplish this by leveraging the Monotone Convergence Theorem (MCT) [89]: it states that a monotone sequence is guaranteed to have a limit if and only if the sequence is bounded. For *KNet*, since ψ is the feature map of GK, $\mathcal{K}_{W_l}(r_i, r_j)$ is naturally constrained between $[0, 1]$. Therefore, given our greedy strategy, $\mathcal{H}$-*Sequence* converges if we can achieve $\mathcal{H}_l \geq \mathcal{H}_{l-1}$ at every layer.

Second, while the MCT guarantees convergence, it does not guarantee the quality of its limit. Namely, the improvement at each layer could be so small such that the overall gain is trivial. To answer this question, we must investigate the potential contribution from each layer to identify the point of convergence. In the most ideal case, as $L \to \infty$ the network should converge towards an optimal kernel that achieves the theoretical upper limit of HSIC, or $\mathcal{H}^* = \sum_{i,j \in \mathcal{S}} \Gamma_{i,j}$. Moreover, we should be able to achieve this within a finite number of layers.

We investigate these criteria using HSIC as the empirical risk of an MLP and prove that by using the feature map of a GK as ψ, there exists a sequence of weights $\{W_l\}_{l=1}^{L}$ where these considerations are simultaneously satisfied. We state this theorem below and provide its proof in App. D.1.

Theorem 6.1. *For any $\mathcal{H}_0$, there exists a Kernel Sequence $\{\phi_{l^\circ}\}_{l=1}^{L}$ parameterized by a set of weights W_l and a set of bandwidths σ_l such that:*

I. $\mathcal{H}_L$ can approach arbitrarily close to $\mathcal{H}^*$ such that for any $L > 1$ and $\delta > 0$ we can achieve

$$\mathcal{H}^* - \mathcal{H}_L \leq \delta, \tag{6.6}$$

II. as $L \to \infty$, the $\mathcal{H}$-Sequence converges to the global optimum, that is

$$\lim_{L \to \infty} \mathcal{H}_L = \mathcal{H}^*, \tag{6.7}$$

III. the convergence is strictly monotonic where

$$\mathcal{H}_l > \mathcal{H}_{l-1} \quad \forall l \geq 1. \tag{6.8}$$

To simplify the proof, Thm. 6.1 use the average directions of each class W_s at each layer when W_s is not guaranteed to be an optimal solution. In spite of this deficit, Thm. 6.1 proves that a global optimum is *attainable given a minimum of two layers*, i.e., for any $L > 1$. While this remarkable

feat is theoretically possible, depending on the data, the σ required for the GK may be extremely small, leading to an undesirably sharp ϕ that is overfitting the noise. Fortunately, this issue is resolved by simply spreading the monotonic improvement across more layers. We have additionally observed experimentally that by replacing the suboptimal W_s with an optimal W_l^* from ISM (where $\partial\mathcal{H}/\partial W(W_l^*) = 0$), large σ values can be used to still achieve convergence with few layers.

6.3 Relating $\mathcal{H}^*$ to Classification.

As $\mathcal{H}_l \to \mathcal{H}^*$, the maximization of Eq. (6.5) demonstrates how we learn the similarity function. However, it may be unclear why learning this function also induces an optimal classification. To understand this deeper, let us first clarify some key notations and concepts. We refer to the image of ψ and ϕ as the Reproducing Kernel Hilbert Space (RKHS) and the image of $\mathcal{W}$ as the Images of the RKHS Dual Space (IDS). This distinction is crucial because each space dictates the geometric orientations that lead to classification. Keeping this in mind, each layer's output can be measured by the within S_w^l and between S_b^l class scatter matrices defined as

$$S_w^l = \sum_{i,j\in\mathcal{S}} W_l^T(r_i - r_j)(r_i - r_j)^T W_l \quad \text{and} \quad S_b^l = \sum_{i,j\in\mathcal{S}^c} W_l^T(r_i - r_j)(r_i - r_j)^T W_l. \tag{6.9}$$

These matrices are historically important [90, 91] because their trace ratio ($\mathcal{T} = \mathrm{Tr}(S_w)/\,\mathrm{Tr}(S_b)$) measures class separability, i.e., a small $\mathcal{T}$ implies a tight grouping of the same class under Euclidean distance. Classification is consequently achieved by mapping different classes into these spatial separations. Crucially, by maximizing $\mathcal{H}$, $\mathcal{T}$ is minimized as a byproduct. In fact, we prove that as $\mathcal{H}_l \to \mathcal{H}^*$, $\mathcal{T}$ approaches 0, and ϕ will map samples of different classes into separated points.

Note that since $\mathcal{T}$ is computed with samples from the image of $\mathcal{W}$, this particular relationship resides in IDS. Concurrently, since the inner product defines similarity in RKHS, samples are simultaneously being partitioned via the angular distance. Indeed, our proof indicates that as $\mathcal{H}_l \to \mathcal{H}^*$, RKHS samples within $\mathcal{S}$ achieves perfect alignment while samples in $\mathcal{S}^c$ become orthogonal to each other, separated by a maximum angle of $\pi/2$. This dual relationship between IDS and RKHS produces a vastly different network output right before and immediately after the final activation function, where different notions of distance (Euclidean and angular) are employed to partition samples. We formally state these results in the following theorem with the proof in App. D.2.

Theorem 6.2. *As $l \to \infty$ and $\mathcal{H}_l \to \mathcal{H}^*$, the following properties are satisfied:*

I the scatter trace ratio $\mathcal{T}$ approaches 0 where

$$\lim_{l\to\infty} \frac{\mathrm{Tr}(S_w^l)}{\mathrm{Tr}(S_b^l)} = 0 \tag{6.10}$$

II the Kernel Sequence converges to the following kernel:

$$\lim_{l\to\infty} \mathcal{K}(x_i, x_j)^l = \mathcal{K}^*(x_i, x_j)^l = \begin{cases} 0 & \forall i, j \in \mathcal{S}^c \\ 1 & \forall i, j \in \mathcal{S} \end{cases} . \tag{6.11}$$

As corollaries to Theorem 6.2, the resulting partition of samples under Euclidean and angular distance implicitly satisfies different classification objectives in each space. In IDS, *KNet* will map a dataset of τ classes into τ distinct points. While these τ points may not match the original label, this difference is inconsequential. In contrast, samples in RKHS at convergence will reside along τ orthogonal axes on a unit sphere. By realigning these results to the standard bases, solutions that simulate the softmax are generated to solve CE. Therefore, as $\mathcal{H}_l \to \mathcal{H}^*$, the maximization of Eq. (6.5) minimizes MSE and CE in different spaces without matching the actual labels itself: instead, we match the underlying geometry of the network output. We state the two corollaries below with their proof in App. D.5.

Corollary 2. *Given $\mathcal{H}_l \to \mathcal{H}^*$, the network output in IDS solves MSE via a translation of labels.*

Corollary 3. *Given $\mathcal{H}_l \to \mathcal{H}^*$, the network output in RKHS solves CE via a change of bases.*

6.4 HSIC and Regularization.

Overparameterized MLPs can generalize without any explicit regularizer [19]. This observation defies classical learning theory and has been a longstanding puzzle in the research community [92, 93, 94]. Overparameterized with infinite width, *KNet* experimentally exhibits a resembling generalization behavior. Moreover, Ma et al. [85] have experimentally observed that HSIC can generalize even without the $W^TW = I$ constraint; we seek to better understand this phenomenon

theoretically. Recently, Poggio et al. [23] have proposed that traditional MLPs generalize because gradient methods implicitly regularize the normalized weights. We discovered a similar relationship with HSIC, i.e., the objective can be reformulated to isolate out n functions $[D_1(W_l), ..., D_n(W_l)]$ that act as a penalty term during optimization. Let $\mathcal{S}|i$ be the set of samples that belongs to the i_{th} class and let $\mathcal{S}^c|i$ be its complement, then each function $D_i(W_l)$ is defined as

$$D_i(W_l) = \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}|i} \Gamma_{i,j} \mathcal{K}_{W_l}(r_i, r_j) - \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}^c|i} |\Gamma_{i,j}| \mathcal{K}_{W_l}(r_i, r_j). \tag{6.12}$$

Notice that $D_i(W_l)$ is simply Eq. (6.5) for a single sample scaled by $\frac{1}{\sigma^2}$. Therefore, as we identify better solutions for W_l, this leads to an increase and decrease of $\mathcal{K}_{W_l}(r_i, r_j)$ associated with $\mathcal{S}|i$ and $\mathcal{S}^c|i$ in Eq. (6.12), thereby increasing the size of the penalty term $D_i(W_l)$. To appreciate how $D_i(W_l)$ penalizes $\mathcal{H}$, we propose an equivalent formulation in the theorem below with its derivation in App D.3.

Theorem 6.3. *Eq. (6.5) is equivalent to*

$$\max_{W_l} \sum_{i,j} \frac{\Gamma_{i,j}}{\sigma^2} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i^T W_l W_l^T r_j) - \sum_i D_i(W_l) ||W_l^T r_i||_2. \tag{6.13}$$

Based on Thm. 6.3, $D_i(W_l)$ adds a negative cost to the sample norm in IDS, $||W_l^T r_i||_2$, suggesting that ISM regularizes *KNet* regardless of $W_l^T W_l = I$. In fact, a better W_l imposes a heavier penalty on Eq. (6.13) where the overall $\mathcal{H}$ may actually decrease.

Complexity Analysis. The complexity analysis of a single ISM iteration as reported by Wu et al. [9] is $O(n^3)$. Since ISM is repeated with L layers, *KNet* complexity is simply $O(Ln^3)$. For memory, *KNet* suffers from the same $O(n^2)$ restriction which all kernel methods inherit.

Limitations of Layer-Wise Kernel Dependence Networks. Although our framework presents many theoretical advantages, we caution that much more research would be required for *KNet* to become practically viable. While ISM resolves many existing problems, it also limits the kernels to the ISM family [84]. Therefore, it currently cannot be extended to solve the traditional activation functions such as relu and sigmoid. The computational and memory complexity is another practical obstacle. Therefore, *KNet* at its current maturity is *intended for analysis* and is *not yet suitable for*

large datasets. Although there are already existing solutions [95, 96] to overcome these challenges, these engineering questions are topics we purposely isolate away from the theory for future research.

6.5 Kernel Dependence Network Experimental Results

Datasets. This work focuses on verifying the theoretical guarantees of a greedily trained network against traditional MLPs of comparable complexity trained by BP. Specifically, we confirm the theoretical properties of *KNet* using three synthetic (Random, Adversarial and Spiral) and five popular UCI benchmark datasets: wine, cancer, car, divorce, and face [44]. They are included along with the source code in the supplementary, and their comprehensive download link and statistics are in App. D.6.

Evaluation Metrics. To evaluate the central claim that MLPs can be solved greedily, we report $\mathcal{H}^*$ at convergence along with the training/test accuracy for each dataset. Here, $\mathcal{H}^*$ is normalized to the range between 0 to 1 using the method proposed by Cortes et al. [66]. To corroborate Corollaries 2 and 3, we also record MSE and CE. To evaluate the sample geometry predicted by Eq. (6.10), we recorded the scatter trace ratio $\mathcal{T}$ to measure the compactness of samples within and between classes. The angular distance between samples in $\mathcal{S}$ and $\mathcal{S}^c$ as predicted by Eq. (6.11) is evaluated with the Cosine Similarity Ratio (C). The equations for $\mathcal{H}^*$ and C are

$$\mathcal{H}^* = \frac{\mathcal{H}(\phi(X), Y)}{\sqrt{\mathcal{H}(\phi(X), \phi(X))\mathcal{H}(Y, Y)}} \quad \text{and} \quad C = \frac{\sum_{i,j \in \mathcal{S}^c} \langle \phi(x_i), \phi(x_j) \rangle}{\sum_{i,j \in \mathcal{S}} \langle \phi(x_i), \phi(x_j) \rangle}. \tag{6.14}$$

Experiment Settings. The width of the network is set by ISM to keep 90% of the data variance. The RFF width is set to 300 for all datasets and the σ_l that maximizes $\mathcal{H}^*$ is chosen. The convergence threshold for $\mathcal{H}$-*Sequence* is set at $\mathcal{H}_l > 0.99$. The network structures discovered by ISM for every dataset are recorded and provided in App. D.7. The MLPs that use MSE and CE have weights initialized via the Kaiming method [97]. All datasets are centered to 0 and scaled to a standard deviation of 1. All sources are written in Python using Numpy, Sklearn and Pytorch [67, 68, 98]. All experiments were conducted on an Intel Xeon(R) CPU E5-2630 v3 @ 2.40GHz x 16 with 16 total cores.

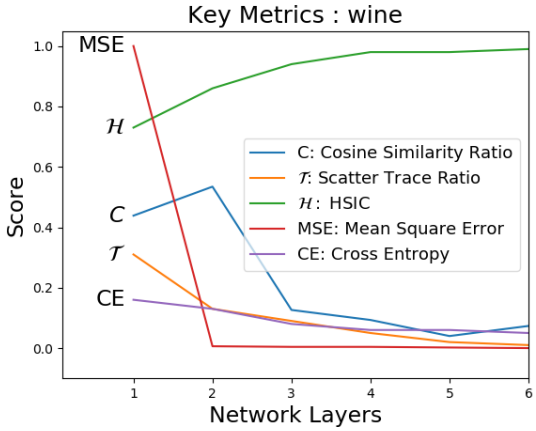

Figure 6.1: Key evaluation metrics at each layer.

Algorithm 5: *KNet* Algorithm

Input : Data $X \in \mathbb{R}^{n \times d}$, Label $Y \in \mathbb{R}^{n \times \tau}$

Output : Network weights $W_1, ..., W_L$

while $\mathcal{H}_l \leq 0.99$ **do**

  Use the output of last layer as input

  Add a new layer

  Initialize layer weight with Eq. (6.3)

  while $\mathcal{Q}_{l^i} \not\approx \mathcal{Q}_{l^{i+1}}$ **do**

    Update $\mathcal{Q}_{l^i} \rightarrow \mathcal{Q}_{l^{i+1}}$ with Eq. (6.4)

  $W_l =$ Most dominant eigenvector of $\mathcal{Q}_l^*$

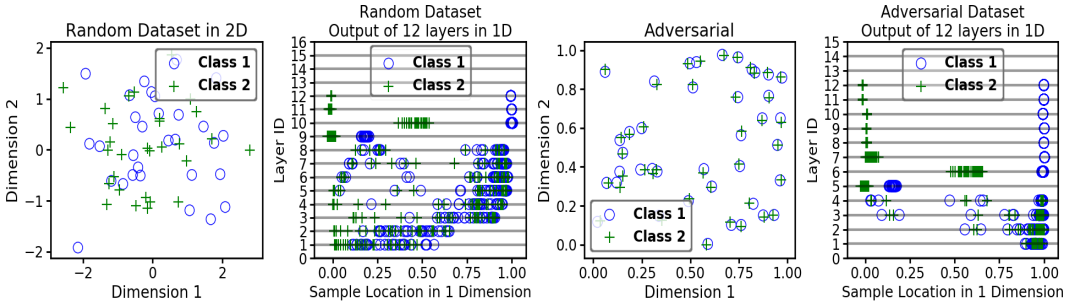

Figure 6.2: Simulation of Thm. 6.1 on Random and Adversarial datasets. The 2D representation is shown, and next to it, the 1D output of each layer is displayed over each line. Both datasets achieved the global optimum $\mathcal{H}^*$ at the 12_{th} layers. Refer to App. D.8 for additional results.

Experimental Results. Since Thm. 6.1 guarantees an optimal convergence for *any* dataset with a suboptimal W_s, we designed an Adversarial dataset to trick the network, i.e., the samples pairs in $\mathcal{S}^c$ are significantly closer than samples pairs in $\mathcal{S}$. We next designed a Random dataset with completely

random labels. We then simulated Thm. 6.1 in Python and plot out the sample behavior in Fig. 6.2. The original 2-dimensional data is shown next to its 1-dimensional IDS results: each line represents the 1D output at that layer. As predicted by the theorem, our network converged at the 12th layer and perfectly separated the samples based on labels. We emphasize that these achievements are acquired purely from the simulation of Thm. 6.1 without resorting to $\sigma \approx 0$ while using a *suboptimal* solution W_s. Namely, the smallest σ values used are 0.15 and 0.03 for Random and Adversarial respectively.

Using the optimal W^* from ISM, we next conduct 10-fold cross-validation across all 8 datasets and reported their mean and the standard deviation for all key metrics. The random and non-random datasets are visually separated. Once our model is trained and has learned its structure, we use the same depth and width to train 2 additional MLPs via SGD, where instead of HSIC, MSE and CE are used as the empirical risk. The results are listed in Table 6.1 with the best outcome in bold.

Can $\mathcal{H}$-*Sequence* be optimized greedily? The $\mathcal{H}^*$ column in Table 6.1 consistently reports results that converge near its theoretical maximum value of 1, thereby corroborating with Thm. 6.1. As predicted by Thm. 6.2, we also report high training accuracies as $\mathcal{H}_l \to \mathcal{H}^*$. Given the overfitting results from Fig. 6.2, will our network generalize? Since smooth mappings are associated with better generalization, we also report the smallest σ value used for each network to highlight the smoothness of ϕ learned by ISM. Correspondingly, with the exception of the two random datasets, our test accuracy consistently performed well across all datasets. *KNet* further differentiates itself on a high dimension Face dataset where it was the only method that avoided overfitting. While we cannot definitively attribute the impressive test results to Thm. 6.3, the experimental evidence appears to be aligned with its implication.

Since Thm. 6.1 also claims that we can achieve $\mathcal{H}^* - \mathcal{H}_l < \delta$ in finite number of layers, we include in Table 6.1 the average length of the $\mathcal{H}$-*Sequence* (L). The table suggests that the $\mathcal{H}$-*Sequence* converges quickly with 9 layers as the deepest network. Additionally, notice in Fig. 6.2 where 12 layers are required for the suboptimal W_s to convergence. In contrast, ISM used much smoother and larger σs (0.38 and 0.5 $\gg$ 0.15 and 0.03) and only requires 3 layers to achieve the same result.

The execution time for each objective is also recorded for reference in Table 6.1. Since *KNet* can be solved via a single forward pass while SGD requires many iterations of BP, *KNet* should be faster. The Time column of Table 6.1 confirmed this expectation by a wide margin. The biggest difference can be observed by comparing the face dataset, $\mathcal{H}$ finished with 0.78 seconds while MSE required

	obj	$\sigma \uparrow$	$L \downarrow$	Train Acc $\uparrow$	Test Acc $\uparrow$	Time(s) $\downarrow$	$\mathcal{H}^* \uparrow$	MSE $\downarrow$	CE $\downarrow$	$C \downarrow$	$\mathcal{T} \downarrow$
random	$\mathcal{H}$	0.38	3.30 ± 0.64	$\mathbf{1.00 \pm 0.00}$	0.38 ± 0.21	$\mathbf{0.40 \pm 0.37}$	$\mathbf{1.00 \pm 0.01}$	$\mathbf{0.00 \pm 0.01}$	0.05 ± 0.00	$\mathbf{0.00 \pm 0.06}$	0.02 ± 0.0
	CE	-	3.30 ± 0.64	$\mathbf{1.00 \pm 0.00}$	0.48 ± 0.17	25.07 ± 5.55	$\mathbf{1.00 \pm 0.00}$	10.61 ± 11.52	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	3.30 ± 0.64	0.98 ± 0.04	$\mathbf{0.63 \pm 0.21}$	23.58 ± 8.38	0.93 ± 0.12	0.02 ± 0.04	0.74 ± 0.03	0.04 ± 0.04	0.08 ± 0.1
adver	$\mathcal{H}$	0.5	3.60 ± 0.92	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.38 \pm 0.10}$	$\mathbf{0.52 \pm 0.51}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.04 \pm 0.00}$	$\mathbf{0.01 \pm 0.08}$	$\mathbf{0.01 \pm 0.0}$
	CE	-	3.60 ± 0.92	0.59 ± 0.04	0.29 ± 0.15	69.54 ± 24.14	0.10 ± 0.07	0.65 ± 0.16	0.63 ± 0.04	0.98 ± 0.03	0.92 ± 0.0
	MSE	-	3.60 ± 0.92	0.56 ± 0.02	0.32 ± 0.20	113.75 ± 21.71	0.02 ± 0.01	0.24 ± 0.01	0.70 ± 0.00	0.99 ± 0.02	0.95 ± 0.0
spiral	$\mathcal{H}$	0.46	5.10 ± 0.30	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.87 \pm 0.08}$	0.98 ± 0.01	0.01 ± 0.00	0.02 ± 0.01	0.04 ± 0.03	0.02 ± 0.0
	CE	-	5.10 ± 0.30	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	11.59 ± 5.52	$\mathbf{1.00 \pm 0.00}$	57.08 ± 31.25	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	5.10 ± 0.30	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	456.77 ± 78.83	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	1.11 ± 0.04	0.40 ± 0.01	$\mathbf{0.00 \pm 0.0}$
wine	$\mathcal{H}$	0.47	6.10 ± 0.54	0.99 ± 0.00	$\mathbf{0.97 \pm 0.05}$	$\mathbf{0.28 \pm 0.04}$	0.98 ± 0.01	0.01 ± 0.00	0.07 ± 0.01	0.04 ± 0.03	0.02 ± 0.0
	CE	-	6.10 ± 0.54	$\mathbf{1.00 \pm 0.00}$	0.94 ± 0.06	3.30 ± 1.24	$\mathbf{1.00 \pm 0.00}$	40.33 ± 35.5	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	6.10 ± 0.54	$\mathbf{1.00 \pm 0.00}$	0.89 ± 0.17	77.45 ± 45.40	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	1.15 ± 0.07	0.49 ± 0.02	$\mathbf{0.00 \pm 0.0}$
cancer	$\mathcal{H}$	0.39	8.10 ± 0.83	0.99 ± 0.00	$\mathbf{0.97 \pm 0.02}$	$\mathbf{2.58 \pm 1.07}$	0.96 ± 0.01	0.02 ± 0.01	0.04 ± 0.01	0.02 ± 0.04	0.04 ± 0.0
	CE	-	8.10 ± 0.83	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.97 \pm 0.01}$	82.03 ± 35.15	$\mathbf{1.00 \pm 0.00}$	2330 ± 2915	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	8.10 ± 0.83	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.97 \pm 0.03}$	151.81 ± 27.27	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.66 ± 0.06	$\mathbf{0.00 \pm 0.0}$	$\mathbf{0.00 \pm 0.0}$
car	$\mathcal{H}$	0.23	4.90 ± 0.30	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.01}$	$\mathbf{1.51 \pm 0.35}$	0.99 ± 0.00	$\mathbf{0.00 \pm 0.00}$	0.01 ± 0.00	0.04 ± 0.03	0.01 ± 0.0
	CE	-	4.90 ± 0.30	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	25.79 ± 18.86	$\mathbf{1.00 \pm 0.00}$	225.11 ± 253	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	4.90 ± 0.30	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	503.96 ± 116.64	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	1.12 ± 0.07	0.40 ± 0.00	$\mathbf{0.00 \pm 0.0}$
face	$\mathcal{H}$	0.44	4.00 ± 0.00	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.78 \pm 0.08}$	0.97 ± 0.00	$\mathbf{0.00 \pm 0.00}$	0.17 ± 0.00	0.01 ± 0.00	$\mathbf{0.00 \pm 0.0}$
	CE	-	4.00 ± 0.00	$\mathbf{1.00 \pm 0.00}$	0.79 ± 0.31	23.70 ± 8.85	$\mathbf{1.00 \pm 0.00}$	16099 ± 16330	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	4.00 ± 0.00	0.92 ± 0.10	0.52 ± 0.26	745.17 ± 281.56	0.94 ± 0.07	0.11 ± 0.12	3.50 ± 0.28	0.72 ± 0.01	$\mathbf{0.00 \pm 0.0}$
divorce	$\mathcal{H}$	0.41	4.10 ± 0.54	0.99 ± 0.01	0.98 ± 0.02	$\mathbf{0.71 \pm 0.41}$	0.99 ± 0.01	0.01 ± 0.01	0.03 ± 0.00	$\mathbf{0.00 \pm 0.05}$	0.02 ± 0.0
	CE	-	4.10 ± 0.54	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.99 \pm 0.02}$	2.62 ± 1.21	$\mathbf{1.00 \pm 0.00}$	14.11 ± 12.32	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.0}$
	MSE	-	4.10 ± 0.54	$\mathbf{1.00 \pm 0.00}$	0.97 ± 0.03	47.89 ± 24.31	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.73 ± 0.07	$\mathbf{0.00 \pm 0.01}$	0.01 ± 0.0

Table 6.1: Each dataset contains 3 rows comparing the greedily trained *KNet* using $\mathcal{H}$ against traditional MLPs trained using MSE and CE via SGD given the same network width and depth. The best results are in bold with $\uparrow$ / $\downarrow$ indicating larger/smaller values preferred.

745 seconds; that is almost 1000 times difference. While the execution times reflect our expectation, techniques that vastly accelerate kernel computations [95, 96] would be required for larger datasets.

Predicted by Thm. 6.2, *KNet* induces low $\mathcal{T}$ and C as shown in Table 6.1, implying that samples in $\mathcal{S}$ and $\mathcal{S}^c$ are being pulled together and pushed apart based on Euclidean and angular distance in IDS and RKHS. Given this geometry, will its optimal arguments also induce a low MSE and CE? We evaluate these predictions from Corollaries 2 and 3 by keeping the same network weights while replacing the final objective with MSE and CE. Our corroborating results are highlighted in the columns of MSE and CE in Table 6.1. Interestingly, while using HSIC induces a low MSE and CE, training via BP using MSE or CE does not necessarily translate to good results for each other.

Within the network, Fig. 6.1 plots out all key metrics at each layer during training. Here, the $\mathcal{H}$-*Sequence* is clearly monotonic and converging towards a global optimal of 1. Moreover, the trends

for $\mathcal{T}$ and C indicate an incremental clustering of samples into separate partitions. We are unsure why C consistently forms a hunchback pattern, this implies an initial expansion in the dimension of the data following compression. However, we note that this behavior is also observed with traditional networks by Ansuini et al. [99]. Corresponding to low $\mathcal{T}$ and C values, the low MSE and CE errors at convergence further reinforces the claims of Corollaries 2 and 3. Note that these identical patterns are consistent and repeatable across all datasets as shown in App. D.10.

We lastly highlight a visual pattern for the *Kernel Sequence* in Fig. 6.3. We rearrange the samples of the same class to be adjacent to each other. This allows us to quickly evaluate the kernel quality via its block diagonal structure. Since GK is restricted to values between 0 and 1, we let white and dark blue be 0 and 1 respectively where the gradients reflect values in between. Our proof predicts that the *Kernel Sequence* should converge to the optimal kernel $\mathcal{K}^*$, i.e., the *Kernel Sequence* should evolve from an uninformative kernel into a highly discriminating kernel of perfect block diagonal structures. Corresponding to the top row, the bottom row plots out the samples in IDS at each layer. As predicted by Thm. 6.2, the samples of the same class incrementally converge towards a single point in IDS. Again, this pattern is observable on all datasets, and the complete collection of the *kernel sequences* for each dataset can be found in App. D.9.

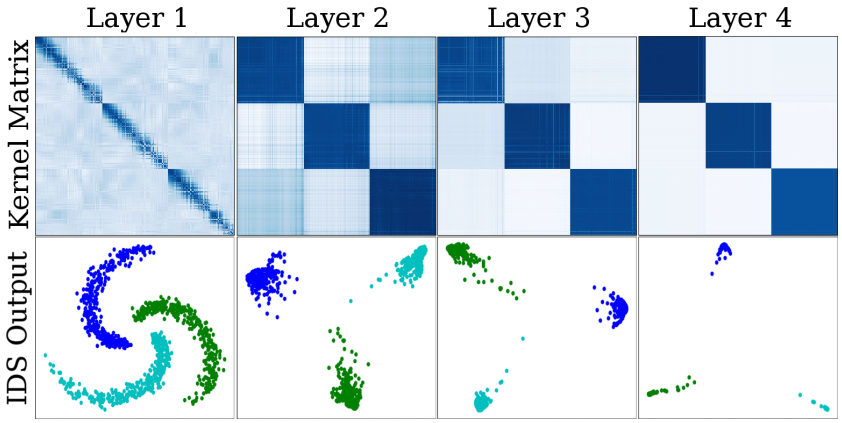

Figure 6.3: A visual confirmation of Thm. 6.2. The kernel matrices per layer produced by the *Kernel Sequence* are displayed in the top row with their corresponding outputs in IDS in the bottom row.

6.6 Kernel Dependence Network Summary.

We have presented a new model of MLP for classification that bypasses BP and SGD. Our model, *KNet*, is guaranteed to reach the global optimum of HSIC in finite steps. The resulting geometric orientation of the samples minimizes the scatter ratio while produces an optimal kernel, $\mathcal{K}^*$. This results in a geometric orientation that consequently solves MSE and CE in different spaces. Indeed, these patterns are predictable by our theorems and experimentally reproducible. Therefore, *KNet* opens the door to a new perspective to analyze MLPs.

Chapter 7

Conclusion

By studying alternative data representations for dimensionality reduction for a variety of downstream tasks, we found a common formulation for Interpretable Kernel Dimension Reduction (IKDR) via HSIC. We presented a general IKDR formulation for the various machine learning paradigms: supervised, unsupervised, and semi-supervised. We developed an efficient method for solving general IKDR problems – Iterative Spectram Method (ISM). We provide theoretical 1st and 2nd order convergence guarantees for our ISM method for a general class of kernels (we call the ISM family of kernels). We demonstrated its usage across various domains and have experimentally shown to it perform well.

While investigating the IKDR formulation, we made the realization that it can be used to model a single layer of a Multi-layered perceptron. We designed a deep neural network by using IKDR formulation to model each layer. This perspective allowed us to introduce a new function class for supervised classification, which builds a kernel neural network in a greedy layer-wise fashion, we call *Kernel Dependence Network* (*KNet*). Each layer is a composition of a linear subspace projection and an infinite-dimensional feature map. A key advantage is that the network can be trained greedily, layer by layer. By greedily solving each layer with ISM, the complete network can also be solved without the need for gradient descent backpropagation, thereby avoiding exploding/vanishing gradients. As a further consequence of our formulation, we demonstrate how the natural choice of the network width and depth also emerges.

In this thesis, we provided the theoretical characterization of the richness of the function class

described by our architecture. We theoretically show that given a minimum of just two infinitely wide layers, the global optimal of the HSIC objective can be achieved. As a consequence of maximizing the HSIC between the data and the label, we show that traditional classification objectives like MSE and Cross-Entropy is also solved. As the final theoretical contribution, we investigate why a regularizer may not be necessary for the HSIC formulation. From this research, we demonstrated that our architecture and training procedure perform an implicit regularization, similar to the weight penalization arguments of Poggio et al. [23], and we describe the mechanisms by which this is achieved.

As a consequence of our work, we have opened up a new direction of research for new networks that are based on the Kernel Dependence Network. The immediate two major problems we still need to resolve are the memory and computational complexity of working with kernels and performing ISM. This requires us to discover an alternative to constructing the kernel matrix, as well as finding a faster method to solve the eigendecomposition problem. Since the problem currently only solves classification, extensions to analysis for regression would be interesting future research.

Bibliography

[1] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[2] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[3] Nandakishore Kambhatla and Todd K Leen. Dimension reduction by local principal component analysis. *Neural computation*, 9(7):1493–1516, 1997.

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

[5] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[6] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, 2011.

[7] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[8] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[9] Chieh Wu, Stratis Ioannidis, Mario Sznaier, Xiangyu Li, David Kaeli, and Jennifer Dy. Iterative spectral method for alternative clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 115–123, 2018.

[10] Kenji Fukumizu, Francis R Bach, Michael I Jordan, et al. Kernel dimension reduction in regression. *The Annals of Statistics*, 37(4):1871–1905, 2009.

[11] Mahdokht Masaeli, Jennifer G Dy, and Glenn M Fung. From transformation-based dimensionality reduction to feature selection. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 751–758, 2010.

[12] Donglin Niu, Jennifer G Dy, and Michael I Jordan. Dimensionality reduction for spectral clustering. In *AISTATS*, pages 552–560, 2011.

[13] Mehrdad J Gangeh, Safaa MA Bedawi, Ali Ghodsi, and Fakhri Karray. Semi-supervised dictionary learning based on hilbert-schmidt independence criterion. In *International Conference Image Analysis and Recognition*, pages 12–19. Springer, 2016.

[14] Yale Chang, Junxiang Chen, Michael H Cho, Peter J Castaidi, Edwin K Silverman, and Jennifer G Dy. Clustering with domain-specific usefulness scores. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 207–215. SIAM, 2017.

[15] Donglin Niu, Jennifer G Dy, and Michael I Jordan. Multiple non-redundant spectral clustering views. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 831–838, 2010.

[16] Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13(May):1393–1434, 2012.

[17] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[18] Donglin Niu, Jennifer G Dy, and Michael I Jordan. Iterative discovery of multiple alternativeclustering views. *IEEE transactions on pattern analysis and machine intelligence*, 36(7): 1340–1353, 2014.

[19] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[20] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

[21] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. In *Advances in Neural Information Processing Systems*, pages 1962–1974, 2019.

[22] Guillermo Valle-Pérez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

[23] Tomaso Poggio, Qianli Liao, and Andrzej Banburski. Complexity control by gradient descent in deep networks. *Nature Communications*, 11(1):1–5, 2020.

[24] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35:67–68, 1999.

[25] David Gondek and Thomas Hofmann. Non-redundant data clustering. *Knowledge and Information Systems*, 12(1):1–24, 2007.

[26] Ying Cui, Xiaoli Z Fern, and Jennifer G Dy. Learning multiple nonredundant clusterings. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(3):15, 2010.

[27] Xuan Hong Dang and James Bailey. Generation of alternative clusterings using the cami approach. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 118–129. SIAM, 2010.

[28] Ian Davidson and Zijie Qi. Finding alternative clusterings using constraints. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 773–778. IEEE, 2008.

[29] Ying Cui, Xiaoli Z Fern, and Jennifer G Dy. Non-redundant multi-view clustering via orthogonalization. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 133–142. IEEE, 2007.

[30] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

[31] Max Vladymyrov and Miguel Carreira-Perpinan. The variational nystrom method for large-scale spectral problems. In *International Conference on Machine Learning*, pages 211–220, 2016.

[32] Peter Richtárik. Generalized power method for sparse principal component analysis.

[33] Qi Lei, Kai Zhong, and Inderjit S Dhillon. Coordinate-wise power method. In *Advances in Neural Information Processing Systems*, pages 2064–2072, 2016.

[34] CMU CMU. universities webkb data, 1997. 4.

[35] Stephen D Bay, Dennis Kibler, Michael J Pazzani, and Padhraic Smyth. The uci kdd archive of large data sets for data mining research and experimentation. *ACM SIGKDD Explorations Newsletter*, 2(2):81–85, 2000.

[36] Particles of tessellations. `http://en.tessellations-nicolas.com/`. Accessed: 2017-04-25.

[37] A Strehl and J Chosh. Knowledge reuse framework for combining multiple partitions. *Journal of Machine learning Research*, 33(3):583–617.

[38] Le Song, Alex Smola, Arthur Gretton, Karsten M Borgwardt, and Justin Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, pages 823–830. ACM, 2007.

[39] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[40] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2): 214–225, 2004.

[41] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[42] Roger A Horn, Roger A Horn, and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.

[43] William H Wolberg. Wisconsin breast cancer dataset. *University of Wisconsin Hospitals*, 1992.

[44] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[45] Yufei Ding, Yue Zhao, Xipeng Shen, Madanlal Musuvathi, and Todd Mytkowicz. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In *International Conference on Machine Learning*, pages 579–587, 2015.

[46] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002.

[47] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican Congress on Pattern Recognition*, pages 117–124. Springer, 2013.

[48] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487, 2016.

[49] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self augmented training. *arXiv preprint arXiv:1702.08720*, 2017.

[50] Uri Shaham, Kelly Stanton, Henry Li, Ronen Basri, Boaz Nadler, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJ_aoCyRZ`.

[51] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[52] James Ross and Jennifer Dy. Nonparametric mixture of gaussian processes with constraints. In *International Conference on Machine Learning*, pages 1346–1354, 2013.

[53] Taiji Suzuki and Masashi Sugiyama. Sufficient dimension reduction via squared-loss mutual information estimation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 804–811, 2010.

[54] Florian Wickelmaier. An introduction to mds. *Sound Quality Research Unit, Aalborg University, Denmark*, 46(5):1–26, 2003.

[55] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[56] Le Song, Arthur Gretton, Karsten Borgwardt, and Alex J Smola. Colored maximum variance unfolding. In *Advances in neural information processing systems*, pages 1385–1392, 2008.

[57] Max Vladymyrov and Miguel A Carreira-Perpinán. Locally linear landmarks for large-scale manifold learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 256–271. Springer, 2013.

[58] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

[59] Ioan Mackenzie James. *The topology of Stiefel manifolds*, volume 24. Cambridge University Press, 1976.

[60] Yasunori Nishimori and Shotaro Akaho. Learning algorithms utilizing quasi-geodesic flows on the stiefel manifold. *Neurocomputing*, 67:106–135, 2005.

[61] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[62] Nicolas Boumal and Pierre-antoine Absil. Rtrmc: A riemannian trust-region method for low-rank matrix completion. In *Advances in neural information processing systems*, pages 406–414, 2011.

[63] Fabian J Theis, Thomas P Cason, and P-A Absil. Soft dimension reduction for ica by joint diagonalization on the stiefel manifold. In *International Conference on Independent Component Analysis and Signal Separation*, pages 354–361. Springer, 2009.

[64] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[65] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. URL `http://www.manopt.org`.

[66] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

[67] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed ¡today¿].

[68] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[69] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[70] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[71] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[72] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[73] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[74] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *ArXiv*, abs/1711.00165, 2018.

[75] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[76] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[77] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. *arXiv preprint arXiv:1902.06853*, 2019.

[78] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

[79] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[80] David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *Artificial Intelligence and Statistics*, pages 202–210, 2014.

[81] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[82] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

[83] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018.

[84] Chieh Wu, Jared Miller, Yale Chang, Mario Sznaier, and Jennifer Dy. Solving interpretable kernel dimension reduction. *arXiv preprint arXiv:1909.03093*, 2019.

[85] Wan-Duo Kurt Ma, JP Lewis, and W Bastiaan Kleijn. The hsic bottleneck: Deep learning without back-propagation. *arXiv preprint arXiv:1908.01580*, 2019.

[86] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

[87] Stanislav Fort and Stanislaw Jastrzebski. Large scale structure of neural network loss landscapes. *arXiv preprint arXiv:1906.04724*, 2019.

[88] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[89] John Bibby. Axiomatisations of the average and a further generalisation of monotonic sequences. *Glasgow Mathematical Journal*, 15(1):63–65, 1974.

[90] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[91] Geoffrey J McLachlan. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons, 2004.

[92] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 10835–10845, 2019.

[93] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

[94] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[95] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pages 14622–14632, 2019.

[96] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems*, pages 3888–3898, 2017.

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[98] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[99] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *arXiv preprint arXiv:1905.12784*, 2019.

[100] Andrew V Knyazev and Peizhen Zhu. Principal angles between subspaces and their tangents. 2012.

[101] GrÃŠgoire Montavon, Mikio L Braun, and Klaus-Robert MÃtller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12(Sep):2563–2581, 2011.

Appendix A

This section provides further detail to the chapter on Alternative Clustering. Here, we provide the derivation that relates the Alternative Clustering to the ISM algorithm. We also provide the proof for the theoretical guarantees of ISM. Namely, we demonstrate why the eigenvectors associated with the smallest eigenvalue of the Φ matrix is the solution of Alternative Clustering.

A.1 Derivation for Equation 3.4

Given the objective function for Alternative Clustering as

$$\begin{aligned} \max_{U,W} \quad & \text{HSIC}(XW, U) - \lambda \, \text{HSIC}(XW, Y) \\ s.t \quad & W^T W = I, U^T U = I, \end{aligned}$$

we note that this objective can be greatly compressed. Here we provide the derivation on an alternative representation of the objective.

Using the HSIC measure defined, the objective function can be rewritten as

$$\begin{aligned} \text{HSIC}(XW, U) - \lambda \, \text{HSIC}(XW, Y) &= \text{Tr}(HUU^T H D^{\frac{-1}{2}} K_{XW} D^{\frac{-1}{2}}) \\ &\quad -\lambda \, \text{Tr}(HYY^T H D^{\frac{-1}{2}} K_{XW} D^{\frac{-1}{2}}) \\ &= \text{Tr}(D^{\frac{-1}{2}} H(UU^T - \lambda YY^T) H D^{\frac{-1}{2}} K_{XW}) \\ &= \text{Tr}(\gamma K_{XW}) \\ &= \textstyle\sum_{i,j} \gamma_{i,j} K_{X_{i,j}}. \end{aligned}$$

where γ is a symmetric matrix and $\gamma = H(UU^T - \lambda YY^T)H$. By substituting the Gaussian kernel for $K_{X_{i,j}}$, the objective function becomes

$$\min_{W} \quad -\sum_{i,j} \gamma_{i,j} e^{-\frac{\mathrm{Tr}[W^T A_{i,j} W]}{2\sigma^2}} \qquad s.t \quad W^T W = I.$$

A.2 Proof for Lemma 3.2.

To solve Alternative Clustering, we identified the ISM algorithm to solve this objective. For the first part of the proof, we showed that the algorithm yields eigenvector of the Φ matrix and consequently the solution.

Lemma 3.2 *Let W^* be a fixed point of Algorithm 2. Then it satisfies:$\Phi(W^*)W^* = W^*\Lambda^*$, where $\Lambda^* \in \mathcal{R}^{q\times q}$ is a diagonal matrix containing the q smallest eigenvalues of $\Phi(W^*)$ and ${W^*}^T W^* = I$.*

Proof. Algorithm 2 sets the smallest q eigenvectors of $\Phi(W_k)$ as W_{k+1}. Since a fixed point W^* is reached when $W_k = W_{k+1}$, therefore W^* consists of the smallest eigenvectors of $\Phi(W^*)$ and Λ^* corresponds with a diagonal matrix of eigenvavlues. Since the eigenvectors of $\Phi(W^*)$ are orthonormal , ${W^*}^T W^* = I$ is also satisfied. □

A.3 Proof for Lemma 3.3

For the ISM algorithm, we have shown that the eigenvector is the stationary point. Here, we provide the proof for this portion of the proof.

Lemma 3.3. *If W^* is a fixed point and Λ^* is as defined in Lemma 3.2, then W^*, Λ^* satisfy the 1st order conditions (C.45)(3.9b) of Lemma 3.1.*

Proof. Using Equation (3.4) as the objective function, the corresponding Lagrangian and its gradient is written as

$$\mathcal{L}(W,\Lambda) = -\sum_{i,j} \gamma_{i,j} e^{-\frac{\mathrm{Tr}(W^T A_{i,j} W)}{2\sigma^2}} - \frac{1}{2}\mathrm{Tr}(\Lambda(W^T W - I)), \tag{A.1}$$

and

$$\nabla_W \mathcal{L}(W,\Lambda) = \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}(W^T A_{i,j} W)}{2\sigma^2}} A_{i,j} W - W\Lambda. \tag{A.2}$$

By setting the gradient of the Lagrangian to zero, and using the definition of $\Phi(W)$ from Equation (3.8), Equation (D.128) can be written as

$$\Phi(W)W = W\Lambda. \tag{A.3}$$

The gradient with respect to Λ is

$$\nabla_\Lambda \mathcal{L}(W,\Lambda) = W^T W - I. \tag{A.4}$$

Setting this gradient of the Lagrangian also to zero, condition (3.9b) is equivalent to

$$W^T W = I. \tag{A.5}$$

By Lemma 3.2, a fixed point W^* and its corresponding Λ^* satisfy (A.3) and (A.5), and the lemma follows. □

A.4 Proof for Lemma 3.4

For the ISM algorithm, we first show that the eigenvector is the stationary point. However, we cannot use any eigenvector, instead, we must use the eigenvectors associated with the smallest eigenvalues. Here, we provide the proof for this portion of the proof.

Lemma 3.4. *If W^* is a fixed point, Λ^* is as defined in Lemma 3.2, and $\Phi(W^*)$ is full rank, then given a large enough σ (satisfying Inequality* (3.10)*), W^* and Λ^* satisfy the 2nd order condition (C.54) of Lemma 3.1.*

APPENDIX A.

The proof for Lemma 3.4 relies on the following three sublemmas. The first two sublemmas demonstrate how the 2nd order conditions can be rewritten into a simpler form. With the simpler form, the third lemma demonstrates how the 2nd order conditions of a local minimum are satisfied given a large enough σ.

Lemma A.0.1. *Let the directional derivative in the direction of Z be defined as*

$$\mathcal{D}f(W)[Z] := \lim_{t \to 0} \frac{f(W + tZ) - f(W)}{t}. \tag{A.6}$$

Then the 2nd order condition of Lemma 3.4 can be written as

$$\operatorname{Tr}(Z^T \mathcal{D}\nabla\mathcal{L}[Z]) = \left\{ \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\operatorname{Tr}((W^*)^T A_{i,j} W^*)}{2\sigma^2}} \left[\operatorname{Tr}(Z^T A_{i,j} Z) - \frac{1}{\sigma^2} \operatorname{Tr}(Z^T A_{i,j} W^*)^2 \right] \right\} - \operatorname{Tr}(Z^T Z \Lambda^*), \tag{A.7}$$

for all Z such that

$$Z^T W^* + (W^*)^T Z = 0. \tag{A.8}$$

Proof. Observe first that

$$\nabla^2_{W^* W^*} \mathcal{L}(W^*, \Lambda^*) Z = \mathcal{D}\nabla\mathcal{L}[Z], \tag{A.9}$$

where the directional derivative of the gradient $\mathcal{D}\nabla\mathcal{L}[Z]$ is given by

$$\mathcal{D}\nabla\mathcal{L}[Z] = \lim_{t \to 0} \frac{\partial}{\partial t} \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\operatorname{Tr}((W^* + tZ)^T A_{i,j} (W^* + tZ))}{2\sigma^2}} A_{i,j}(W^* + tZ) - (W^* + tZ)\Lambda.$$

This can be written as

$$\mathcal{D}\nabla\mathcal{L}[Z] = T_1 + T_2 - T_3,$$

where

$$T_1 = \lim_{t \to 0} \frac{\partial}{\partial t} \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}((W^*+tZ)^T A_{i,j}(W^*+tZ))}{2\sigma^2}} A_{i,j} W^* \tag{A.10}$$

$$= \lim_{t \to 0} \frac{\partial}{\partial t} \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^* + tZ^T A_{i,j} W^* + t{W^*}^T A_{i,j} Z + t^2 Z^T A_{i,j} Z)}{2\sigma^2}} A_{i,j} W^* \tag{A.11}$$

$$= -\sum_{i,j} \frac{\gamma_{i,j}}{2\sigma^4} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^*)}{2\sigma^2}} \mathrm{Tr}(Z^T A_{i,j} W^* + {W^*}^T A_{i,j} Z) A_{i,j} W^* \tag{A.12}$$

$$= -\sum_{i,j} \frac{\gamma_{i,j}}{\sigma^4} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^*)}{2\sigma^2}} \mathrm{Tr}(Z^T A_{i,j} W^*) A_{i,j} W^* \qquad \text{as } A_{i,j} = A_{i,j}^T, \tag{A.13}$$

$$T_2 = \lim_{t \to 0} \frac{\partial}{\partial t} \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} t e^{-\frac{\mathrm{Tr}((W^*+tZ)^T A_{i,j}(W^*+tZ))}{2\sigma^2}} A_{i,j} Z \tag{A.14}$$

$$= \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^*)}{2\sigma^2}} A_{i,j} Z, \tag{A.15}$$

$$T_3 = \lim_{t \to 0} \frac{\partial}{\partial t} (W^* + tZ)\Lambda \tag{A.16}$$

$$= Z\Lambda. \tag{A.17}$$

Hence, putting all three terms together yields

$$\mathcal{D}\nabla\mathcal{L}[Z] = \left\{ \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^*)}{2\sigma^2}} \left[A_{i,j} Z - \frac{1}{\sigma^2} \mathrm{Tr}(Z^T A_{i,j} W^*) A_{i,j} W^* \right] \right\} - Z\Lambda. \tag{A.18}$$

Hence,

$$\mathrm{Tr}(Z^T \nabla^2_{W^* W^*} \mathcal{L}(W^*, \Lambda^*) Z) = \mathrm{Tr}(Z^T \mathcal{D}\nabla\mathcal{L}[Z]), \tag{A.19}$$

$$= \left\{ \sum_{i,j} \frac{\gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}({W^*}^T A_{i,j} W^*)}{2\sigma^2}} \left[\mathrm{Tr}(Z^T A_{i,j} Z) - \frac{1}{\sigma^2} \mathrm{Tr}(Z^T A_{i,j} W^*)^2 \right] \right\} - \mathrm{Tr}(Z^T Z \Lambda_W). \tag{A.20}$$

Next, let Z be such that $Z \neq 0$ and $\nabla h(W^*)^T Z = 0$, where

$$h(W^*) = {W^*}^T W^* - I. \tag{A.21}$$

Therefore, the constraint condition can be written on Z in (C.54) can be written as

$$\begin{aligned} \nabla h(W^*)^T Z = & \lim_{t \to 0} \frac{\partial}{\partial t} \frac{(W^* + tZ)^T (W^* + tZ) - {W^*}^T W^*}{t} \\ & = Z^T W^* + {W^*}^T Z = 0. \end{aligned} \tag{A.22}$$

Using Equations (A.20) and (A.22) lemma A.0.1 follows. □

Recall from Lemma 3.2 that W^* consists of the q eigenvectors of $\Phi(W^*)$ with the smallest eigenvalues. We define $\bar{W}^* \in \mathcal{R}^{d \times d-q}$ as all other eigenvectors of $\Phi(W^*)$. Because Z has the same dimension as W^*, each column of Z resides in the space of $\mathcal{R}^d$. Since the eigenvectors of $\Phi(W^*)$ span $\mathcal{R}^d$, each column of Z can be represented as a linear combination of the eigenvectors of $\Phi(W^*)$. In other words, each column z_i can therefore be written as $z_i = W^* P_W^{(i)} + \bar{W}^* P_{\bar{W}^*}^{(i)}$, where $P_{W^*}^{(i)} \in \mathcal{R}^{q \times 1}$ and $P_{\bar{W}^*}^{(i)} \in \mathcal{R}^{d-q \times 1}$ represents the coordinates for the two sets of eigenvectors. Using the same notation, we also define $\Lambda^* \in \mathcal{R}^{q \times q}$ as the eigenvalues corresponding to W^* and $\bar{\Lambda}^* \in \mathcal{R}^{d-q \times d-q}$ as the eigenvalues corresponding to $\bar{W}^*$. The entire matrix Z can therefore be represented as

$$Z = \bar{W}^* P_{\bar{W}^*} + W^* P_{W^*}. \tag{A.23}$$

Furthermore, it can be easily shown that P_{W^*} is a skew symmetric matrix, or $-P_{W^*} = P_{W^*}^T$. By setting Z from Equation (A.8) into (A.23), the constraint can be rewritten as

$$[P_{\bar{W}^*}^T {\bar{W}^*}^T + P_W^{*T} {W^*}^T] W^* + {W^*}^T [\bar{W}^* P_{\bar{W}^*} + W^* P_{W^*}] = 0. \tag{A.24}$$

Simplifying the equation yields the relationship

$$P_W^{*T} + P_{W^*} = 0. \tag{A.25}$$

Using these definitions, we define the following sublemma.

Lemma A.0.2. *Given a fixed point W^* and a Z satisfying condition (A.8), the condition* $\mathrm{Tr}(Z^T\mathcal{D}\nabla\mathcal{L}[Z]) \geq 0$ *is equivalent to*

$$\mathrm{Tr}(P_{\bar{W}^*}^T\bar{\Lambda}^*P_{\bar{W}^*}) - \mathrm{Tr}(P_{\bar{W}^*}\Lambda^*P_{\bar{W}^*}^T) \geq C_2, \tag{A.26}$$

where

$$C_2 = \sum_{i,j}\frac{\gamma_{i,j}}{\sigma^4}e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}\mathrm{Tr}(Z^TA_{i,j}W^*)^2, \tag{A.27}$$

$P_{W^*}, P_{\bar{W}^*}$ are given by Equation (A.23), and $\Lambda^*, \bar{\Lambda}^*$ are the diagonal matrices containing the bottom and top eigenvalues of $\Phi(W^*)$ respectively.

Proof. By condition (A.7),

$$\mathrm{Tr}(Z^T\mathcal{D}\nabla\mathcal{L}[Z]) = C_1 - C_2 + C_3, \tag{A.28}$$

where

$$\begin{aligned}
C_1 &= \mathrm{Tr}\left(Z^T\sum_{i,j}\frac{\gamma_{i,j}}{\sigma^2}e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}A_{i,j}Z\right),\\
C_2 &= \sum_{i,j}\frac{\gamma_{i,j}}{\sigma^4}e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}\mathrm{Tr}(Z^TA_{i,j}W^*)^2,\\
C_3 &= -\mathrm{Tr}(Z^TZ\Lambda^*).
\end{aligned}$$

C_1 can be written as

$$\begin{aligned}
C_1 &= \mathrm{Tr}\left(Z^T\sum_{i,j}\frac{\gamma_{i,j}}{\sigma^2}e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}A_{i,j}Z\right)\\
&= \mathrm{Tr}(Z^T\Phi(W^*)[\bar{W}^*P_{\bar{W}^*} + W^*P_{W^*}])\\
&= \mathrm{Tr}(Z^T[\Phi(W^*)\bar{W}^*P_{\bar{W}^*} + \Phi(W^*)W^*P_{W^*}])\\
&= \mathrm{Tr}(Z^T[\bar{W}^*\bar{\Lambda}P_{\bar{W}^*} + W^*\Lambda P_{W^*}]) && \text{By definition of eigenvalues.}\\
&= \mathrm{Tr}([P_{\bar{W}^*}^T\bar{W}^{*^T} + P_W^{*^T}W^{*^T}][\bar{W}^*\bar{\Lambda}P_{\bar{W}^*} + W^*\Lambda P_{W^*}]) && \text{Substitute for } Z\\
&= \mathrm{Tr}(P_{\bar{W}^*}^T\bar{\Lambda}P_{\bar{W}^*}) + \mathrm{Tr}(P_{W^*}^T\Lambda P_W) && \text{Given } W^{*^T}W^* = I,\ \bar{W}^{*^T}W^* = 0.
\end{aligned}$$

Similarly

$$\begin{aligned} C_3 &= -\operatorname{Tr}(Z^T Z\Lambda) \\ &= -\operatorname{Tr}([P_{\bar{W}^*}^T \bar{W}^{*^T} + P_{W^*}^T W^{*^T}][\bar{W}^* P_{\bar{W}^*} + W^* P_{W^*}]\Lambda) \\ &= -\operatorname{Tr}([P_{\bar{W}^*}^T P_{\bar{W}^*} + P_{W^*}^T P_{W^*}]\Lambda) \\ &= -\operatorname{Tr}(P_{\bar{W}^*}^T P_{\bar{W}^*}\Lambda) - \operatorname{Tr}(P_{W^*}^T P_{W^*}\Lambda). \end{aligned}$$

Because P_{W^*} is a square skew symmetric matrix, the diagonal elements of $P_{W^*}P_{W^*}^T$ is the same as the diagonal of $P_{W^*}P_{W^*}^T$. From this observation, we conclude that $\operatorname{Tr}(P_{W^*}P_{W^*}^T\Lambda) = \operatorname{Tr}(P_{W^*}^T P_{W^*}\Lambda)$. Hence,

$$C_3 = -\operatorname{Tr}(P_{\bar{W}^*}\Lambda P_{\bar{W}^*}^T) - \operatorname{Tr}(P_{W^*}^T\Lambda P_{W^*}).$$

Putting all 3 parts together yields

$$\begin{aligned} \operatorname{Tr}(Z^T\mathcal{D}\nabla\mathcal{L}[Z]) &= \operatorname{Tr}(P_{\bar{W}^*}^T\bar{\Lambda}P_{\bar{W}^*}) + \operatorname{Tr}(P_{W^*}^T\Lambda P_{W^*}) - C_2 - \operatorname{Tr}(P_{\bar{W}^*}\Lambda P_{\bar{W}^*}^T) - \operatorname{Tr}(P_{W^*}^T\Lambda P_{W^*}) \\ &= \operatorname{Tr}(P_{\bar{W}^*}^T\bar{\Lambda}P_{\bar{W}^*}) - \operatorname{Tr}(P_{\bar{W}^*}\Lambda P_{\bar{W}^*}^T) - C_2. \end{aligned} \tag{A.29}$$

The 2nd order condition (C.54) is, therefore, satisfied, when

$$\operatorname{Tr}(P_{\bar{W}^*}^T\bar{\Lambda}P_{\bar{W}^*}) - \operatorname{Tr}(P_{\bar{W}^*}\Lambda P_{\bar{W}^*}^T) \geq C_2. \tag{A.30}$$

□

Lemma A.0.3. *Given W^*,$\bar{W}^*$,$\bar{\Lambda}^*$, and Λ^* as defined in Equation (A.23), if the corresponding smallest eigenvalue of $\bar{\Lambda}^*$ is larger than the largest eigenvalue of Λ^*, then given a large enough σ the condition (C.54) of Lemma 3.1 is satisfied.*

Proof. To proof sublemma (A.0.3), we provide bounds on each of the terms in (A.30). Starting with C_2 defined at (A.27). It has a trace term, $(\operatorname{Tr}(Z^T A_{ij} W^*))^2$ that can be rewritten as

$$(\mathrm{Tr}(A_{ij}W^*Z^T))^2 = (\mathrm{Tr}(A_{ij}W^*P_{W^*}^T W^{*^T} + A_{ij}W^*P_{\bar{W}^*}^T \bar{W}^{*^T}))^2. \tag{A.31}$$

Since A_{ij} is symmetric and $W^*P_{W^*}^T W^{*^T}$ is skew-symmetric, then $\mathrm{Tr}(A_{ij}W^*P_{W^*}^T W^{*^T}) = 0$. Hence

$$(\mathrm{Tr}(Z^T A_{ij}W^*))^2 = (\mathrm{Tr}(A_{ij}W^*Z^T))^2 = (\mathrm{Tr}(A_{ij}W^*P_{\bar{W}^*}^T \bar{W}^{*^T}))^2 \tag{A.32}$$

$$\leq \mathrm{Tr}(A_{i,j}^T A_{ij})\,\mathrm{Tr}(P_{\bar{W}^*}^T P_{\bar{W}^*}) \tag{A.33}$$

where the last inequality follows from Cauchy-Schwartz inequality and that fact that $W^{*^T}W^* = I$ and $\bar{W}^{*^T}\bar{W}^* = I$. Thus, C_2 in (A.29) is bounded by

$$C_2 \leq \sum_{i,j} \frac{|\gamma_{i,j}|}{\sigma^4} e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}\,\mathrm{Tr}(A_{i,j}^T A_{ij})\,\mathrm{Tr}(P_{\bar{W}^*}^T P_{\bar{W}^*}) \tag{A.34}$$

Similarly, the remaining terms in (A.28) can be bounded by

$$C_1 = \mathrm{Tr}(P_{\bar{W}^*}^T \bar{\Lambda}^* P_{\bar{W}^*}) \geq \min_i(\bar{\Lambda}^*{}_i)\,\mathrm{Tr}(P_{\bar{W}^*}P_{\bar{W}^*}^T) \tag{A.35}$$

$$C_3 = -\,\mathrm{Tr}(P_{\bar{W}^*}\Lambda^* P_{\bar{W}^*}^T) \geq -\max_i(\Lambda_i^*)\,\mathrm{Tr}(P_{\bar{W}^*}^T P_{\bar{W}^*}). \tag{A.36}$$

Using the bounds for each term, the Equation (A.30) can be rewritten as

$$[\min_i(\bar{\Lambda}^*{}_i) - \max_j(\Lambda_j^*)]\,\mathrm{Tr}(P_{\bar{W}^*}^T P_{\bar{W}^*}) \geq \sum_{i,j} \frac{|\gamma_{i,j}|}{\sigma^4} e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}\,\mathrm{Tr}(A_{i,j}^T A_{ij})\,\mathrm{Tr}(P_{\bar{W}^*}^T P_{\bar{W}^*}) \tag{A.37}$$

$$[\min_i(\bar{\Lambda}^*{}_i) - \max_j(\Lambda_j^*)] \geq \sum_{i,j} \frac{|\gamma_{i,j}|}{\sigma^4} e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}\,\mathrm{Tr}(A_{i,j}^T A_{ij}) \tag{A.38}$$

It should be noted that Λ^* is a function of $\frac{1}{\sigma^2}$. This relationship could be removed by multiplying both sides of the inequality by σ^* to yield

$$\sigma^2[\min_i(\bar{\Lambda}^*{}_i) - \max_j(\Lambda_j^*)] \geq \sum_{i,j} \frac{|\gamma_{i,j}|}{\sigma^2} e^{-\frac{\mathrm{Tr}((W^{*^T}A_{i,j}W^*)}{2\sigma^2}}\,\mathrm{Tr}(A_{i,j}^T A_{ij}). \tag{A.39}$$

Since σ^2 is always a positive value, as long as all the eigenvalues from $\bar{\Lambda}^*$ is larger than all the eigenvalues from Λ^*, the left hand side of the equation will always be greater than 0. As $\sigma \to \infty$, the right hand side approaches 0, and the condition (C.54) of Lemma 3.1 is satisfied. □

As a side note, the eigen gap between $\min(\bar{\Lambda}^*)$ and $\max(\Lambda^*)$ controls the range of potential σ values —i.e. the larger the eigen gap the easier for σ to satisfy (A.39). Therefore, the ideal cutoff point should have a large eigen gap.

Appendix B

This section provides further detail to the chapter on KNet Clustering. Here, we provide the derivation that relates HSIC to Spectral Clustering and supply futher experimental results.

B.1 Relating HSIC to Spectral Clustering Derivation

Spectral Clustering is capable of clustering non-convex datasets. The key is to choose the appropriate similarity measure to form its Adjacency Matrix. It turns out that the HSIC objective performs a very similar task. Here, we provide the proof of their relationship.

Proof. Using Eq. (4.4) to compute HSIC emprically, Eq (4.5) can be rewritten as

$$Y^* = \arg\max_Y Tr(D^{-1/2}K_X D^{-1/2} H K_Y H), \tag{B.1}$$

where $D^{-1/2}K_X D^{-1/2}$ and K_Y are the kernel matrices computed from X and Y. As shown by [12], if we let K_Y be a linear kernel such that $K_Y = YY^T$, add the constraint such that $Y^TY = I$ and rotate the trace terms we get

$$[Y^*] = \arg\max_Y Tr(Y^T H D^{-1/2} K_X D^{-1/2} H Y) \tag{B.2a}$$

$$\text{s.t} : Y^TY = I. \tag{B.2b}$$

By setting the Laplacian as $\mathcal{L} = HD^{-1/2}K_X D^{-1/2}H$, the formulation becomes identical to Spectral

Clustering as

$$[Y^*] = \arg\max_{Y} Tr(Y^T \mathcal{L} Y) \tag{B.3a}$$

$$\text{s.t} : Y^T Y = I. \tag{B.3b}$$

□

B.2 Derivation for Eq. (4.11)

By ignoring the reconstruction error, the loss from Eq. (4.8) becomes

$$\mathbb{H}(\Psi_\theta(X), U). \tag{B.4}$$

From [8], this expression can be expanded into

$$Tr(K_{\Psi_\theta(X)} D^{-1/2} H K_U H D^{-1/2}). \tag{B.5}$$

Since $D^{-1/2} H K_U H D^{-1/2}$ is a constant matrix, we let it equal to Γ, and the objective becomes

$$Tr(\Gamma K_{\Psi_\theta(X)}). \tag{B.6}$$

The trace term can be converted into a matrix sum as

$$\sum_{i,j} \Gamma_{i,j} K_{\Psi_\theta(X),i,j}. \tag{B.7}$$

By replacing the kernel with the Gaussian kernel, Eq. (4.11) emerges as

$$\textstyle\sum_{i,j} \Gamma_{i,j} e^{-\frac{1}{2\sigma^2}||\psi(\mathbf{x}_i;\theta) - \psi(\mathbf{x}_j\theta)||^2}, \tag{B.8}$$

B.3 Moon and Spiral dataset Statistics for KNet Clustering

In this appendix, we illustrate that for the Moon and the Spiral datasets, we are able to learn (a) convex images for the clusters and (b) kernels that produces block diagonal structures. The kernel matrix is constructed with a Gaussian kernel, therefore the values are between 0 to 1. The kernel

matrices shown in the figures below use white as 0 and dark blue as 1; all values in between are shown as a gradient between the two colors.

In Figure B.1, the Moon dataset X is plotted in Fig. B.1(a) and its kernel block structure in Fig. B.1(b). After training Ψ, the image of Ψ is shown in Fig. B.1(c) along with its block diagonal structure in Fig. B.1(d). Using the same Ψ trained on X, we distorted X with Gaussian noise and plot it in Fig. B.2(a) along with its kernel matrix in Fig. B.2(b). We then pass the distorted X into Ψ and plot the resulting image in Fig. B.2(c) along with its kernel matrix in Fig. B.2(d). From this example, we demonstrate KNet's ability to embed data into convex clusters even under Gaussian noise.

In Figure B.3, a subset of 300 samples of the Spiral dataset is plotted in Fig. B.3(a) and its kernel block structure in Fig. B.3(b). After training Ψ, the image of Ψ is shown in Fig. B.3(c) along with its block diagonal structure in Fig. B.3(d). The full dataset is shown in (a) of Figure B.4 along with its kernel matrix in Fig. B.4(b). Using the same Ψ trained from Fig. B.3(a), we pass the full dataset into Ψ and plot the resulting image in Fig. B.3(b) along with its kernel matrix in Fig. B.3(d). From this example, we demonstrate KNet's ability to generalize convex embedding using only 1% of the data.

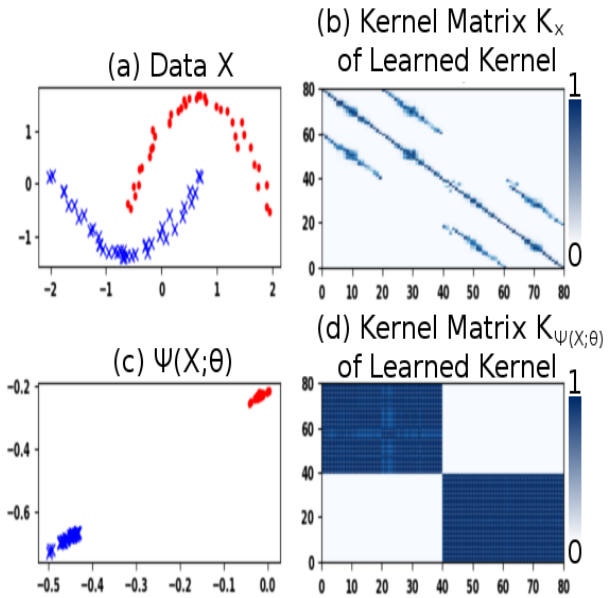

Figure B.1: This figure plots out the effect of training Ψ on the Moon dataset. The original data and its kernel matrix is shown in (a) and (b) respectively. The embedding of the data with Ψ and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's ability to maps non-convex clustering into convex representation.

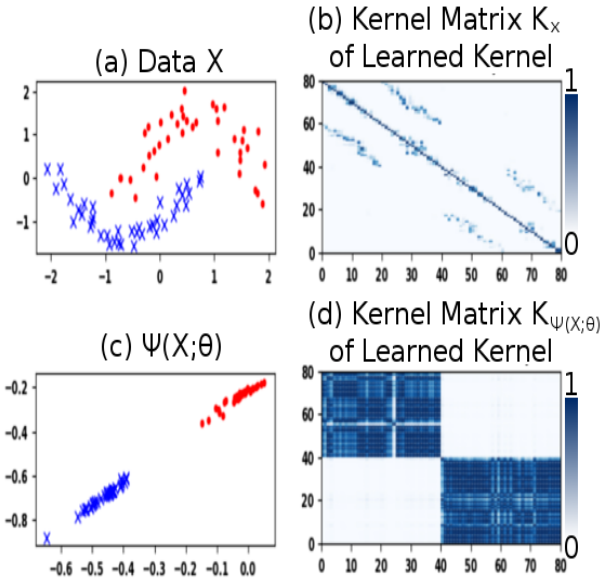

Figure B.2: This figure plots out the effect of applying a trained Ψ on a distorted Moon dataset. The distorted data and its kernel matrix is shown in (a) and (b) respectively. The embedding of the distorted data with Ψ and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's robustness in mapping non-convex clustering into convex representation under Gaussian distortion.

B.4 KNet Clustering Algorithm Experimental Hyperparameters

The hyperparameters for each network are set as outlined in the respective papers. In the case of SN, the hyper-parameters included the number of neighbors for calculation, the number of neighbors to use for graph Laplacian affinity matrix, the number of neighbors to use to calculate the scale of the Gaussian graph Laplacian, and the threshold for calculating the closest neighbors in the Siamese network. They were set by doing a grid search over values ranging from 2 to 10 and by using the loss over 10% of the training data.

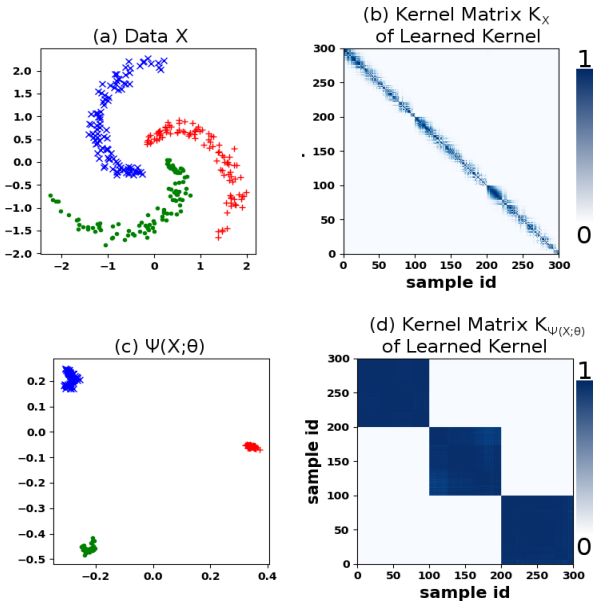

Figure B.3: This figure plots out the effect of training and applying Ψ on a small subset of the Spiral dataset. The subset and its kernel matrix is shown in (a) and (b) respectively. The embedding of the subset and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's ability to map non-convex cluster into convex representation with only a small subset of the data.

Algorithm	**Learning Rate**	**Batch size**	**Optimizer**
AEC	-	100	SGD
DEC	0.01	256	SGDSolver
IMSAT	0.002	250	Adam
SN	0.001	128	RMSProp
KNet	0.001	5	Adam

Table B.1: Here we include the typically used learning rates and batch sizes, and the optimizer type for each algorithm. These were set as recommended by the respective papers, except in the case of AEC which is silent on what learning rate needs to be set, the available implementation sets the learning rate with a line search. We use the above mentioned settings generally and only change them if the batch size is too big for a dataset or we notice the preset learning rate not leading to convergence.

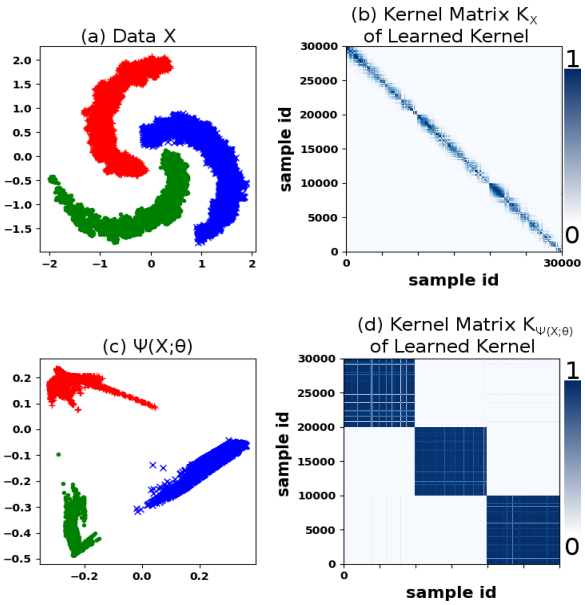

Figure B.4: This figure plots out the effect of applying Ψ trained on a small subset to the full Spiral dataset. The full dataset and its kernel matrix is shown in (a) and (b) respectively. The embedding of the full dataset and its kernel matrix is shown in (c) and (d) respectively. This figure demonstrates KNet's ability to generate convex representation on a large scale while training only on a small subset of the data. In this particular case, 1%.

Appendix C

In this Appendix, we provide further detail on IKDR background. The derivation for the Φ matrix for the entire ISM family is provided here. We also provide the proof that extends the original ISM proof to a generalized ISM formulation. Lastly, we provide the proof to demonstrate how conic combination of of ISM kernels is still an ISM kernel.

C.1 Kernel Definitions

For IKDR applications, we have discovered a family kernels for the ISM algorithm. We refer to this family as the ISM family. Here we provide the definition of several common kernels with relation to the projection matrix W in terms of the kernel and as a function of $\beta = \mathbf{a}WW^T\mathbf{b}$.

Linear Kernel

$$k(x_i, x_j) = x_i^T WW^T x_j, \qquad f(\beta) = \beta. \tag{C.1}$$

Polynomial Kernel

$$k(x_i, x_j) = (x_i^T WW^T x_j + c)^p, \qquad f(\beta) = (\beta + c)^p. \tag{C.2}$$

Gaussian Kernel

$$k(x_i, x_j) = e^{-\frac{(x_i - x_j)^T WW^T (x_i - x_j)}{2\sigma^2}}, \qquad f(\beta) = e^{-\frac{\beta}{2\sigma^2}}. \tag{C.3}$$

Squared Kernel

$$k(x_i, x_j) = (x_i - x_j)^T WW^T (x_i - x_j), \qquad f(\beta) = \beta. \tag{C.4}$$

Multiquadratic Kernel

$$k(x_i, x_j) = \sqrt{(x_i - x_j)^T W W^T (x_i - x_j) + c^2}, \qquad f(\beta) = \sqrt{\beta + c^2}. \tag{C.5}$$

C.2 Derivation for Approximating each Φ_0

For ISM, we have generalized the Φ matrix to include other kernels besides Gaussian. To initialize Φ, we approximate this matrix using the 2nd order Taylor's approximation. Here, we provide the derivation of this approximation for several common kernels within the ISM family.

Using Eq. (C.99), we know that

$$\Phi_0 = \text{sign}(\mu) \sum_{i,j} \Gamma_{i,j} A_{i,j}. \tag{C.6}$$

If $\mathbf{a}$ and $\mathbf{b}$ are both defined as $x_i - x_j$, then

$$\Phi_0 = \text{sign}(4\mu) X^T (D_\Gamma - \Gamma) X. \tag{C.7}$$

However, if $\mathbf{a}$ and $\mathbf{b}$ are defined as (x_i, x_j), then

$$\Phi_0 = \text{sign}(2\mu) X^T \Gamma X. \tag{C.8}$$

Therefore, to compute Φ_0, the key is to first determine the ($\mathbf{a}$, $\mathbf{b}$) based on the kernel and then find μ to determine the sign.

Φ_0 for the Linear Kernel: With a Linear Kernel, $(\mathbf{a}, \mathbf{b})$ uses (x_i, x_j), therefore Eq. (C.8) is use. Since $f(\beta) = \beta$, the sign of the gradient with respect to β is

$$\text{sign}(2\nabla_\beta f(\beta)) = \text{sign}(2) = 1. \tag{C.9}$$

Therefore,

$$\Phi_0 = X^T \Gamma X. \tag{C.10}$$

Φ_0 for the Polynomial Kernel: With a Polynomial Kernel, $(\mathbf{a}, \mathbf{b})$ uses (x_i, x_j), therefore Eq. (C.8) is use. Since $f(\beta) = (\beta + c)^p$, the sign of the gradient with respect to β is

$$\text{sign}(2\nabla_\beta f(\beta)) = \text{sign}(2p(\beta + c)^{p-1}) = 1. \tag{C.11}$$

Therefore,

$$\Phi_0 = X^T \Gamma X. \tag{C.12}$$

Φ_0 for the Gaussian Kernel: With a Gaussian Kernel, $(\mathbf{a}, \mathbf{b})$ uses $x_i - x_j$, therefore Eq. (C.7) is use. Since $f(\beta) = e^{-\frac{\beta}{2\sigma^2}}$, the sign of the gradient with respect to β is

$$\text{sign}(4\nabla_\beta f(\beta)) = \text{sign}(-\frac{4}{2\sigma^2} e^{-\frac{\beta}{2\sigma^2}}) = -1. \tag{C.13}$$

Therefore,

$$\Phi_0 = -X^T (D_\Gamma - \Gamma) X. \tag{C.14}$$

Φ_0 for the RBF Relative Kernel: With a RBF Relative Kernel, it is easier to start with the Lagrangian once we have approximated relative Kernel with the 2nd order Taylor expansion as

$$\mathcal{L} \approx -\sum_{i,j} \Gamma_{i,j} \left[1 + \text{Tr}(W^T (-\frac{1}{\sigma_i \sigma_j} A_{i,j}) W) \right] - \text{Tr}\left[\Lambda (W^T W - I) \right]. \tag{C.15}$$

The gradient of the Lagrangian is therefore

$$\nabla_W \mathcal{L} \approx \left[\sum_{i,j} \Gamma_{i,j} (\frac{2}{\sigma_i \sigma_j} A_{i,j}) \right] W - 2W\Lambda. \tag{C.16}$$

Setting the gradient to 0, we get

$$\left[\sum_{i,j} (\frac{1}{\sigma_i \sigma_j} \Gamma_{i,j} A_{i,j}) \right] W = W\Lambda. \tag{C.17}$$

If we let $\Sigma_{i,j} = \frac{1}{\sigma_i \sigma_j}$ and $\Psi = \Sigma \odot \Gamma$, then we end up with

$$4\left[X^T (D_\Psi - \Psi) X \right] W = W\Lambda. \tag{C.18}$$

This equation requires W to be the eigenvectors associated with the smallest eigenvalues. We flip the sign so the most dominant eigenvectors are the solution. Therefore, we define Φ as

$$\Phi = -X^T (D_\Psi - \Psi) X \tag{C.19}$$

Φ_0 for the Squared Kernel: With a Squared Kernel, $(\mathbf{a}, \mathbf{b})$ uses $x_i - x_j$, therefore Eq. (C.7) is use. Since $f(\beta) = \beta$, the sign of the gradient with respect to β is

$$\text{sign}(4\nabla_\beta f(\beta)) = \text{sign}(4) = 1. \tag{C.20}$$

Therefore,

$$\Phi_0 = X^T(D_\Gamma - \Gamma)X. \tag{C.21}$$

Φ_0 for the Multiquadratic Kernel: With a Multiquadratic Kernel, $(\mathbf{a}, \mathbf{b})$ uses $x_i - x_j$, therefore Eq. (C.7) is use. Since $f(\beta) = \sqrt{\beta + c^2}$, the sign of the gradient with respect to β is

$$\text{sign}(4\nabla_\beta f(\beta)) = \text{sign}(\frac{4}{2}(\beta + c^2)^{-1/2}) = 1. \tag{C.22}$$

Therefore,

$$\Phi_0 = X^T(D_\Gamma - \Gamma)X. \tag{C.23}$$

C.3 Derivation for each Φ

For ISM, we have generalized the Φ matrix to include other kernels besides Gaussian. Here, we provide the derivation of this matrix for several common kernels within the ISM family.

Using Eq. (5.2), we know that

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\nabla_\beta f(\beta)]A_{i,j}. \tag{C.24}$$

If we let $\Psi = \Gamma_{i,j}[\nabla_\beta f(\beta)]$ then Φ can also be written as

$$\Phi = \frac{1}{2}\sum_{i,j}\Psi_{i,j}A_{i,j}. \tag{C.25}$$

If $\mathbf{a}$ and $\mathbf{b}$ are both defined as $x_i - x_j$, then

$$\Phi = 2X^T(D_\Psi - \Psi)X. \tag{C.26}$$

However, if $\mathbf{a}$ and $\mathbf{b}$ are defined as (x_i, x_j), then

$$\Phi = X^T\Psi X. \tag{C.27}$$

Therefore, to compute Φ, the key is to first determine the ($\mathbf{a}$, $\mathbf{b}$) based on the kernel and then find the appropriate Ψ.

Φ for the Linear Kernel: With a Linear Kernel, $(\mathbf{a}, \mathbf{b})$ uses (x_i, x_j), therefore Eq. (C.27) is use. Since $f(\beta) = \beta$, Φ becomes

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\nabla_\beta f(\beta)]A_{i,j} = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}A_{i,j}. \tag{C.28}$$

Since, we are only interested in the eigenvectors of Φ only the sign of the constants are necessary. Therefore,

$$\Phi = \text{sign}(1)X^T\Gamma X = X^T\Gamma X. \tag{C.29}$$

Φ for the Polynomial Kernel: With a Polynomial Kernel, $(\mathbf{a}, \mathbf{b})$ uses (x_i, x_j), therefore Eq. (C.27) is use. Since $f(\beta) = (\beta + c)^p$, Φ becomes

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\nabla_\beta f(\beta)]A_{i,j} = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[p(\beta + c)^{p-1}]A_{i,j}. \tag{C.30}$$

Since p is a constant, and $K_{XW,p-1} = (\beta + c)^{p-1}$ is the polynomial kernel itself with power of $(p-1)$, Ψ becomes

$$\Psi = \Gamma \odot K_{XW,p-1}, \tag{C.31}$$

and

$$\Phi = \text{sign}(p)X^T\Psi X = X^T\Psi X \tag{C.32}$$

Φ for the Gaussian Kernel: With a Gaussian Kernel, $(\mathbf{a}, \mathbf{b})$ uses $x_i - x_j$, therefore Eq. (C.7) is use. Since $f(\beta) = e^{-\frac{\beta}{2\sigma^2}}$, Φ becomes

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\nabla_\beta f(\beta)]A_{i,j} = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[-\frac{1}{2\sigma^2}e^{-\frac{\beta}{2\sigma^2}}]A_{i,j} = -\frac{1}{4\sigma^2}\sum_{i,j}\Gamma_{i,j}[K_{XW}]_{i,j}A_{i,j}. \tag{C.33}$$

If we let $\Psi = \Gamma \odot K_{XW}$, then

$$\Phi = \text{sign}(-\frac{2}{4\sigma^2})X^T(D_\Psi - \Psi)X = -X^T(D_\Psi - \Psi)X. \tag{C.34}$$

Φ for the Squared Kernel: With a Squared Kernel, $(\mathbf{a}, \mathbf{b})$ uses $x_i - x_j$, therefore Eq. (C.7) is use. Since $f(\beta) = \beta$, Φ becomes

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\nabla_\beta f(\beta)]A_{i,j} = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}A_{i,j}. \tag{C.35}$$

Therefore,

$$\Phi = \text{sign}(1)X^T(D_\Gamma - \Gamma)X = X^T(D_\Gamma - \Gamma)X. \tag{C.36}$$

Φ for the Multiquadratic Kernel: With a Multiquadratic Kernel, $(\mathbf{a}, \mathbf{b})$ uses $x_i - x_j$, therefore Eq. (C.7) is use. Since $f(\beta) = \sqrt{\beta + c^2}$, Φ becomes

$$\Phi = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\nabla_\beta f(\beta)]A_{i,j} = \frac{1}{2}\sum_{i,j}\Gamma_{i,j}[\frac{1}{2}(\beta + c^2)^{-1/2}]A_{i,j} = \frac{1}{4}\sum_{i,j}\Gamma_{i,j}[K_{XW}]^{(-1)}_{i,j}A_{i,j}. \tag{C.37}$$

If we let $\Psi = \Gamma \odot K^{(-1)}_{XW}$, then

$$\Phi = \text{sign}(\frac{1}{4})X^T(D_\Psi - \Psi)X = X^T(D_\Psi - \Psi)X. \tag{C.38}$$

Φ_0 for the RBF Relative Kernel: With a RBF Relative Kernel, we start with the initial Lagrangian

$$\mathcal{L} = \sum_{i,j}\Gamma_{i,j}e^{-\frac{Tr(W^TA_{i,j}W)}{2\sigma_i\sigma_j}} - \text{Tr}(\Lambda(W^TW - I)) \tag{C.39}$$

where the gradient becomes

$$\nabla_W\mathcal{L} = -\sum_{i,j}\frac{1}{\sigma_i\sigma_j}\Gamma_{i,j}e^{-\frac{Tr(W^TA_{i,j}W)}{2\sigma_i\sigma_j}}A_{i,j}W - 2W\Lambda. \tag{C.40}$$

If we let $\Sigma_{i,j} = \frac{1}{\sigma_i\sigma_j}$ then we get

$$\nabla_W\mathcal{L} = -\sum_{i,j}\Psi_{i,j}A_{i,j}W - 2W\Lambda, \tag{C.41}$$

where $\Psi_{i,j} = \Sigma_{i,j}\Gamma_{i,j}K_{XW_{i,j}}$. If we apply Appendix C.8 and set the gradient to 0, then we get

$$-4\left[X^T(D_\Psi - \Psi)X\right]W = 2W\Lambda. \tag{C.42}$$

From here, we see that it has the same form as the Gaussian kernel, with Ψ defined as $\Psi = \Sigma \odot \Gamma \odot K_{XW}$.

This equation requires W to be the eigenvectors associated with the smallest eigenvalues. We flip the sign so the most dominant eigenvectors are the solution. Therefore, we define Φ as

$$\Phi = X^T(D_\Psi - \Psi)X \tag{C.43}$$

C.4 Proof for Theorem 5.1

To extend ISM to a family of kernels, we show that the original proof can be generalized to a common formulation. Here, we provide the proof of this extension.

The main body of the proof is organized into two lemmas where the 1st lemma will prove the 1st order condition and the 2nd lemma will prove the 2nd order condition. For convenience, we included the 2nd Order Necessary Condition [24] as Theorem. 2.1 in the Background Chapter. We also convert the optimization problem into a standard minimization form where we solve

$$\min_{W} -\operatorname{Tr}(\Gamma K_{XW}) \quad \text{s.t. } W^T W = I. \tag{C.44}$$

The proof is initialized by manipulating the different kernels into a common form. If we let $\beta = a(x_i, x_j)WW^T b(x_i, x_j)$, then the kernels in this family can be expressed as $f(\beta)$. This common form allows a universal proof that works for all kernels that belongs to the ISM family. Depending on the kernel, the definition of f, $a(x_i, x_j)$ and $b(x_i, x_j)$ are listed in Table C.1. Kernels in this form are functions of the Grassmannian WW^T.

Name	$f(\beta)$	$a(x_i, x_j)$	$b(x_i, x_j)$
Linear	β	x_i	x_j
Polynomial	$(\beta + c)^p$	x_i	x_j
Gaussian	$e^{\frac{-\beta}{2\sigma^2}}$	$x_i - x_j$	$x_i - x_j$
Squared	β	$x_i - x_j$	$x_i - x_j$

Table C.1: Common components of different Kernels.

Lemma C.1. *Given $\mathcal{L}$ as the Lagrangian of Eq. (5.1), if W^* is a fixed point of Algorithm 4, and Λ^* is a diagonal matrix of its corresponding eigenvalues, then*

$$\nabla_W \mathcal{L}(W^*, \Lambda^*) = 0, \tag{C.45}$$

$$\nabla_\Lambda \mathcal{L}(W^*, \Lambda^*) = 0. \tag{C.46}$$

Proof. Since $\operatorname{Tr}(\Gamma K_{XW}) = \sum_{i,j} \Gamma_{i,j} K_{XW_{i,j}}$, where the subscript indicates the i, jth element of the associated matrix. If we let $\mathbf{a} = a(x_i, x_j), \mathbf{b} = b(x_i, x_j)$, the Lagrangian of Eq. (5.1) becomes

$$\mathcal{L}(W, \Lambda) = -\sum_{ij} \Gamma_{ij} f(\mathbf{a}^T W W^T \mathbf{b}) - \operatorname{Tr}[\Lambda(W^T W - I)]. \tag{C.47}$$

The gradient of the Lagrangian with respect to W is

$$\nabla_W \mathcal{L}(W, \Lambda) = -\sum_{ij} \Gamma_{ij} f'(\mathbf{a}^T W W^T \mathbf{b})(\mathbf{b}\mathbf{a}^T + \mathbf{a}\mathbf{b}^T)W - 2W\Lambda. \tag{C.48}$$

If we let $A_{i,j} = \mathbf{b}\mathbf{a}^T + \mathbf{a}\mathbf{b}^T$ then setting $\nabla_W \mathcal{L}(W, \Lambda)$ of Eq. (C.48) to 0 yields the relationship

$$0 = \left[-\frac{1}{2} \sum_{ij} \Gamma_{ij} f'(\mathbf{a}^T W W^T \mathbf{b}) A_{i,j} \right] W - W\Lambda. \tag{C.49}$$

Since $f'(\mathbf{a}^T W W^T \mathbf{b})$ is a scalar value that depends on indices i, j, we multiply it by $-\frac{1}{2}\Gamma_{i,j}$ to form a new variable $\Psi_{i,j}$. Then Eq. (C.49) can be rewritten as

$$\left[\sum_{ij} \Psi_{ij} A_{i,j} \right] W = W\Lambda. \tag{C.50}$$

To match the form shown in Table 5.2, Appendix C.7 further showed that if $\mathbf{a}$ and $\mathbf{b}$ is equal to x_i and x_j, then

$$\left[\sum_{ij} \Psi_{ij} A_{i,j} \right] = 2X^T \Psi X. \tag{C.51}$$

From Appendix C.8, if $\mathbf{a}$ and $\mathbf{b}$ are equal to $x_i - x_j$, then

$$\left[\sum_{ij} \Psi_{ij} A_{i,j} \right] = 4X^T [D_\Psi - \Psi] X. \tag{C.52}$$

If we let $\Phi = \left[\sum_{ij} \Psi_{ij} A_{i,j} \right]$, it yields the relationship $\Phi W = W\Lambda$ where the eigenvectors of Φ satisfies the 1st order condition of $\nabla_W \mathcal{L}(W^*, \Lambda^*) = 0$. The gradient with respect to Λ yields the expected constraint

$$\nabla_\Lambda \mathcal{L} = W^T W - I. \tag{C.53}$$

Since the eigenvectors of Φ is orthonormal, the condition $\nabla_\Lambda \mathcal{L} = 0 = W^T W - I$ is also satisfied. Observing these 2 properties, Lemma C.1 confirms that the eigenvectors of Φ also satisfies the 1st order condition from Eq. (5.1).

□

Lemma C.2. *Given a full rank Φ, an eigengap defined by Eq. (C.73), and W^* as the fixed point of Algorithm 4, then*

$$\begin{aligned} &\mathrm{Tr}(Z^T \nabla^2_{WW} \mathcal{L}(W^*, \Lambda^*) Z) \geq 0 \\ &\qquad \text{for all } Z \neq 0, \text{with } \nabla h(W^*)^T Z = 0. \end{aligned} \tag{C.54}$$

Proof. To proof Lemma 2, we must relate the concept of eigengap to the conditions of

$$\mathrm{Tr}(Z^T \nabla^2_{WW} \mathcal{L}(W^*, \Lambda^*) Z) \geq 0 \quad \forall \quad Z \neq 0 \quad \text{with} \quad \nabla h(W^*)^T Z = 0 \ . \tag{C.55}$$

Given the constraint $h(W) = W^T W - I$, we start by computing the constrain $\nabla h(W^*)^T Z = 0$. Given

$$\nabla h(W^*)^T Z = \lim_{t \to 0} \frac{\partial}{\partial t} h(W + tZ), \tag{C.56}$$

the constraint becomes

$$\begin{aligned} \nabla h(W^*)^T Z = 0 \quad &= \quad \lim_{t \to 0} \frac{\partial}{\partial t} [(W + tZ)^T (W + tZ) - I], \\ 0 \quad &= \quad \lim_{t \to 0} \frac{\partial}{\partial t} [(W^T W + t W^T Z + t Z^T W + t^2 Z^T Z) - I], \\ 0 \quad &= \quad \lim_{t \to 0} W^T Z + Z^T W + 2t Z^T Z. \end{aligned} \tag{C.57}$$

By setting the limit to 0, an important relationship emerges as

$$0 \quad = \quad W^T Z + Z^T W. \tag{C.58}$$

Given a full rank operator Φ, its eigenvectors must span the complete $\mathcal{R}^d$ space. If we let W and $\bar{W}$ represent the eigenvectors chosen and not chosen respectively from Algorithm 1, and let B and $\bar{B}$ be scambling matrices, then the matrix $Z \in \mathcal{R}^{d \times q}$ can be rewritten as

$$Z = WB + \bar{W}\bar{B}. \tag{C.59}$$

It should be noted that since W and $\bar{W}$ are eigenvalues of the symmetric matrix Φ, they are orthogonal to each other, i.e., $W^T \bar{W} = 0$. Furthermore, if we replace Z in Eq. (C.58) with Eq. (C.59), we get the condition

$$\begin{aligned} 0 \quad &= \quad W^T (WB + \bar{W}\bar{B}) + (WB + \bar{W}\bar{B})^T W \\ 0 \quad &= \quad B + B^T. \end{aligned} \tag{C.60}$$

From Eq. (C.60), we observe that B must be a antisymmetric matrix because $B = -B^T$. Next, we work to compute the inequality of of Eq. (C.55) by noting that

$$\nabla^2_{WW} \mathcal{L}(W, \Lambda) Z = \lim_{t \to 0} \frac{\partial}{\partial t} \nabla \mathcal{L}(W + tZ). \tag{C.61}$$

Also note that Lemma 1 has already computed $\nabla_W \mathcal{L}(W)$ as

$$\nabla_W \mathcal{L}(W) = -\frac{1}{2}\left[\sum_{i,j} \Gamma_{i,j} f'(\beta) A_{i,j}\right] W - W\Lambda. \tag{C.62}$$

Since we need $\nabla_W \mathcal{L}$ to be a function of $W + tZ$ with t as the variable, we change $\beta(W)$ into $\beta(W + tZ)$ with

$$\begin{aligned} \beta(W+tZ) &= \boldsymbol{a}(W+tZ)(W+tZ)^T\boldsymbol{b}, \\ &= \boldsymbol{a}^T W W^T \boldsymbol{b} + [\boldsymbol{a}^T(WZ^T + ZW^T)\boldsymbol{b}]t + [\boldsymbol{a}^T Z Z^T \boldsymbol{b}]t^2, \\ &= \beta + c_1 t + c_2 t^2, \end{aligned} \tag{C.63}$$

where β, c_1, and c_2 are constants with respect to t. Using the β from Eq. (C.63) with $\nabla_W \mathcal{L}$, we get

$$\nabla^2_{WW}\mathcal{L}(W,\Lambda)Z = \lim_{t \to 0} \frac{\partial}{\partial t}\left[-\frac{1}{2}\sum_{i,j} \Gamma_{i,j} f'(\beta + c_1 t + c_2 t^2) A_{i,j}\right](W+tZ) - (W+tZ)\Lambda. \tag{C.64}$$

If we take the derivative with respect to t and then set the limit to 0, we get

$$\nabla^2_{WW}\mathcal{L}(W,\Lambda)Z = \left[-\frac{1}{2}\sum_{i,j} \Gamma_{i,j} f''(\beta) c_1 A_{i,j}\right] W + \left[-\frac{1}{2}\sum_{i,j} \Gamma_{i,j} f'(\beta) A_{i,j}\right] Z - Z\Lambda. \tag{C.65}$$

Next, we notice the definition of $\Phi = -\frac{1}{2}\sum \Gamma_{i,j} f'(\beta) A_{i,j}$ from Lemma 1, the term $\mathrm{Tr}(Z^T \nabla^2_{WW}\mathcal{L}(W,\Lambda)Z)$ can now be expressed as 3 separate terms as

$$\mathrm{Tr}(Z^T \nabla^2_{WW}\mathcal{L}(W,\Lambda)Z) = \mathcal{T}_1 + \mathcal{T}_2 + \mathcal{T}_3, \tag{C.66}$$

where

$$\mathcal{T}_1 = \mathrm{Tr}\left(Z^T\left[-\frac{1}{2}\sum_{i,j} \Gamma_{i,j} f''(\beta) c_1 A_{i,j}\right] W\right), \tag{C.67}$$

$$\mathcal{T}_2 = \mathrm{Tr}(Z^T \Phi Z), \tag{C.68}$$

$$\mathcal{T}_3 = -\mathrm{Tr}(Z^T Z \Lambda). \tag{C.69}$$

Since $\mathcal{T}_1$ cannot be further simplified, the concentration will be on $\mathcal{T}_2$ and $\mathcal{T}_3$. If we let $\bar{\Lambda}$ and Λ be the corresponding eigenvlaue matrices associated with $\bar{W}$ and W, by replacing Z in $\mathcal{T}_2$ from

Eq. (C.68), we get

$$\begin{aligned}
\mathrm{Tr}(Z^T\Phi Z) &= \mathrm{Tr}((WB+\bar{W}\bar{B})^T\Phi(WB+\bar{W}\bar{B})) \\
&= \mathrm{Tr}(B^TW^T\Phi WB + \bar{B}^T\bar{W}^T\Phi WB + B^TW^T\Phi\bar{W}\bar{B} + \bar{B}^T\bar{W}^T\Phi\bar{W}\bar{B}) \\
&= \mathrm{Tr}(B^TW^TW\Lambda B + \bar{B}^T\bar{W}^TW\Lambda B + B^TW^T\bar{W}\bar{\Lambda}\bar{B} + \bar{B}^T\bar{W}^T\bar{W}\bar{\Lambda}\bar{B}) \\
&= \mathrm{Tr}(B^T\Lambda B + 0 + 0 + \bar{B}^T\bar{\Lambda}\bar{B}) \\
&= \mathrm{Tr}(B^T\Lambda B + \bar{B}^T\bar{\Lambda}\bar{B}).
\end{aligned}$$

By replacing Z from $\mathcal{T}_3$ from Eq. (C.69), we get

$$\begin{aligned}
-\mathrm{Tr}(Z^TZ\Lambda) &= -\mathrm{Tr}((WB+\bar{W}\bar{B})^T(WB+\bar{W}\bar{B})\Lambda) \\
&= -\mathrm{Tr}(B^TW^TWB\Lambda + \bar{B}^T\bar{W}^TWB\Lambda + B^TW^T\bar{W}\bar{B}\Lambda + \bar{B}^T\bar{W}^T\bar{W}\bar{B}\Lambda) \\
&= -\mathrm{Tr}(B^TB\Lambda + 0 + 0 + \bar{B}^T\bar{B}\Lambda) \\
&= -\mathrm{Tr}(B\Lambda B^T + \bar{B}\Lambda\bar{B}^T).
\end{aligned}$$

The inequality that satisfies the 2nd order condition can now be written as

$$\mathrm{Tr}(B^T\Lambda B) + \mathrm{Tr}(\bar{B}^T\bar{\Lambda}\bar{B}) - \mathrm{Tr}(B\Lambda B^T) - \mathrm{Tr}(\bar{B}\Lambda\bar{B}^T) + \mathcal{T}_1 \geq 0. \tag{C.70}$$

Since B is an antisymmetric matrix, $B^T = -B$, and therefore $\mathrm{Tr}(B\Lambda B^T) = \mathrm{Tr}(B^T\Lambda B)$. From this Eq. (C.70) can be rewritten as

$$\mathrm{Tr}(B^T\Lambda B) - \mathrm{Tr}(B^T\Lambda B) + \mathrm{Tr}(\bar{B}^T\bar{\Lambda}\bar{B}) - \mathrm{Tr}(\bar{B}\Lambda\bar{B}^T) + \mathcal{T}_1 \geq 0. \tag{C.71}$$

With the first two terms canceling each other out, the inequality can be rewritten as

$$\mathrm{Tr}(\bar{B}^T\bar{\Lambda}\bar{B}) - \mathrm{Tr}(\bar{B}\Lambda\bar{B}^T) \geq -\mathcal{T}_1. \tag{C.72}$$

With this inequality, the terms can be further bounded by

$$\mathrm{Tr}(\bar{B}^T\bar{\Lambda}\bar{B}) \geq \min_i \bar{\Lambda}_i \,\mathrm{Tr}(\bar{B}\bar{B}^T)$$

$$\mathrm{Tr}(\bar{B}\Lambda\bar{B}^T) \leq \max_j \Lambda_j \,\mathrm{Tr}(\bar{B}^T\bar{B})$$

Noting that since $\mathrm{Tr}(\bar{B}\bar{B}^T) = \mathrm{Tr}(\bar{B}^T\bar{B})$, we treat it as a constant value of α. With this, the inequality can be rewritten as

$$\left(\min_i \bar{\Lambda}_i - \max_j \Lambda_j\right) \geq -\frac{1}{\alpha}\mathcal{T}_1.$$

Here, since $-\frac{1}{\alpha}\mathcal{T}_1$ is simply a constant, we denote it as $\mathcal{C}$ to yield the final conclusion that

$$\left(\min_i \bar{\Lambda}_i - \max_j \Lambda_j \right) \geq \mathcal{C}. \tag{C.73}$$

Eq. (C.73) concludes that to satisfy the 2nd order condition, the eigengap must be greater than $\mathcal{C}$. Therefore, given the choice of q eigenvectors, the eigengap is maximized when the eigenvectors associated with the q smallest eigenvalues are chosen as W. □

We note that it is customary for machine learning algorithms to look for the most dominant eigenvectors, crucially, many KDR algorithms follow this standard. Therefore, to maintain consistency, the Φ defined within the paper is actually the negative Φ from the proof. By flipping the sign, the eigenvectors associated with the smallest eigenvalues is now the most dominant eigenvectors. Hence, Φ within the paper is defined as

$$\Phi = \frac{1}{2}\sum_{ij} \Gamma_{ij} f'(\mathbf{a}^T W W^T \mathbf{b}) A_{i,j}. \tag{C.74}$$

C.5 Computing the Hessian for the Taylor Series

To approximate the initial Φ matrix for ISM, we compute the Hessian Matrix for the Taylor expansion of the kernel. Here, we provide the derivation for this process.

First we compute the gradient and the Hessian for $\beta(W)$ where

$$\beta(W) = a^T W W^T b, \tag{C.75}$$

$$\beta(W) = \mathrm{Tr}(W^T b a^T W), \tag{C.76}$$

$$\nabla_W \beta(W) = [ba^T + ab^T]W, \tag{C.77}$$

$$\nabla_{W,W} \beta(W) = [ba^T + ab^T], \tag{C.78}$$

$$\nabla_{W,W} \beta(W = 0) = [ba^T + ab^T]. \tag{C.79}$$

$$\tag{C.80}$$

Next, we compute the gradient and Hessian for $f(\beta(W))$ where

$$f(\beta(W)) = f(a^T W W^T b), \tag{C.81}$$

$$f(\beta(W)) = f(\mathrm{Tr}(W^T b a^T W)), \tag{C.82}$$

$$f'(\beta(W)) = \nabla_\beta f(\beta(W))[ba^T + ab^T]W = \nabla_\beta f(\beta(W))\nabla_W \beta(W) \tag{C.83}$$

$$f''(\beta(W) = \nabla_{\beta,\beta} f(\beta(W))[ba^T + ab^T]W(...) + \nabla_\beta f(\beta(W))[ba^T + ab^T] \tag{C.84}$$

$$f''(\beta(W=0)) = 0 + \nabla_\beta f(\beta(W))\nabla_{W,W}\beta(W=0) \tag{C.85}$$

$$f''(\beta(W=0)) = \nabla_\beta f(\beta(W))\nabla_{W,W}\beta(W=0) \tag{C.86}$$

$$f''(0) = \mu A_{i,j}. \tag{C.87}$$

Using Taylor Series the gradient of the Lagrangian is approximately

$$\nabla_W \mathcal{L} \approx -\sum_{i,j} \Gamma_{i,j} f''(0) W - 2W\Lambda, \tag{C.88}$$

$$\nabla_W \mathcal{L} \approx -\mu \sum_{i,j} \Gamma_{i,j} A_{i,j} W - 2W\Lambda. \tag{C.89}$$

Setting the gradient of the Lagrangian to 0 and combining the constant 2 to μ, it yields the relationship

$$\left[-\mu \sum_{i,j} \Gamma_{i,j} A_{i,j}\right] W = W\Lambda. \tag{C.90}$$

Here we note that μ is a constant. Therefore, only the sign will affect the eigenvector selection. With this, it yields

$$\left[-\mathrm{sign}(\mu) \sum_{i,j} \Gamma_{i,j} A_{i,j}\right] W = W\Lambda. \tag{C.91}$$

With this, the terms within the bracket become the initial Φ_0 as

$$\Phi_0 W = W\Lambda. \tag{C.92}$$

C.6 Derivation for Approximated Φ

We first convert the optimization problem into a standard minimization form where we solve

$$\min_W -\mathrm{Tr}(\Gamma K_{XW}) \quad \text{s.t. } W^T W = I. \tag{C.93}$$

Since the objective Lagrangian is non-convex, a solution can be achieved faster and more accurately if the algorithm is initialized at an intelligent starting point. Ideally, we wish to have a closed-form solution that yields the global optimal without any iterations. However, this is not possible since Φ is a function of W. ISM circumvents this problem by approximating the kernel using Taylor Series up to the 2nd order while expanding around 0. This approximation has the benefit of removing the dependency of W for Φ, therefore, a global minimum can be achieved using the approximated kernel. The ISM algorithm uses the global minimum found from the approximated kernel as the initialization point. Here, we provide a generalized derivation for the ISM kernel functions that are twice differentiable. First, we note that the 2nd order Taylor expansion for $f(\beta(W))$ around 0 is $f(\beta(W)) \approx f(0) + \frac{1}{2!}\operatorname{Tr}(W^T f''(0) W)$, where the 1st order expansion around 0 is equal to 0. Therefore, the ISM Lagrangian can be approximated with

$$\mathcal{L} = -\sum_{i,j} \Gamma_{i,j} \left[f(0) + \frac{1}{2!}\operatorname{Tr}(W^T f''(0) W) \right] - \operatorname{Tr}(\Lambda(W^T W - I)), \tag{C.94}$$

where the gradient of the Lagrangian is

$$\nabla_W \mathcal{L} = -\sum_{i,j} \Gamma_{i,j} f''(0) W - 2W\Lambda. \tag{C.95}$$

Next, we look at the kernel function $f(\beta(W))$ more closely. The Hessian is computed as

$$f'(\beta(W)) = \nabla_\beta f(\beta(W)) \nabla_W \beta(W), \tag{C.96}$$

$$f''(\beta(W=0)) = \nabla_\beta f(\beta(0)) \nabla_{W,W} \beta(0). \tag{C.97}$$

Since we skipped several steps for the computation of the Hessian, refer to Appendix C.5 for more detail. Because $\nabla_\beta f(\beta(0))$ is just a constant, we can bundle all constants into this term and refer to it as μ. Since $\nabla_{W,W}\beta(0) = A_{i,j}$, the Hessian is simply $\mu A_{i,j}$ regardless of the kernel. By combining constants setting the gradient of Eq. (C.95) to 0, we get the expression

$$\left[-\operatorname{sign}(\mu) \sum_{i,j} \Gamma_{i,j} A_{i,j} \right] W = W\Lambda, \tag{C.98}$$

where if we let $\Phi = -\operatorname{sign}(\mu)\sum_{i,j}\Gamma_{i,j}A_{i,j}$, we get a Φ that is not dependent on W. Therefore, a closed-form global minimum of the second-order approximation can be achieved. It should be noted that while the magnitude of μ can be ignored, the sign of μ cannot be neglected since it flips the sign of the eigenvalues of Ψ. Following Eq. (C.98), the initial Φ_0 for each kernel is shown in Table 5.1. We also provide detailed proofs for each kernel in Appendix C.2.

It is important to note that based on proof of Theorem 5.1 in Appendix C.4, the Φ as defined from Eq. (C.98) requires the optimal W to be the eigenvectors of Φ that is associated with the smallest eigenvalues. This is equivalent to the most dominant eigenvectors of negative Φ. To maintain consistency, the Φ defined with the paper is the negative Φ_0 from this derivation, and therefore the Φ_0 defined within the paper is

$$\Phi = \text{sign}(\mu) \sum_{i,j} \Gamma_{i,j} A_{i,j}. \tag{C.99}$$

C.7 Derivation for $\sum_{i,j} \Psi_{i,j} A_{i,j}$ if $A_{i,j} = x_i x_j^T + x_j x_i^T$

Since Ψ is a symmetric matrix and $A_{i,j} = (x_i x_j^T + x_j x_i^T)$, we first note that while $x_i x_j^T \neq x_j x_i^T$, it still hold that

$$\sum_{i,j} \Psi_{i,j} x_i x_j^T = \sum_{i,j} \Psi_{i,j} x_j x_i^T. \tag{C.100}$$

Therefore, we can rewrite the expression into

$$\sum_{i,j} \Psi_{i,j} A_{i,j} = 2 \sum_{i,j}^{n} \Psi_{i,j} x_i x_j^T.$$

If we expand the summation for $i = 1$, we get

$$\begin{aligned}
[\Psi_{1,1} x_1 x_1^T + \ldots + \Psi_{1,n} x_1 x_n^T] &= x_1 [\Psi_{1,1} x_1^T + \ldots + \Psi_{1,n} x_n^T] \\
&= x_1 \left[\begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} \begin{bmatrix} \Psi_{1,1} \\ . \\ \Psi_{1,n} \end{bmatrix} \right]^T \\
&= x_1 \left[\begin{bmatrix} \Psi_{1,1} & \ldots & \Psi_{1,n} \end{bmatrix} \begin{bmatrix} x_1^T \\ . \\ x_n^T \end{bmatrix} \right].
\end{aligned}$$

Now if we sum up all i, we get

$$\begin{aligned}
\Psi_{i,j}x_i x_j^T &= x_1\left[\begin{bmatrix}\Psi_{1,1} & \dots & \Psi_{1,n}\end{bmatrix}\begin{bmatrix}x_1^T\\ .\\ x_n^T\end{bmatrix}\right] + \dots + x_n\left[\begin{bmatrix}\Psi_{n,1} & \dots & \Psi_{n,n}\end{bmatrix}\begin{bmatrix}x_1^T\\ .\\ x_n^T\end{bmatrix}\right], \\
&= \left[x_1\begin{bmatrix}\Psi_{1,1} & \dots & \Psi_{1,n}\end{bmatrix} + \dots + x_n\begin{bmatrix}\Psi_{n,1} & \dots & \Psi_{n,n}\end{bmatrix}\right]\begin{bmatrix}x_1^T\\ .\\ x_n^T\end{bmatrix}, \\
&= \left[\begin{bmatrix}x_1 & \dots & x_n\end{bmatrix}\begin{bmatrix}\Psi_{1,1}\\ .\\ \Psi_{n,1}\end{bmatrix} + \dots + \begin{bmatrix}x_1 & \dots & x_n\end{bmatrix}\begin{bmatrix}\Psi_{1,n}\\ .\\ \Psi_{n,n}\end{bmatrix}\right]\begin{bmatrix}x_1^T\\ .\\ x_n^T\end{bmatrix}, \\
&= \begin{bmatrix}x_1 & \dots & x_n\end{bmatrix}\begin{bmatrix}\begin{bmatrix}\Psi_{1,1}\\ .\\ \Psi_{n,1}\end{bmatrix} & \dots & \begin{bmatrix}\Psi_{1,n}\\ .\\ \Psi_{n,n}\end{bmatrix}\end{bmatrix}\begin{bmatrix}x_1^T\\ .\\ x_n^T\end{bmatrix}.
\end{aligned}$$

Given that $X = \begin{bmatrix}x_1 & \dots & x_n\end{bmatrix}^T$, the final expression becomes.

$$2\sum_{i,j}^{n}\Psi_{i,j}x_i x_j^T = 2X^T\Psi X.$$

C.8 Derivation for $\sum_{i,j}\Psi_{i,j}A_{i,j}$ if $A_{i,j} = (x_i - x_j)(x_i - x_j)^T + (x_i - x_j)(x_i - x_j)^T$

Since Ψ is a symmetric matrix, and $A_{i,j} = (x_i - x_j)(x_i - x_j)^T + (x_i - x_j)(x_i - x_j)^T = 2(x_i - x_j)(x_i - x_j)^T$, we can rewrite the expression into

$$\begin{aligned}
\textstyle\sum_{i,j}\Psi_{i,j}A_{i,j} &= 2\textstyle\sum_{i,j}\Psi_{i,j}(x_i - x_j)(x_i - x_j)^T \\
&= 2\textstyle\sum_{i,j}\Psi_{i,j}(x_i x_i^T - x_j x_i^T - x_i x_j^T + x_j x_j^T) \\
&= 4\textstyle\sum_{i,j}\Psi_{i,j}(x_i x_i^T - x_j x_i^T) \\
&= \left[4\textstyle\sum_{i,j}\Psi_{i,j}(x_i x_i^T)\right] - \left[4\textstyle\sum_{i,j}\Psi_{i,j}(x_j x_i^T)\right].
\end{aligned}$$

If we expand the 1st term where $i = 1$, we get

$$\sum_{i=1,j}^{n}\Psi_{1,j}(x_1 x_1^T) = \Psi_{1,1}(x_1 x_1^T) + \dots + \Psi_{1,n}(x_1 x_1^T) = \left[\sum_{i=1,j}^{n}\Psi_{1,j}\right]x_1 x_1^T.$$

From here, we notice that $\left[\sum_{i=1,j}^{n} \Psi_{1,j}\right]$ is the degree $d_{i=1}$ of $\Psi_{i=1}$. Therefore, if we sum up all i values we get

$$\sum_{i,j} \Psi_{i,j}(x_i x_i^T) = d_1 x_1 x_1^T + \ldots + d_n x_n x_n^T.$$

If we let D_Ψ be the degree matrix of Ψ, then this expression becomes

$$4\sum_{i,j} \Psi_{i,j}(x_i x_i^T) = 4X^T D_\Psi X.$$

Since Appendix C.7 has already proven the 2nd term, together we get

$$4\sum_{i,j} \Psi_{i,j}(x_i x_i^T) - 4\sum_{i,j} \Psi_{i,j}(x_j x_i^T) = 4X^T D_\Psi X - 4X^T \Psi X = 4X^T[D_\Psi - \Psi]X.$$

C.9 IKDR Experimental Dataset Statistics

Wine. This dataset has 13 features and 178 samples. The features are continuous and heavily unbalanced in magnitude. During the experiments, the dimension is reduced down to 3 prior to performing supervised or unsupervised tasks. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/wine`.

Cancer. This dataset has 9 features and 683 samples. The features are discrete and unbalanced in magnitude. During the experiments, the dimension is reduced down to 2 prior to performing supervised or unsupervised tasks. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`.

Face. This dataset consists of images of 20 people in various poses. The 624 images are vectorized into 960 features. During the experiments, the dimension is reduced down to 20 prior to performing supervised or unsupervised tasks. This dataset is commonly used for alternative clustering since it can be clustered by the identity or the pose of the individuals. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/CMU+Face+Images`.

MNIST. This dataset consists of 10,000 images of 10 characters in various orientations. The images are vectorized into 785 features. During the experiments, the dimension is reduced down to 10 prior to performing supervised or unsupervised tasks. The original MNIST dataset consists of 60,000 training samples and 10,000 test samples. We have decided to use the 10,000 test samples

as our dataset. Since ISM have a memory complexity of $O(n^2)$, storing matrix size of 60,000 $\times$ 60,000 was beyond our computer's capability. We are actively conducting research into using the concept of coresets to alleviate the memory bottleneck. The dataset can be downloaded at `http://yann.lecun.com/exdb/mnist/`

Flower. The Flower image is a dataset that allows for alternative ways to perform image segmentation. It is an image of 350x256 pixel. The RGB values of each pixel is taken as a single sample, with repeated samples removed. This results in a dataset of 256 samples and 3 features. The image is segmented into group of 2, represented by black and white. The dataset can be downloaded at `http://en.tessellations-nicolas.com/`

C.10 Proof of Reformulating Eq. (5.1) into Quadratic Optimization

Given

$$\min_{W} \operatorname{Tr}(W^T \Phi W) \quad \text{s.t.} \quad W^T W = I. \tag{C.101}$$

Here we proof that the local minimum for Eq. (C.101) is equivalent to a local minimum for Eq. (5.1). From Theorem 5.1, we establish that the q minimizing eigenvectors of $\Phi \in \mathcal{R}^{d \times d}$ is a local minimum of Eq. (5.1). Therefore, the strategy of this proof is to show that the optimal solution for Eq. (C.101) is also the minimizing eigenvectors of Φ.

Proof. Given Eq. (C.101), the Lagrangian of the objective is

$$\mathcal{L}(W) = \operatorname{Tr}(W^T \Phi W) - \operatorname{Tr}\left[\Lambda(W^T W - I)\right]. \tag{C.102}$$

Therefore, given a symmetric Φ, the gradient of the Lagrangian becomes

$$\nabla_W \mathcal{L}(W) = 2\Phi W - 2W\Lambda. \tag{C.103}$$

Here, by setting the gradient to 0, we arrive to the definition of eigenvector where

$$\Phi W = W\Lambda, \tag{C.104}$$

thereby proving that the eigenvector of Φ is also a stationary point for Eq. (C.101). The proof ends here if $W \in \mathcal{R}^{d \times d}$, however, if $W \in \mathcal{R}^{d \times q}$ where $q < d$, then we must also determine the

appropriate q eigenvectors to minimize the objective. Given $\bar{W} \in \mathcal{R}^{d \times d}$ as the full set eigenvectors, we replace W from Eq. (C.102) with $\bar{W}$ to get

$$\mathcal{L}(\bar{W}) = \mathrm{Tr}(\bar{W}^T \Phi \bar{W}) - \mathrm{Tr}\left[\Lambda(\bar{W}^T \bar{W} - I)\right]. \tag{C.105}$$

Since $\bar{W}^T \bar{W} = I$ and $\Phi \bar{W} = W\Lambda$, we substitute these terms into Eq. (C.105) to get

$$\mathcal{L}(\bar{W}) = \mathrm{Tr}(\bar{W}^T \bar{W} \Lambda). \tag{C.106}$$

If we let $w_1, w_2, ..., w_d$ be the set of individual eigenvectors of Φ within $\bar{W}$ and $\lambda_1, \lambda_2, ..., \lambda_d$ be their corresponding eigenvalues, then Eq. (C.106) can be rewritten as

$$\mathcal{L}(\bar{W}) = \lambda_1 w_1^T w_1 + \lambda_2 w_2^T w_2 + ... + \lambda_n w_d^T w_d. \tag{C.107}$$

Since the inner product of any eigenvector with itself ($w_i^T w_i$) is always equal to 1, the Lagrangian becomes the summation of its eigenvalues where

$$\mathcal{L}(\bar{W}) = \lambda_1 + \lambda_2 + ... + \lambda_n. \tag{C.108}$$

Therefore, the selection of a subset of eigenvectors is equivalent to keeping a subset of eigenvalues while setting the rest to 0 in Eq. (C.108). To minimize the Lagrangian, therefore, implies that the eigenvectors corresponding to the smallest eigenvalues should be chosen. Here, we have proven that the minimizing eigenvectors of Φ is a local minimum for both Eq. (5.1) and (C.101). □

C.11 Proof for Corollary 1

Proof. The optimization of Eq. (5.1) using a conic combination of m kernels becomes

$$\min_W - \mathrm{Tr}\left(\Gamma[\mu_1 K_1 + \mu_2 K_2 + ... + \mu_m K_m]\right) \quad \text{s.t. } W^T W = I. \tag{C.109}$$

The trace term can be separated into dividual terms where

$$\min_W - \mathrm{Tr}(\mu_1 \Gamma K_1) - \mathrm{Tr}(\mu_2 \Gamma K_2) - ... - \mathrm{Tr}(\mu_m \Gamma K_m) \quad \text{s.t. } W^T W = I. \tag{C.110}$$

Therefore, the Lagrangian can be written as

$$\mathcal{L} = -\mathrm{Tr}(\mu_1 \Gamma K_1) - \mathrm{Tr}(\mu_2 \Gamma K_2) - ... - \mathrm{Tr}(\mu_m \Gamma K_m) - mTr(\Lambda[W^T W - I]). \tag{C.111}$$

From Lemma C.1, we have shown that the gradient of the Lagrangian becomes

$$\nabla_W \mathcal{L} = \left[-\mu_1\Phi_1 - \mu_2\Phi_2 - ... - \mu_m\Phi_m\right] W - mW\Lambda, \tag{C.112}$$

where each Φ_i is the Φ matrix corresponding to each kernel. Setting the gradient to 0, it yields the relationship

$$\frac{1}{m}\left[-\mu_1\Phi_1 - \mu_2\Phi_2 - ... - \mu_m\Phi_m\right] W = W\Lambda, \tag{C.113}$$

Therefore, optimizing a conic combination of kernels for Eq. (5.1) is equivalent to using a conic combination of the corresponding Φs with the same coefficients. □

C.12 Proof for Proposition 1

Proof. For a kernel to belong to the ISM family, it must satisfy the following 3 conditions.

- The kernel function must be twice differentiable.
- The kernel function can be written in terms of $f(\beta)$.
- The kernel matrix from $f(\beta)$ must be symmetric positive semi-definite.

To satisfy the 1st condition, given a kernel K that is a conic combination of n ISM kernels where

$$K = \mu_1 K_1 + \mu_2 K_2 + ... + \mu_n K_n. \tag{C.114}$$

Since each kernel K_i is twice differentiable, the conic combination is still twice differentiable. Therefore, K is a twice differentiable function.

To satisfy the 2nd condition, given a kernel K from Eq. (C.114) where each kernel is from the ISM family, the trivial case of when $\beta = a(x_i, x_j)WW^t b(x_i, x_j)$ is defined identically between kernels:

$$K = \mu_1 f_1(\beta) + \mu_2 f_2(\beta) + ... + \mu_n f_n(\beta). \tag{C.115}$$

From Eq. (C.115), it is obvious that K itself can also be written in terms of β. However, in the cases where the functions $a(x_i, x_j)$ and $b(x_i, x_j)$ are defined differently, β must be defined differently.

Here, we define the following

$$a(x_i, x_j) = \textbf{Diag}([a_1(x_i, x_j), a_2(x_i, x_j), .., a_n(x_i, x_j)]) \tag{C.116}$$

$$b(x_i, x_j) = \textbf{Diag}([b_1(x_i, x_j), b_2(x_i, x_j), .., b_n(x_i, x_j)]) \tag{C.117}$$

$$W = \textbf{Diag}([W, W, ..., W]) \tag{C.118}$$

$$W^T = \textbf{Diag}([W^T, W^T, ..., W^T]) \tag{C.119}$$

$$\beta = \textbf{Diag}([a_1^T W W^T b_1, a_2^T W W^T b_2, ..., a_n^T W W^T b_n]) \tag{C.120}$$

where **Diag** puts the element of the vector on the diagonal of a matrix with both the upper and lower triangle as 0s. Given β is a matrix, each kernel function can always multiply β by a one-hot vector on both sides to choose the appropriate sub-β value. Therefore, the joint kernel K can always be written in terms of β.

For the 3rd condition, we know that conic combinations of symmetric positive semi-definite matrices are still symmetric positive semi-definite. □

C.13 Proof of Theorem 5.2

Proof. The original ISM leverages Bolzano-Weierstrass theorem to prove that a sequence generated using the Gaussian kernel is bounded, therefore ISM has a convergent subsequence. Since the generalized ISM extends the guarantee to other kernels, here we demonstrate that the extension of ISM to other kernels does not have any effect on the convergence guarantee.

ISM over arbitrary kernels solves an optimization problem over the Grassmannian manifold $G(n, d)$, as parametized by the subspace WW^T. The Grassmannian manifold is a quotient of the Stiefel Manifold $G(n, d) = V(n, d)/O(n)$. The Grassmann manifold inherits compactness and an induced metric from the Stiefel manifold. For metric spaces, compact and sequentially compact topological spaces are equivalent. Therefore, sequences $\{WW^T\}^k$ will have convergent subsequences. While W may not converge (choice of frame), its subspace description will. Termination criteria in Algorithm 1 is independent of the frame W. □

C.14 ISM Convergence Criteria

Since the objective is to discover a linear subspace, the rotation of the space does not affect the solution. Therefore, instead of constraining the solution on the Stiefel Manifold, the manifold can be relaxed to a Grassmann Manifold. This implies that Algorithm 4 can reach convergence as long as the columns space spanned by W are identical. To identify the overlapping span of two spaces, we can append the two matrices into $\mathcal{W} = [W_k W_{k+1}]$ and observe the rank of $\mathcal{W}$. In theory, the rank should equal to q, however, a hard threshold on rank often suffers from numerical inaccuracies.

One approach is to study the principal angles ('angles between flats') between the subspaces spanned by W_k and W_{k+1}. This is based on the observation that if the maximal principal angle $\theta_{\text{max}} = 0$, then the two subspaces span the same space. The maximal principal angle between subspaces spanned by W_k and W_{k+1} can be found by computing $U\Sigma V^T = W_k^T W_{k+1}$ [100]. The cosines of the principal angles between W_k and W_{k+1} are the singular values of Σ, thus $\theta_{\text{max}} = \cos^{-1}(\sigma_{\text{min}})$. Computation of θ_{max} requires two matrix multiplications to form $V\Sigma^2 V^T = (W_k^T W_{k+1})^T (W_k^T W_{k+1})$ and then a round of inverse iteration to find σ_{min}^2. Although this approach confirms the convergence definitively, in practice, we avoid this extra computation by using the convergence of eigenvalues (of Φ) between iterations as a surrogate. Since eigenvalues are already computed during the algorithm, no additional computations are required. Although tracking eigenvalue of Φ for convergence is vulnerable to false positive errors, in practice, it works consistently well. Therefore, we recommend to use the eigenvalues as a preliminary check before defaulting to principal angles.

Appendix D

This Appendix focuses on providing further detail on the Kernel Dependence Network. As the main theoretical contribution, we have proven that our network is guaranteed to generate a monotonic sequence of dependence improvements that approaches the global optimum. We have shown how this sequence leads to an optimum classification result, and why this result generalizes. This appendix provides to proof to these claims along with additional experimental results.

D.1 Proof for Theorem 6.1

Theorem 6.1: *For any $\mathcal{H}_0$, there exists a Kernel Sequence $\{\phi_{l^\circ}\}_{l=1}^{L}$ parameterized by a set of weights W_l and a set of bandwidths σ_l such that:*

I. $\mathcal{H}_L$ can approach arbitrarily close to $\mathcal{H}^*$ such that for any $L > 1$ and $\delta > 0$ we can achieve

$$\mathcal{H}^* - \mathcal{H}_L \leq \delta, \tag{D.1}$$

II. as $L \to \infty$, the $\mathcal{H}$-*Sequence* converges to the global optimum, that is

$$\lim_{L\to\infty} \mathcal{H}_L = \mathcal{H}^*, \tag{D.2}$$

III. the convergence is monotonic where

$$\mathcal{H}_l > \mathcal{H}_{l-1} \quad \forall l. \tag{D.3}$$

Lemma D.1. *Given σ_0 and σ_1 as the σ values from the last layer and the current layer, then there exists a lower bound for $\mathcal{H}_l$, denoted as $\mathscr{L}(\sigma_0, \sigma_1)$ such that*

$$\mathcal{H}_l \geq \mathscr{L}(\sigma_0, \sigma_1). \tag{D.4}$$

Basic Background, Assumptions, and Notations.

1. The simulation of this theorem for Adversarial and Random data is also publicly available on `https://github.com/anonymous`.
2. Here we show that this bound can be established given the last 2 layers.
3. σ_0 is the σ value of the previous layer
4. σ_1 is the σ value of the current layer
5. τ is the number of classes
6. n is total number of samples
7. n_i is number of samples in the i^{th} class
8. $\mathcal{S}$ is a set of all i, j sample pairs where r_i and r_j belong to the same class.
9. $\mathcal{S}^c$ is a set of all i, j sample pairs where r_i and r_j belong to different same classes.
10. $\mathcal{S}^\beta$ is a set of all i, j sample pairs that belongs to the same β^{th} classes.
11. $r_i^{(\alpha)}$ is the i^{th} sample in the α^{th} class among τ classes.
12. We assume no $r_i \neq r_j$ pair are equal $\forall i \neq j$.
13. Among all $r_i \neq r_j$ pairs, there exists an optimal r_i^*, r_j^* pair where $\langle r_i^*, r_j^* \rangle \geq \langle r_i, r_j \rangle \; \forall r_i \neq r_i^*$ and $r_j \neq r_j^*$. We denote this maximum inner product as

$$u_{\sigma_0} = \langle r_i^*, r_j^* \rangle. \tag{D.5}$$

14. In *KNet*, each r_i sample is assumed to be a sample in the RKHS of the Gaussian kernel, therefore all inner products are bounded such that

$$0 \leq \langle r_i, r_j \rangle \leq u_{\sigma_0}. \tag{D.6}$$

15. We let W be

$$W_s = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & \dots & \sum_\iota r_\iota^{(\tau)} \end{bmatrix}. \tag{D.7}$$

Instead of using an optimal W^* defined as $W^* = \arg\max_W H_l(W)$, we use a suboptimal W_s where each dimension is simply the average direction of each class: $\frac{1}{\sqrt{\zeta}}$ is a normalizing constant $\zeta = ||W_s||_2^2$ that ensures $W_s^T W_s = I$. By using W_s, this implies that the $\mathcal{H}$ we obtain is already a lower bound compare $\mathcal{H}$ obtained by W^*. But, we will use this suboptimal W_s to identify an even lower bound. Note that based on the definition W^*, we have the property $\mathcal{H}(W^*) \geq \mathcal{H}(W) \, \forall W$.

16. We note that the objective $\mathcal{H}$ is

$$\mathcal{H} = \underbrace{\sum_{i,j \in \mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}}}_{\mathscr{W}} - \underbrace{\sum_{i,j \in \mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}}}_{\mathscr{B}} \tag{D.8}$$

where we let $\mathscr{W}$ be the summation of terms associated with the within cluster pairs, and let $\mathscr{B}$ be the summation of terms associated with the between cluster pairs.

Proof.

The equation is divided into smaller parts organized into multiple sections.

For sample pairs in $\mathcal{S}$. The first portion of the function can be split into multiple classes where

$$\mathscr{W} = \underbrace{\sum_{\mathcal{S}^1} \Gamma_{i,j} e^{-\frac{(r_i^{(1)} - r_j^{(1)})^T W W^T (r_i^{(1)} - r_j^{(1)})}{2\sigma_1^2}}}_{\mathscr{W}_1} + \dots + \underbrace{\sum_{\mathcal{S}^\tau} \Gamma_{i,j} e^{-\frac{(r_i^{(\tau)} - r_j^{(\tau)})^T W W^T (r_i^{(\tau)} - r_j^{(\tau)})}{2\sigma_1^2}}}_{\mathscr{W}_\tau} \tag{D.9}$$

Realize that to find the lower bound, we need to determine the minimum possible value of each term which translates to **maximum** possible value of each exponent. Without of loss of generality we can find the lower bound for one term and generalize its results to other terms due to their similarity. Let us focus on the numerator of the exponent from $\mathscr{W}_1$. Given W_s as W, our goal is identify the **maximum** possible value for

$$\underbrace{(r_i^{(1)} - r_j^{(1)})^T W}_{\Pi_1} \underbrace{W^T (r_i^{(1)} - r_j^{(1)})}_{\Pi_2}. \tag{D.10}$$

Zoom in further by looking only at Π_1, we have the following relationships

$$\Pi_1 = \underbrace{r_i^{(1)^T} W}_{\xi_1} - \underbrace{r_j^{(1)^T} W}_{\xi_2} \tag{D.11}$$

$$\xi_1 = \frac{1}{\sqrt{\zeta}} r_i^{(1)^T} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)} \end{bmatrix} \tag{D.12}$$

$$= \frac{1}{\sqrt{\zeta}} r_i^{(1)^T} \begin{bmatrix} (r_1^{(1)} + ... + r_{n_1}^{(1)}) & ... & (r_1^{(\tau)} + ... + r_{n_\tau}^{(\tau)}) \end{bmatrix} \tag{D.13}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} r_j^{(1)^T} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)} \end{bmatrix} \tag{D.14}$$

$$= \frac{1}{\sqrt{\zeta}} r_j^{(1)^T} \begin{bmatrix} (r_1^{(1)} + ... + r_{n_1}^{(1)}) & ... & (r_1^{(\tau)} + ... + r_{n_\tau}^{(\tau)}) \end{bmatrix} \tag{D.15}$$

By knowing that the inner product is constrained between $[0, u_{\sigma_0}]$, we know the maximum possible value for ξ_1 and the minimum possible value for ξ_2 to be

$$\xi_1 = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} 1 + (n_1 - 1)u_{\sigma_0} & n_2 u_{\sigma_0} & n_3 u_{\sigma_0} & ... & n_\tau u_{\sigma_0} \end{bmatrix} \tag{D.16}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} 1 & 0 & 0 & ... & 0 \end{bmatrix}. \tag{D.17}$$

Which leads to

$$\Pi_1 = \frac{1}{\sqrt{\zeta}}(\xi_1 - \xi_2) = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} (n_1 - 1)u_{\sigma_0} & n_2 u_{\sigma_0} & n_3 u_{\sigma_0} & ... & n_\tau u_{\sigma_0} \end{bmatrix} \tag{D.18}$$

Since $\Pi_2^T = \Pi_1$ we have

$$\Pi_1 \Pi_2 = \frac{1}{\zeta}[(n_1 - 1)^2 u_{\sigma_0}^2 + n_2^2 u_{\sigma_0}^2 + n_3^2 u_{\sigma_0}^2 + ... + n_\tau^2 u_{\sigma_0}^2] \tag{D.19}$$

$$= \frac{1}{\zeta}[(n_1 - 1)^2 + n_2^2 + n_3^2 + ... + n_\tau^2] u_{\sigma_0}^2 \tag{D.20}$$

The lower bound for just the $\mathscr{W}_1$ term emerges as

$$\mathscr{W}_1 \geq \sum_{\mathcal{S}^1} \Gamma_{i,j} e^{-\frac{[(n_1-1)^2 + n_2^2 + n_3^2 + ... + n_\tau^2] u_{\sigma_0}^2}{2\zeta\sigma_1^2}}. \tag{D.21}$$

To further condense the notation, we define the following constant

$$\mathscr{N}_g = \frac{1}{2\zeta}[n_1^2 + n_2^2 + ... + (n_g - 1)^2 + ... + n_\tau^2]. \tag{D.22}$$

Therefore, the lower bound for $\mathscr{W}_1$ can be simplified as

$$\mathscr{W}_1 \geq \sum_{\mathcal{S}^1} \Gamma_{i,j} e^{-\frac{\mathscr{N}_1 u_{\sigma_0}^2}{\sigma_1^2}} \tag{D.23}$$

and the general pattern for any $\mathscr{W}_g$ becomes

$$\mathscr{W}_g \geq \sum_{\mathcal{S}^i} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}}. \tag{D.24}$$

The lower bound for the entire set of $\mathcal{S}$ then becomes

$$\sum_{i,j \in \mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}} = \mathscr{W}_1 + ... + \mathscr{W}_\tau \geq \underbrace{\sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}}}_{\text{Lower bound}}. \tag{D.25}$$

For sample pairs in $\mathcal{S}^c$. To simplify the notation, we note that

$$-\mathscr{B}_{g_1,g_2} = -\sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{(r_i^{(g_1)} - r_j^{(g_2)})^T W W^T (r_i^{(g_1)} - r_j^{(g_2)})}{2\sigma_1^2}} \tag{D.26}$$

$$= -\sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathrm{Tr}(W^T ((r_i^{(g_1)} - r_j^{(g_1)}))((r_i^{(g_1)} - r_j^{(g_2)}))^T W)}{2\sigma_1^2}} \tag{D.27}$$

$$= -\sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(g_1,g_2)} W)}{2\sigma_1^2}} \tag{D.28}$$

$$\tag{D.29}$$

We now derived the lower bound for the sample pairs in $\mathcal{S}^c$. We start by writing out the entire

summation sequence for $\mathscr{B}$.

$$\begin{aligned}\mathscr{B} = &\underbrace{-\sum_{i\in\mathcal{S}^1}\sum_{j\in\mathcal{S}^2}|\Gamma_{i,j}|e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(1,2)}W)}{2\sigma_1^2}}}_{\mathscr{B}_{1,2}} - \underbrace{...}_{\mathscr{B}_{g_1\neq g_2}} \underbrace{-\sum_{i\in\mathcal{S}^1}\sum_{j\in\mathcal{S}^\tau}|\Gamma_{i,j}|e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(1,\tau)}W)}{2\sigma_1^2}}}_{\mathscr{B}_{1,\tau}} \\ &\underbrace{-\sum_{i\in\mathcal{S}^2}\sum_{j\in\mathcal{S}^1}|\Gamma_{i,j}|e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(2,1)}W)}{2\sigma_1^2}}}_{\mathscr{B}_{2,1}} - \underbrace{...}_{\mathscr{B}_{g_1\neq g_2}} \underbrace{-\sum_{i\in\mathcal{S}^2}\sum_{j\in\mathcal{S}^\tau}|\Gamma_{i,j}|e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(2,\tau)}W)}{2\sigma_1^2}}}_{\mathscr{B}_{2,\tau}} \\ &... \\ &\underbrace{-\sum_{i\in\mathcal{S}^\tau}\sum_{j\in\mathcal{S}^1}|\Gamma_{i,j}|e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(\tau,1)}W)}{2\sigma_1^2}}}_{\mathscr{B}_{\tau,1}} - \underbrace{...}_{\mathscr{B}_{g_1\neq g_2}} \underbrace{-\sum_{i\in\mathcal{S}^{\tau-1}}\sum_{j\in\mathcal{S}^\tau}|\Gamma_{i,j}|e^{-\frac{\mathrm{Tr}(W^T A_{i,j}^{(\tau-1,\tau)}W)}{2\sigma_1^2}}}_{\mathscr{B}_{\tau-1,\tau}}\end{aligned} \tag{D.30}$$

Using a similar approach with the terms from $\mathscr{W}$, note that $\mathscr{B}$ is a negative value, so we need to maximize this term to obtain a lower bound. Consequently, the key is to determine the **minimal** possible values for each exponent term. Since every one of them will behave very similarly, we can simply look at the numerator of the exponent from $\mathscr{B}_{1,2}$ and then arrive to a more general conclusion. Given W_s as W, our goal is to identify the **minimal** possible value for

$$\underbrace{(r_i^{(1)} - r_j^{(2)})^T W}_{\Pi_1}\underbrace{W^T(r_i^{(1)} - r_j^{(2)})}_{\Pi_2}. \tag{D.31}$$

Zoom in further by looking only at Π_1, we have the following relationships

$$\Pi_1 = \underbrace{r_i^{(1)^T} W}_{\xi_1} - \underbrace{r_j^{(2)^T} W}_{\xi_2} \tag{D.32}$$

$$\xi_1 = \frac{1}{\sqrt{\zeta}} r_i^{(1)^T}\begin{bmatrix}\sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)}\end{bmatrix} \tag{D.33}$$

$$= \frac{1}{\sqrt{\zeta}} r_i^{(1)^T}\begin{bmatrix}(r_1^{(1)} + ... + r_{n_1}^{(1)}) & ... & (r_1^{(\tau)} + ... + r_{n_\tau}^{(\tau)})\end{bmatrix} \tag{D.34}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} r_j^{(2)^T}\begin{bmatrix}\sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)}\end{bmatrix} \tag{D.35}$$

$$= \frac{1}{\sqrt{\zeta}} r_j^{(2)^T}\begin{bmatrix}(r_1^{(1)} + ... + r_{n_1}^{(1)}) & ... & (r_1^{(\tau)} + ... + r_{n_\tau}^{(\tau)})\end{bmatrix} \tag{D.36}$$

By knowing that the inner product is constrained between $[0, u_{\sigma_0}]$, we know the **minimum** possible value for ξ_1 and the **maximum** possible value for ξ_2 to be

$$\xi_1 = \frac{1}{\sqrt{\zeta}}\begin{bmatrix}1 & 0 & 0 & ... & 0\end{bmatrix} \tag{D.37}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}}\begin{bmatrix}n_1 u_{\sigma_0} & 1+(n_2-1)u_{\sigma_0} & n_3 u_{\sigma_0} & ... & n_\tau u_{\sigma_0}\end{bmatrix} \tag{D.38}$$

Which leads to

$$\Pi_1 = \frac{1}{\sqrt{\zeta}}(\xi_1 - \xi_2) = \frac{1}{\sqrt{\zeta}}\begin{bmatrix}1-n_1 u_{\sigma_0} & -(1+(n_2-1)u_{\sigma_0}) & -n_3 u_{\sigma_0} & ... & -n_\tau u_{\sigma_0}\end{bmatrix} \tag{D.39}$$

Since $\Pi_2^T = \Pi_1$ we have

$$\Pi_1\Pi_2 = \frac{1}{\zeta}[(1-n_1 u_{\sigma_0})^2 + (1+(n_2-1)u_{\sigma_0})^2 + n_3^2 u_{\sigma_0}^2 + ... + n_\tau^2 u_{\sigma_0}^2]. \tag{D.40}$$

The lower bound for just the $\mathscr{B}_{1,2}$ term emerges as

$$-\mathscr{B}_{1,2} \geq -\sum_{\mathcal{S}^1}\sum_{\mathcal{S}^2}|\Gamma_{i,j}|e^{-\frac{(1-n_1 u_{\sigma_0})^2+(1+(n_2-1)u_{\sigma_0})^2+n_3^2u_{\sigma_0}^2+...+n_\tau^2 u_{\sigma_0}^2}{2\zeta\sigma_1^2}}. \tag{D.41}$$

To further condense the notation, we define the following function

$$\begin{aligned}\mathscr{N}_{g_1,g_2}(u_{\sigma_0}) = \frac{1}{2\zeta}[&n_1^2u_{\sigma_0}^2 + n_2^2u_{\sigma_0}^2 + ... \\ &+ (1-n_{g_1}u_{\sigma_0})^2 + ... + (1+(n_{g_2}-1)u_{\sigma_0})^2 \\ &+ ... + n_\tau^2 u_{\sigma_0}^2].\end{aligned} \tag{D.42}$$

Note that while for $\mathcal{S}$, the u_{σ_0} term can be separated out. But here, we cannot, and therefore $\mathscr{N}$ here must be a function of u_{σ_0}. Therefore, the lower bound for $\mathscr{B}_{1,2}$ can be simplified into

$$-\mathscr{B}_{1,2} \geq -\sum_{\mathcal{S}^1}\sum_{\mathcal{S}^2}|\Gamma_{i,j}|e^{-\frac{\mathscr{N}_{1,2}(u_{\sigma_0})}{\sigma_1^2}} \tag{D.43}$$

and the general pattern for any $\mathscr{B}_{g_1,g_2}$ becomes

$$-\mathscr{B}_{g_1,g_2} \geq -\sum_{\mathcal{S}^{g1}}\sum_{\mathcal{S}^{g2}}\Gamma_{i,j}e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}. \tag{D.44}$$

The lower bound for the entire set of $\mathcal{S}^c$ then becomes

$$-\sum_{i,j\in\mathcal{S}^c}|\Gamma_{i,j}|e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma_1^2}} = -\mathscr{B}_{1,2} - \mathscr{B}_{1,3} - ... - \mathscr{B}_{\tau-1,\tau} \tag{D.45}$$

$$\geq \underbrace{-\sum_{g_1\neq g_2}^{\tau}\sum_{i\in\mathcal{S}^{g_1}}\sum_{j\in\mathcal{S}^{g_2}}|\Gamma_{i,j}|e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}}_{\text{Lower bound}}. \tag{D.46}$$

Putting $\mathcal{S}$ and $\mathcal{S}^c$ Together.

$$\mathcal{H} = \mathscr{W} + \mathscr{B} \tag{D.47}$$

$$\geq \underbrace{\sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}}}_{\text{Lower bound of } \mathscr{W}} - \underbrace{\sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}}_{\text{Lower bound of } \mathscr{B}}. \tag{D.48}$$

Therefore, we have identified a lower bound that is a function of σ_0 and σ_1 where

$$\mathscr{L}(\sigma_0, \sigma_1) = \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}. \tag{D.49}$$

From the lower bound, it is obvious why it is a function of σ_1. The lower bound is also a function of σ_0 because u_{σ_0} is actually a function of σ_0. To specifically clarify this point, we have the next lemma.

□

Lemma D.2. *The u_{σ_0} used in Lemma D.1 is a function of σ_0 where u_{σ_0} approaches to zero as σ_0 approaches to zero, i.e.*

$$\lim_{\sigma_0 \to 0} u_{\sigma_0} = 0. \tag{D.50}$$

Assumptions and Notations.

1. We use Fig. D.1 to help clarify the notations. We here only look at the last 2 layers.
2. We let $\mathcal{H}_0$ be the $\mathcal{H}$ of the last layer, and $\mathcal{H}_1$, the $\mathcal{H}$ of the current layer.
3. The input of the data is X with each sample as x_i, and the output of the previous layer are denoted as r_i. ψ_{σ_0} is the feature map of the previous layer using σ_0 and ψ_{σ_1} corresponds to the current layer.
4. As defined from Lemma D.1, among all $r_i \neq r_j$ pairs, there exists an optimal r_i^*, r_j^* pair where $\langle r_i^*, r_j^* \rangle \geq \langle r_i, r_j \rangle \; \forall r_i \neq r_i^*$ and $r_j \neq r_j^*$. We denote this maximum inner product as

$$u_{\sigma_0} = \langle r_i^*, r_j^* \rangle. \tag{D.51}$$

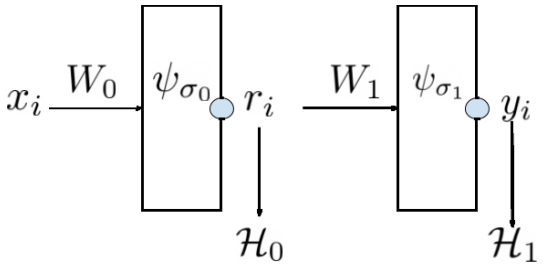

Figure D.1: Figure of a 2 layer network.

Proof.

Given Fig. D.1, the equation for $\mathcal{H}_0$ is

$$\mathcal{H}_0 = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} e^{-\frac{(x_i-x_j)^T WW^T(x_i-x_j)}{2\sigma_0^2}} - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(x_i-x_j)^T WW^T(x_i-x_j)}{2\sigma_0^2}} \tag{D.52}$$

$$= \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} \langle \psi_{\sigma_0}(x_i), \psi_{\sigma_0}(x_j)\rangle - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}| \langle \psi_{\sigma_0}(x_i), \psi_{\sigma_0}(x_j)\rangle \tag{D.53}$$

Notice that as $\sigma_0 \to 0$, we have

$$\lim_{\sigma_0\to 0} \langle \psi_{\sigma_0}(x_i), \psi_{\sigma_0}(x_j)\rangle = \begin{cases} 0 & \forall i \neq j \\ 1 & \forall i = j \end{cases} . \tag{D.54}$$

In other words, as $\sigma_0 \to 0$, the samples r_i in the RKHS of a Gaussian kernel approaches orthogonal to all other samples. Given this fact, it also implies that the σ_0 controls the inner product magnitude in RKHS space of the maximum sample pair r_i^*, r_j^*. We define this maximum inner product as

$$\langle \psi_{\sigma_0}(x_i^*), \psi_{\sigma_0}(x_j^*)\rangle \geq \langle \psi_{\sigma_0}(x_i), \psi_{\sigma_0}(x_j)\rangle \tag{D.55}$$

or equivalently

$$\langle r_i^*, r_j^*\rangle \geq \langle r_i, r_j\rangle \tag{D.56}$$

Therefore, given a σ_0, it controls the upper bound of the inner product. Notice that as $\sigma_0 \to 0$, every sample in RKHS becomes orthogonal. Therefore, the upper bound of $\langle r_i, r_j\rangle$ also approaches 0 when $r_i \neq r_j$. From this, we see the relationship

$$\lim_{\sigma_0\to 0} u_{\sigma_0} = \lim_{\sigma_0\to 0} \exp -(|.|/\sigma_0^2) = 0 \tag{D.57}$$

, where —.— is bounded and has a minimum and maximum, because we have finite number of samples.

□

Lemma D.3. *Given any fixed $\sigma_1 > 0$, the lower bound $\mathscr{L}(\sigma_0, \sigma_1)$ is a function with respect to σ_0 and as $\sigma_0 \to 0$, $\mathscr{L}(\sigma_0, \sigma_1)$ approaches the function*

$$\mathscr{L}(\sigma_1) = \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{1}{\zeta \sigma_1^2}}. \tag{D.58}$$

At this point, if we let $\sigma_1 \to 0$, we have

$$\lim_{\sigma_1 \to 0} \mathscr{L}(\sigma_1) = \sum_{i,j \in \mathcal{S}} \Gamma_{i,j} \tag{D.59}$$

$$= \mathcal{H}^*. \tag{D.60}$$

Proof.

Given Lemma D.2, we know that

$$\lim_{\sigma_0 \to 0} u_{\sigma_0} = 0. \tag{D.61}$$

Therefore, having $\sigma_0 \to 0$ is equivalent to having $u_{\sigma_0} \to 0$. Since Lemma D.1 provide the equation of a lower bound that is a function of u_{σ_0}, this lemma is proven by simply evaluating $\mathscr{L}(\sigma_0, \sigma_1)$ as $u_{\sigma_0} \to 0$. Following these steps, we have

$$\mathscr{L}(\sigma_1) = \lim_{u_{\sigma_0} \to 0} \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} e^{-\frac{\mathcal{N}_g u_{\sigma_0}^2}{\sigma_1^2}} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathcal{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}, \tag{D.62}$$

$$= \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{1}{\zeta \sigma_1^2}}. \tag{D.63}$$

At this point, as $\sigma_1 \to 0$, our lower bound reaches the global maximum

$$\lim_{\sigma_1 \to 0} \mathscr{L}(\sigma_1) = \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} = \sum_{i,j \in \mathcal{S}} \Gamma_{i,j} \tag{D.64}$$

$$= \mathcal{H}^*. \tag{D.65}$$

□

Lemma D.4. *Given any $\mathcal{H}_{l-2}$, $\delta > 0$, there exists a $\sigma_0 > 0$ and $\sigma_1 > 0$ such that*

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta. \tag{D.66}$$

Proof.

Observation 1.

Note that the objective of $\mathcal{H}_l$ is

$$\begin{aligned}\mathcal{H}_l = \max_{W} \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i^{(\mathcal{S})}-r_j^{(\mathcal{S})})^T WW^T (r_i^{(\mathcal{S})}-r_j^{(\mathcal{S})})}{2\sigma_1^2}} \\ - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i^{(\mathcal{S}^c)}-r_j^{(\mathcal{S}^c)})^T WW^T (r_i^{(\mathcal{S}^c)}-r_j^{(\mathcal{S}^c)})}{2\sigma_1^2}}.\end{aligned} \tag{D.67}$$

Since the Gaussian kernel is bounded between 0 and 1, the theoretical maximum of $\mathcal{H}^*$ is when the kernel is 1 for $\mathcal{S}$ and 0 for $\mathcal{S}^c$ with the theoretical maximum as $\mathcal{H}^* = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}$. Therefore Eq. (D.66) inequality is equivalent to

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j} - \mathcal{H}_l \leq \delta. \tag{D.68}$$

Observation 2.

If we choose a σ_0 such that

$$\mathscr{L}^*(\sigma_1) - \mathscr{L}(\sigma_0, \sigma_1) \leq \frac{\delta}{2} \quad \text{and} \quad \mathcal{H}^* - \mathscr{L}^*(\sigma_1) \leq \frac{\delta}{2} \tag{D.69}$$

then we have identified the condition where $\sigma_0 > 0$ and $\sigma_1 > 0$ such that

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j} - \mathscr{L}(\sigma_0, \sigma_1) \leq \delta. \tag{D.70}$$

Note that the $\mathscr{L}^*(\sigma_1)$ is a continuous function of σ_1. Therefore, a σ_1 exists such that $\mathscr{L}^*(\sigma_1)$ can be set arbitraty close to $\mathcal{H}^*$. Hence, we choose an σ_1 that has the following property:

$$\mathcal{H}^* - \mathscr{L}^*(\sigma_1) \leq \frac{\delta}{2}. \tag{D.71}$$

We next fix σ_1, we also know $\mathscr{L}(\sigma_0, \sigma_1)$ is a continuous function of σ_0, and it has a limit $\mathscr{L}^*(\sigma_1)$ as σ_0 approaches to 0, hence there exits a σ_0, where

$$\mathscr{L}^*(\sigma_1) - \mathscr{L}(\sigma_0, \sigma_1) \leq \frac{\delta}{2} \tag{D.72}$$

Then we have:

$$\mathscr{L}^*(\sigma_1) - \mathscr{L}(\sigma_0, \sigma_1) \leq \frac{\delta}{2} \quad \text{and} \quad \mathcal{H}^* - \mathscr{L}^*(\sigma_1) \leq \frac{\delta}{2}. \tag{D.73}$$

By adding the two $\frac{\delta}{2}$, we conclude the proof.

□

Lemma D.5. *There exists a Kernel Sequence $\{\phi_{l^\circ}\}_{l=1}^{L}$ parameterized by a set of weights W_l and a set of bandwidths σ_l such that*

$$\lim_{l\to\infty} \mathcal{H}_l = \mathcal{H}^*, \quad \mathcal{H}_{l+1} > \mathcal{H}_l \quad \forall l \tag{D.74}$$

Before, the proof, we use the following figure, Fig. D.2, to illustrate the relationship between *Kernel Sequence* $\{\phi_{l^\circ}\}_{l=1}^{L}$ that generates the $\mathcal{H}$*-Sequence* $\{\mathcal{H}_l\}_{l=1}^{L}$. By solving a network greedily, we separate the network into L separable problems. At each additional layer, we rely on the weights learned from the previous layer. At each network, we find σ_{l-1}, σ_l, and W_l for the next network. We also note that since we only need to prove the existence of a solution, this proof is done by ***Proof by Construction***, i.e, we only need to show an example of its existence. Therefore, this proof consists of us constructing a $\mathcal{H}$*-Sequence* which satisfies the lemma.

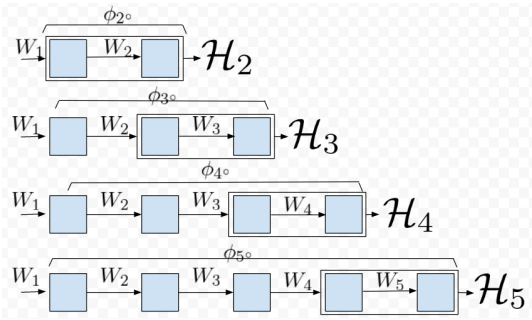

Figure D.2: Relating *Kernel Sequence* to $\mathcal{H}$*-Sequence*.

Proof.

We first note that from Lemma D.4, we have previously proven given any $\mathcal{H}_{l-2}$, $\delta > 0$, there exists a $\sigma_0 > 0$ and $\sigma_1 > 0$ such that

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l. \tag{D.75}$$

This implies that based on Fig. D.2, at any given layer, we could reach arbitrarily close to $\mathcal{H}^*$. Given this, we list the 2 steps to build the $\mathcal{H}$-*Sequence*.

Step 1: Define $\{\mathcal{E}_n\}_{n=1}^{\infty}$ as a sequence of numbers $\mathcal{H}^* - \frac{\mathcal{H}^* - \mathcal{H}_0}{n}$ on the real line. We have the following properties for this sequence:

$$\lim_{n\to\infty} \mathcal{E}_n = \mathcal{H}^*, \quad \mathcal{E}_1 = \mathcal{H}_0. \tag{D.76}$$

Using these two properties, for any $\mathcal{H}_{l-1} \in [\mathcal{H}_0, \mathcal{H}^*]$ there exist an unique n, where

$$\mathcal{E}_n \leq \mathcal{H}_{l-1} < \mathcal{E}_{n+1}. \tag{D.77}$$

Step 2: For any given l, we choose δ_l to satisfies Eq. (D.75) by the following procedure, First find an n that satisfies

$$\mathcal{E}_n \leq \mathcal{H}_{l-1} < \mathcal{E}_{n+1}, \tag{D.78}$$

and second define δ_l to be

$$\delta_l = \mathcal{H}^* - \mathcal{E}_{n+1}. \tag{D.79}$$

To satisfy Eq. (D.75), the following must be true.

$$\mathcal{H}^* - \mathcal{H}_{l-1} \leq \delta_{l-1}. \tag{D.80}$$

and further we found n such that

$$\mathcal{E}_n \leq \mathcal{H}_{l-1} < \mathcal{E}_{n+1} \implies \mathcal{H}^* - \mathcal{E}_n \geq \mathcal{H}^* - \mathcal{H}_{l-1} > \mathcal{H}^* - \mathcal{E}_{n+1}. \tag{D.81}$$

Thus combining Eq. (D.79), Eq. (D.80), and Eq. (D.81) we have

$$\delta_{l-1} > \delta_l. \tag{D.82}$$

Therefore, $\{\delta_l\}$ is a decreasing sequence.

Step 3: Note that $\{\mathcal{E}_n\}$ is a converging sequence where

$$\lim_{n\to\infty} \mathcal{H}^* - \frac{\mathcal{H}^* - \mathcal{H}_0}{n} = \mathcal{H}^*. \tag{D.83}$$

Therefore, $\{\Delta_n\} = \mathcal{H}^* - \{\mathcal{E}_n\}$ is also a converging sequence where

$$\lim_{n\to\infty} \mathcal{H}^* - \mathcal{H}^* + \frac{\mathcal{H}^* - \mathcal{H}_0}{n} = 0 \tag{D.84}$$

and $\{\delta_l\}$ is a subsequence of $\{\Delta_l\}$. Since any subsequence of a converging sequence also converges to the same limit, we know that

$$\lim_{l\to\infty} \delta_l = 0. \tag{D.85}$$

Following this construction, if we always choose $\mathcal{H}_l$ such that

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l. \tag{D.86}$$

As $l \to \infty$, the inequality becomes

$$\mathcal{H}^* - \lim_{l\to\infty} \mathcal{H}_l \leq \lim_{l\to\infty} \delta_l, \tag{D.87}$$

$$\leq 0. \tag{D.88}$$

Since we know that

$$\mathcal{H}^* - \mathcal{H}_l \geq 0 \, \forall l. \tag{D.89}$$

The condition of

$$0 \leq \mathcal{H}^* - \lim_{l\to\infty} \mathcal{H}_l \leq 0 \tag{D.90}$$

is true only if

$$\mathcal{H}^* - \lim_{l\to\infty} \mathcal{H}_l = 0. \tag{D.91}$$

This allows us to conclude

$$\mathcal{H}^* = \lim_{l\to\infty} \mathcal{H}_l. \tag{D.92}$$

Proof of the Monotonic Improvement.

Given Eq. (D.77) and Eq. (D.79), at each step we have the following:

$$\mathcal{H}_{l-1} < \mathcal{E}_{n+1} \tag{D.93}$$

$$\leq \mathcal{H}^* - \delta_l. \tag{D.94}$$

Rearranging this inequality, we have

$$\delta_l < \mathcal{H}^* - \mathcal{H}_{l-1}. \tag{D.95}$$

By combining the inequalities from Eq. (D.95) and Eq. (D.86), we have the following relationships.

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l < \mathcal{H}^* - \mathcal{H}_{l-1} \tag{D.96}$$

$$\mathcal{H}^* - \mathcal{H}_l < \mathcal{H}^* - \mathcal{H}_{l-1} \tag{D.97}$$

$$-\mathcal{H}_l < -\mathcal{H}_{l-1} \tag{D.98}$$

$$\mathcal{H}_l > \mathcal{H}_{l-1}, \tag{D.99}$$

$$\tag{D.100}$$

which concludes the proof of theorem.

□

Lemma D.6. *Given $\frac{1}{\sqrt{\zeta}}$ as a normalizing constant for $W_s = \frac{1}{\sqrt{\zeta}} \sum_\alpha r_\alpha$ such that $W^T W = I$, then W_s is not guaranteed to be the optimal solution for the HSIC objective.*

Proof. We start with the Lagrangian

$$\mathcal{L} = -\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} - \operatorname{Tr}(\Lambda(W^T W - I)). \tag{D.101}$$

If we now take the derivative with respect to the Lagrange, we get

$$\nabla \mathcal{L} = \frac{1}{\sigma^2} \sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i - r_j)(r_i - r_j)^T W - 2W\Lambda. \tag{D.102}$$

By setting the gradient to 0, we have

$$\left[\frac{1}{2\sigma^2} \sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i - r_j)(r_i - r_j)^T \right] W = W\Lambda. \tag{D.103}$$

$$\mathcal{Q}_l W = W\Lambda. \tag{D.104}$$

From Eq. (D.104), we see that W is only the optimal solution when W is the eigenvector of Q_l. Therefore, by setting W to $W_s = \frac{1}{\sqrt{\zeta}} \sum_\alpha r_\alpha$, it is not guaranteed to yield an optimal.

□

D.2 Proof for Theorem 6.2

Theorem 6.2: *As $l \to \infty$ and $\mathcal{H}_l \to \mathcal{H}^*$, the following properties are satisfied:*

I the scatter ratio approaches 0 where

$$\lim_{l\to\infty} \frac{\mathrm{Tr}(S_w^l)}{\mathrm{Tr}(S_b^l)} = 0 \tag{D.105}$$

II the *Kernel Sequence* converges to the following kernel:

$$\lim_{l\to\infty} \mathcal{K}(x_i, x_j)^l = \mathcal{K}^* = \begin{cases} 0 & \forall i,j \in \mathcal{S}^c \\ 1 & \forall i,j \in \mathcal{S} \end{cases} . \tag{D.106}$$

Proof. We start by proving condition II starting from the $\mathcal{H}$ objective using a GK

$$\max_W \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}\mathcal{K}_W(r_i, r_j) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_W(r_i, r_j) \tag{D.107}$$

$$\max_W \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} \tag{D.108}$$

Given that $\mathcal{H}_l \to \mathcal{H}^*$, and the fact that $0 \leq \mathcal{K}_W \leq 1$, this implies that the following condition must be true:

$$\mathcal{H}^* = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}(1) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|(0). \tag{D.109}$$

Based on Eq. (D.75), our construction at each layer ensures to satisfy

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l. \tag{D.110}$$

Substituting the definition of $\mathcal{H}^*$ and $\mathcal{H}_l$, we have

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}(1) - \left[\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}\mathcal{K}_W(r_i, r_j) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_W(r_i, r_j)\right] \leq \delta_l \tag{D.111}$$

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}(1 - \mathcal{K}_W(r_i, r_j)) + \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_W(r_i, r_j) \leq \delta_l. \tag{D.112}$$

Since every term within the summation in Eq. (D.112) is positive, this implies

$$1 - \mathcal{K}_W(r_i, r_j) \leq \delta_l \quad i,j \in \mathcal{S} \tag{D.113}$$

$$\mathcal{K}_W(r_i, r_j) \leq \delta_l \quad i,j \in \mathcal{S}^c. \tag{D.114}$$

So as $l \to \infty$ and $\delta_l \to 0$, every component getting closer to limit Kernel, i.e, taking the limit from both sides and using the fact that is proven is theorem 1 $\lim_{l\to\infty} \delta_l = 0$ leads to

$$\lim_{l\to\infty} 1 \leq \mathcal{K}_W(r_i, r_j) \quad i, j \in \mathcal{S} \tag{D.115}$$

$$\lim_{l\to\infty} \mathcal{K}_W(r_i, r_j) \leq 0 \quad i, j \in \mathcal{S}^c \tag{D.116}$$

both terms must instead be strictly equality. Therefore, we see that at the limit point $\mathcal{K}_W$ would have the form

$$\mathcal{K}^* = \begin{cases} 0 & \forall i, j \in \mathcal{S}^c \\ 1 & \forall i, j \in \mathcal{S} \end{cases} . \tag{D.117}$$

First Property:

Using Eq. (D.113) and Eq. (D.114) we have:

$$1 - \delta_l \leq e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} \quad i, j \in \mathcal{S} \tag{D.118}$$

$$e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} \leq \delta_l \quad i, j \in \mathcal{S}^c. \tag{D.119}$$

As $\lim_{l\to\infty} \delta_l = 0$, taking the limit from both side leads to:

$$\begin{cases} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} = 1 & \forall i, j \in \mathcal{S} \\ e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} = 0 & \forall i, j \in \mathcal{S}^c \end{cases} . \tag{D.120}$$

If we take the log of the conditions, we get

$$\begin{cases} \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = 0 & \forall i, j \in \mathcal{S} \\ \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = \infty & \forall i, j \in \mathcal{S}^c \end{cases} . \tag{D.121}$$

This implies that as $l \to \infty$ we have

$$\lim_{l\to\infty} \sum_{i,j\in\mathcal{S}} \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = \lim_{l\to\infty} \mathrm{Tr}(S_w) = 0. \tag{D.122}$$

$$\lim_{l\to\infty} \sum_{i,j\in\mathcal{S}^c} \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = \lim_{l\to\infty} \mathrm{Tr}(S_b) = \infty, \tag{D.123}$$

This yields the ratio

$$\lim_{\mathcal{H}_l \to \mathcal{H}^*} \frac{\mathrm{Tr}(S_w)}{\mathrm{Tr}(S_b)} = \frac{0}{\infty} = 0. \tag{D.124}$$

□

D.3 Proof for Theorem 6.3

Theorem 6.3: *Eq. (6.5) objective is equivalent to*

$$\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i^T W W^T r_j) - \sum_i D_i(W) ||W^T r_i||_2. \tag{D.125}$$

Proof. Let $A_{i,j} = (r_i - r_j)(r_i - r_j)^T$. Given the Lagranian of the HSIC objective as

$$\mathcal{L} = -\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} - \mathrm{Tr}[\Lambda(W^T W - I)]. \tag{D.126}$$

Our layer wise HSIC objective becomes

$$\min_W -\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} - \mathrm{Tr}[\Lambda(W^T W - I)]. \tag{D.127}$$

We take the derivative of the Lagrangian, the expression becomes

$$\nabla_W \mathcal{L}(W, \Lambda) = \sum_{i,j} \frac{\Gamma_{i,j}}{\sigma^2} e^{-\frac{\mathrm{Tr}(W^T A_{i,j} W)}{2\sigma^2}} A_{i,j} W - 2W\Lambda. \tag{D.128}$$

Setting the gradient to 0, and consolidate some scalar values into $\hat{\Gamma}_{i,j}$, we get the expression

$$\left[\sum_{i,j} \frac{\Gamma_{i,j}}{2\sigma^2} e^{-\frac{\mathrm{Tr}(W^T A_{i,j} W)}{2\sigma^2}} A_{i,j} \right] W = W\Lambda \tag{D.129}$$

$$\left[\frac{1}{2} \sum_{i,j} \hat{\Gamma}_{i,j} A_{i,j} \right] W = W\Lambda \tag{D.130}$$

$$\mathcal{Q}W = W\Lambda. \tag{D.131}$$

From here, we see that the optimal solution is an eigenvector of $\mathcal{Q}$. Based on ISM, it further proved that the optimal solution is not just any eigenvector, but the eigenvectors associated with the smallest

values of $\mathcal{Q}$. From this logic, ISM solves objective (D.127) with a surrogate objective

$$\min_{W} \quad \text{Tr}\left(W^T\left[\frac{1}{2}\sum_{i,j}\hat{\Gamma}_{i,j}A_{i,j}\right]W\right) \quad \text{s.t.}\, W^TW = I. \tag{D.132}$$

Given $D_{\hat{\Gamma}}$ as the degree matrix of $\hat{\Gamma}$ and $R = [r_1, r_2, ...]^T$, ISM further shows that Eq. (D.132) can be written into

$$\min_{W} \quad \text{Tr}\left(W^TR^T\left[D_{\hat{\Gamma}} - \hat{\Gamma}\right]RW\right) \quad \text{s.t.}\, W^TW = I \tag{D.133}$$

$$\max_{W} \quad \text{Tr}\left(W^TR^T\left[\hat{\Gamma} - D_{\hat{\Gamma}}\right]RW\right) \quad \text{s.t.}\, W^TW = I \tag{D.134}$$

$$\max_{W} \quad \text{Tr}\left(W^TR^T\hat{\Gamma}RW\right) - \text{Tr}\left(W^TR^TD_{\hat{\Gamma}}RW\right) \quad \text{s.t.}\, W^TW = I \tag{D.135}$$

$$\max_{W} \quad \text{Tr}\left(\hat{\Gamma}RWW^TR^T\right) - \text{Tr}\left(D_{\hat{\Gamma}}RWW^TR^T\right) \quad \text{s.t.}\, W^TW = I \tag{D.136}$$

$$\max_{W} \quad \sum_{i,j}\hat{\Gamma}_{i,j}[RWW^TR^T]_{i,j} - \sum_{i,j}D_{\hat{\Gamma}_{i,j}}[RWW^TR^T]_{i,j} \quad \text{s.t.}\, W^TW = I. \tag{D.137}$$

Since the jump from Eq. (D.132) can be intimidating for those not familiar with the literature, we included a more detailed derivation in App. D.4.

Note that the degree matrix $D_{\hat{\Gamma}}$ only have non-zero diagonal elements, all of its off diagonal are 0. Given $[RWW^TR^T]_{i,j} = (r_i^TWW^Tr_j)$, the objective becomes

$$\max_{W} \quad \sum_{i,j}\hat{\Gamma}_{i,j}(r_i^TWW^Tr_j) - \sum_{i}D_i(W)||W^Tr_i||_2 \quad \text{s.t.}\, W^TW = I. \tag{D.138}$$

Here, we treat D_i as a penalty weight on the norm of the W^Tr_i for every sample. □

To better understand the behavior of $D_i(W)$, note that $\hat{\Gamma}$ matrix looks like

$$\hat{\Gamma} = \frac{1}{\sigma^2}\begin{bmatrix} \left[\Gamma_{\mathcal{S}}e^{-\frac{(r_i-r_j)^TWW^T(r_i-r_j)}{2\sigma^2}}\right] & \left[-|\Gamma_{\mathcal{S}^c}|e^{-\frac{(r_i-r_j)^TWW^T(r_i-r_j)}{2\sigma^2}}\right] & ... \\ \left[-|\Gamma_{\mathcal{S}^c}|e^{-\frac{(r_i-r_j)^TWW^T(r_i-r_j)}{2\sigma^2}}\right] & \left[\Gamma_{\mathcal{S}}e^{-\frac{(r_i-r_j)^TWW^T(r_i-r_j)}{2\sigma^2}}\right] & ... \\ ... & ... & ... \end{bmatrix}. \tag{D.139}$$

The diagonal block matrix all $\Gamma_{i,j}$ elements that belong to $\mathcal{S}$ and the off diagonal are elements that belongs to $\mathcal{S}^c$. Each penalty term is the summation of its corresponding row. Hence, we can write out the penalty term as

$$D_i(W_l) = \frac{1}{\sigma^2}\sum_{j\in\mathcal{S}|i}\Gamma_{i,j}\mathcal{K}_{W_l}(r_i, r_j) - \frac{1}{\sigma^2}\sum_{j\in\mathcal{S}^c|i}|\Gamma_{i,j}|\mathcal{K}_{W_l}(r_i, r_j). \tag{D.140}$$

From this, it shows that as W improve the objective, the penalty term is also increased. In fact, at its extreme as $\mathcal{H}_l \to \mathcal{H}^*$, all the negative terms are gone and all of its positive terms are maximized and this matrix approaches

$$\hat{\Gamma}^* = \frac{1}{\sigma^2} \begin{bmatrix} \begin{bmatrix} \Gamma_{\mathcal{S}} \end{bmatrix} & \begin{bmatrix} 0 \end{bmatrix} & ... \\ \begin{bmatrix} 0 \end{bmatrix} & \begin{bmatrix} \Gamma_{\mathcal{S}} \end{bmatrix} & ... \\ ... & ... & ... \end{bmatrix}. \tag{D.141}$$

From the matrix $\hat{\Gamma}^*$ and the definition of $D_i(W_l)$, we see that as $\mathcal{K}_W$ from $\mathcal{S}$ increase,

Since $D_i(W)$ is the degree matrix of $\hat{\Gamma}$, we see that as $\mathcal{H}_l \to \mathcal{H}^*$, we have

$$D_i^*(W) > D_i(W). \tag{D.142}$$

D.4 Derivation for $\sum_{i,j} \Psi_{i,j}(x_i - x_j)(x_i - x_j)^T = 2X^T(D_\Psi - \Psi)X$

Since Ψ is a symmetric matrix, and $A_{i,j} = (x_i - x_j)(x_i - x_j)^T$, we can rewrite the expression into

$$\begin{aligned} \textstyle\sum_{i,j} \Psi_{i,j} A_{i,j} &= \textstyle\sum_{i,j} \Psi_{i,j}(x_i - x_j)(x_i - x_j)^T \\ &= \textstyle\sum_{i,j} \Psi_{i,j}(x_i x_i^T - x_j x_i^T - x_i x_j^T + x_j x_j^T) \\ &= 2\textstyle\sum_{i,j} \Psi_{i,j}(x_i x_i^T - x_j x_i^T) \\ &= \left[2\textstyle\sum_{i,j} \Psi_{i,j}(x_i x_i^T)\right] - \left[2\textstyle\sum_{i,j} \Psi_{i,j}(x_i x_j^T)\right]. \end{aligned}$$

If we expand the 1st term, we get

$$2\sum_i^n \sum_j^n \Psi_{i,j}(x_i x_i^T) = 2\sum_i \Psi_{i,1}(x_i x_i^T) + \ldots + \Psi_{i,n}(x_i x_i^T) \tag{D.143}$$

$$= 2\sum_i^n [\Psi_{1,1} + \Psi_{1,2} + ...] x_i x_i^T \tag{D.144}$$

$$= 2\sum_i^n d_i x_i x_i^T \tag{D.145}$$

$$= 2X^T D_\Psi X \tag{D.146}$$

Given Ψ_i as the ith row, next we look at the 2nd term

$$2\sum_i\sum_j \Psi_{i,j}x_i x_j^T = 2\sum_i \Psi_{i,1}x_i x_1^T + \Psi_{i,2}x_i x_2^T + \Psi_{i,3}x_i x_3^T + ... \tag{D.147}$$

$$= 2\sum_i x_i(\Psi_{i,1}x_1^T) + x_i(\Psi_{i,2}x_2^T) + x_i(\Psi_{i,3}x_3^T) + ... \tag{D.148}$$

$$= 2\sum_i x_i\left[(\Psi_{i,1}x_1^T) + (\Psi_{i,2}x_2^T) + (\Psi_{i,3}x_3^T) + ...\right] \tag{D.149}$$

$$= 2\sum_i x_i\left[X^T\Psi_i^T\right]^T \tag{D.150}$$

$$= 2\sum_i x_i\left[\Psi_i X\right] \tag{D.151}$$

$$= 2\left[x_1\Psi_1 X + x_2\Psi_2 X + x_3\Psi_3 X + ...\right] \tag{D.152}$$

$$= 2\left[x_1\Psi_1 + x_2\Psi_2 + x_3\Psi_3 + ...\right]X \tag{D.153}$$

$$= 2X^T\Psi X \tag{D.154}$$

$$\tag{D.155}$$

Putting both terms together, we get

$$\sum_{i,j}\Psi_{i,j}A_{i,j} = 2X^T D_\Psi X - 2X^T\Psi X a \tag{D.156}$$

$$= 2X^T[D_\Psi - \Psi]X \tag{D.157}$$

$$\tag{D.158}$$

D.5 Proof for Corollary 2 and 3

Corollary 2: *Given $\mathcal{H}_l \to \mathcal{H}^*$, the network output in IDS solves MSE via a translation of labels.*

Proof.

As $\mathcal{H}_l \to \mathcal{H}^*$, Thm. 6.2 shows that sample of the same class are mapped into the same point. Assuming that ϕ has mapped the sample into c points $\alpha = [\alpha_1, ..., \alpha_c]$ that's different from the truth label $\xi = [\xi_1, ..., \xi_c]$. Then the MSE objective is minimized by translating the ϕ output by

$$\xi - \alpha. \tag{D.159}$$

□

Corollary 3: *Given $\mathcal{H}_l \to \mathcal{H}^*$, the network output in RKHS solves CE via a change of bases.*

Assumptions, and Notations.

1. n is the number of samples.
2. τ is the number of classes.
3. $y_i \in \mathbb{R}^\tau$ is the ground truth label for the i^{th} sample. It is one-hot encoded where only the j^{th} element is 1 if x_i belongs to the j^{th} class, all other elements would be 0.
4. We denote ϕ as the network, and $\hat{y}_i \in \mathbb{R}^\tau$ as the network output where $\hat{y}_i = \phi(x_i)$. We also assume that $\hat{y}_i$ is constrained on a probability simplex where $1 = \hat{y}_i^T \mathbf{1}_n$.
5. We denote the j^{th} element of y_i, and $\hat{y}_i$ as $y_{i,j}$ and $\hat{y}_{i,j}$ respectively.
6. We define

   Orthogonality Condition: A set of samples $\{\hat{y}_1, ..., \hat{y}_n\}$ satisfies the orthogonality condition if

   $$\begin{cases} \langle \hat{y_i}, \hat{y_j} \rangle = 1 & \forall \quad i, j \text{ same class} \\ \langle \hat{y_i}, \hat{y_j} \rangle = 0 & \forall \quad i, j \text{ not in the same class} \end{cases}. \tag{D.160}$$

7. We define the Cross-Entropy objective as

   $$\arg\min_{\phi} - \sum_{i=1}^{n} \sum_{j=1}^{\tau} y_{i,j} \log(\phi(x_i)_{i,j}). \tag{D.161}$$

Proof.

From Thm. 6.2, we know that the network ϕ output, $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$, satisfy the orthogonality condition at $\mathcal{H}^*$. Then there exists a set of orthogonal bases represented by $\Xi = [\xi_1, \xi_2, ..., \xi_c]$ that maps $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$ to simulate the output of a softmax layer. Let $\xi_i = \hat{y}_j, j \in \mathcal{S}^i$, i.e., for the i_{th} class we arbitrary choose one of the samples from this class and assigns ξ_i of that class to be equal to the sample's output. Realize in our problem we have $< \hat{y}_i, \hat{y}_i >= 1$, so if $< \hat{y}_i, \hat{y}_j >= 1$, then subtracting these two would lead to $< \hat{y}_i, \hat{y}_i - \hat{y}_j >= 0$, which is the same

as $\hat{y}_i = \hat{y}_j$. So this representation is well-defined and its independent of choices of the sample from each group if they satisfy orthogonality condition. Now we define transformed labels, Y as:

$$Y = \hat{Y}\Xi. \tag{D.162}$$

Note that $Y = [y_1, y_2, ..., y_n]^T$ which each y_i is a one hot vector representing the class membership of i sample in c classes. Since given Ξ as the change of basis, we can match $\hat{Y}$ to Y exactly, CE is minimized.

□

D.6 Dataset Details

No samples were excludes from any of the dataset.

Wine. This dataset has 13 features, 178 samples, and 3 classes. The features are continuous and heavily unbalanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/wine`.

Divorce. This dataset has 54 features, 170 samples, and 2 classes. The features are discrete and balanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set`.

Car. This dataset has 6 features, 1728 samples and 2 classes. The features are discrete and balanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`.

Cancer. This dataset has 9 features, 683 samples, and 2 classes. The features are discrete and unbalanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`.

Face. This dataset consists of images of 20 people in various poses. The 624 images are vectorized into 960 features. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/CMU+Face+Images`.

Random. This dataset has 2 features, 80 samples and 2 classes. It is generate with a gaussian distribution where half of the samples are randomly labeled as 1 or 0.

Adversarial. This dataset has 2 features, 80 samples and 2 classes. It is generate with the following code:

```
#!/usr/bin/env python


n = 40
X1 = np.random.rand(n,2)
X2 = X1 + 0.01*np.random.randn(n,2)
```

```
X = np.vstack((X1,X2))
Y = np.vstack(( np.zeros((n,1)), np.ones((n,1)) ))
```

D.7 W_l Dimensions for each 10 Fold of each Dataset

We report the input and output dimensions of each W_l for every layer of each dataset in the form of (α, β); the corresponding dimension becomes $W_l \in \mathbb{R}^{\alpha \times \beta}$. Since each dataset consists of 10-folds, the network structure for each fold is reported. We note that the input of the 1st layer is the dimension of the original data. However, after the first layer, the width of the RFF becomes the output of each layer; here we use 300.

The β value is chosen during the ISM algorithm. By keeping only the most dominant eigenvector of the Φ matrix, the output dimension of each layer corresponds with the rank of Φ. It can be seen from each dataset that the first layer significantly expands the rank. The expansion is generally followed by a compression of fewer and fewer eigenvalues. These results conform with the observations made by Montavon et al. [101] and Ansuini et al. [99].

Data	Layer 1	Layer 2	Layer 3	Layer 4
adversarial 1	(2, 2)	(300, 61)	(300, 35)	
adversarial 2	(2, 2)	(300, 61)	(300, 35)	
adversarial 3	(2, 2)	(300, 61)	(300, 8)	(300, 4)
adversarial 4	(2, 2)	(300, 61)	(300, 29)	
adversarial 5	(2, 2)	(300, 61)	(300, 29)	
adversarial 6	(2, 2)	(300, 61)	(300, 7)	(300, 4)
adversarial 7	(2, 2)	(300, 61)	(300, 34)	
adversarial 8	(2, 2)	(300, 12)	(300, 61)	(300, 30)
adversarial 9	(2, 2)	(300, 61)	(300, 33)	
adversarial 10	(2, 2)	(300, 61)	(300, 33)	

Data	Layer 1	Layer 2	Layer 3
Random 1	(3, 3)	(300, 47)	(300, 25)
Random 2	(3, 3)	(300, 46)	(300, 25)
Random 3	(3, 3)	(300, 46)	(300, 25)
Random 4	(3, 3)	(300, 47)	(300, 4)
Random 5	(3, 3)	(300, 47)	(300, 25)
Random 6	(3, 3)	(300, 45)	(300, 23)
Random 7	(3, 3)	(300, 45)	(300, 25)
Random 8	(3, 3)	(300, 45)	(300, 21)
Random 9	(3, 3)	(300, 45)	(300, 26)
Random 10	(3, 3)	(300, 47)	(300, 25)

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
spiral 1	(2, 2)	(300, 15)	(300, 6)	(300, 7)	(300, 6)	
spiral 2	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 3	(2, 2)	(300, 12)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 4	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 5	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	
spiral 6	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	
spiral 7	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	
spiral 8	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 9	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	
spiral 10	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
wine 1	(13, 11)	(300, 76)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 2	(13, 11)	(300, 76)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 3	(13, 11)	(300, 75)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 4	(13, 11)	(300, 76)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 5	(13, 11)	(300, 74)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 6	(13, 11)	(300, 74)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 7	(13, 11)	(300, 74)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 8	(13, 11)	(300, 75)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 9	(13, 11)	(300, 75)	(300, 6)	(300, 8)	(300, 6)	(300, 6)
wine 10	(13, 11)	(300, 76)	(300, 6)	(300, 7)	(300, 6)	(300, 6)

APPENDIX D.

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
car 1	(6, 6)	(300, 96)	(300, 6)	(300, 8)	(300, 6)	
car 2	(6, 6)	(300, 96)	(300, 6)	(300, 8)	(300, 6)	
car 3	(6, 6)	(300, 91)	(300, 6)	(300, 8)	(300, 6)	
car 4	(6, 6)	(300, 88)	(300, 6)	(300, 8)	(300, 6)	(300, 6)
car 5	(6, 6)	(300, 94)	(300, 6)	(300, 8)	(300, 6)	
car 6	(6, 6)	(300, 93)	(300, 6)	(300, 7)		
car 7	(6, 6)	(300, 92)	(300, 6)	(300, 8)	(300, 6)	
car 8	(6, 6)	(300, 95)	(300, 6)	(300, 7)	(300, 6)	
car 9	(6, 6)	(300, 96)	(300, 6)	(300, 9)	(300, 6)	
car 10	(6, 6)	(300, 99)	(300, 6)	(300, 8)	(300, 6)	

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
divorce 1	(54, 35)	(300, 44)	(300, 5)	(300, 5)	
divorce 2	(54, 35)	(300, 45)	(300, 4)	(300, 4)	
divorce 3	(54, 36)	(300, 49)	(300, 6)	(300, 6)	
divorce 4	(54, 36)	(300, 47)	(300, 7)	(300, 6)	
divorce 5	(54, 35)	(300, 45)	(300, 6)	(300, 6)	
divorce 6	(54, 36)	(300, 47)	(300, 6)	(300, 6)	
divorce 7	(54, 35)	(300, 45)	(300, 6)	(300, 6)	(300, 4)
divorce 8	(54, 36)	(300, 47)	(300, 6)	(300, 7)	(300, 4)
divorce 9	(54, 36)	(300, 47)	(300, 5)	(300, 5)	
divorce 10	(54, 36)	(300, 47)	(300, 6)	(300, 6)	

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9	Layer 10
cancer 1	(9, 8)	(300, 90)	(300, 5)	(300, 6)	(300, 6)	(300, 5)	(300, 4)	(300, 5)	(300, 6)	(300, 6)
cancer 2	(9, 8)	(300, 90)	(300, 6)	(300, 7)	(300, 8)	(300, 11)	(300, 8)	(300, 4)		
cancer 3	(9, 8)	(300, 88)	(300, 5)	(300, 6)	(300, 7)	(300, 7)	(300, 6)	(300, 4)		
cancer 4	(9, 8)	(300, 93)	(300, 6)	(300, 7)	(300, 9)	(300, 11)	(300, 8)			
cancer 5	(9, 8)	(300, 93)	(300, 9)	(300, 10)	(300, 10)	(300, 11)	(300, 9)	(300, 7)		
cancer 6	(9, 8)	(300, 92)	(300, 7)	(300, 8)	(300, 8)	(300, 7)	(300, 7)			
cancer 7	(9, 8)	(300, 90)	(300, 4)	(300, 4)	(300, 5)	(300, 6)	(300, 6)	(300, 6)	(300, 6)	
cancer 8	(9, 8)	(300, 88)	(300, 5)	(300, 6)	(300, 7)	(300, 8)	(300, 7)	(300, 6)		
cancer 9	(9, 8)	(300, 88)	(300, 5)	(300, 7)	(300, 7)	(300, 7)	(300, 7)			
cancer 10	(9, 8)	(300, 97)	(300, 9)	(300, 11)	(300, 12)	(300, 13)	(300, 6)			

Data	Layer 1	Layer 2	Layer 3	Layer 4
face 1	(960, 233)	(300, 74)	(300, 73)	(300, 46)
face 2	(960, 231)	(300, 75)	(300, 73)	(300, 43)
face 3	(960, 231)	(300, 76)	(300, 73)	(300, 44)
face 4	(960, 232)	(300, 76)	(300, 74)	(300, 44)
face 5	(960, 231)	(300, 77)	(300, 73)	(300, 43)
face 6	(960, 232)	(300, 74)	(300, 72)	(300, 47)
face 7	(960, 232)	(300, 76)	(300, 73)	(300, 45)
face 8	(960, 230)	(300, 74)	(300, 74)	(300, 44)
face 9	(960, 233)	(300, 76)	(300, 76)	(300, 45)
face 10	(960, 231)	(300, 76)	(300, 70)	(300, 43)

D.8 Sigma Values used for Random and Adversarial Simulation

The simulation of Thm. 6.1 as shown in Fig. 6.2 spread the improvement across multiple layers. The σ_l and $\mathcal{H}_l$ values are recorded here. We note that σ_l are reasonably large and not approaching 0 and the improvement of $\mathcal{H}_l$ is monotonic.

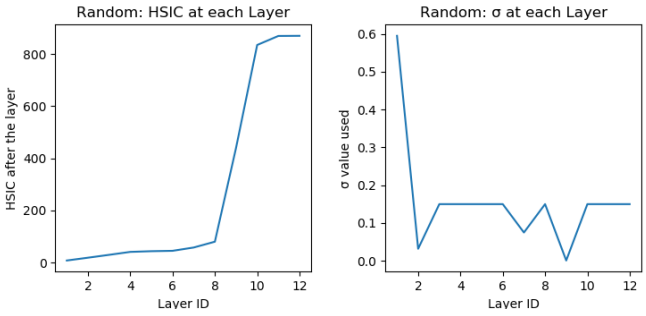

Figure D.3

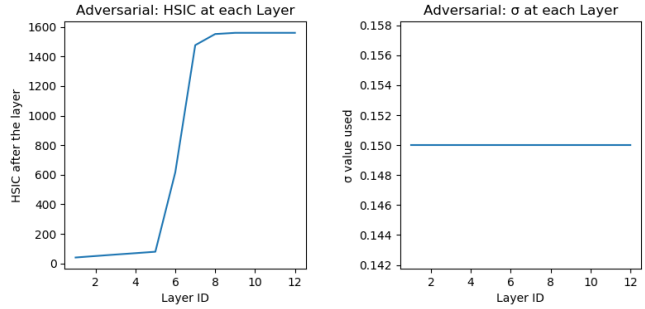

Figure D.4

Given a sufficiently small σ_0 and σ_1, Thm. 6.1 claims that it can come arbitrarily close to the global optimal using a minimum of 2 layers. We here simulate 2 layers using a relatively small σ values ($\sigma_0 = 10^{-5}$) on the Random (left) and Adversarial (right) data and display the results of the 2 layers below. Notice that given 2 layer, it generated a clearly separable clusters that are pushed far apart.

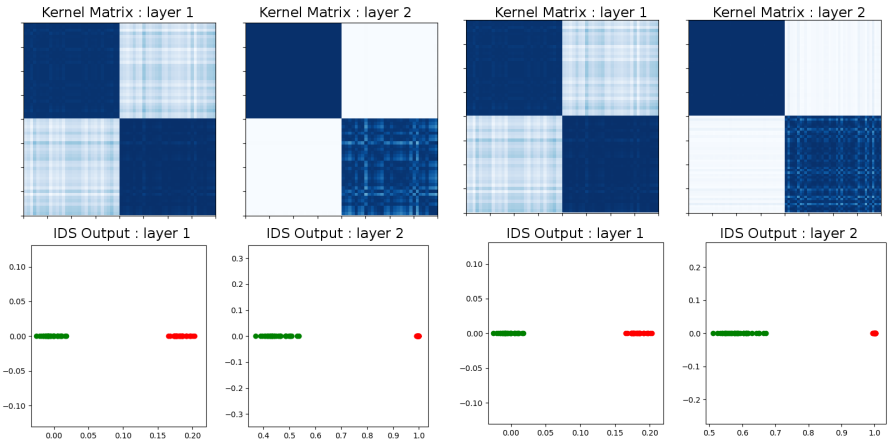

Figure D.5: Random Dataset with 2 layers and $\sigma = 10^{-5}$

Figure D.6: Adversarial Dataset with 2 layers and $\sigma = 10^{-5}$

D.9 Graphs of Kernel Sequences

A representation of the *Kernel Sequence* are displayed in the figures below for each dataset. The samples of the kernel matrix are previously organized to form a block structure by placing samples of the same class adjacent to each other. Since the Gaussian kernel is restricted to values between 0 and 1, we let white and dark blue be 0 and 1 respectively where the gradients reflect values in between. Our theorems predict that the *Kernel Sequence* will evolve from an uninformative kernel into a highly discriminating kernel of perfect block structures.

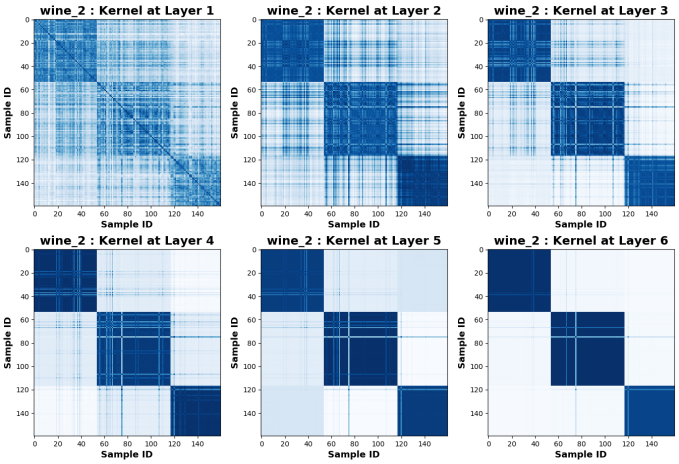

Figure D.7: The kernel sequence for the wine dataset.

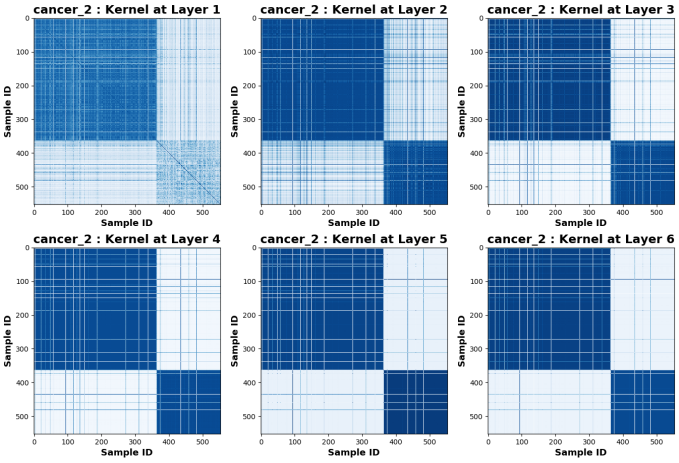

Figure D.8: The kernel sequence for the cancer dataset.

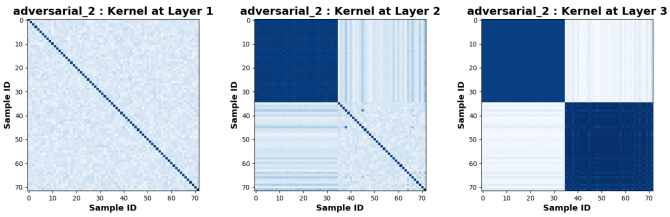

Figure D.9: The kernel sequence for the Adversarial dataset.

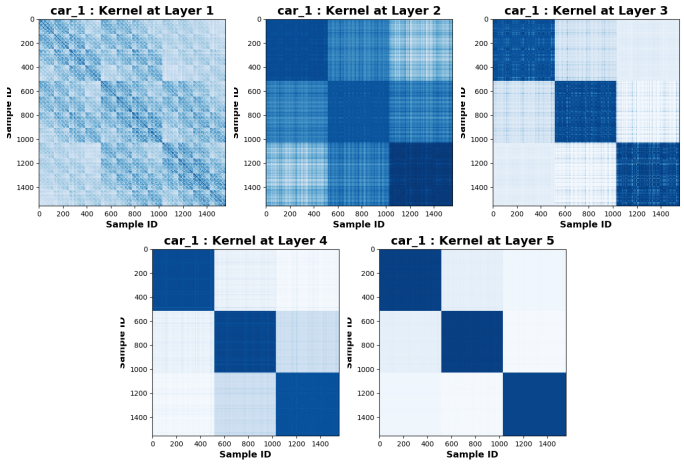

Figure D.10: The kernel sequence for the car dataset.

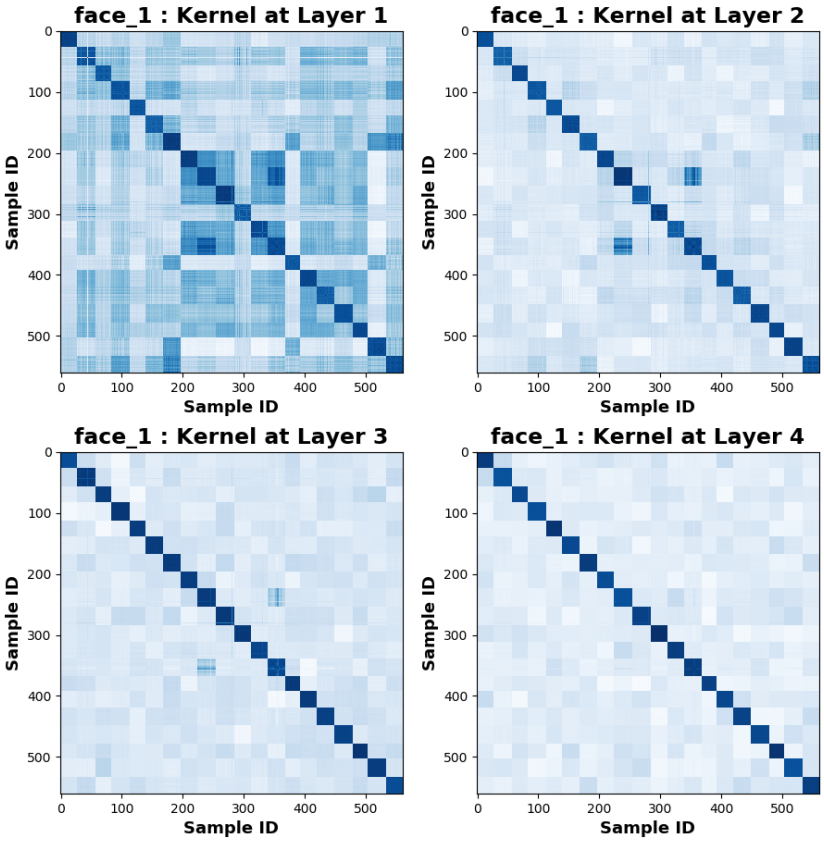

Figure D.11: The kernel sequence for the face dataset.

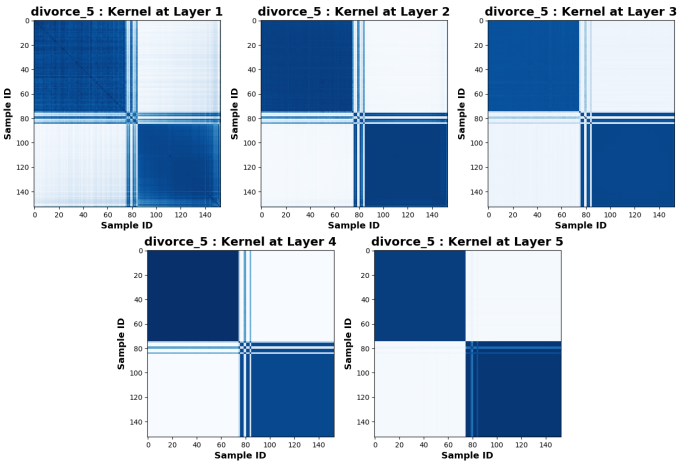

Figure D.12: The kernel sequence for the divorce dataset.

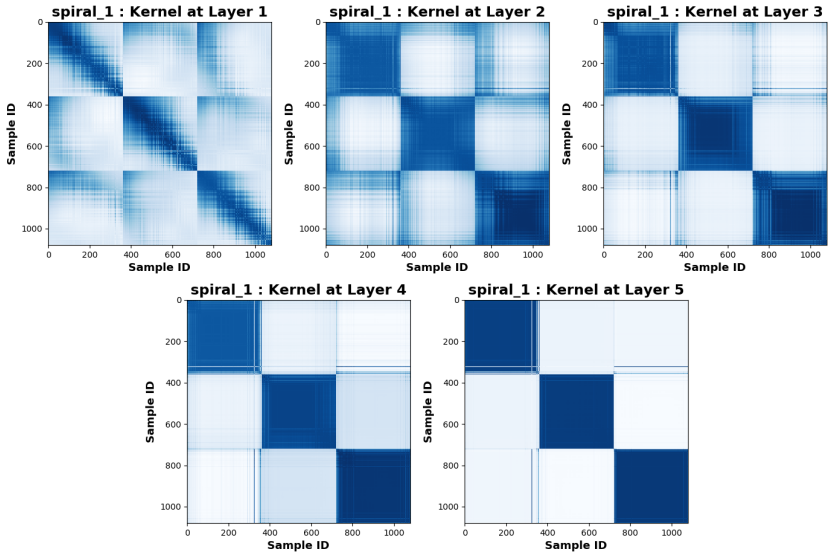

Figure D.13: The kernel sequence for the spiral dataset.

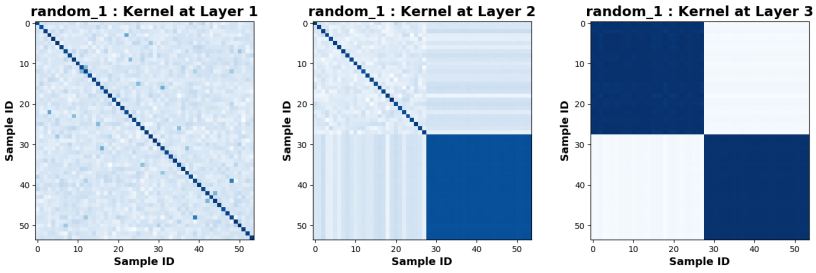

Figure D.14: The kernel sequence for the Random dataset.

D.10 Evaluation Metrics Graphs

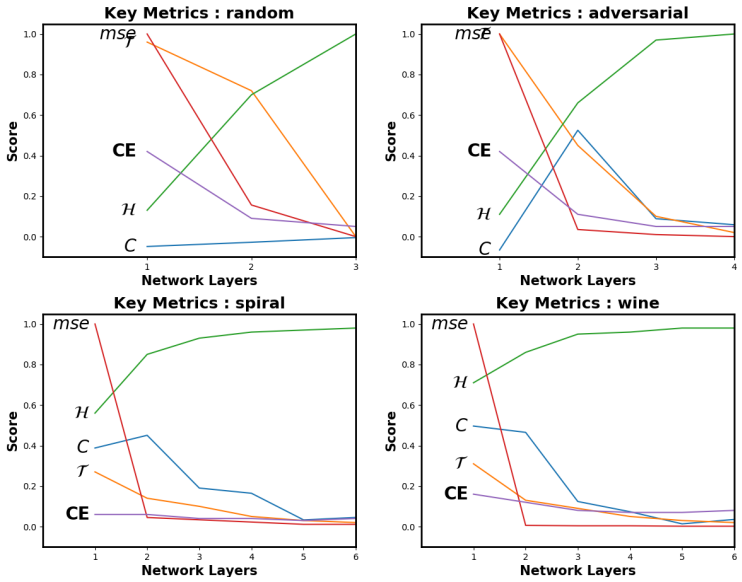

Figure D.15

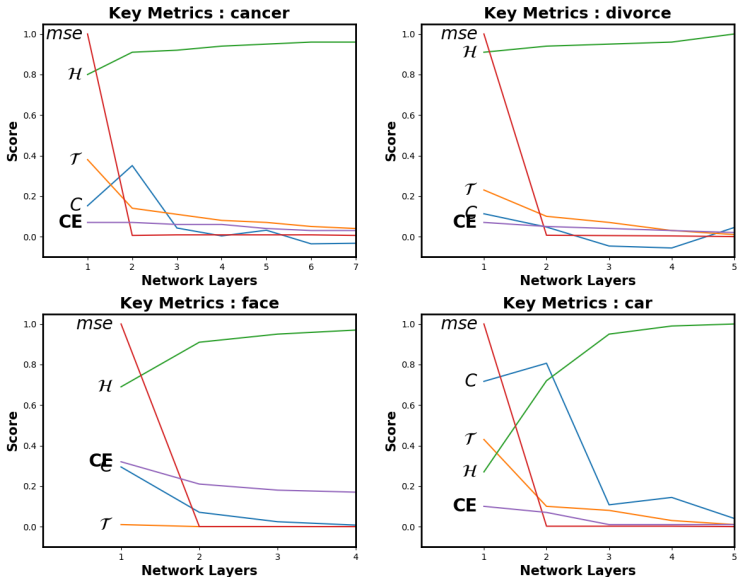

Figure D.16: Figures of key metrics for all datasets as samples progress through the network. It is important to notice the uniformly and monotonically increasing $\mathcal{H}$-*Sequence* for each plot since this guarantees a converging kernel/risk sequence. As the $\mathcal{T}$ approach 0, samples of the same/difference classes in IDS are being pulled into a single point or pushed maximally apart respectively. As C approach 0, samples of the same/difference classes in RKHS are being pulled into 0 or $\frac{\pi}{2}$ cosine similarity respectively.

D.11 Optimal Gaussian σ for Maximum Kernel Separation

Although the Gaussian kernel is the most common kernel choice for kernel methods, its σ value is a hyperparameter that must be tuned for each dataset. This work proposes to set the σ value based on the maximum kernel separation. The source code is made publicly available on `https://github.com/anonamous`.

Let $X \in \mathbb{R}^{n \times d}$ be a dataset of n samples with d features and let $Y \in \mathbb{R}^{n \times \tau}$ be the corresponding one-hot encoded labels where τ denotes the number of classes. Given $\kappa_X(\cdot,\cdot)$ and $\kappa_Y(\cdot,\cdot)$ as two kernel functions that applies respectively to X and Y to construct kernel matrices $K_X \in \mathbb{R}^{n \times n}$ and $K_Y \in \mathbb{R}^{n \times n}$. Given a set $\mathcal{S}$, we denote $|\mathcal{S}|$ as the number of elements within the set. Also let $\mathcal{S}$ and $\mathcal{S}^c$ be sets of all pairs of samples of (x_i, x_j) from a dataset X that belongs to the same and different

classes respectively, then the average kernel value for all (x_i, x_j) pairs with the same class is

$$d_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{i,j \in \mathcal{S}} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{D.163}$$

and the average kernel value for all (x_i, x_j) pairs between different classes is

$$d_{\mathcal{S}^c} = \frac{1}{|\mathcal{S}^c|} \sum_{i,j \in \mathcal{S}^c} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}. \tag{D.164}$$

We propose to find the σ that maximizes the difference between $d_{\mathcal{S}}$ and $d_{\mathcal{S}^c}$ or

$$\max_{\sigma} \quad \frac{1}{|\mathcal{S}|} \sum_{i,j \in \mathcal{S}} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} - \frac{1}{|\mathcal{S}^c|} \sum_{i,j \in \mathcal{S}^c} e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}. \tag{D.165}$$

It turns out that is expression can be computed efficiently. Let $g = \frac{1}{|\mathcal{S}|}$ and $\bar{g} = \frac{1}{|\mathcal{S}^c|}$, and let $\mathbf{1}_{n \times n} \in \mathbb{R}^{n \times n}$ be a matrix of 1s, then we can define Q as

$$Q = -gK_Y + \bar{g}(\mathbf{1}_{n \times n} - K_Y). \tag{D.166}$$

Or Q can be written more compactly as

$$Q = \bar{g}\mathbf{1}_{n \times n} - (g + \bar{g})K_Y. \tag{D.167}$$

Given Q, Eq. (D.165) becomes

$$\min_{\sigma} \quad \mathrm{Tr}(K_X Q). \tag{D.168}$$

This objective can be efficiently solved with BFGS.

Below in Fig. D.17, we plot out the average within cluster kernel and the between cluster kernel values as we vary σ. From the plot, we can see that the maximum separation is discovered via BFGS.

Relation to HSIC. From Eq. (D.168), we can see that the σ that causes maximum kernel separation is directly related to HSIC. Given that the HSIC objective is normally written as

$$\min_{\sigma} \quad \mathrm{Tr}(K_X H K_Y H), \tag{D.169}$$

by setting $Q = HK_Y H$, we can see how the two formulations are related. While the maximum kernel separation places the weight of each sample pair equally, HSIC weights the pair differently. We also notice that the $Q_{i,j}$ element is positive/negative for (x_i, x_j) pairs that are with/between

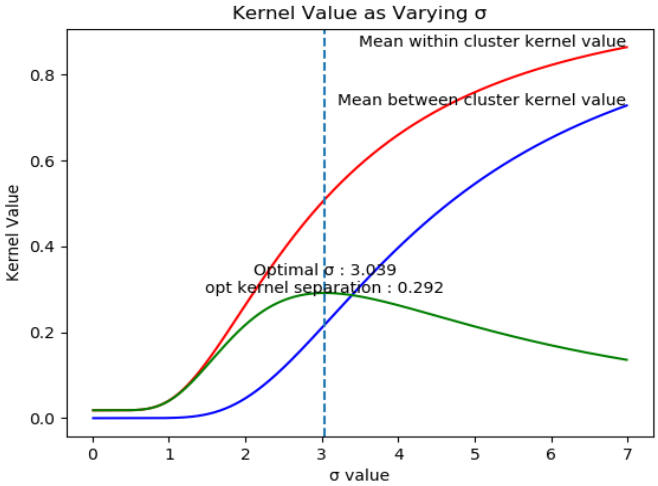

Figure D.17: Maximum Kernel separation.

classes respectively. Therefore, the argument for the global optimum should be relatively close for both objectives. Below in Figure D.18, we show a figure of HSIC values as we vary σ. Notice how the optimal σ is almost equivalent to the solution from maximum kernel separation. For the purpose of *KNet*, we use σ that maximizes the HSIC value.

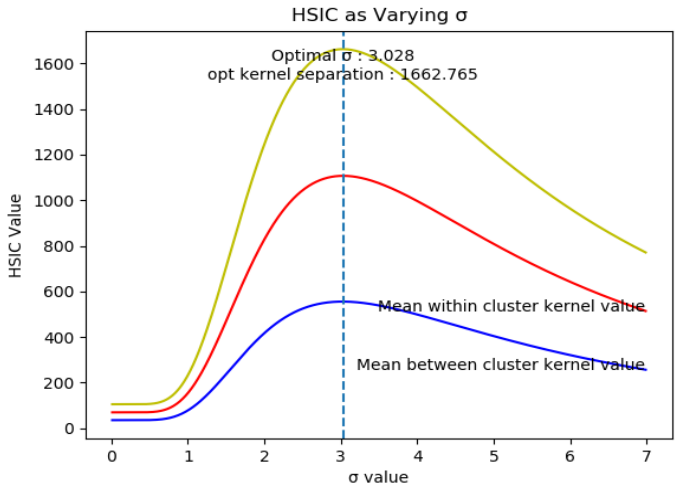

Figure D.18: Maximal HSIC.