DISSERTATION

LONG-TERM LEARNING FOR ADAPTIVE UNDERWATER UXO CLASSIFICATION

Submitted by

John Joseph Hall

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2022

Doctoral Committee:

Advisor: Mahmood R. Azimi-Sadjadi

Ali Pezeshki
J. Rockey Luo
Michael Kirby

ABSTRACT

LONG-TERM LEARNING FOR ADAPTIVE UNDERWATER UXO CLASSIFICATION

Classification of underwater objects such as unexploded ordnances (UXO) and mines from sonar datasets poses a difficult problem. Among factors that complicate classification of these objects are: variations in the operating and environmental conditions, presence of spatially varying clutter, variations in target shape, composition, orientation and burial conditions. Furthermore, collection of large quantities of *real* and representative data for training and testing in various background conditions is very difficult and impractical in many cases. In this dissertation, we build on our previous work in [1] where sparse-reconstruction based classification models were trained on synthetically generated sonar datasets to perform classification on real datasets. While this earlier work helped address issues of data poverty that are intrinsic to the underwater mine-hunting problem, in this work we change course to focus on the adaptation of such models. Particularly, we investigate approaches to adapting linear and kernelized forms of sparse reconstruction based classifiers (SRCs) to function in a *lifelong learning* setting in order to perform classification as environmental parameters are constantly evolving, without sacrificing performance on previously encountered environments.

In this dissertation, we try to address several key questions for designing robust classifiers for UXO and munitions classification from low frequency sonar in a Lifelong learning setting. These include: (1) What are the most viable mechanisms to allow an unmanned underwater vehicle to accumulate and incorporate novel labeled or un-labeled data into its target identification system without sacrificing performance in old environments? (2) What are the most viable mechanisms for allowing an underwater ATR system to extract class labels despite varying environmental conditions? (3) What are the advantages, shortcomings, and major differences, of compressed-sensing based approaches to target identification,

such as the modified MSC with incremental dictionaries, versus popular alternatives such as multi-task learning approaches? (4) How can the modified MSC framework from [1, 2] be extended to allow for kernelized solutions in an efficient manner?

In this work, we propose several novel algorithms in order to address the problems of kernelizing compressed-sensing systems and transitioning these systems to an efficient incremental learning that does not depend on the full kernel matrix of all training samples. By kernelizing the sparse reconstruction classifier, the benefits of: sparse representations and non-linear embedding of samples can be coupled. Among the novel algorithms presented in this dissertation include: an incremental linearized kernel embedding (LKE) that leverages Nyström approximation [3–5] for useful geometric interpretation in the embedded space; A novel algorithm for updating the eigen-decomposition of a growing kernel matrix which leverages fast arrowhead matrix eigendecompositions; and a method for optimizing a custom kernel function for M-ary discrimination tasks. A major technical question that is addressed in this work pertains to whether or not the Matched Subspace Classifier (MSC) [2, 6] can successfully be kernelized and converted into an adaptive form for use in a lifelong learning setting.

The comprehensive testing of the incremental kernelized MSC and its application to the classification of munitions using low frequency sonar is another primary objective of this work. To this end, we test the hypothesis that the non-linearly mapped spectral features captured in the acoustic color (AC) data [2,7,8], extracted from the sonar back-scattered from various objects, display unique features providing superior discrimination between different classes of detected objects to the standard features. In this dissertation we present new classification results using three variants of a kernelized MSC, including an incremental linearized kernel embedding (LKE) MSC with uniform and ridge-leverage score (RLS) sampling, along with an incremental version of the linear version of the modified MSC from [2]. These classifier systems are applied to real sonar datasets, namely the TREX13 and PondEX09-10, to test the generalization ability of classifiers whose baseline training is performed on synthetic

(i.e model generated) sonar datasets generated by a fast ray model (FRM), also known as the Target in environment response (TIER) model [8, 9]. In the incremental cases, a very limited number of labeled samples are utilized to augment the signal models when moving into a new operating environment.

The methods presented here have provided extremely promising results so far, with the incremental LKE based MSC system providing $P_{CC} = 94.6\%$, $P_{FA} = 5.4\%$, and $P_{CC} = 99.3\%$, $P_{FA} = 0.7\%$, when using seven aspects (AC features) per decision, for the TREX13 and PondEX09-10 respectively.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Mahmood Azimi-Sadjadi, he has been a constant source of guidance, support, and inspiration; and has provided an excellent filter through which I may clarify and refine my ideas. His time, effort, mentoring, and friendship in and out of the research environment are so valued. You're the man Mo, thanks for believing in me.

I would also like to thank my committee members, Dr. Ali Pezeshki, Dr. Michael Kirby, and Dr. J. Rockey Luo for their time and assistance over the course of my research. Each of you encouraged and guided me to a more complete view of the problem at hand.

I would like to thank the Strategic Environmental Research and Development Program (SERDP) for providing the funding used in this project under contract number W912HQ-17-C-0002. Thanks to my collaborators on this project, namely the researchers at APL-UW, Dr. Steve Kargl, Dr. Timothy Marston, and Dr. Kevin Williams. I would also like to thank Dr. Nick H. Klausner at Numerica, his guidance and expertise early in my research made this work possible.

I would like to thank my colleagues in the Signal and Image Processing Lab at CSU. They have provided a great environment for discussing work and providing help when most needed. Thanks to Chris Robbiano, Nathan Larson, Yifan Yang, Miles Brim, and Saeid Ahmadinia. Special thanks to Chris for his much needed advice and insight while we were developing the incremental form of the linearized kernel embedding among other tricky problems.

Perhaps most importantly, I would like to thank all of the teachers, mentors, and educators who have helped me see things clearly throughout my education. Thank you to all of my Keith Country Day, Loyola University, Miami University, and Colorado State University professors and teachers. In particular, thank you: Mrs. Lori Walsh, Snra. Apryle Evans, Mr. Greg Farnham, Mr. David Bye, Dr. Beth Lipton, Mr. Leo Dombrowski, Mrs. Ann Wiermanski, Mrs. Christine Brown, Mr. Tom Nuccio, Mrs. Lou Ann Johnson, Mr. Brad Stott,

Dr. Ron Lee, Mrs. Raeann Jurgens, Mr. Jim Radloff, Mrs. Carolyn Cadigan, Mr. Tae Eun Ha, Dr. Katherine Bird, Dr. Özgür Martin, Dr. Katherine Magurn, Dr. Bruce Magurn, Dr. Olga Brezhneva, Dr. Beata Randrianantoanina, Dr. Peter Jamieson, Dr. Qihou Zhou, Dr. Chi-Hao Cheng, Dr. Dmitriy Garmatyuk, Dr. Venkatachalam Chandrasekar, Dr. Edwin Chong, Dr. Ali Pezeshki, Dr. Dan Cooley, Dr. Don Estep, Dr. Margaret Cheney, and extra special thanks to my dear mentors who advised me during graduate school: Dr. Jade Morton and Dr. Mahmood Azimi-Sadjadi. I am so blessed to have met each and every one of you, thank you a million times over. Special thanks to Dr. Edwin Chong who provided me with Lemma 1 in Chapter 5 and for all of his careful guidance during my graduate education.

Finally, I would like to thank my family for their consistent support and guidance throughout my entire education. Thank you to my parents, Renée Neary and Scott Hall, my incredible wife Dr. Yu Jiao, and my four brilliant siblings, Katherine, Jessica, Edward, and Kenneth.

DEDICATION

This work is dedicated to my dear wife, my guiding light, Dr. Yu Jiao.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Motivations

The problem of underwater object classification in sonar imagery is rather complicated due to the numerous factors which inhibit repeatable and reliable Automatic Target Recognition (ATR). These factors include: variations in operating and environmental conditions, presence of spatially varying clutter, as well as burial depth and variations in target shapes, compositions and orientation [1, 2, 7, 8]. Moreover, bottom features such as coral reefs, man-made artificial reefs, sand formations, and vegetation may obscure a target or confuse a classification system. Consequently, a robust ATR system should be able to quantify changes between the returns from the bottom features or structured clutter and any target activity in sonar data, while at the same time extract useful features for classification. Typically, the underwater unexploded ordnance (UXO) hunting problem is addressed with a combination of unmanned underwater vehicles (UUV), trained divers, and surface-based surveying tow-bodies, resulting in very costly reclamation efforts.

In the context of lifelong learning, these constantly changing environmental factors can pose many obstacles to stable and confusion-free ATR. A lifelong learning machine is one that learns continuously, accumulates the knowledge learned in previous tasks, and uses it to help future learning. In the process, the learner becomes more and more knowledgeable and effective at learning. Furthermore, a lifelong learning system should allow for this continuous learning without sacrificing the performance on previously learned tasks. This phenomenon, sometimes referred to as "catastrophic forgetting", where performance is partially or completely compromised on previously encountered datasets or environments, should be avoided

at all costs. To further complicate matters, in the context of UXO hunting problems, access to extensive training datasets is nearly impossible due to the nature of the problem and though several rather pristine datasets exist [9–12] often they are either completely unlabeled or only partially labeled.

In this work we will investigate lifelong learning solutions for underwater ATR problems and introduce several new techniques for developing classification models that can adapt over time to accommodate for and circumvent the aforementioned variations and difficulties.

1.2 Fundamental Questions to be Addressed

The work presented here follows in the vein of our previous studies discussed in [1]. In this continuing work on classification for shallow water sonar systems, the following fundamental questions regarding lifelong learning for classification of UXOs in sonar data will be addressed:

1. What are the most viable mechanisms to allow an unmanned underwater vehicle to accumulate and incorporate novel labeled or un-labeled data into its target identification system without sacrificing performance in old environments?

2. What are the most viable mechanisms for allowing an underwater ATR system to extract class labels despite varying environmental conditions?

3. What are the advantages, shortcomings, and major differences, of compressed-sensing based approaches to target identification, such as the modified MSC with incremental dictionaries, versus popular alternatives such as multi-task learning approaches [13,14]?

4. How can the modified MSC framework be extended to allow for kernelized solutions in an efficient manner?

Particularly, we propose several novel algorithms in order to address the problems of kernelizing compressed-sensing systems and transitioning these systems to an efficient incremental learning that does not depend on the full kernel matrix of all training samples. By

kernelizing the sparse reconstruction classifier, the benefits of: sparse representations and non-linear embedding of samples can be coupled. Among the novel algorithms presented in this dissertation include: an incremental linearized kernel embedding (LKE) [4] that leverages Nyström approximation [3, 5] for useful geometric interpretation in the embedded space; A novel algorithm for updating the eigen-decomposition of a growing kernel matrix; and two methods for optimizing a custom kernel function for M-ary discrimination tasks, inspired by the work in [15]. The classification algorithms developed in this work will be compared to other state-of-the-art approaches, e.g. those in [1, 12, 16], and relative strengths and weaknesses of all systems will be compared. The developed methods could be useful in a multitude of remote sensing applications in which sonar is used to search or survey underwater areas including environmental and oceanographic studies, undersea exploration, the search for wreckage on the seafloor, and underwater mine-hunting.

1.3 Research Objectives and Contributions

The main objective of this work is the development of efficient methods for the classification of military munitions in shallow underwater environments using data collected from low-frequency sonar systems [9–12]. Specifically, the technical questions that are addressed in this work pertain to whether or not the Matched Subspace Classifier (MSC) [2,6] can successfully be kernelized and converted into an adaptive form for use in a lifelong learning setting. This work builds on our previous work in [1] which investigated the feasibility of training a sparse reconstruction classification (SRC) style classifier on model-generated sonar data of various UXO-like and non-UXO like objects and then applying the classifier to real sonar datasets to discriminate the objects with sufficient accuracy. As was demonstrated in [1], the development of systems that can be trained on model-generated data with guaranteed performance on real data can provide a significant contribution toward solving this difficult problem. While, using model data on its own can provide a decent baseline for real environments, the work in [1] also demonstrated that a variety of incremental dictionary update

strategies [17, 18] can be utilized to augment the learned dictionaries with limited labeled data from a new environment resulting in greatly improved discrimination accuracy in the new environment without sacrificing performance on old environment data samples. In this work, the focus is now centered on mechanisms for extracting empirical kernel map features incrementally for the MSC framework and again incorporating the incremental learning in a manner that is consistent with the kernelized framework.

The comprehensive testing of the kernelized MSC and its application to the classification of munitions using low frequency sonar is another primary objective of this work. To this end, we test the hypothesis that the *non-linearly* mapped spectral features captured in the acoustic color (AC) data [2, 7, 8], extracted from the sonar back-scattered from various objects, display unique features providing superior discrimination between different classes of detected objects when compared to the standard features. In this dissertation we present new classification results using several forms of a kernelized MSC, we first explore a solution to fully kernelizing the modified MSC from [2] and then we develop several variants of a more efficient solution that leverages a linearized kernel embedding (LKE) with the modified MSC [4]. We then present an incremental version of the second form. These classifier systems are applied to the TREX13 and PondEX09-10 real sonar datasets to test the generalization ability of the classifiers whose baseline training is performed on datasets generated by a fast ray model (FRM), also known as the Target in environment response (TIER) model [8, 9]. In the incremental cases, a very limited number of labeled samples are utilized to augment the signal models when moving into a new operating environment.

The work presented here will develop a thorough analysis of the usage of standard (linear) and kernelized subspace matching for the problem of underwater UXO classification and some of the benefits and pitfalls of current dictionary learning methods, e.g. [4, 19–21], when applied to this difficult problem. Test results on one *synthetic* and two different *real* low frequency sonar datasets are presented. The first dataset contains all objects for which FRM models exist from the TREX13 exercise [22] and the corresponding FRM generated data for

this controlled low frequency (3-30 kHz) sonar experiment. The second dataset, from the PondEX09-10 collections [10], features several of the same objects as the former and was collected in a similar fixed range and fixed motion controlled experiment with significantly reduced clutter and vegetation when compared to the TREX collection. Classification results and analysis for each of the experiments are provided in terms of performance metrics such as receiver operating characteristic (ROC) curve and confusion matrices.

Additionally, this work compares other state of the art approaches to lifelong learning and their effectiveness for underwater UXO classification. In these discussions, comparisons will be made with the incremental kernelized MSC developed in this work. Furthermore, insights from the kenerlized MSC approach lead to several suggested improvements to (a) fully kernelized K-SVD learning using Kernel K-SVD [21]. (b) the incremental update procedure for the linearized kernel dictionary learning (LKDL) method; (c) importance sampling for the Nyström Approximation used in the LKE modified MSC, and (d) a discriminative kernel learning method allowing for generation of custom kernel functions tailored for *M-class* discrimination.

1.4 Organization of the Dissertation

This dissertation is organized as follows: In Chapter 2 we begin with an overview of the FRM, a physical model which characterizes an objects scattering behavior as well as the water sediment interactions of an underwater object. An overview of the acoustic color (AC) feature generation process is also given for the synthetic and real datasets. In this chapter, a list of the AC samples from objects used in testing and training is also given. Chapter 3 reviews the kernel K-SVD from [21,23] and discusses the solution from [21] to fully kernelizing a sparse MSC that is based on K-SVD dictionaries [19, 24]. In Chapter 4, an alternative to the full kernel solution for dictionary learning is presented which uses important samples to first approximate the full kernel matrix via the Nyström approximation [3, 5], and then embeds all samples into the subspace spanned by the important samples when they are

mapped to the latent feature space. Chapter 5 first reviews common sampling approaches to Nyström method and then presents a method for recursively selecting important samples based on their approximate ridge leverage scores [5, 25] which provide the strongest known bounds for approximation of the kernel matrix via Nyström method, a critical building block in the *incremental* linearized kernel embedding (LKE) modified MSC to come in Chapter 6. The methods from Chapter 4 are then extended in Chapter 6 to consider a scenario where new important samples are discovered and we would like to incorporate them into the model without retraining from scratch. Chapter 7 presents data-driven approach to learning a kernel function by optimizing the weights of a conformal kernel function with respect to the Fisher scalar [15, 26]. Additionally, an M-Ary extension of this method is developed in this chapter. Lastly, a multi-kernel approach is presented that uses a convex combination of kernel functions rather than a conformal kernel and which seeks to optimize the same Fisher scalar objective function. In Chapter 8 we present a selection of classification results on underwater UXO classification datasets utilizing the methods reviewed in the prior chapters. Finally, in Chapter 9 we give concluding remarks on this work and suggestions for future developments.

CHAPTER 2

FAST RAY MODEL AND SONAR DATASETS

2.1 Introduction

For a classifier system to perform reliably, the system must be trained on samples that are representative of different object classes. In many practical situations, the acquisition of a complete and exhaustive, labeled, training sample library is nearly impossible to achieve. Furthermore, training biases that are introduced from specific properties of a sonar hydrophone element or the environment from which training samples were collected can often lead to poor generalization capabilities in sonar classification systems. As an alternative, several physical models seek to emulate the interactions of a solid object reflecting sound pressure underwater in order to predict what target responses we might expect in conditions we have yet to encounter. These models can be utilized in order to provide some indications of what response we can expect to receive from objects of a known class, with certain shape and composition, in different bottom conditions.

In this work, we utilize an empirically validated acoustic scattering model which can quickly and reliably generate synthetic, yet realistic, sonar datasets with desired environmental conditions and sonar path geometry. This chapter begins by reviewing the fast ray model (FRM) [27] utilized in this work in Section 2.2. The four-path model and its underlying parameters are also given. We also discuss how the FRM data are used to generate Acoustic Color (AC) features for classification. An overview of the process used for creating AC plots for the real sonar datasets is also given. Section 2.3 provides descriptions of the real sonar datasets used for testing of the trained MSC classifier, namely the TREX13 and PondEX09-10 datasets, and their collection experiments. Section 2.4 gives the concluding remarks and the benefits and drawbacks of the FRM for data generation.

2.2 Fast Ray Modeling of the Acoustic Response from Munitions

In order to construct template signals needed to reliably represent the various UXOs in our classification system, the work of [8, 10, 22, 27–29] on fast ray modeling of scattering from objects at a water-sediment interface has been utilized. The scattering model allows for monostatic synthetic aperture sonar (SAS) data sets to be simulated via a fast ray model that combines an acoustic ray approximation for propagation in a fluid-filled halfspace with scattering from a target in a number of conditions and media. This fast ray modeling is beneficial for generating large data sets for dictionary matrix construction. In this section, we will discuss this scattering model. Under typical operation for a short-range SAS platform,

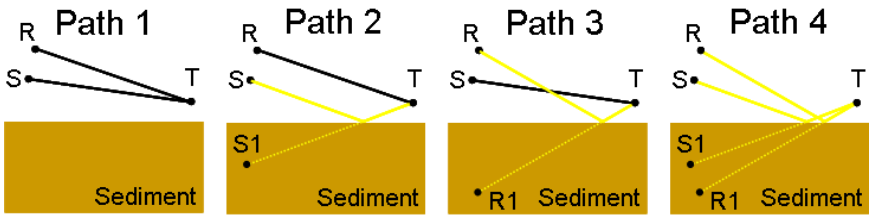


Figure 2.1: Four Ray Path Model.

air-water scattering paths can be ignored, because paths that interact with the air-water interface are either removed by time-gating the received signals or are naturally suppressed by the directivity of the source and receiver. In addition, the separation distance between the actual source and receiver is much smaller than the distance between the interface and the target, so the source and receiver can be considered to be co-located. Under these conditions, only the four ray paths shown in Figure 2.1 contribute to the scattered pressure. The actual source, receiver, and target are denoted by S, R, and T, while $S1$ and $R1$ are image source and receivers. In this figure, Path 1 is a direct reflection path. Paths 2 and 3 interact with the sediment once and scatter from the target in a bistatic direction, and Path 4 is a back-scattering path with two bottom interactions. In this model, the source, receiver, and target are located at $\mathbf{r}_s$, $\mathbf{r}_r$, and $\mathbf{r}_t$, and an image source is located at $\mathbf{r}_{s_i}$ with an

image receiver at $\mathbf{r}_{r_i}$, where these location vectors contain the 3-D position coordinates. To distinguish Path 2 and Path 3, the source and receiver are shown at distinct locations; and with our assumption of co-located source and receiver, Paths 2 and 3 are reciprocal and Paths 1 and 4 are backscattered. With the specification of a source and receiver, the scattering from a target has been reduced to a superposition of 4 free-field scattering problems. Under operational conditions, the distance associated with each path satisfies $d \gg \lambda$ where λ is the wavelength of the pressure. The scattered steady-state pressure can then be written as

$$p_s = p_0 A\left(\mathbf{k}_s, \mathbf{k}_i, \omega\right) \frac{\exp(ikr)}{r} \tag{2.1}$$

where p_0 is the amplitude of the incident pressure, r is the range from a field point to the target, A is the scattering amplitude, $\exp(ikr)/r$ is a spherically diverging wave, $k = 2\pi/\lambda$ is the wavenumber, ω is the angular frequency, and $\mathbf{k}_i$ and $\mathbf{k}_s$ are the unit vectors associated with the direction of the incident and scattered fields, respectively. The scattering amplitude contains useful information concerning the material properties of the target and the directionality of the scattered field. The scattering amplitudes can be determined from analytic solutions to scattering problems (e.g., scattering from a spherical target), direct measurements from actual targets, or numerical simulations e.g., a finite element (FE) model for a given target [27].

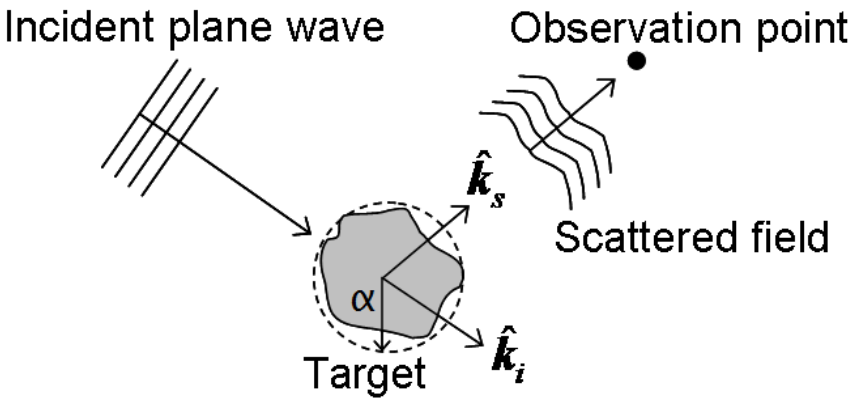


Figure 2.2: Free-field scattering assumes a portion of an incident wave is scattered to a distant target.

Combining the ray model paradigm with free-field scattering as given in (2.1), the spectrum of the total scattered pressure can be written as

$$P(\omega) = \left(\frac{A_1(\omega)}{d_1 d_2} e^{i\omega t_1} + \frac{V(\theta_g) A_2(\omega)}{d_2 d_3} e^{i\omega t_2} + \frac{V(\theta_g) A_3(\omega)}{d_1 d_4} e^{i\omega t_3} + \frac{V^2(\theta_g) A_4(\omega)}{d_3 d_4} e^{i\omega t_4} \right) r_0 P_{src}(\omega) \tag{2.2}$$

with $d_1 = |\mathbf{r}_s - \mathbf{r}_t|$, $d_2 = |\mathbf{r}_t - \mathbf{r}_r|$, $d_3 = |\mathbf{r}_{s_i} - \mathbf{r}_t|$, and $d_4 = |\mathbf{r}_t - \mathbf{r}_{r_i}|$. The time delays are then $t_1 = (d_1 + d_2)/c_1$, $t_2 = (d_2 + d_3)/c_1$, $t_3 = (d_1 + d_4)/c_1$, and $t_4 = (d_3 + d_4)/c_1$; with c_1 being the speed of sound in water. The pressure spectrum $P_{src}(\omega)$ represents the frequency spectrum of the transmitted wave packet from the source, and $r_0 = 1$ m is a reference distance. The scattering amplitudes $A_k(\omega)$ in (2.2) also depend on the locations of the sources, receivers, and target. Note the indices of A_k correspond to the path enumeration depicted in Figure 2.1. The reflection coefficient at the water-sediment interface, $V(\theta_g)$, is a function of the grazing angle θ_g, and is defined as follows

$$V(\theta_g) = \frac{\rho \sin(\theta_g) - (\kappa^2 - \cos^2(\theta_g))^{1/2}}{\rho \sin(\theta_g) + (\kappa^2 - \cos^2(\theta_g))^{1/2}} \tag{2.3}$$

where $\rho = \rho_2/\rho_1$ and $\kappa = (1 + j\delta)/\nu$ with $\nu = c_2/c_1$. Here ρ_1 is the density of water. The density, sound speed, and loss parameter for the sediment are ρ_2, c_2 and δ, respectively. An inverse Fourier transform of $P(\omega)$ thus gives a generated sonar signal that includes the four primary acoustic propagations for a target near an interface. The four paths represented in this model take into account the interaction of the sonar with the target and the environment (seafloor). If one were to throw out the three paths that reflect from the seafloor and only keep the first path, then one would have "free-field scattering" from the target only. The term free-field scattering implies that the target is so far from any boundary that it can be thought of as suspended in an infinite volume of water.

2.2.1 Acoustic Color for Synthesized Data

In order to produce AC features for the synthetically generated data to train the MSC classifier [1, 2, 6], raw sonar returns generated by the fast ray model must first be pre-processed. Generation of AC features amounts to forming the intensity magnitude of the returned spectral power over the entire range of aspect angles that are modeled in either linear path SAS (LSAS) or circular path SAS (CSAS) runs. This is accomplished by the following procedure: First, a FE model [29] is implemented to produce scattering amplitude information for the intended target. These scattering amplitudes are modeled for acoustic transmissions and returns in the low frequency range of 1-30 kHz. Next, the half-space model including the four described ray paths in (2.2) is utilized to generate a raw sonar return data set by generating the modeled returns of a target using the inverse FFT of (2.2) over a pre-generated coordinate set representing the various positions along a circular or linear path making soundings. A copy of the transmitted pulse used during data collection was used, which provides spectral information of an object's backscatter in the desired frequency range. Next, these raw soundings were pulse-compressed with the original transmit signal. Finally, the magnitude value of the FFT of the pulse-compressed data is computed and the result is windowed to $3 - 30$ kHz to remove the unused frequency portions and isolate the frequency range of interest. As an arbitrary trajectory of a SAS platform can be modeled with the FRM [27], the model was used to generate synthetic LSAS sonar data with a target located at the center of the LSAS path for a set of ranges and objects collected in all real datasets used in this research. Model parameters such as sonar interface elevation and water conditions were set to match those measured during data collection experiments. This was repeated for all objects present in the tested datasets. Lists of objects used in training and their characteristics are given in Table 2.1. An example of these AC features generated for a 21-m long simulated LSAS run at a range of 10 m and with an interface elevation of 3.8 m is given in Figure 2.3 (c) for an aluminum UXO object. Plots (a) and (b) of this figure demonstrate a real data AC and FEM modeled AC for the same object. Examples of the

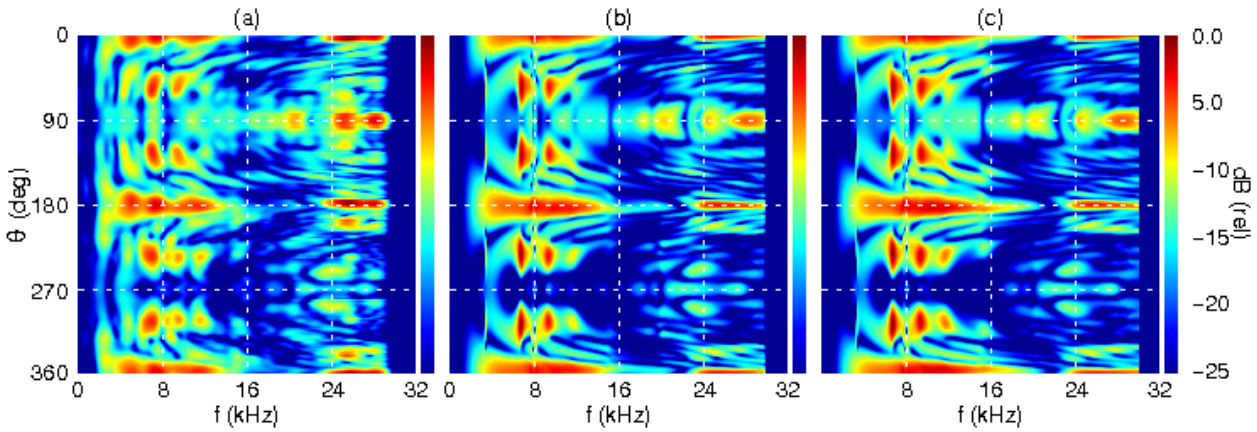
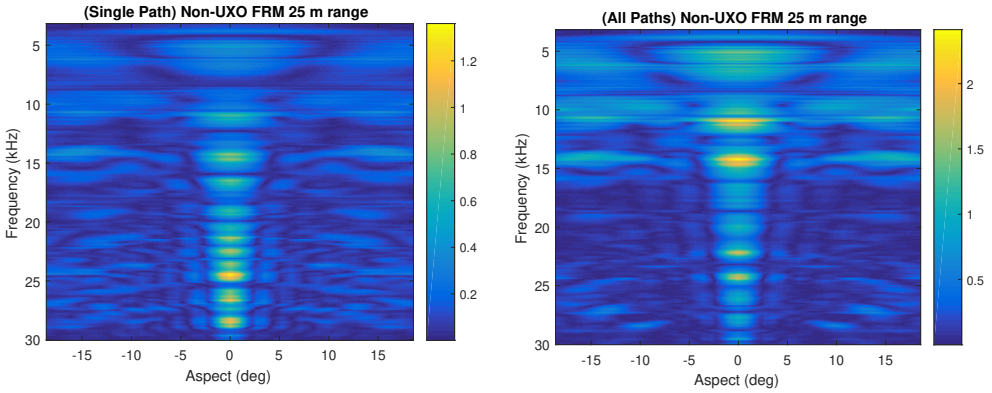


Figure 2.3: AC data of a bullet-shaped aluminum UXO replica at 10 m range generated via (a) Data collected during PondEX10 (SERDP MR-1665). (b) FEM and Kirchoff-Helmholtz integral. (c) FRM with scattering form function derived from the scattered pressure computed in (b).

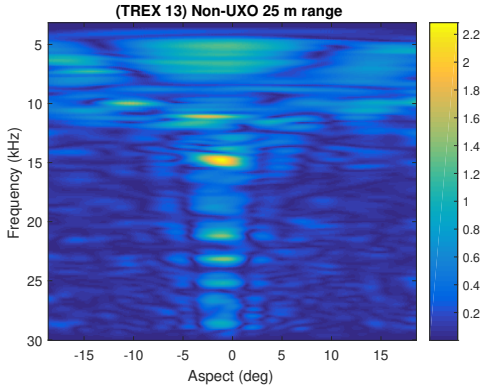
acoustic color matrices for a non-UXO object generated for a 40 m long simulated LSAS run at a target range of 25 m and with an interface elevation of 3.8 m are given in Figures 2.4 (a) and (b) for a single path, i.e. using only the first term in (2.2), and for all paths, i.e. using all four terms in (2.2), respectively. For this particular run configuration, the target was at 0° orientation (i.e. broadside view of the target). Note that although the maximum angle for this target range is ±38.7°, the portion that contains nonzero signal is only ≈ ±20°. Figure 2.4 (c), on the other hand, shows the acoustic color matrix for the same non-UXO object under the same conditions in the experimental TREX13 data set. One can clearly see a closer match between the aspect-dependent spectral features in Figure 2.4 (b) and the corresponding ones in Figure 2.4 (c), especially in the frequency range of 10 to 20 kHz when compared with those of the single path case in Figure 2.4 (a).

AC features generated for template signals via the described scattering model were then utilized as training data for our various MSC classifiers. AC feature vectors were generated for all proud objects and ranges that were encountered in the actual test datasets. The major benefit of utilizing the ray model developed in [27,29] is that, after free-field scattering amplitudes for a desired object are collected or modeled via FE methods, the regeneration of the ray model simulating various aspects and orientations is far simpler and faster than



(a) Fast Ray Model-Single Path

(b) Fast Ray Model-All Paths



(c) TREX13 Data

Figure 2.4: Acoustic Color Plots for Non-UXO Object using LSAS-Model Generated vs. TREX Data.

re-running these variations with the FE method [29].

The validated performance of the FRM can be visualized in Figure 2.4. Figure 2.4(c) displays the real AC data of an aluminum pipe collected from the TREX13 exercise [22], while Figures 2.4(a) and 2.4(b) display the result of the FRM for the same objects response using one Path (direct reflection) and 4 paths respectively. Comparing the AC plots in Figures 2.3(b)-(c), one can clearly observe that the AC plot in Figure 2.3(b), generated via the FRM [27], captures the essential AC features of the real sonar data of Figure 2.4(c) for the same object collected in TREX13 experiment. A great deal of spectral information for object discernment seems to be present in these AC plots. We shall present in Section 2.3.3

the AC data generation for real sonar datasets.

2.2.2 Synthesized Training Data Set

Table 2.1: Fast Ray Model Combined Training Data Set.

No.	Class	Object Description	Ranges
1	non-UXO	2 ft Aluminum Cylinder	10 m
2	non-UXO	3 ft Aluminum Cylinder	$(10), 30, 35, 40$ m
3	non-UXO	2 ft Aluminum Pipe Section	$(10), 15, 25, 30$ m
4	UXO	100 mm Aluminum Rocket Round	$10, 15, 30$ m
5	UXO	100 mm Solid Steel Rocket Round	$10, 15, 25, 30$ m
6	UXO	105 mm Bullet (Air fill)	$15, 20, 25$ m
7	UXO	105 mm Bullet (H_2O fill)	$15, 20, 35$ m
8	UXO	155 mm Howitzer w/ Cap (Air fill)	$10^*, 15, 20, 35$ m
9	UXO	155 mm Howitzer w/ Cap (H_2O fill)	$15, 25, 30$ m
10	UXO	155 mm Howitzer no Cap	$25, 30, 40$ m

As stated before, a specific objective of this work and our previous work in [1] is to determine if model-generated AC features, generated for a wide variety of UXO and non-UXO objects with known geometrical and physical characteristics at different sonar range and orientation, can be used to successfully construct dictionary matrices for an MSC-based classification system which provide good performance on real datasets. Similar to the real datasets described in Sections 2.3.1 - 2.3.2, object orientations were created that ranged from -80° to $+80^\circ$ in 20° increments, with 0° orientation corresponding to broadside view of the object and 90° corresponding to an end-on view of objects. In order to have comparable features to the TREX13 dataset, these features were truncated to include only the frequency components between $3-30$ kHz. For each object in the real sonar data collection experiments, the AC features generated were decimated along the frequency dimension to have $N = 271$ frequency bins spanning the 3-30 kHz frequency range corresponding to approximately 100 Hz separation of frequency bins and along the aspect dimension to have 0.5° aspect resolution (i.e. 721 aspect vectors $\propto 360^\circ$). This database of AC features was created to match aspect resolution and frequency resolution of those generated for the real sonar data. Table 2.1 gives the list of all the objects and their range information for which synthesized data were

generated to match those in the TREX13 experiment. This synthesized training set is given with range information in order to provide a reference for the FRM types used for the TREX13 and PondEX09-10 experiments performed with the developed classifiers.

2.3 Sonar Data Descriptions

In this section we overview the sonar datasets and the UXO and non-UXO objects that were used for testing the developed classifiers. The object characteristics and their ranges from the sonar platform are also provided. After the test datasets are overviewed, a summary of the training objects synthesized via the previously described FRM model is given. In the following, we will describe the TREX and PondEX real sonar datasets. Figure 2.5

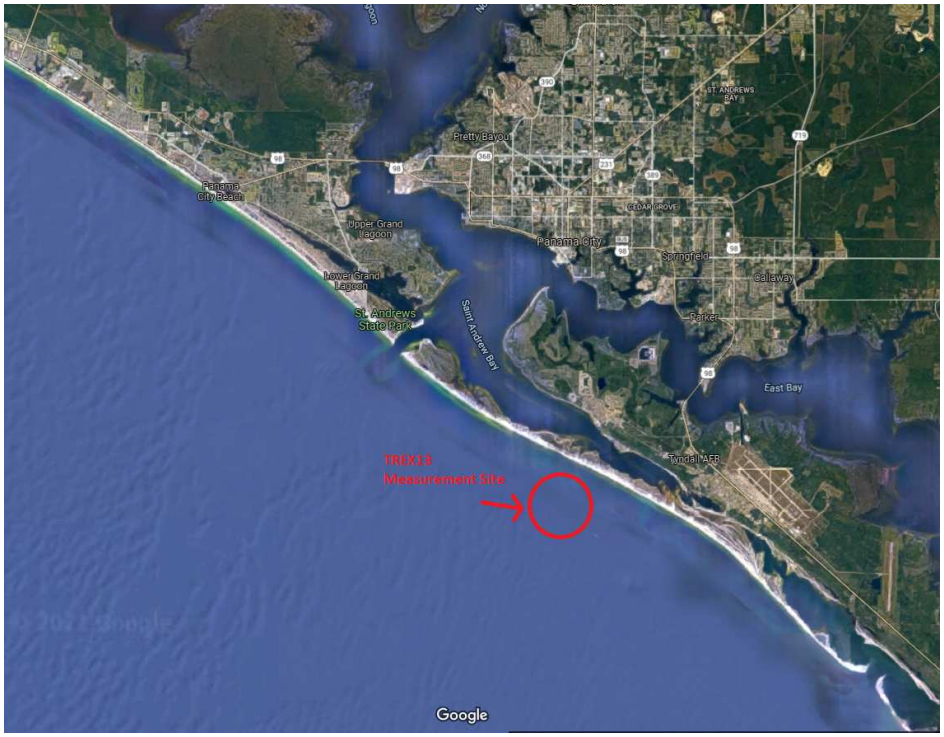


Figure 2.5: TREX 2013 Collection site.

depicts the approximate location of the TREX13 collection experiment. The collection was conducted in the Gulf of Mexico off the coast of Panama City, Florida where the water depth is approximately 20 m.

2.3.1 TREX13 Data Descriptions

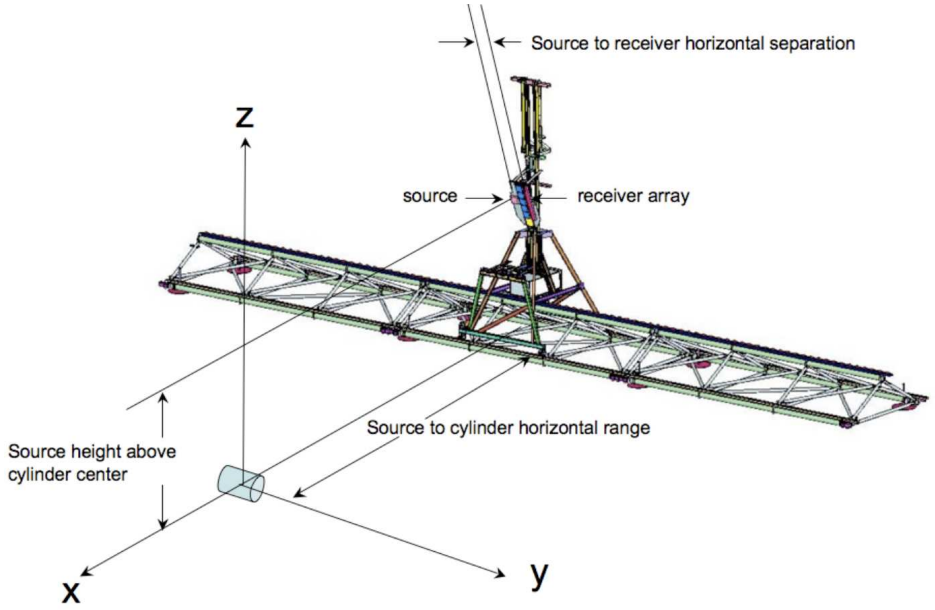


Figure 2.6: APL-UW's Rail Based Sonar Collection System.

The experimental setup for TREX13 was similar to PondEx10 experiment described in [10] and [30]. As mentioned before, for TREX13, the target field was located in the Gulf of Mexico off the coast of Panama City, Florida in waters with an approximate depth of 20 m. The target field contained several objects with varying shapes, sizes, and compositions, all of which were located between 10 to 40 m horizontally from the rail system and are proud on the sandy bottom. Table 2.2 gives the list and some properties of these objects, together with range information for each object. The rail that the sonar system was mounted on was fixed to minimize platform motion as the sonar tower traversed along its track. The length of the rail during TREX13 was approximately 40 m (42 m total with a 40 m linear path for collection). Figure 2.6 depicts the rail system utilized in TREX13 data collection. A similar rail-based system was used in the PondEX09-10 collection discussed in the next section. The sonar transmit signal was a 6 msec linear frequency modulated (LFM) pulse over 3-30 kHz with a 10% taper between the leading and trailing edges to minimize ringing effects in the transmitted signals. Sonar backscatter was received by a 6-element linear array that was

Table 2.2: TREX13 available proud targets used for testing.

No.	Class	**APL-UW #**	**FRM Name**	Ranges
1	non-UXO	Target 17	alcyl2ft	10 m
2	non-UXO	Target 7	alcyl3ft	30, 35, 40 m
3	non-UXO	Target 16	alpipe	15, 25, 30 m
4	UXO	Target 20	aluxo	10, 15, 30, 40 m
5	UXO	Target 21	ssuxo	10, 15, 25, 30 m
6	UXO	Target 25	bullet_105mm_air	15, 20, 25 m
7	UXO	Target 29	bullet_105mm_h2o	15, 25, 30 m
8	UXO	Target 9	howitzer_cap_air	10, 15, 20, 35 m
9	UXO	Target 28	howitzer_cap_h2o	15, 25, 30, 40 m
10	UXO	Target 8	howitzer_nocap	10, 25, 30, 40 m

approximately normal to the seafloor. Figure 2.7 demonstrates the approximate layout of the target field for the TREX13 data collection study. Targets were placed at range bins A-H corresponding to ranges from 5 m to 40 m in 5 m increments. Each range bin had between 3 and 7 along-track positions where it could be located. Specific placements can be found in [22]. This layout was typical for the study and other field configurations used similar placements. All targets from the TREX13 set were proud on the sandy seafloor. The sonar data sets used in this study were collected during ten runs through the target field. Each run differed in the orientation of all the objects, with each object having the same orientation for a given run. The object orientations varied from $-80°$ to $+80°$ in $20°$ increments. However, each run of data consisted of approximately 1600 pings in which the sonar platform moved along the fixed rail in increments of 0.025 m, transmitting and receiving once for each sonar position. The data was sampled at 100 kHz and the sonar platform was tilted at either a $10°$ or $20°$ grazing angle depending on range to the targets being observed.

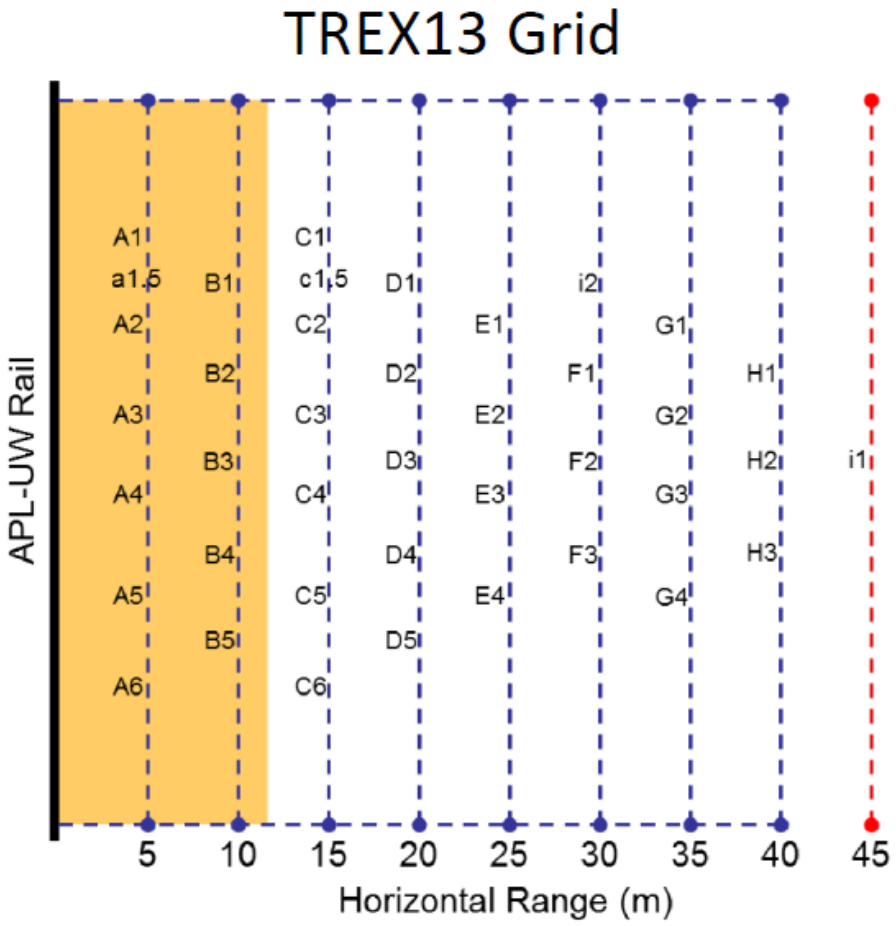


Figure 2.7: Typical target layout of TREX13.

2.3.2 PondEX Data Description

Figure 2.8 shows the layout of the PondEX10 experiment which also matches the setup that was used in the earlier PondEX09 experiment including the relative locations of the rail-mounted sonar system and the objects in the target field. The 21 m rail the sonar system is mounted on is fixed to eliminate platform motion as the sonar interface moves along its track. The sonar transmit signal is a 6 msec linear frequency modulated (LFM) pulse over 0.5-30 kHz with a 10% taper between the leading and trailing edges to minimize ringing in the transmitted signals. Sonar backscatter is received with $L = 6$ hydrophone elements that are arranged in a linear array approximately normal to the seafloor. For the formation of AC data, only the 3rd hydrophone element data was used. As can be seen from

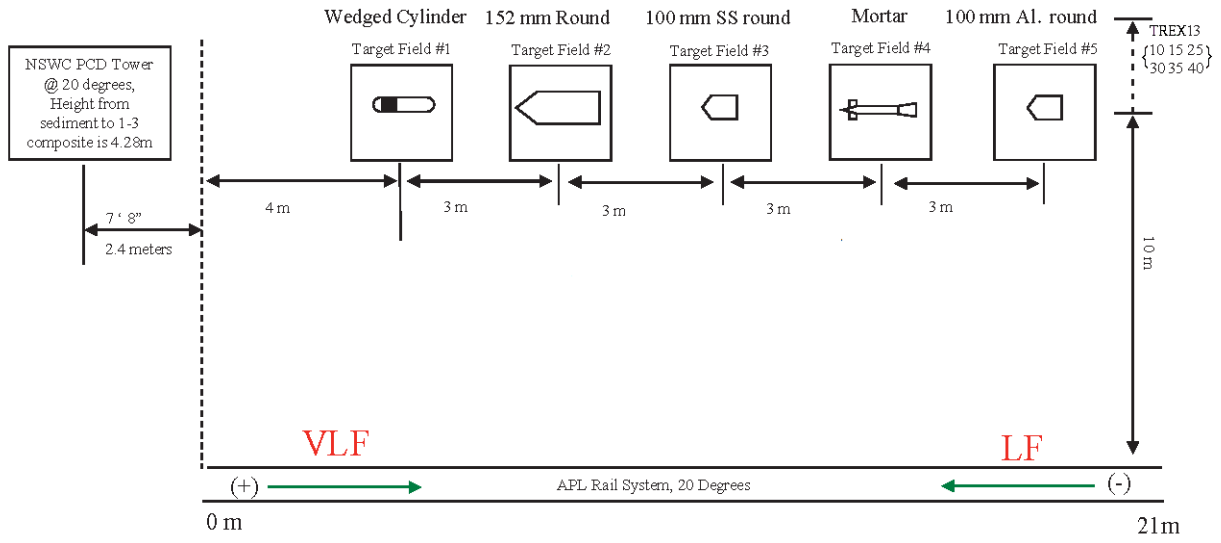


Figure 2.8: Layout of the Field for PondEx10 Data Collection.

Figure 2.8, for the PondEX experiments, the target field contained several objects at a time with varying shapes, sizes, and compositions, all of which were located approximately 10 m horizontally from the rail system and are proud on the sandy bottom. Table 2.3 enumerates the tested objects for which FRM models exist. This data set was expected to provide more of a challenge because there is the potential for the backscattered frequency structure of a man-made non-UXO object such as an aluminum pipe to be similar to that of aluminum UXO due to the regularity and similarity of the shapes and materials.

The sonar data sets from this study were collected during ten runs through the target field with the target field for each run containing the listed objects. Each run differs in the orientation of all the objects, with each object having the same orientation for a given run. Ten total object orientations were used, ranging from $-80°$ to $+80°$ in $20°$ increments, with $0°$ orientation corresponding to broadside view of the object and $90°$ corresponding to an end-on view of objects. Each run consists of 769 pings in which the sonar platform moved along the fixed rail in increments of 0.025 m, transmitting and receiving once for each sonar position. The data was sampled at 1 MHz and the sonar platform was tilted at a fixed $20°$ grazing angle for all runs. Since the useful spectral information in the collected data

Table 2.3: PondEX09-10 available proud targets used for testing.

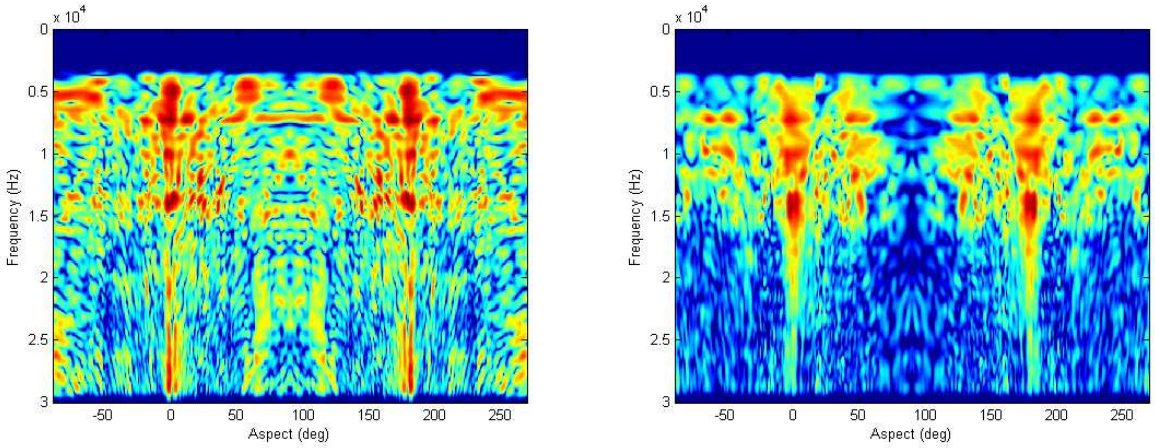
No.	Class	**APL-UW #**	**FRM Name**	Ranges
1	non-UXO	Target 17	alcyl2ft	10 m
3	non-UXO	Target 16	alpipe	10 m
4	UXO	Target 20	aluxo	10 m
5	UXO	Target 21	ssuxo	10 m
8	UXO	Target 9	howitzer_cap_air	10 m

has a Nyquist frequency well below the sampled rate, the 1 MHz data was down-sampled in time by a factor of 10 resulting in data sampled at 100 kHz. Then after AC data was generated, the frequency dimension has $N = 301$ bins spanning the 0-30 kHz frequency range corresponding to approximately 100 Hz in each frequency bin. In order to have comparable features to the TREX13 dataset, these features were truncated to include only the frequency components between $3 - 30$ kHz resulting in features with dimension $N = 271$.

As the MSC classifier is to be implemented on AC data, testing was performed on AC's generated from filtered data from these experiments [10]. Using the method described in Section 2.3.3, AC Template testing data was formed by composing the useful aspects from multiple linear SAS runs with the field-centered target taking various rotational poses.

2.3.3 AC for Real Sonar Data

The procedure for generating AC features for real raw sonar data experiments is similar to the procedure performed on the synthesized sonar data described in Section 2.2.1. Raw sonar data time series collected from these real sonar datasets are first pulse compressed with the transmit signal that was used in the data collection experiment and the result is further processed to remove returns from the neighboring objects to isolate the object of interest. This processing utilizes a reversible SAS imaging process, a spatial filtering process using a 2-D Tukey window, and a pseudo-inverse filtering [31]. This inverse filter maps the SAS image back to the pulse-compressed version that has less interface scattering noise. These filtered pulse-compressed signals are then transformed to the frequency domain via FFT



(a) Air-Filled Howitzer (object 8)

(b) Aluminum UXO Shell (object 4)

Figure 2.9: AC Features for two UXO objects in TREX13 dataset

and the magnitude of the FFT was utilized as the testing feature vectors. This process is repeated for all aspects of a given LSAS run and the amplitude spectrum is then generated and plotted to display AC for each object. Examples of these AC plots for two real UXO objects in the TREX13 test set, the air-filled Howitzer with a cap and the aluminum UXO, are shown in Figures 2.9(a) and 2.9(b). From these two figures, it is easy to verify that the frequency response of the air filled Howitzer and aluminum UXO object types are distinctly different despite similarities in shape and object fill. Furthermore, when comparing to the Aluminum UXO plot of Figure 2.3 (a), some discrepancies and interference appears to be present in the TREX13 observation given in Figure 2.9(b).

2.4 Conclusion

In this chapter we first introduced the FRM [27] as a framework for modeling object scattering in a water sediment interface. Using the ray model, target scattering is reduced to a convolution of a free-field scattering amplitude and an incident acoustic field at the target location. A simulated or measured scattered free-field pressure from a complicated target can be reduced to a complex scattering, and this is then used within the ray model via interpolation. Using this model, we explained how the four-path wave guide model could be

used along with pre-saved complex scattering coefficients, found via FEM methods, to rapidly generate a customizable and complete library of synthetic training samples. This section was then followed by an explanation of the AC data features used for target classification in this work and the procedure for generating AC features for synthesized and real sonar datasets. Examples of a patch of AC vectors generated from a synthesized LSAS run were also given in this chapter. Finally, overview of the testing and training datasets to be used by the various classifiers were given also. Using the model-generated data in conjunction with the subspace learning methods presented in the following chapter, we attempt to learn a robust signal model for different object classes. The two testing datasets, namely, TREX13 and PondEX09-10, are then used to determine the effectiveness of different incremental learning methods developed in this work

CHAPTER 3

KERNELIZED DICTIONARY LEARNING AND KERNELIZATION OF SPARSE MSC

3.1 Introduction

In [1] we investigated a fully kernelized matched subspace classifier (KMSC) system, which utilized kernelized least squares (LS) estimates for the purposes of discriminating underwater targets, this technique was originally published in [32]. This approach, has many appealing attributes and displayed competitive results with the traditional matched subspace classifier (MSC) [6] and the sparse reconstruction variants of MSC that utilized both K-SVD and LP-KSVD dictionary learning [19, 33]. A natural question then arises when comparing the benefits of the non-linear mappings associated with kernel methods to sparse or compressed sensing variants of the traditional subspace matching classifier. That is, is there any way that we can combine the benefits of both sparse estimates and kernelized learning solutions? We address exactly this question in this chapter and discuss a solution to this question by integrating the kernelized dictionary learning of [21, 23] into the MSC framework in the same manner with which we previously integrated K-SVD based subspace matrices and OMP estimates into the traditional (linear) MSC [1, 2].

As we discussed above, investigations into kernelizing the modified MSC classifier from [2] have revealed the need for a dictionary learning strategy that simultaneously learns atoms which perform good discrimination when used in sparse estimates. Additionally it should provide such learning for samples in the reproducing kernel Hilbert space (RKHS) associated with the implicit mapping of a given kernel function.

3.1.1 Chapter Organization

To address the problem of extending K-SVD learning to a kernelized form, the method presented in [21] provides a solution that constructs atoms as a linear combination of the training examples in the feature space associated with a given kernel. The theory of this method and the two associated algorithms are presented in this chapter. In this chapter, and throughout the rest of this dissertation, we use "fully kernelized" to distinguish this solution from the empirical kernel maps utilized in Chapters 4 and 6. As we will later see, the fully kernelized solution has some drawbacks, namely space and time requirements, that make it a less-than-ideal solution to combining sparse estimates with kernelized features. In this chapter we begin by describing the kernel K-SVD problem. Then we review a technique for implementing sparse coding to solve the coding phase of the K-SVD learning. We then explain the procedure to perform dictionary updates for the kernel K-SVD. Lastly we discuss how to utilize this kernel K-SVD learning to construct a fully kernelized modified MSC.

3.2 Kernel K-SVD Method

As with the original K-SVD, the kernel K-SVD learning method [21, 23] involves two stages: sparse coding and dictionary update. The authors in [21] assume dictionary $\mathbf{H} = \mathbf{BA}$, where $\mathbf{B}$ is some predefined base dictionary and $\mathbf{A}$ is the atom representation dictionary. The base dictionary $\mathbf{B}$ can be chosen such that it incorporates some prior knowledge about the data. This model provides adaptivity via modification of the matrix $\mathbf{A}$. Let $\Phi(\mathbf{Z})$ denote the matrix whose columns are obtained by embedding the input signals $\mathbf{Z} = [\mathbf{z}_1, ..., \mathbf{z}_N] \in \mathbb{R}^{d \times N}$ into some feature space using the mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$, where usually $d \ll D$. That is, $\Phi(\mathbf{Z}) = [\Phi(\mathbf{z}_1), ..., \Phi(\mathbf{z}_N)] \in \mathbb{R}^{D \times N}$. Denote learned dictionary in feature space as $\Phi(\mathbf{H}) \in \mathbb{R}^{D \times K}$. Since dictionary atoms lie within the subspace spanned by the input data, we can write $\Phi(\mathbf{H}) = \Phi(\mathbf{Z})\mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{N \times K}$ is the atom representation dictionary and $\Phi(\mathbf{Z})$ is the mapped training dataset. That is, $\Phi(\mathbf{Z})$ acts as the basis with which we construct our atoms through linear combinations described by the columns of $\mathbf{A}$. Given

$$\mathbf{Z} \in \mathbb{R}^{d\times N},\ \Phi(\mathbf{Z}) \in \mathbb{R}^{D\times N},\ \mathbf{A} \in \mathbb{R}^{N\times K},\ \Phi(\mathbf{H}) \in \mathbb{R}^{D\times K},\ \boldsymbol{\Theta} \in \mathbb{R}^{K\times N}$$

The objective of the kernel K-SVD is to find the best dictionary $\Phi(\mathbf{H})$ via $\mathbf{A}$ to represent the data in the feature space $\{\Phi(\mathbf{z}_i)\}_{i=1}^N$ as sparse compositions by solving the following optimization problem [21, 23]

$$\underset{\mathbf{A},\boldsymbol{\Theta}}{\operatorname{argmin}} ||\Phi(\mathbf{Z}) - \Phi(\mathbf{Z})\mathbf{A}\boldsymbol{\Theta}||_F^2 \ s.t.\ ||\boldsymbol{\theta}_i||_0 \leq \tau, \forall i \tag{3.1}$$

The objective function in (3.1) can be rewritten in terms of the kernel matrix $\mathbb{K}$, i.e.

$$||\Phi(\mathbf{Z}) - \Phi(\mathbf{Z})\mathbf{A}\boldsymbol{\Theta}||_F^2 = \operatorname{tr}((\mathbf{I} - \mathbf{A}\boldsymbol{\Theta})^\top \mathbb{K}(\mathbf{Z},\mathbf{Z})(\mathbf{I} - \mathbf{A}\boldsymbol{\Theta})) \tag{3.2}$$

where $\mathbb{K}(\mathbf{Z},\mathbf{Z}) \in \mathbb{R}^{N\times N}$ is a positive semi-definite (PSD) matrix whose elements $[\mathbb{K}(\mathbf{Z},\mathbf{Z})]_{ij} = [\langle \Phi(\mathbf{Z}), \Phi(\mathbf{Z})\rangle]_{ij} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$ satisfy the Mercer condition [34].

3.3 Sparse Coding

Since K-SVD [19,24] uses sparse coding, next we review a kernelized version of the Orthogonal Matching Pursuit (OMP) [24, 35]. The kernel OMP (KOMP) procedure [21] assumes the existence of a fixed dictionary through which individual samples can be represented. For our description of $\mathbf{H}$ this is equivalent to assuming that $\mathbf{A}$ is fixed. We seek sparse codes contained in the matrix $\boldsymbol{\Theta}$. Note that, the penalty term in (3.1) can be re-written as

$$||\Phi(\mathbf{Z}) - \Phi(\mathbf{Z})\mathbf{A}\boldsymbol{\Theta}||_F^2 = \sum_{i=1}^{N} ||\Phi(\mathbf{z}_i) - \Phi(\mathbf{Z})\mathbf{A}\boldsymbol{\theta}_i||_2^2. \tag{3.3}$$

Since each squared error term in this sum depends on only one $\boldsymbol{\theta}_i$ and each term is convex in its respective $\boldsymbol{\theta}_i$, the problem in (3.1) can be reformulated as solving N different problems of the following form

$$\min_{\boldsymbol{\theta}_i} ||\Phi(\mathbf{z}_i) - \Phi(\mathbf{Z})\mathbf{A}\boldsymbol{\theta}_i||_2^2 \ s.t.\ ||\boldsymbol{\theta}_i||_0 \leq \tau \tag{3.3}$$

for $i = 1, ..., N$. We next show how the well-known OMP algorithm can be generalized using kernels to solve (3.3).

Given signal $\mathbf{z} \in \mathbb{R}^d$ and kernel dictionary represented via $\mathbf{A}$, we seek a sparse combinations of dictionary atoms that approximate the signal in the feature space:

$$\Phi(\mathbf{z}) = \Phi(\mathbf{Z})\hat{\mathbf{z}}_s + \mathbf{r}_s$$

Where $\hat{\mathbf{z}}_s \in \mathbb{R}^N$ indicates current estimate of signal $\mathbf{z}$, and $\mathbf{r}_s$ is the current residual. We can represent the inner product between this residual and an individual dictionary atom like so,

$$\begin{aligned}
\mathbf{r}_s^\top(\Phi(\mathbf{Z})\mathbf{a}_i) &= (\Phi(\mathbf{z}) - \Phi(\mathbf{Z})\hat{\mathbf{z}}_s)^\top(\Phi(\mathbf{Z})\mathbf{a}_i) \\
&= \Phi(\mathbf{z})^\top\Phi(\mathbf{Z})\mathbf{a}_i - \hat{\mathbf{z}}_s^\top\Phi(\mathbf{Z})^\top\Phi(\mathbf{Z})\mathbf{a}_i \\
&= (\mathbb{K}(\mathbf{z}, \mathbf{Z}) - \hat{\mathbf{z}}_s^\top\mathbb{K}(\mathbf{Z}, \mathbf{Z}))\mathbf{a}_i
\end{aligned} \tag{3.4}$$

The KOMP algorithm selects new dictionary atom that gives largest projection coefficient in (3.4). Let $\mathbf{A}_S$ indicate the set of dictionary atoms whose indices are from the set S. To project the signal $\Phi(\mathbf{z})$ onto the subspace spanned by the selected dictionary atoms $\Phi(\mathbf{Z})\mathbf{A}_S$. The projection coefficients are simply obtained as follows:

$$\begin{aligned}
\boldsymbol{\theta}_s &= ((\Phi(\mathbf{Z})\mathbf{A}_S)^\top(\Phi(\mathbf{Z})\mathbf{A}_S))^{-1}(\Phi(\mathbf{Z})\mathbf{A}_S)^\top\Phi(\mathbf{z}) \\
&= (\mathbf{A}_S^\top\mathbb{K}(\mathbf{Z}, \mathbf{Z})\mathbf{A}_S)^{-1}(\mathbb{K}(\mathbf{z}, \mathbf{Z})\mathbf{A}_S)^\top
\end{aligned} \tag{3.5}$$

Once the coefficients $\boldsymbol{\theta}_s$ are found, the approximating signals $\hat{\mathbf{z}}_s$ are updated as in step 5 of the procedure in Algorithm 1 below. The procedure is repeated until τ atoms are selected.

Algorithm 1 KOMP Algorithm

Input: Signal $\mathbf{z}$, $\mathbf{A}$, τ

Output Sparse vector $\boldsymbol{\theta} \in \mathbb{R}^K$ satisfying $\boldsymbol{\theta}(S(j)) = \boldsymbol{\theta}_s(j), \forall j \in \{1, ..., \tau\}$

Initialization $s = 0$, $S_0 = \emptyset$, $\boldsymbol{\theta}_0 = \mathbf{0}$, $\hat{\mathbf{z}} = \mathbf{0}$

Procedure:

1: $r_i = (\mathbb{K}(\mathbf{z}, \mathbf{Z}) - \hat{\mathbf{z}}_s^\top \mathbb{K}(\mathbf{Z}, \mathbf{Z}))\mathbf{a}_i, \forall i \notin S_{s-1}$

2: $i_{max} = \text{argmax}_i |r_i|, \forall i \notin S_{s-1}$

3: Update the index set $S = S_{s-1} \cup i_{max}$

4: $\boldsymbol{\theta}_s = (\mathbf{A}_S^\top \mathbb{K}(\mathbf{Z}, \mathbf{Z})\mathbf{A}_S)^{-1}(\mathbb{K}(\mathbf{z}, \mathbf{Z})\mathbf{A}_S)^\top$

5: $\hat{\mathbf{z}}_s = \mathbf{A}_S \boldsymbol{\theta}_s$

6: $s \leftarrow s + 1$; Repeat steps 1-6 τ times

3.4 Dictionary Updating using Kernel K-SVD

In this section we describe the kernel K-SVD dictionary update procedure [21, 23].

Let $\mathbf{a}_k$ and $\boldsymbol{\theta}_T^j$ be the k-th column of $\mathbf{A}$ and the j-th row of $\boldsymbol{\Theta}$, respectively. The cost function $||\Phi(\mathbf{Z}) - \Phi(\mathbf{Z})\mathbf{A}\boldsymbol{\Theta}||_F^2$ in (3.3) can be re-written as:

$$\begin{aligned} ||\Phi(\mathbf{Z}) - \Phi(\mathbf{Z}) \sum_{j=1}^{K} \mathbf{a}_j \boldsymbol{\theta}_T^j||_F^2 &= ||\Phi(\mathbf{Z}) \left(\mathbf{I} - \sum_{j \neq k} \mathbf{a}_j \boldsymbol{\theta}_T^j \right) - \Phi(\mathbf{Z})(\mathbf{a}_k \boldsymbol{\theta}_T^k)||_F^2 \\ &= ||\Phi(\mathbf{Z})\mathbf{E}_k - \Phi(\mathbf{Z})\mathbf{M}_k||_F^2 \end{aligned} \tag{3.6}$$

where, $\mathbf{E}_k = \left(\mathbf{I} - \sum_{j \neq k} \mathbf{a}_j \boldsymbol{\theta}_T^j \right)$, and $\mathbf{M}_k = (\mathbf{a}_k \boldsymbol{\theta}_T^k)$. Error matrix $\mathbf{E}_k$ denotes error between approximated and true signals when removing the k-th atom. Matrix $\mathbf{M}_k$, on the other hand, indicates the contribution of the k-th atom to the estimated signals. We assume that only $(\mathbf{a}_k, \boldsymbol{\theta}_T^k)$ are variables and the rest are fixed, hence $\mathbf{E}_k$ is also constant for each k. Minimization of the above problem is equivalent to finding $(\mathbf{a}_k, \boldsymbol{\theta}_T^k)$ such that the rank-1 matrix $\Phi(\mathbf{Z})\mathbf{M}_k$ best approximates $\Phi(\mathbf{Z})\mathbf{E}_k$. The optimal solution can be obtained via SVD, but this would yield a dense vector $\boldsymbol{\theta}_T^k$, increasing the number of non-zeros in the representation $\boldsymbol{\Theta}$. Furthermore, this matrix may have infinite row dimension making it

intractable.

To minimize the objective function while keeping the sparsities of all the representations fixed, we work only with a subset of columns. Since the columns of $\mathbf{M}_k$ associated with zero value elements of $\boldsymbol{\theta}_T^k$ are zero and don't affect the minimization this means we can shrink $\mathbf{E}_k$ and $\mathbf{M}_k$ by discarding zero columns as is done in the traditional K-SVD [19]. When working with the reduced matrices, only non-zero coefficients in $\boldsymbol{\theta}_T^k$ are allowed to vary to ensure that sparsity is preserved. Define ω_k as the group of indices pointing to examples $\{\Phi(\mathbf{z}_i)\}$ that use the atom $(\Phi(\mathbf{Z})\mathbf{A})_k : \omega_k = \{i|\ 1 \leq i \leq K,\ \boldsymbol{\theta}_T^k(i) \neq 0\}$. Let Ω_k be a matrix of size $N \times |\omega_k|$, with ones on the $(\omega_k(i), i)$-th entries and zeros elsewhere. When multiplied by Ω_k, all zeros in row vector $\boldsymbol{\theta}_T^k$ will be discarded resulting in the row vector $\boldsymbol{\theta}_R^k$ of dimension $|\omega_k|$. The column-reduced matrices are obtained as $\mathbf{E}_k^R = \mathbf{E}_k\Omega_k; \mathbf{M}_k^R = \mathbf{M}_k\Omega_k$. For a more detailed explanation see [7, 21] and Appendix A.

We can now modify the cost function in (3.6) so that its solution has the same support with the original $\boldsymbol{\theta}_T^k$:

$$||\Phi(\mathbf{Z})\mathbf{E}_k^R - \Phi(\mathbf{Z})\mathbf{M}_k^R||_F^2 = ||\Phi(\mathbf{Z})\mathbf{E}_k^R - \Phi(\mathbf{Z})\mathbf{a}_k\boldsymbol{\theta}_R^k||_F^2. \tag{3.7}$$

It may appear that we can find the optimal solution of (3.7) by first decomposing $\Phi(\mathbf{Z})\mathbf{E}_k^R$ using SVD, as

$$\Phi(\mathbf{Z})\mathbf{E}_k^R = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \tag{3.8}$$

and then equating $\Phi(\mathbf{Z})\mathbf{a}_k\boldsymbol{\theta}_R^k$ to the rank-1 matrix corresponding to the largest singular value, $\sigma_1 = \boldsymbol{\Sigma}(1,1)$. That is

$$\Phi(\mathbf{Z})\mathbf{a}_k\boldsymbol{\theta}_R^k = \sigma_1\mathbf{u}_1\mathbf{v}_1^\top,$$

To guarantee that the resulting dictionary atom on the feature space is normalized to unit norm, one would choose

$$\boldsymbol{\theta}_R^k = \sigma_1\mathbf{v}_1^\top \tag{3.9}$$

$$\Phi(\mathbf{Z})\mathbf{a}_k = \mathbf{u}_1. \tag{3.10}$$

However, as mentioned before, we can not perform direct SVD decomposition on $\Phi(\mathbf{Z})\mathbf{E}_k^R$ as $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ since the matrix $\mathbf{U}$ can have infinitely large row dimension. We address this issue next.

A remedy for this issue comes from the fact that SVD decomposition above is closely related to the eigendecomposition of the Gram matrix, $(\Phi(\mathbf{Z})\mathbf{E}_k^R)^\top(\Phi(\mathbf{Z})\mathbf{E}_k^R)$ which is of size $n \times n$. It is easy to confirm

$$(\Phi(\mathbf{Z})\mathbf{E}_k^R)^\top(\Phi(\mathbf{Z})\mathbf{E}_k^R) = (\mathbf{E}_k^R)^\top\mathbb{K}(\mathbf{Z},\mathbf{Z})(\mathbf{E}_k^R) = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top$$

Now, denoting $\mathbf{v}_1$ as the first column of $\mathbf{V}$, and $\sigma_1 = \boldsymbol{\Sigma}(1,1)$, $\boldsymbol{\theta}_R^k$ can be found using the relation in (3.9). To solve for $\mathbf{a}_k$, we first observe that by multiplying (3.8) from the right by $\mathbf{V}$ and considering only the first column, we get

$$\Phi(\mathbf{Z})\mathbf{E}_k^R\mathbf{v}_1 = \sigma_1\mathbf{u}_1. \tag{3.11}$$

The solution for $\mathbf{a}_k$ is obtained by substituting (3.10) into (3.11) $\Phi(\mathbf{Z})\mathbf{E}_k^R\mathbf{v}_1 = \sigma_1\Phi(\mathbf{Z})\mathbf{a}_k$. Now, multiplying from the left by $\Phi(\mathbf{Z})^\top$ we have:

$$\begin{aligned}
\Phi(\mathbf{Z})^\top\Phi(\mathbf{Z})\mathbf{E}_k^R\mathbf{v}_1 &= \sigma_1\Phi(\mathbf{Z})^\top\Phi(\mathbf{Z})\mathbf{a}_k \\
\mathbb{K}(\mathbf{Z},\mathbf{Z})\mathbf{E}_k^R\mathbf{v}_1 &= \sigma_1\mathbb{K}(\mathbf{Z},\mathbf{Z})\mathbf{a}_k \\
\implies \mathbf{E}_k^R\mathbf{v}_1 &= \sigma_1\mathbf{a}_k
\end{aligned} \tag{3.12}$$

Note that $\mathbb{K}(\mathbf{Z},\mathbf{Z})$ is typically invertible for Mercer kernels. Hence, $\mathbf{a}_k = \sigma_1^{-1}\mathbf{E}_k^R\mathbf{v}_1$. One can easily verify that this updating procedure of $\mathbf{a}_k$ results in a dictionary atom of unit-norm on the feature space. The final Kernelized K-SVD process is described below in Algorithm 2.

Algorithm 2 The Kernel K-SVD Algorithm

Input: set of signals $\mathbf{Z}$, kernel function $\kappa(\cdot,\cdot)$

Output: $\mathbf{A}$ and $\boldsymbol{\Theta}$

Initialization: Set τ random elements of each column in $\boldsymbol{\Theta}$ to be 1. Set $J = 1$.

Stage 1: Sparse Coding

Use KOMP algorithm to obtain sparse coefficients $\boldsymbol{\Theta}$ given a fixed dictionary mixing $\mathbf{A}$.

Stage 2: Dictionary Update

1. For $k = 0, 1, ..., K$ in $\mathbf{A}^{(J-1)}$
2. (a) Let $\omega_k = \{i | 1 \leq i \leq N, \boldsymbol{\theta}_T^k(i) \neq 0\}$
   (b) Generate $\mathbf{E}_k$, according to (3.6).
   (c) Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to ω_k, and obtain $\mathbf{E}_k^R$ as $\mathbf{E}_k^R = \mathbf{E}_k \Omega_k$
   (d) Apply SVD decomposition to get $(\mathbf{E}_k^R)^\top \mathbb{K}(\mathbf{Z}, \mathbf{Z})(\mathbf{E}_k^R) = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top$. Choose updated $\mathbf{a}_k = \sigma_1^{-1}\mathbf{E}_k^R\mathbf{v}_1$, where $\mathbf{v}_1$ is the first vector of $\mathbf{V}$ corresponding to the largest singular value $\sigma_1^2 = \boldsymbol{\Sigma}^2(1,1)$.

$J = J + 1$

3.5 Kernel modified MSC

With the kernel K-SVD algorithm established, it is simple to extend the modified MSC framework, outlined in Appendix C, to a kernel modified MSC test. Building on the notations from this appendix, we present the kernel version of the modified MSC.

In order to obtain the kernel version of the modified MSC from Appendix C [2,6], we now consider all model components under a mapping Φ associated with a chosen Mercer kernel function. Applying this assumption, our signal model becomes:

$$\begin{aligned} \Phi(\mathbf{Z}) &= \mathbf{H}_m\boldsymbol{\Theta}_m + \mathbf{N} \qquad m \in [1, M] \\ &= \Phi(\mathbf{Z})\mathbf{A}_m\boldsymbol{\Theta}_m + \mathbf{N} \end{aligned} \tag{3.13}$$

where $\Phi(\mathbf{Z}) \in \mathbb{R}^{D\times n}$ is the mapped data matrix containing observation vectors $\Phi(\mathbf{z}_i)$ as its columns, $\mathbf{H}_m = \Phi(\mathbf{Z})\mathbf{A}_m \in \mathbb{R}^{D\times K}$ is the dictionary matrix whose columns are the basis vectors that span the subspace associated with the m^{th} class, $\mathbf{\Theta}_m$ is an unknown matrix with the columns being the parameter vector associated with mapped data vectors $\Phi(\mathbf{z}_i)$, and $\mathbf{N} \in \mathbb{R}^{D\times n}$ denotes an additive zero-mean noise matrix in the mapped feature space.

For the kernel modified MSC, the decision-making is carried out by determining the class that satisfies

$$\begin{aligned} m^* &= \text{argmin}_{m\in[1,M]}\{||\Phi(\mathbf{Z}) - \mathbf{H}_m\hat{\mathbf{\Theta}}_m||_F^2\} \\ &= \text{argmin}_{m\in[1,M]}\{\text{tr}((\mathbf{I} - \mathbf{A}_m\hat{\mathbf{\Theta}}_m)^\top \mathbb{K}(\mathbf{Z},\mathbf{Z})(\mathbf{I} - \mathbf{A}_m\hat{\mathbf{\Theta}}_m))\} \end{aligned} \tag{3.14}$$

where $\hat{\mathbf{\Theta}}_m$ is Kernel Orthogonal Matching Pursuit (KOMP) estimate of $\mathbf{\Theta}_m$ when dictionary matrix $\mathbf{H}_m$ is used [6,21]. That is, the correct class m^* is the one which makes the magnitude of the reconstruction error the smallest. Training of the MSC amounts to constructing class-dependent dictionary matrices $\mathbf{H}_m, m = 1,\ldots,M$ from representative training data sets in each class m. Such training can be performed using the kernel K-SVD method presented earlier in this chapter. For individual samples $\Phi(\mathbf{z})$ we can form the decision rule as follows:

$$\begin{aligned} m^* &= \text{argmin}_{m\in[1,M]}\{||\Phi(\mathbf{z}) - \mathbf{H}_m\hat{\theta}_m||_F^2\} \\ &= \text{argmin}_{m\in[1,M]}\{\text{tr}((\Phi(\mathbf{z}) - \Phi(\mathbf{Z})\mathbf{A}_m\hat{\theta}_m)^\top(\Phi(\mathbf{z}) - \Phi(\mathbf{Z})\mathbf{A}_m\hat{\theta}_m))\} \\ &= \text{argmin}_{m\in[1,M]}\{\mathbb{K}(\mathbf{z},\mathbf{z}) - 2\mathbb{K}(\mathbf{z},\mathbf{Z})\mathbf{A}_m\hat{\theta}_m + \hat{\theta}_m^\top\mathbf{A}_m^\top\mathbb{K}(\mathbf{Z},\mathbf{Z})\mathbf{A}_m\hat{\theta}_m\}. \end{aligned} \tag{3.15}$$

Therefore, running the kernel modified MSC test amounts to computing the error term in (3.15) for each of the learned class dictionaries and the model m^* which admits the smallest reconstruction error is chosen as the class label of sample $\mathbf{z}$.

3.6 Conclusion

In this chapter we reviewed a fully kernelized solution to the K-SVD dictionary learning problem. Furthermore, we explained how this kernel K-SVD algorithm can be used to

construct a *kernel* modified MSC test, extending the modified MSC from [2] to allow for non-linear mapping of data samples, much like the work in [32]. The solution in Algorithm 2 relies on forming the full kernel matrix $\mathbb{K}(\mathbf{Z}, \mathbf{Z})$ using all N samples available in the training set which is obviously computationally inefficient. In Lifelong learning applications, N can grow to an enormous size as the system collects more and more data samples and it would be ideal to have a solution that depends on only a handful of *important* samples. A similar problem exists with the kernel modified MSC solution in (3.15), where the final term in the test statistic relies on a multiplication with the full kernel matrix for each sample and across each potential hypothesis. In the following chapters we introduce an approach to greatly reduce the computational cost of learning kernelized dictionaries and constructing a modified MSC statistic using the compact features.

CHAPTER 4

LINEARIZED KERNEL EMBEDDING AND EMPIRICAL KERNEL MAPS

4.1 Introduction

In this chapter an alternative to the fully kernelized SRC developed in [21] is presented. This method utilizes an empirical kernel embedding method [4] which relies on the Nyström approximation [3, 5], leveraging *important* samples that are representative of the generative distribution in order to create *virtual* features. These features are faithful to the relative geometry in the mapped space associated with the kernel function of choice. By using this linearized kernel embedding approach, linear learning algorithms can be applied without the need for storing *all* training samples, traditionally required in kernel methods [21, 36–38], while preserving the benefits of a fully kernelized solution. We begin by discussing the empirical linearized kernel embedding, also known as linearized kernel dictionary learning [4].

Kernel machines are widely used for many complex pattern recognition and machine learning applications. However, most kernel machines require the formation of an exhaustive kernel matrix utilizing all training data samples. In [4], a linearized Kernel Dictionary Learning (LKDL) was proposed for reducing the complexity of representations. The LKDL method finds new lower-dimensional data representations based on the Nyström method. In this way, LKDL allows for linear dictionary learning methods to be applied to non-linearly mapped virtual features without increasing the dimensionality of the data representation. That is, discriminative benefits of kernel machines can be gained without the need for the full kernel matrix. The inherent sampling in this method is highly beneficial to developing adaptive classification systems with long-term incremental learning. Owing to all these benefits we have adopted, and further developed, this method in this dissertation.

4.2 Linearized Kernel Embedding and Nyström Approximation – A Review

In order to allow for linear (dictionary or other) learning on kernelized representatives of data samples, a popular approach is to factorize the kernel matrix into a set of vector samples whose Gram matrix is approximately equal to the original kernel matrix [4,5]. This can be achieved using a low-rank projection of the full mapped training data set via the Nyström approximation [3–5] which is briefly reviewed next.

4.2.1 Nyström Method for Kernel Matrix Approximation

Let $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \cdots\ \mathbf{z}_N]$ be a data matrix containing N data (feature) vectors $\mathbf{z}_i \in \mathbb{R}^d$ and $\mathbf{\Phi} = [\phi(\mathbf{z}_1)\ \phi(\mathbf{z}_2)\ \cdots\ \phi(\mathbf{z}_N)] \in \mathbb{R}^{D\times N}$ be the corresponding nonlinearly mapped data matrix with columns $\boldsymbol{\phi}(\mathbf{z}_i) \in \mathbb{R}^D$ where usually $D \gg d$. Then, the associated kernel Gram matrix of the nonlinearly mapped data is $\mathbf{K} = \mathbf{\Phi}^\top\mathbf{\Phi}$ where each element $\mathbf{K}_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j) = \boldsymbol{\phi}(\mathbf{z}_i)^\top\boldsymbol{\phi}(\mathbf{z}_j)$ and $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ is an appropriate kernel function [36].

We assume that $c \leq N$ columns of $\mathbf{K}$ are chosen according to a sampling scheme e.g. uniform sampling or column-norm sampling [39]. The kernel matrix $\mathbf{K}$ can be partitioned as,

$$\mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{B} \end{bmatrix},$$

where $\mathbf{W}$ is the kernel matrix associated with the retained *important* data samples, $\mathbf{B} \in \mathbb{R}^{(N-c)\times(N-c)}$ is the one associated with unimportant samples, and $\mathbf{A}$ is a matrix of inner products between the c chosen samples and $N-c$ unimportant samples. That is, if the data matrix associated with the retained samples is denoted by $\mathbf{Z}_R = [\mathbf{z}_1\ \ldots\ \mathbf{z}_c] \in \mathbb{R}^{d\times c}$ and its mapped version in the feature space by $\mathbf{\Phi}_R = [\phi(\mathbf{z}_1)\ \ldots\ \phi(\mathbf{z}_c)] \in \mathbb{R}^{D\times c}$, then $\mathbf{W} = \mathbf{\Phi}_R^\top\mathbf{\Phi}_R \in \mathbb{R}^{c\times c}$.

Now, denoting $\mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{A} \end{bmatrix} \in \mathbb{R}^{N\times c}$ and using the Nyström method [3–5], the kernel

matrix $\mathbf{K}$ can be approximated as

$$\begin{aligned}\mathbf{K} \approx \hat{\mathbf{K}} &= \mathbf{C}\mathbf{W}^{-1}\mathbf{C}^{\top} \\ &= \mathbf{\Phi}^{\top}\mathbf{\Phi}_R(\mathbf{\Phi}_R^{\top}\mathbf{\Phi}_R)^{-1}\mathbf{\Phi}_R^{\top}\mathbf{\Phi} \\ &= \mathbf{\Phi}^{\top}\mathbf{P}_{\mathbf{\Phi}_R}\mathbf{\Phi}, \end{aligned} \tag{4.1}$$

where $\mathbf{W}$ is assumed to be full rank and $\mathbf{P}_{\mathbf{\Phi}_R}$ denotes the orthogonal projection matrix onto the span of $\mathbf{\Phi}_R$. Since $\mathbf{P}_{\mathbf{\Phi}_R}$ is idempotent, (4.1) implies that $\hat{\mathbf{K}}$ is the Gram matrix after projecting all samples in $\mathbf{\Phi}$ onto the subspace $\langle \mathbf{\Phi}_R \rangle$.

4.2.2 Linearized Kernel Embedding

The method presented in [4] finds a linear realization of the kernel matrix with features that capture the benefits of the nonlinear mapping in an efficient way. More specifically, using this method the kernel Gram matrix $\mathbf{K}$ can be factored into $\mathbf{K} \approx \mathbf{F}^{\top}\mathbf{F}$ leading to the *virtual feature* matrix $\mathbf{F}$ that yields the full kernel matrix.

To see this, let us apply eigenvalue decomposition to the symmetric full rank kernel matrix $\mathbf{W}$ such that $\mathbf{W} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^{\top}$ with $\mathbf{\Sigma}$ being a diagonal matrix containing all the eigenvalues and $\mathbf{V}$ is an orthogonal matrix containing the associated eigenvectors as its columns. Using this eigenvalue decomposition, (4.1) can be expressed as,

$$\mathbf{K} \approx \mathbf{C}\mathbf{W}^{-1}\mathbf{C}^{\top} = \mathbf{F}^{\top}\mathbf{F} = \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{V}^{\top}\mathbf{C}^{\top},$$

This yields the virtual data matrix $\mathbf{F}$ as,

$$\mathbf{F} = \mathbf{\Sigma}^{-1/2}\mathbf{V}^{\top}\mathbf{C}^{\top}. \tag{4.2}$$

Thus, for a particular data sample $\mathbf{z}$ and its mapped version $\boldsymbol{\phi}(\mathbf{z})$, the corresponding virtual feature vector is,

$$\mathbf{f} = \mathbf{\Sigma}^{-1/2}\mathbf{V}^{\top}\mathbf{\Phi}_R^{\top}\boldsymbol{\phi}(\mathbf{z}) \tag{4.3}$$

$$= \mathbf{\Sigma}^{-1/2}\mathbf{V}^{\top}\left[\kappa(\mathbf{z}_1, \mathbf{z}) \ \ldots \ \kappa(\mathbf{z}_c, \mathbf{z})\right]^{\top}. \tag{4.4}$$

The dimension of the virtual samples in $\mathbf{F}$ can be reduced by choosing the k largest eigenvalues, $\mathbf{\Sigma}_k = diag[\sigma_1\ \sigma_2\ \cdots\ \sigma_k]$ of $\mathbf{W}$ and the associated eigenvectors $\mathbf{V}_k = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_k]$. Then, the reduced dimensional *virtual data matrix* [4] is given by

$$\mathbf{F}_k = \mathbf{\Sigma}_k^{-1/2}\mathbf{V}_k^\top\mathbf{C}^\top. \tag{4.5}$$

Given the matrices $\mathbf{\Sigma}_k^{-1/2}$ and $\mathbf{V}_k$, any test sample $\mathbf{z}$ can be mapped to the corresponding virtual feature vector using,

$$\mathbf{f} = \mathbf{\Sigma}_k^{-1/2}\mathbf{V}_k^\top\left[\kappa(\mathbf{z}_1,\mathbf{z})\ \ldots\ \kappa(\mathbf{z}_c,\mathbf{z})\right]^\top.$$

where $\mathbf{z}_1 \cdots\ \mathbf{z}_c$ are the c important samples, chosen via sampling, and $\kappa(\mathbf{x},\mathbf{y})$ is the chosen kernel function.

4.2.3 Geometric Perspective of Kernel Embedding

At first glance, one might assume that the approximation $\hat{\mathbf{K}} = \mathbf{F}^\top\mathbf{F}$ gives a solution that is unrelated to the original embedded signals and their geometry. As explained above, the Nyström method can be derived by projecting the full data set onto the subspace defined by the important samples [40]. In this section, we illustrate how the coordinates $\mathbf{F}$ are in fact the exact local coordinates (i.e., using c-dimensional coordinates rather than D) of *all* samples in the feature space when projected onto the principal axes of the important samples, i.e., the $c \leq N$ samples selected for approximating the kernel matrix, in the feature space.

Proposition 4.2.1. *Given a data sample $\boldsymbol{z}$, the principal component vector of the mapped data, $\boldsymbol{\phi}(\boldsymbol{z})$, is its associated virtual feature vector $\boldsymbol{f}$, i.e., $\boldsymbol{f} = \boldsymbol{U}^\top\boldsymbol{\phi}(\boldsymbol{z})$ where $\boldsymbol{U}^\top$ is the mapping matrix.*

Proof. The important samples in the kernel feature space, $\mathbf{\Phi}_R$, can be decomposed using SVD as

$$\mathbf{\Phi}_R = \mathbf{U}\mathbf{\Sigma}^{1/2}\mathbf{V}^\top \in \mathbb{R}^{D\times c} \tag{4.6}$$

where $\mathbf{U} \in \mathbb{R}^{D\times c}$ is a semi-orthogonal matrix that contains the orthonormal left singular vectors of $\mathbf{\Phi}_R$, i.e., $\mathbf{U}^\top\mathbf{U} = \mathbf{I}_{c\times c}$, $\mathbf{\Sigma}^{1/2}$ is a $c \times c$ matrix of singular values, and $\mathbf{V}$ is a $c \times c$ orthonormal matrix containing the right singular vectors.

Now, using (4.6) it is evident that the matrix $\mathbf{U}$ diagonalizes the rank-c correlation matrix, $\mathbf{\Phi}_R\mathbf{\Phi}_R^\top$. Hence, the matrix $\mathbf{U}^\top$ maps the nonlinearly mapped data $\boldsymbol{\phi}(\mathbf{z})$ to the corresponding principal components. Furthermore, $\mathbf{f} = \mathbf{U}^\top\boldsymbol{\phi}(\mathbf{z})$ can easily be confirmed by replacing $\mathbf{\Phi}_R$ in (4.3) with its SVD. □

Remark II.1

For any pair of data samples $\mathbf{z}_i, \mathbf{z}_j \in \mathbf{Z}_R$, using (4.3) and (4.6), we get

$$\begin{aligned}\mathbf{f}_i^\top\mathbf{f}_j &= \boldsymbol{\phi}(\mathbf{z}_i)^\top\mathbf{\Phi}_R\mathbf{V}\mathbf{\Sigma}^{-1/2}\mathbf{\Sigma}^{-1/2}\mathbf{V}^\top\mathbf{\Phi}_R^\top\boldsymbol{\phi}(\mathbf{z}_j)\\ &= \boldsymbol{\phi}(\mathbf{z}_i)^\top\mathbf{U}\mathbf{U}^\top\boldsymbol{\phi}(\mathbf{z}_j).\end{aligned}$$

However, we also know that $\mathbf{P}_{\mathbf{\Phi}_R} = \mathbf{U}\mathbf{U}^\top$, i.e., the projection matrix onto the subspace spanned by $\mathbf{U}$, and also since $\boldsymbol{\phi}(\mathbf{z}_i)$ is in $\langle\mathbf{\Phi}_R\rangle$ then $\mathbf{P}_{\mathbf{\Phi}_R}\boldsymbol{\phi}(\mathbf{z}_i) = \boldsymbol{\phi}(\mathbf{z}_i)$. Thus, $\mathbf{f}_i^\top\mathbf{f}_j = \boldsymbol{\phi}(\mathbf{z}_i)^\top\boldsymbol{\phi}(\mathbf{z}_j)$.

Furthermore, we have $||\boldsymbol{\phi}(\mathbf{z}_i) - \boldsymbol{\phi}(\mathbf{z}_j)||_2 = ||\mathbf{f}_i - \mathbf{f}_j||_2 \ \forall\ i, j \in \{1, ..., c\}$, in other words, this embedding preserves distance and angle properties, and $\mathbf{f}_i$ simply gives a c-dimensional subspace coordinates representation for $\boldsymbol{\phi}(\mathbf{z}_i) \in \langle\mathbf{\Phi}_R\rangle$. Similarly, if $k < c$ then $\mathbf{f}_i$'s represent the coordinates of the samples projected onto the first k principal axes of the important samples in $\mathbf{\Phi}_R$. This means that representations $\boldsymbol{\phi}(\mathbf{z}_i)$ and $\mathbf{f}_i$ are isometric for important samples.

4.3 Modified MSC using Virtual Features

Building off the notations from Appendix C where the traditional and modified Matched Subspace Classifiers (MSC) are introduced, in this section, we briefly discuss how to implement the modified MSC classifier using virtual features. The key idea is to use the virtual

feature matrix $\mathbf{F}$ instead of the actual observation matrix $\mathbf{Z}$ to take full benefit of the non-linear kernel mapping without increasing the dimension of the representation. The inherent sampling in the LKE method covered in the previous sections significantly reduces the number of useful samples required to generate highly discriminative kernelized features when compared to the fully kernelized modified MSC from Chapter 3.

When using the virtual features in the modified MSC method [2], the decision-making rule becomes,

$$m^* = \text{argmin}_{m\in[1,M]}\{||\mathbf{F} - \mathbf{H}_m\hat{\mathbf{\Theta}}_m||_F^2\} \tag{4.7}$$

where $\hat{\mathbf{\Theta}}_m$ is generated using a pursuit method such as the Orthogonal Matching Pursuit (OMP) or Basis Pursuit (BP) algorithms [41] when the dictionary $\mathbf{H}_m$ is used. Here, we will use the fast OMP method in [24] which does not require any matrix inversion. Training this modified MSC is similar to that of the original modified MSC, however, dictionary $\mathbf{H}_m$ is trained using the class m virtual samples, $\mathbf{F}_m$, rather than the class m input samples, $\mathbf{Z}_m$.

4.4 Conclusion

In this chapter we reviewed a linearized kernel embedding technique that is used along with the Nyström method for kernel matrix approximation. Furthermore we provided analysis on a convenient and illuminating geometric interpretation of this embedding technique that is closely related to the geometry of the fully kernelized solution. In the following chapter we will begin a discussion of techniques that can be utilized in order to expand the embedding when novel important samples are discovered. These techniques are developed in a manner that is consistent with the original embedding geometry and which simply inflates the subspace spanned by the mapped important samples $\mathbf{\Phi}_R$.

CHAPTER 5

SAMPLING TECHNIQUES FOR THE NYSTRÖM METHOD

5.1 Introduction

In order to choose the important samples for the linearized kernel embedding (LKE) from Chapter 4 there are a variety of sampling methods that are available based on various criteria for optimality [42, 43]. In this chapter we will begin with a discussion of various sampling techniques that are commonly applied when using the Nyström method for kernel matrix approximation. The methods surveyed in the introduction of this chapter are considered in more detail in [39]. Then we review a reconstructions based approach known as ridge-leverage score (RLS) sampling [5, 44] and discuss a recent recursive algorithm that was developed in order to solve the RLS scores, used in this sampling, more efficiently [5].

5.1.1 Motivations for investigating Sampling Approaches

The basic approach behind the Nyström method [3–5] is to choose a subset of s columns (or equivalently, rows) of $\mathbf{K}$, denoted as $\mathbf{C} = \mathbf{KS}$, where $\mathbf{S} \in \mathbb{R}^{n \times s}$ has a single non-zero entry in each column equal to 1, to approximate the full matrix $\mathbf{K}$ via a closely related low-rank matrix $\mathbf{CW}^{+}\mathbf{C}^{\top} = \mathbf{KS}(\mathbf{S}^{\top}\mathbf{KS})^{+}\mathbf{S}^{\top}\mathbf{K}$. As a result, the method by which these columns are selected can *significantly* influence the accuracy of the approximation. In this section, we will briefly discuss various sampling techniques that aim to select informative columns from $\mathbf{K}$.

5.1.2 A Brief Overview of Sampling Methods

We begin with the most common class of sampling techniques that select columns using a fixed probability distribution. The most basic sampling technique, which was originally proposed by Williams and Seeger [3] involves uniform sampling of the columns without replacement. While this method is very efficient to implement, given a fixed column budget c, the representative power of the uniform sampling diminshes as a datasets grow exceedingly large.

Alternatively, non-uniform sampling approaches can be taken. Two commonly used non-uniform sampling approaches are "diagonal sampling" and "column-norm sampling". In diagonal sampling, the ith column can be sampled non-uniformly with weight proportional to its corresponding diagonal element $\mathbf{K}_{ii}$, i.e. by letting $p_i = \mathbf{K}_{ii}^2 / \sum_{i=1}^{n} \mathbf{K}_{ii}^2$. In "column-norm sampling" columns are sampled with their weight proportional to the the L_2 norm of the column, i.e. by letting $p_i = ||\mathbf{k}_i||^2 / ||\mathbf{K}||_F^2$ [42, 43]. There are additional computational costs associated with these non-uniform sampling methods: $\mathcal{O}(n)$ time and space requirements for diagonal sampling and $\mathcal{O}(n^2)$ time and space for column-norm sampling. These non-uniform sampling techniques are often presented using sampling with replacement to simplify theoretical analysis. Column-norm sampling has been used to analyze a general SVD approximation algorithm.

Another sampling approach which was originally presented in [45] for the CUR decomposition (which is quite similar to the Nyström method) is to sample from a distribution over the columns based on "leverage scores" [44, 45]. Until recently, these scores could not be efficiently computed in practice for large-scale applications [5]. In the next sections we will discuss ridge-leverage score sampling and a recently developed recursive algorithm that makes the computation of leverage-scores more practical for very large datasets. This sampling technique provides the strongest approximation guarantees (see equation 5.6 in following section) for reconstruction capabilities among the discussed sampling techniques and, as such, it is a favorable approach.

5.2 Ridge Leverage Score Sampling

The approach discussed in this subsection is concerned with a reconstructive based objective and utilizes a sampling strategy that is closely related to the ridge regression estimator at its core. This technique is known as ridge leverage score (RLS) sampling. In this chapter we discuss a recently developed recursive algorithm for solving the ridge leverage scores, used in this sampling, in competitive time. The method discussed is the method from [5]; we touch on the most important lemmas and theorems for understanding this technique but several of the proofs are excluded as they are rather involved and can be found in the original paper [5].

5.2.1 RLS-Nyström Method

Ridge leverage scores (RLS) sampling is popular for Nyström method because of the associated spectral norm approximation guaratees [5, 44]. In basic RLS-Nyström, samples are chosen to be landmarks (i.e. important samples) with probability proportional to their ridge leverage score. In this section, we briefly describe ridge leverage scores and several ways to view them.

Definition. 1, Ridge Leverage Scores (RLS) [5, 44]:
For any $\lambda > 0$, the λ-ridge leverage score of data point $\mathbf{x}_i$ with respect to the kernel matrix $\mathbf{K}$ is defined as

$$\ell_i^{\lambda}(\mathbf{K}) := (\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1})_{i,i}, \tag{5.1}$$

For any $\mathbf{B} \in \mathbb{R}^{n \times n}$ satisfying $\mathbf{B}\mathbf{B}^{\top} = \mathbf{K}$, we can also write

$$\ell_i^{\lambda}(\mathbf{K}) = \mathbf{b}_i^{\top}(\mathbf{B}^{\top}\mathbf{B} + \lambda \mathbf{I})^{-1}\mathbf{b}_i, \tag{5.2}$$

where $\mathbf{b}_i^{\top} \in \mathbb{R}^{1 \times n}$ is the i^{th} row of $\mathbf{B}$.

To see this, note $\mathbf{b}_i^\top(\mathbf{B}^\top\mathbf{B}+\lambda\mathbf{I})^{-1}\mathbf{b}_i = (\mathbf{B}(\mathbf{B}^\top\mathbf{B}+\lambda\mathbf{I})^{-1}\mathbf{B}^\top)_{i,i}$. Using the SVD to write $\mathbf{B}=\mathbf{U}\Sigma\mathbf{V}^\top$ and accordingly $\mathbf{K}=\mathbf{U}\Sigma^2\mathbf{U}^\top$ confirms that

$$\begin{aligned}
\mathbf{K}(\mathbf{K}+\lambda\mathbf{I})^{-1} &= \mathbf{U}\Sigma^2\mathbf{U}^\top(\mathbf{U}\Sigma^2\mathbf{U}^\top+\lambda\mathbf{I})^{-1} = \mathbf{U}\Sigma^2(\Sigma^2+\lambda\mathbf{I})^{-1}\mathbf{U}^\top \\
\mathbf{B}(\mathbf{B}^\top\mathbf{B}+\lambda\mathbf{I})^{-1}\mathbf{B}^\top &= \mathbf{U}\Sigma\mathbf{V}^\top(\mathbf{V}\Sigma^2\mathbf{V}^\top+\lambda\mathbf{I})^{-1}\mathbf{V}\Sigma\mathbf{U}^\top = \mathbf{U}\Sigma(\Sigma^2+\lambda\mathbf{I})^{-1}\Sigma\mathbf{U}^\top \\
&\overset{*}{=} \mathbf{U}\Sigma^2(\Sigma^2+\lambda\mathbf{I})^{-1}\mathbf{U}^\top \\
\implies \mathbf{K}(\mathbf{K}+\lambda\mathbf{I})^{-1} &= \mathbf{B}(\mathbf{B}^\top\mathbf{B}+\lambda\mathbf{I})^{-1}\mathbf{B}^\top = \mathbf{U}\Sigma^2(\Sigma^2+\lambda\mathbf{I})^{-1}\mathbf{U}^\top
\end{aligned}$$

Alternatively, the ridge scores can be defined:

$$\ell_i^\lambda = \min_{\mathbf{y}\in\mathbb{R}^n} \frac{1}{\lambda}||\mathbf{b}_i^\top - \mathbf{y}^\top\mathbf{B}||_2^2 + ||\mathbf{y}||_2^2. \tag{5.3}$$

To see this we first solve the objective in (5.3). We begin by converting the objective to a standard quadratic:

$$\begin{aligned}
\text{Objective of (5.3)} &= \frac{1}{\lambda}(\mathbf{b}_i^\top\mathbf{b}_i - 2\mathbf{b}_i^\top\mathbf{B}^\top\mathbf{y} + \mathbf{y}^\top\mathbf{B}\mathbf{B}^\top\mathbf{y}) + \mathbf{y}^\top\mathbf{y} \\
&= \mathbf{y}^\top(\mathbf{I}+\frac{1}{\lambda}\mathbf{K})\mathbf{y} - \frac{2}{\lambda}\mathbf{b}_i^\top\mathbf{B}^\top\mathbf{y} + \frac{1}{\lambda}\mathbf{b}_i^\top\mathbf{b}_i
\end{aligned}$$

Now, a quadratic of the form $f(\mathbf{x}) = \mathbf{x}^\top\mathbf{Q}\mathbf{x} - 2\mathbf{b}^\top\mathbf{x} + c$ (where $\mathbf{Q}$ is symmetric and full-rank) can be minimized as follows:

$$\begin{aligned}
Df(\mathbf{x}) &= 2\mathbf{x}^\top\mathbf{Q} - 2\mathbf{b}^\top \\
\implies \nabla_{\mathbf{x}} f &= 2\mathbf{Q}\mathbf{x} - 2\mathbf{b}
\end{aligned}$$

Now we apply the FONC to find candidate minimizers (set $\nabla_{\mathbf{x}} f = 0$)

$$\begin{aligned}
0 &= 2\mathbf{Q}\mathbf{x} - 2\mathbf{b} \\
\implies \mathbf{x}^* &= \mathbf{Q}^{-1}\mathbf{b}
\end{aligned}$$

Applying this solution to our original $f(\mathbf{x})$ we see that the minimum (i.e. f evaluated at the minimizer) is:

$$f(\mathbf{x}^*) = c - \mathbf{b}^\top\mathbf{Q}^{-1}\mathbf{b}$$

Now, looking back at our standard form quadratic $g(\mathbf{y})$, we had $\mathbf{Q} = (\mathbf{I} + \frac{1}{\lambda}\mathbf{K})$, $\mathbf{b} = \frac{1}{\lambda}\mathbf{B}\mathbf{b}_i$ and $c = \frac{1}{\lambda}\mathbf{b}_i^\top \mathbf{b}_i$. So the minimum of $g(\mathbf{y})$ is:

$$g(\mathbf{y}^*) = \frac{1}{\lambda}\mathbf{b}_i^\top \mathbf{b}_i - \frac{1}{\lambda^2}\mathbf{b}_i^\top \mathbf{B}^\top (\mathbf{I} + \frac{1}{\lambda}\mathbf{K})^{-1}\mathbf{B}\mathbf{b}_i \qquad = \frac{1}{\lambda}\mathbf{b}_i^\top (\mathbf{I} - \mathbf{B}^\top (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{B})\mathbf{b}_i \qquad (5.4)$$

The last step of the proof involves the following lemma.

Lemma 1: $\frac{1}{\lambda}(\mathbf{I} - \mathbf{B}^\top (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{B}) = (\mathbf{B}^\top \mathbf{B} + \lambda\mathbf{I})^{-1}$.

Proof: We begin by noting that $\mathbf{K} = \mathbf{B}\mathbf{B}^\top$, $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \mathbf{U}\,\mathrm{diag}(\sigma_i)\mathbf{V}^\top$ so $\mathbf{K} = \mathbf{U}\,\mathrm{diag}(\sigma_i^2)\mathbf{U}^\top$ and lastly, $(\mathbf{K} + \lambda\mathbf{I})^{-1} = \mathbf{U}\,\mathrm{diag}(\frac{1}{\sigma_i^2 + \lambda})\mathbf{U}^\top$.

We now have

$$\begin{aligned}
&\frac{1}{\lambda}(\mathbf{I} - \mathbf{B}^\top (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{B}) \\
&\qquad = \frac{1}{\lambda}(\mathbf{I} - \mathbf{V}\,\mathrm{diag}(\sigma_i)\mathbf{U}^\top \mathbf{U}\,\mathrm{diag}(\frac{1}{\sigma_i^2 + \lambda})\mathbf{U}^\top \mathbf{U}\,\mathrm{diag}(\sigma_i)\mathbf{V}^\top) \\
&\qquad = \frac{1}{\lambda}\mathbf{V}(\mathrm{diag}(\mathbf{1}) - \mathrm{diag}(\frac{\sigma_i^2}{\sigma_i^2 + \lambda}))\mathbf{V}^\top \\
&\qquad = \mathbf{V}(\mathrm{diag}(\frac{1}{\sigma_i^2 + \lambda}))\mathbf{V}^\top \\
&\qquad = \mathbf{V}(\mathrm{diag}(\sigma_i^2) + \lambda\mathbf{I})^{-1}\mathbf{V}^\top \\
&\qquad = (\mathbf{V}\,\mathrm{diag}(\sigma_i)\,\mathrm{diag}(\sigma_i)\mathbf{V}^\top + \lambda\mathbf{I})^{-1} \\
&\qquad = (\mathbf{V}\,\mathrm{diag}(\sigma_i)\mathbf{U}^\top \mathbf{U}\,\mathrm{diag}(\sigma_i)\mathbf{V}^\top + \lambda\mathbf{I})^{-1} \\
&\qquad = (\mathbf{B}^\top \mathbf{B} + \lambda\mathbf{I})^{-1} \qquad \blacksquare
\end{aligned}$$

Confirming that (5.2) and (5.3) are equivalent provides us with a convenient interpretation of the ridge leverage scores. Since $\mathbf{B}\mathbf{B}^\top = \mathbf{K}$, any kernel learning algorithm effectively performs linear learning with $\mathbf{B}$'s rows as data points. The representation in (5.3) shows that ridge scores reflect the relative uniqueness of rows $\mathbf{b}_i^\top$. Note $\ell_i^\lambda \leq 1$ since we can set $\mathbf{y} = \mathbf{e}_i$, the i^{th} standard basis vector. A row $\mathbf{b}_i^\top$ will have ridge score $\ll 1$ (i.e. is less important) when

it is possible to find a more "spread out" $\mathbf{y}$ that uses other rows in $\mathbf{B}$ to approximately reconstruct $\mathbf{b}_i^\top$, indicating that this $\mathbf{b}_i^\top$ is highly linearly dependent on the other rows and does not carry much unique information. The following algorithm describes a basic form of RLS-Nyström sampling algorithm without recursion [44].

Algorithm 1 RLS-Nyström Sampling Algorithm

Input: $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$, Kernel Matrix $\mathbf{K}$, ridge $\lambda > 0$, failure prob. $\delta \in (0, 1/8)$

Output: Kernel matrix approx. $\tilde{\mathbf{K}}$

1: Compute over-approximation $\tilde{\ell}_i^\lambda > \ell_i^\lambda$ for the ridge leverage score of each $\mathbf{x}_1, ..., \mathbf{x}_n$

2: Set $p_i := \min\{1, \tilde{\ell}_i^\lambda \cdot 16 \log(\sum \tilde{\ell}_i^\lambda / \delta)\}$

3: Construct $\mathbf{S} \in \mathbb{R}^{n \times s}$ by sampling $\mathbf{x}_i$ with probability p_i

4: **return** Nyström factors $\mathbf{KS} \in \mathbb{R}^{n \times s}$ and $(\mathbf{S}^\top \mathbf{KS})^+ \in \mathbb{R}^{s \times s}$

The failure probability δ in Algorithm 1 indicates the probability that we will not meet the approximation guarantees of RLS-Nyström (see (5.5)). Two things to note about Algorithm 1 are the following: in step 1 we use over approximations of the true leverage scores, this is done for two reasons. First, because, as we show later, over-approximations can be computed more efficiently than computing the exact ridge leverage scores. Second, because this is a sampling-based approach, rather than selecting the top s leverage scores, we select a given sample $\mathbf{x}_i$ with probability proportional p_i; to be sure we select sufficient samples with respect to the failure probability, we choose leverage scores that boost the probability of selecting a given sample $\mathbf{x}_i$. The second thing to note is that the choice of probabilities in step 2 is not arbitrary, as will be explained further in Lemma 2; in the limiting case where $\delta \mapsto 0$ we will select every sample. Anywhere else in the interval $(0, 1/8)$, we will meet the spectral guarantee with probability $1 - \delta$ (see proof of Theorem 1 in [5]).

In [5] the authors proved the following theorem for the basic RLS-Nyström sampling algorithm which implies the strong approximation guarantees of RLS-Nyström.

Theorem 1 (Spectral error approximation):

For any $\lambda > 0$ and $\delta \in (0, 1/8)$, Algorithm 1 returns an $\mathbf{S} \in \mathbb{R}^{n \times s}$ such that with probability $1 - \delta$, $s \leq 2 \sum_i p_i$ and $\tilde{\mathbf{K}} = \mathbf{K}\mathbf{S}(\mathbf{S}^\top \mathbf{K}\mathbf{S})^+ \mathbf{S}^\top \mathbf{K}$ satisfies:

$$\tilde{\mathbf{K}} \preceq \mathbf{K} \preceq \tilde{\mathbf{K}} + \lambda \mathbf{I} \tag{5.5}$$

When ridge scores are computed exactly, $\sum_i p_i = \mathcal{O}(d_{eff}^\lambda \log(d_{eff}^\lambda / \delta))$, where

$$d_{eff}^\lambda = \mathrm{tr}(\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1}) = \sum_{i=1}^{n} \ell_i^\lambda(\mathbf{K})$$

is the so-called "λ-effective dimensionality" of $\mathbf{K}$ [46, 47].

In (5.5), $\preceq$ denotes the standard Loewner matrix ordering [48] on positive semi-definite Hermitian matrices (i.e. $\mathbf{M} \preceq \mathbf{N}$ if $\mathbf{N} - \mathbf{M}$ is PSD). Note that (5.5) implies the spectral norm guarantee:

$$||\mathbf{K} - \tilde{\mathbf{K}}||_2 \leq \lambda. \tag{5.6}$$

5.2.2 Recursive RLS-Nyström

In this section we discuss the recursive RLS sampling algorithm developed in [5]. Having discussed the strong approximation guarantees for RLS-Nyström, we now demonstrate an efficient implementation of the basic RLS sampling in Algorithm 1. The first step of Algorithm 1 naively requires $\mathcal{O}(n^3)$ time. In [5] it is shown that significant acceleration is possible using a recursive sampling approach, we illustrate this next. The key idea is to approximate the ridge leverage scores of $\mathbf{K}$ using a uniform sample of the data points. To ensure accuracy, the sample must be a constant fraction of the number of points. We later demonstrate how the strategy in [5] recursively approximates this large sample to achieve their final run-times. Next, we discuss the proof from [5] which establishes the following useful Lemma:

Lemma 2 [5]. For any $\mathbf{B} \in \mathbb{R}^{n\times n}$ with $\mathbf{B}\mathbf{B}^\top = \mathbf{K}$ and $\mathbf{S} \in \mathbb{R}^{n\times s}$ chosen by sampling each data point independently with probability 1/2, let

$$\tilde{\ell}_i^\lambda = \mathbf{b}_i^\top (\mathbf{B}^\top \mathbf{S}\mathbf{S}^\top \mathbf{B} + \lambda \mathbf{I})^{-1} \mathbf{b}_i \tag{5.7}$$

and $p_i = \min\{1, 16\tilde{\ell}_i^\lambda \log(\sum_i \tilde{\ell}_i^\lambda/\delta)\}$ for any $\delta \in (0, 1/8)$. Then with probability at least $1-\delta$:

1. $\tilde{\ell}_i^\lambda \geq \ell_i^\lambda$ for all i.
2. $\sum_i p_i \leq 64 \sum_i \ell_i^\lambda \log(\sum_i \ell_i^\lambda/\delta)$.

The first bound guarantees that the approximate leverage scores $\tilde{\ell}_i^\lambda$ suffice for use in Algorithm 1. The second bound ensures that the resulting Nyström approximation will have, up to constant factors, the same number of important samples as if we used the true ridge leverage scores. Note that it is not obvious how one might compute $\tilde{\ell}_i^\lambda$ via (5.7) without explicitly forming $\mathbf{B}$. We discuss how to do this in the next section.

Proof [5]. The first bound is a direct consequence of $\mathbf{B}^\top \mathbf{S}\mathbf{S}^\top \mathbf{B} \preceq \mathbf{B}^\top \mathbf{B}$, for an arbitrary row $\mathbf{b}_i$ it is easy to show:

$$\tilde{\ell}_i^\lambda = \mathbf{b}_i^\top (\mathbf{B}^\top \mathbf{S}\mathbf{S}^\top \mathbf{B} + \lambda \mathbf{I})^{-1} \mathbf{b}_i \geq \mathbf{b}_i^\top (\mathbf{B}^\top \mathbf{B} + \lambda \mathbf{I})^{-1} \mathbf{b}_i = \ell_i^\lambda.$$

Proving the second bound is more involved. The key observation made in [5] is that there exists a diagonal reweighting matrix $\mathbf{W} \in \mathbb{R}^{n\times n}, \mathbf{0} \preceq \mathbf{W} \preceq \mathbf{I}$ such that for all i, $\ell_i^\lambda(\mathbf{W}\mathbf{K}\mathbf{W}) \leq \alpha$ where $\alpha \overset{\text{def}}{=} \frac{1}{2} \cdot \frac{1}{16 \log(\sum \ell_i^\lambda/\delta)}$. This ensures that uniformly sampling rows with probability 1/2 from the *reweighted kernel* $\mathbf{W}\mathbf{K}\mathbf{W}$ is a valid ridge leverage score sampling. Additionally,

$$|\{i : \mathbf{W}_{i,i} < 1\}| \leq 32 \log(\sum \ell_i^\lambda/\delta) \cdot \sum \ell_i^\lambda.$$

That is, we do not need to reweight too many columns to achieve the ridge leverage score bound. Although $\mathbf{W}$ is never actually computed, its existence can be proved algorithmically: we can construct a valid $\mathbf{W}$ by iteratively considering any i with $\ell_i^\lambda(\mathbf{W}\mathbf{K}\mathbf{W}) \geq \alpha$. Since $\lambda > 0$, it is always possible to decrease the ridge leverage score to exactly α by decreasing $\mathbf{W}_{i,i}$

sufficiently. It is clear from the interpretation of Definition 1 given in (5.3) that decreasing $\mathbf{W}_{i,i}$, which corresponds to decreasing the weight of row i of $\mathbf{B}$, only increases the ridge leverage scores of other rows. So, any reweighted row will always maintain leverage score $\geq \alpha$ as other rows are reweighted. Theorem 2 of [49] demonstrates rigorously that the reweighted rows' leverage scores in fact converge to α. Further, since $\mathbf{W} \preceq \mathbf{I}$, it is simple to show:

$$\sum_i \ell_i^\lambda(\mathbf{WKW}) \leq \sum_i \ell_i^\lambda(\mathbf{K}) \stackrel{\text{def}}{=} \sum_i \ell_i^\lambda.$$

Thus, since each reweighted row has $\ell_i^\lambda(\mathbf{WKW}) \geq \alpha$, $\alpha|\{i : \mathbf{W}_{i,i} < 1\}| \leq \sum_i \ell_i^\lambda$ and so:

$$|\{i : \mathbf{W}_{i,i} < 1\}| \leq \frac{1}{\alpha} \sum_i \ell_i^\lambda = 32 \log\left(\sum \ell_i^\lambda/\delta\right) \cdot \sum \ell_i^\lambda. \tag{5.8}$$

We can now bound $\sum_i p_i$. For any i that is reweighted by $\mathbf{W}$ we just trivially bound $p_i \leq 1$. Since $\ell_i^\lambda(\mathbf{WKW}) \leq \frac{1}{2} \cdot \frac{1}{16 \log(\sum \ell_i^\lambda/\delta)}$ for all i, and since $\mathbf{S}$ samples each i with probability $1/2$, by the matrix Bernstein bound (see Lemma 11 and Theorem 12 of [5]), with probability $1 - \delta/2$:

$$\frac{1}{2}(\mathbf{B}^\top \mathbf{W}^2 \mathbf{B} + \lambda \mathbf{I}) \preceq (\mathbf{B}^\top \mathbf{WSS}^\top \mathbf{WB} + \lambda \mathbf{I}) \preceq \frac{3}{2}(\mathbf{B}^\top \mathbf{W}^2 \mathbf{B} + \lambda \mathbf{I}).$$

Hence:

$$\begin{aligned}
\tilde{\ell}_i^\lambda = \mathbf{b}_i^\top (\mathbf{B}^\top \mathbf{SS}^\top \mathbf{B} + \lambda \mathbf{I})^{-1} \mathbf{b}_i &\leq \mathbf{b}_i^\top (\mathbf{B}^\top \mathbf{WSS}^\top \mathbf{WB} + \lambda \mathbf{I})^{-1} \mathbf{b}_i \\
&\leq 2\mathbf{b}_i^\top (\mathbf{B}^\top \mathbf{W}^2 \mathbf{B} + \lambda \mathbf{I})^{-1} \mathbf{b}_i \\
&= 2\ell_i^\lambda(\mathbf{WBB}^\top \mathbf{W}) = 2\ell_i^\lambda(\mathbf{WKW}).
\end{aligned}$$

Again using that $\mathbf{W} \preceq \mathbf{I}$ and Lemma 19 from [5], $\sum_{\{i:\mathbf{W}_{i,i}=1\}} \tilde{\ell}_i^\lambda \leq 2 \sum_i \ell_i^\lambda$. Finally the

second bound is established:

$$\begin{aligned}\sum_i p_i &= \sum_{\{i:\mathbf{W}_{i,i}<1\}} p_i + \sum_{\{i:\mathbf{W}_{i,i}=1\}} p_i \\ &\leq |\{i : \mathbf{W}_{i,i} < 1\}| + 32\log\left(\sum \ell_i^\lambda/\delta\right)\cdot\sum_i \ell_i^\lambda \\ &= 64\log\left(\sum \ell_i^\lambda/\delta\right)\cdot\sum_i \ell_i^\lambda. \quad \blacksquare\end{aligned}$$

5.2.3 Efficiently computing approximate ridge leverage scores for a sample

In order to utilize Lemma 2 we must show how to efficiently compute $\tilde{\ell}_i^\lambda$ via formula (5.7) without explicitly forming either $\mathbf{K}$ or $\mathbf{B}$. We prove the following:

Lemma 3 [5]. For any sampling matrix $\mathbf{S} \in \mathbb{R}^{n\times s}$, and any $\lambda > 0$:

$$\tilde{\ell}_i^\lambda \stackrel{\text{def}}{=} \mathbf{b}_i^\top(\mathbf{B}^\top\mathbf{S}\mathbf{S}^\top\mathbf{B} + \lambda\mathbf{I})^{-1}\mathbf{b}_i = \frac{1}{\lambda}\left(\mathbf{K} - \mathbf{K}\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S} + \lambda\mathbf{I})^{-1}\mathbf{S}^\top\mathbf{K}\right)_{i,i}. \tag{5.9}$$

It follows that we can compute over-estimates $\tilde{\ell}_i^\lambda$ for all i in $\mathcal{O}(ns^2)$ time using just $\mathcal{O}(ns)$ kernel evaluations, to compute $\mathbf{KS}$ and the diagonal of $\mathbf{K}$.

Proof [5]. Using the SVD write $\mathbf{S}^\top\mathbf{B} = \bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^\top$. $\bar{\mathbf{V}} \in \mathbb{R}^{n\times s}$ forms an orthonormal basis for the row span of $\mathbf{S}^\top\mathbf{B}$. Let $\bar{\mathbf{V}}_\perp$ be span for the nullspace of $\mathbf{S}^\top\mathbf{B}$. Then we can rewrite $\tilde{\ell}_i^\lambda$ as:

$$\tilde{\ell}_i^\lambda = \mathbf{b}_i^\top(\mathbf{B}^\top\mathbf{S}\mathbf{S}^\top\mathbf{B} + \lambda\mathbf{I})^{-1}\mathbf{b}_i = \mathbf{b}_i^\top[\bar{\mathbf{V}}, \bar{\mathbf{V}}_\perp](\bar{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1}[\bar{\mathbf{V}}, \bar{\mathbf{V}}_\perp]^\top\mathbf{b}_i.$$

Here we abuse notation a by letting $\bar{\mathbf{\Sigma}}$ represent an $n \times n$ diagonal matrix whose first s entries are the singular values of $\mathbf{S}^\top\mathbf{B}$ and whose remaining entries are all equal to 0. Now:

$$\tilde{\ell}_i^\lambda = \mathbf{b}_i^\top[\bar{\mathbf{V}}, \bar{\mathbf{V}}_\perp](\bar{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1}[\bar{\mathbf{V}}, \bar{\mathbf{V}}_\perp]^\top\mathbf{b}_i = \frac{1}{\lambda}\mathbf{b}_i^\top\bar{\mathbf{V}}_\perp^\top\bar{\mathbf{V}}_\perp\mathbf{b}_i + \mathbf{b}_i^\top\bar{\mathbf{V}}(\bar{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1}\bar{\mathbf{V}}^\top\mathbf{b}_i. \tag{5.10}$$

Focusing on the second term of (5.10),

$$\begin{aligned}\mathbf{b}_i^\top\bar{\mathbf{V}}(\bar{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1}\bar{\mathbf{V}}^\top\mathbf{b}_i &= \mathbf{b}_i^\top\bar{\mathbf{V}}\frac{1}{\lambda}(\mathbf{I} - \bar{\mathbf{\Sigma}}^2(\bar{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1})\bar{\mathbf{V}}^\top\mathbf{b}_i \\ &= \frac{1}{\lambda}\mathbf{b}_i^\top\bar{\mathbf{V}}\bar{\mathbf{V}}^\top\mathbf{b}_i - \frac{1}{\lambda}\mathbf{b}_i^\top\bar{\mathbf{V}}(\bar{\mathbf{\Sigma}}^2(\bar{\mathbf{\Sigma}}^2 + \lambda\mathbf{I})^{-1})\bar{\mathbf{V}}^\top\mathbf{b}_i. \end{aligned}\tag{5.11}$$

Focusing on the second term of (5.11),

$$\begin{aligned}\mathbf{b}_i^\top \bar{\mathbf{V}}(\bar{\mathbf{\Sigma}}^2(\bar{\mathbf{\Sigma}}^2+\lambda\mathbf{I})^{-1})\bar{\mathbf{V}}^\top\mathbf{b}_i &= \mathbf{b}_i^\top\bar{\mathbf{V}}\bar{\mathbf{\Sigma}}\bar{\mathbf{U}}^\top\bar{\mathbf{U}}(\bar{\mathbf{\Sigma}}^2+\lambda\mathbf{I})^{-1}\bar{\mathbf{U}}^\top\bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^\top\mathbf{b}_i^\top \\ &= \mathbf{b}_i^\top\mathbf{B}^\top\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S}+\lambda\mathbf{I})^{-1}\mathbf{S}^\top\mathbf{B}\mathbf{b}_i.\end{aligned}$$

Substituting back into (5.11) and then (5.10), we conclude that:

$$\begin{aligned}\tilde{\ell}_i^\lambda &= \frac{1}{\lambda}\mathbf{b}_i^\top\bar{\mathbf{V}}_\perp^\top\bar{\mathbf{V}}_\perp\mathbf{b}_i + \frac{1}{\lambda}\mathbf{b}_i^\top\bar{\mathbf{V}}\bar{\mathbf{V}}^\top\mathbf{b}_i - \frac{1}{\lambda}\mathbf{b}_i^\top\mathbf{B}^\top\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S}+\lambda\mathbf{I})^{-1}\mathbf{S}^\top\mathbf{B}\mathbf{b}_i \\ &= \frac{1}{\lambda}\mathbf{b}_i^\top\mathbf{b}_i - \frac{1}{\lambda}\mathbf{b}_i^\top\mathbf{B}^\top\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S}+\lambda\mathbf{I})^{-1}\mathbf{S}^\top\mathbf{B}\mathbf{b}_i \\ &= \frac{1}{\lambda}\mathbf{K}_{i,i} - \frac{1}{\lambda}\left(\mathbf{K}\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S}+\lambda\mathbf{I})^{-1}\mathbf{S}^\top\mathbf{K}\right)_{i,i}.\end{aligned}$$

We can compute $(\mathbf{S}^\top\mathbf{K}\mathbf{S}+\lambda\mathbf{I})^{-1}$ in $\mathcal{O}(s^3) \leq \mathcal{O}(ns^2)$ time and $\mathcal{O}(s^2) \leq \mathcal{O}(ns)$ kernel evaluations. Given this inverse, computing the diagonal entries of $\mathbf{K}\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S}+\lambda\mathbf{I})^{-1}\mathbf{S}^\top\mathbf{K}$ requires just $\mathcal{O}(ns)$ kernel evaluations to form $\mathbf{K}\mathbf{S}$ and $\mathcal{O}(ns^2)$ time to perform the necessary multiplications. Finally, computing the diagonal entries of $\mathbf{K}$ requires n additional kernel evaluations. ■

Now we are ready to use Lemmas 2 and 3 to give an efficient recursive method for ridge leverage score Nyström approximation. We show that the output of Algorithm 2 (below), $\mathbf{S}$, is sampled according to approximate ridge leverage scores for $\mathbf{K}$ and so satisfies the approximation bound of (5.5).

Theorem 2 (Main Result from [5]). Let $\mathbf{S} \in \mathbb{R}^{n\times s}$ be computed by Algorithm 2. With probability $1-3\delta$, $s \leq 384 d_{eff}^\lambda \log(d_{eff}^\lambda/\delta)$, $\mathbf{S}$ is sampled by overestimates of the λ -ridge leverage scores of $\mathbf{K}$, and thus by Theorem 1, the Nyström approximation $\tilde{\mathbf{K}} = \mathbf{K}\mathbf{S}(\mathbf{S}^\top\mathbf{K}\mathbf{S})^+\mathbf{S}^\top\mathbf{K}$ satisfies:

$$\tilde{\mathbf{K}} \preceq \mathbf{K} \preceq \tilde{\mathbf{K}} + \lambda\mathbf{I}.$$

Algorithm 2 uses $\mathcal{O}(ns)$ kernel evaluations and $\mathcal{O}(ns^2)$ computation

Proof. The proof of this result extends beyond the scope of this work but can be found to be a corollary of Theorem 8 from [5].

A recursive version of Algorithm 1 is presented below [5]. This algorithm uses over-approximations of true ridge leverage scores allowing the same convergence guarantees as traditional RLS-Nyström sampling but admitting much lower computational complexity. This algorithm is summarized below.

Algorithm 2 RECURSIVE RLS-NYSTRÖM Algorithm

Input: $\mathbf{x}_1, ..., \mathbf{x}_m \in \mathcal{X}$, Kernel Matrix $\mathbf{K}$, ridge $\lambda > 0$, failure prob. $\delta \in (0, 1/32)$

Output Weighted sampling matrix $\mathbf{S} \in \mathbb{R}^{m \times s}$

1: **if** $m \leq 192 \log(1/\delta)$ **then**

  return $\mathbf{S} := \mathbf{I}_{m \times m}$

  end if

2: Let $\bar{\mathcal{S}}$ be a random subset of $\{1, ..., m\}$, with each i included indep. with prob. $1/2$

  Let $\bar{\mathbf{X}} = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, ..., \mathbf{x}_{i|\bar{\mathcal{S}}|}\}$ for $i_j \in \bar{\mathcal{S}}$ be the data sample corresponding to $\bar{\mathcal{S}}$

  Let $\bar{\mathbf{S}} = [\mathbf{e}_{i1}, \mathbf{e}_{i2}, ..., \mathbf{e}_{i|\bar{\mathcal{S}}|}]$

3: $\tilde{\mathbf{S}} :=$ RECURSIVE RLS-NYSTRÖM$(\bar{\mathbf{X}}, K, \lambda, \delta/3)$

4: $\hat{\mathbf{S}} := \bar{\mathbf{S}}\tilde{\mathbf{S}}$

5: Set $\tilde{\ell}_i^{\lambda} := \frac{3}{2\lambda}\left(\mathbf{K} - \mathbf{K}\hat{\mathbf{S}}\left(\hat{\mathbf{S}}^\top \mathbf{K}\hat{\mathbf{S}} + \lambda \mathbf{I}\right)^{-1}\hat{\mathbf{S}}^\top \mathbf{K}\right)_{i,i}$ for each $i \in \{1, ..., m\}$

6: Set $p_i := \min\{1, \tilde{\ell}_i^{\lambda} \cdot 16 \log(\sum \tilde{\ell}_i^{\lambda}/\delta)\}$ for each $i \in \{1, ..., m\}$

7: Initially set weighted sampling matrix $\mathbf{S}$ to be empty. For each $i \in \{1, ..., m\}$, with probability p_i, append the column $\frac{1}{\sqrt{p_i}}\mathbf{e}_i$ onto $\mathbf{S}$

8: **return S**.

Alternatively, if we have a fixed sampling budget s in mind and we want to utilize this full budget, we can pick elements with this fixed sample size s, by controlling λ using the following fact [5]:

For any $\mathbf{K}$ and integer k, setting $\lambda = \frac{1}{k}\sum_{i=k+1}^{n}\sigma_i(\mathbf{K})$, ensures $d_{eff}^{\lambda} \leq 2k$.

If we choose k such that $s \approx k\log k$ then setting λ as above will yield an RLS-Nyström approximation with approximately s sampled columns. The fixed sample budget version of this recursive algorithm is described below:

Algorithm 3 RECURSIVE RLS-NYSTRÖM Sampling Algorithm, Fixed Sample Size

---

Input: $\mathbf{x}_1, ..., \mathbf{x}_m \in \mathcal{X}$, Kernel Function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, sample size s, failure prob. $\delta \in (0, 1/32)$

Output Weighted sampling matrix $\mathbf{S} \in \mathbb{R}^{m \times s'}$

1: **if** $m \leq s$ **then**

return $\mathbf{S} := \mathbf{I}_{m\times m}$

end if

2: Let $\bar{\mathcal{S}}$ be a random subset of $\{1, ..., m\}$, with each i included indep. with prob. $1/2$

Let $\bar{\mathbf{X}} = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, ..., \mathbf{x}_{i|\bar{\mathcal{S}}|}\}$ for $i_j \in \bar{\mathcal{S}}$ be the data sample corresponding to $\bar{\mathcal{S}}$

Let $\bar{\mathbf{S}} = [\mathbf{e}_{i1}, \mathbf{e}_{i2}, ..., \mathbf{e}_{i|\bar{\mathcal{S}}|}]$

3: $\tilde{\mathbf{S}} :=$ RECURSIVERLS-NYSTRÖM$(\bar{\mathbf{X}}, K, s, \delta/3)$

4: $\hat{\mathbf{S}} := \bar{\mathbf{S}}\tilde{\mathbf{S}}$

5: Set k to the maximum integer with $ck\log(2k/\delta) \leq s$, where c is some fixed constant.

6: $\tilde{\lambda} := \frac{1}{k}\sum_{i=k+1}^{n}\sigma_i(\hat{\mathbf{S}}^\top\mathbf{K}\hat{\mathbf{S}})$

7: Set $\tilde{\ell}_i^{\lambda} := \frac{5}{\tilde{\lambda}}\left(\mathbf{K} - \mathbf{K}\hat{\mathbf{S}}\left(\hat{\mathbf{S}}^\top\mathbf{K}\hat{\mathbf{S}} + \tilde{\lambda}\mathbf{I}\right)^{-1}\hat{\mathbf{S}}^\top\mathbf{K}\right)_{i,i}$ for each $i \in \{1, ..., m\}$

8: Set $p_i := \min\{1, \tilde{\ell}_i^{\lambda} \cdot 16\log(2k/\delta)\}$ for each $i \in \{1, ..., m\}$

9: Initially set weighted sampling matrix $\mathbf{S}$ to be empty. For each $i \in \{1, ..., m\}$, with probability p_i, append the column $\frac{1}{\sqrt{p_i}}\mathbf{e}_i$ onto $\mathbf{S}$

10: **return S**.

5.3 Numerical Examples

In this section we demonstrate numerical examples which exemplify the reconstructive capabilities of the recursive RLS sampling strategies. A comparison is made between the norm of the spectral error in the approximation $||\mathbf{K} - \tilde{\mathbf{K}}||_2$ for a uniform sampling approach versus an ridge leverage score sampling approach. The comparison is made over a range of fixed sample budgets $s \in [200, 4000]$ with steps of 200 samples. The datasets utilized in this testing are the MNIST digits and the TREX13 sonar datasetThe full kernel matrix, i.e. the matrix we are approximating, features inner-products among 10000 samples from a given dataset. Sampling experiments were run a total of 5 times and the plots in Figures 5.1-5.2 display the average Spectral Norm of the error for the Uniform Sampling Approach and the Recursive RLS sampling approach. In these experiments, Algorithm 3 is utilized so that the number of samples can be fixed and fair comparison can be made between the Uniform and RLS sampling approaches. For each experiment, a gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) = \exp(\frac{-||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ was utilized. The spread parameter σ was set to $\sigma = 0.1$ for the MNIST dataset and to $\sigma = 2.0$ for the TREX13 dataset.

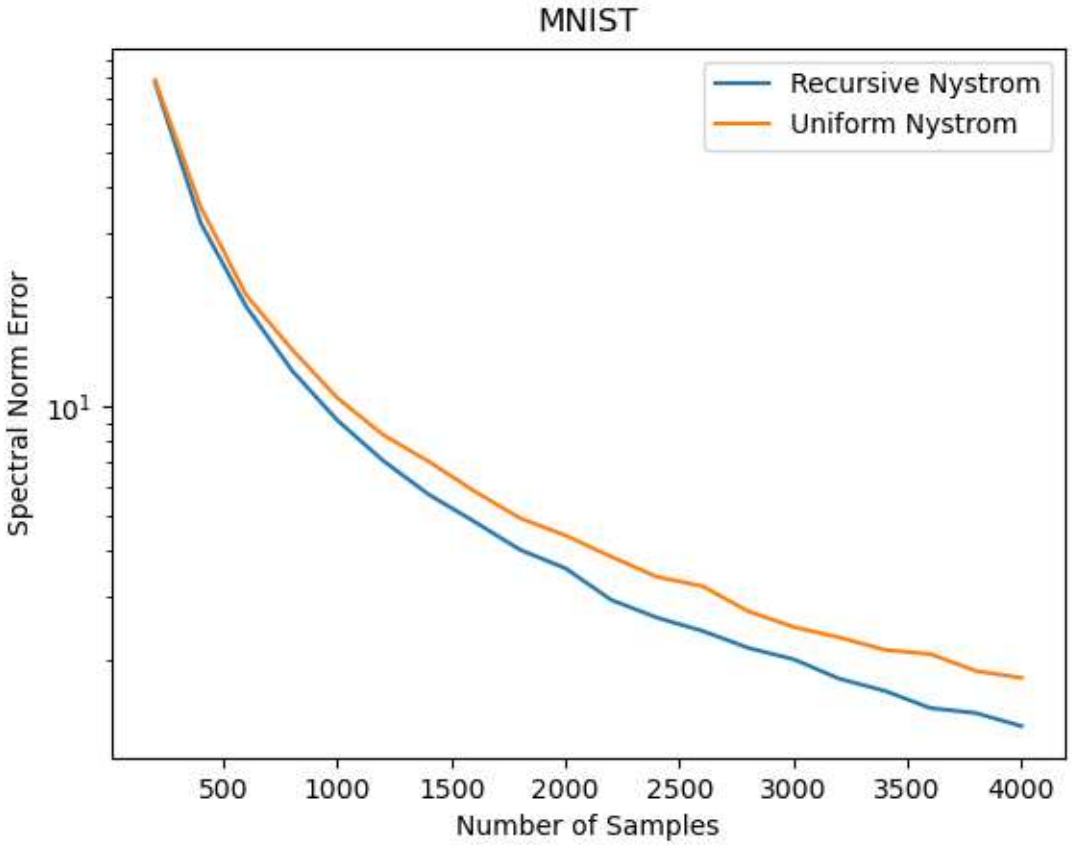


Figure 5.1: Spectral Error for Sample Budget $s \in [200, 4000]$ MNIST Dataset

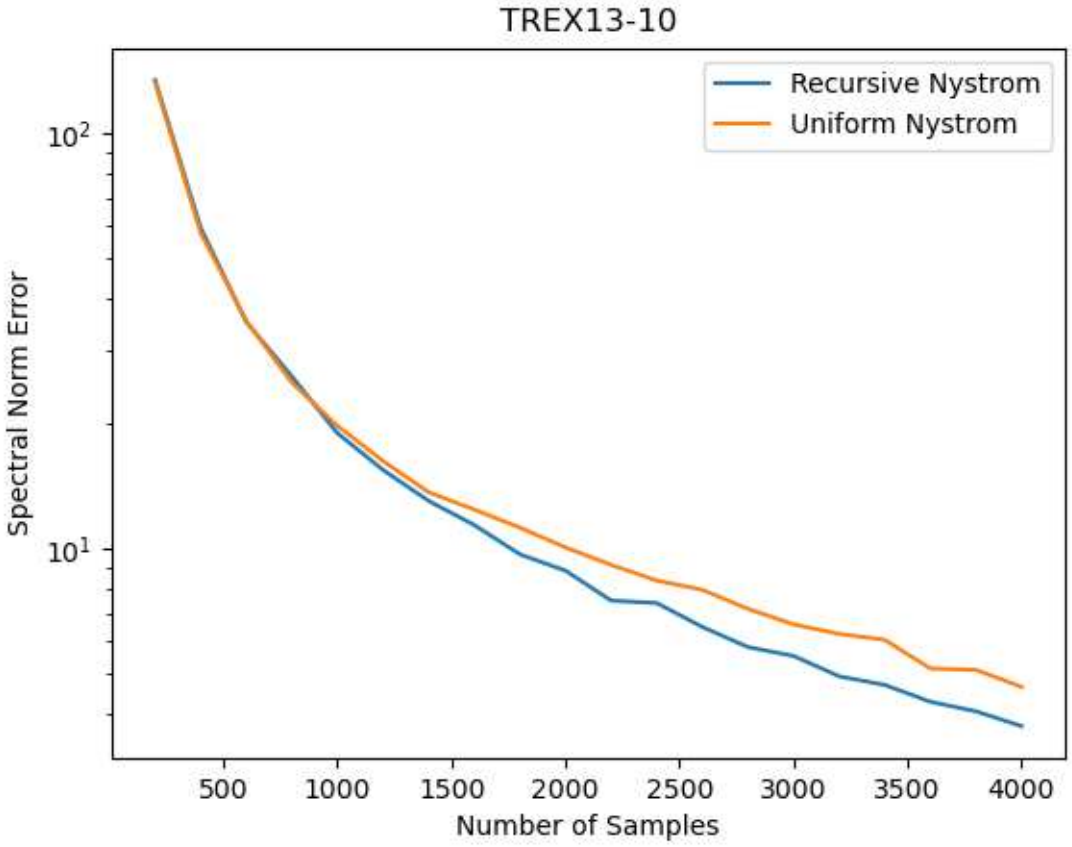


Figure 5.2: Spectral Error for Sample Budget $s \in [200, 4000]$ TREX13 Sonar Dataset

From Figures 5.1-5.2 it is clear that the RLS sampling strategy consistently out-performs that of a uniform sampling approach when datasets grow large and have considerable variability as is the case with each of the datasets that were tested.

5.4 Conclusion

In this chapter we provided an overview of several approaches to performing column sampling for the Nyström approximation in Chapter 4. We reviewed 3 simple approaches that are commonly used which involve sampling from a variety of uniform and non-uniform distributions. Then, we described a novel technique from [5] which allows ridge leverage scores to be compute in $\mathcal{O}(ns)$ kernel evaluations and $\mathcal{O}(ns^2)$ additional runtime. Leverage score sampling has long been known to give strong theoretical guarantees for the Nyström approximation, by employing a recursive sampling scheme [5], the algorithm described is the first to make the approach scalable and practical for exceedingly large datasets. Furthermore, through several numerical examples we demonstrated empirically that this technique finds more accurate, low-rank kernel approximations than the traditional uniform sampling approach from [3]. Considering all of these benefits, RLS-Nyström is a strong contender for sampling in a lifelong learning setting.

CHAPTER 6

INCREMENTAL LINEARIZED KERNEL EMBEDDING

6.1 Introduction

In the realm of machine vision and pattern recognition, there are countless problems which require a machine to continuously adapt over its lifetime to accommodate distribution changes of data samples coming from environments with varying conditions. Lifelong learning approaches seek to provide mechanisms for a machine to continually adapt to arriving data without inducing catastrophic forgetting or interference on examples from previous environments. The scenario primarily addressed in this work is the problem of underwater target classification [2, 8, 50–52] where a system faces the challenging task of identifying buried, or partially buried targets from sonar returns at several observation points. The targets can be found in a wide variety of environmental and operational conditions, making the need for model adaptation a concern of paramount importance.

To this end, this chapter presents an enhancement to the theory of LKDL for Lifelong Learning. In this chapter, we discuss two incremental procedures for updating the linearized embedding when new important samples are made available. The first is the method of Hallgren and Northrop [53] which relies upon two rank-one eigenupdates. The second is our novel method, inspired by the work in [53], which we refer to as Symmetric Arrowhead Updating (SAU). This method uses Arrowhead eigendecompositions [54] for faster updating of the linearized kernel embedding. Complexity analysis of the method and a comparison with that of the method in [53] is also presented. The proposed method, like that of [53], can also be applied to standard kernel PCA [55]. Lastly we demonstrate the consistency of our

incremental eigenupdates procedure with a standard batch SVD of growing gram matrices.

6.2 Updating Embedding with New Important Samples

In this section, we consider a scenario where the system trained with the original c samples needs to be updated when a new important sample $\mathbf{z}_{\text{new}} = \mathbf{z}_{c+1}$ is added. That is, the augmented data matrix of the chosen samples and its nonlinearly mapped version become $\tilde{\mathbf{Z}}_R = [\mathbf{Z}_R, \mathbf{z}_{c+1}] \in \mathbb{R}^{d\times(c+1)}$ and $\tilde{\boldsymbol{\Phi}}_R = \begin{bmatrix}\boldsymbol{\phi}(\mathbf{z}_1) & \dots & \boldsymbol{\phi}(\mathbf{z}_c) & \boldsymbol{\phi}(\mathbf{z}_{c+1})\end{bmatrix} \in \mathbb{R}^{D\times(c+1)}$, respectively. It should be noted that the procedures presented here are not limited to an important samples (Nyström approximation) approach, but can be used for standard kernel PCA as well.

Now, we desire to find a new set of virtual samples, $\tilde{\mathbf{F}}$, that satisfies

$$\tilde{\mathbf{F}} = \tilde{\boldsymbol{\Sigma}}^{-1/2}\tilde{\mathbf{V}}^\top \tilde{\mathbf{C}}^\top, \tag{6.1}$$

where $\tilde{\boldsymbol{\Sigma}}$ and $\tilde{\mathbf{V}}$ are traditionally computed by applying the eigenvalue decomposition (EVD) to the corresponding kernel matrix $\tilde{\mathbf{W}} = \tilde{\boldsymbol{\Phi}}_R^\top \tilde{\boldsymbol{\Phi}}_R$. More specifically,

$$\tilde{\mathbf{W}} = \tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}\tilde{\mathbf{V}}^\top = \begin{bmatrix}\mathbf{W} & \boldsymbol{\rho} \\ \boldsymbol{\rho}^\top & \kappa_{c+1,c+1}\end{bmatrix}, \tag{6.2}$$

where $\boldsymbol{\rho} = \mathbf{k}(\mathbf{Z}_R, \mathbf{z}_{c+1}) = [\kappa_{1,c+1}\ \kappa_{2,c+1}\ \dots\ \kappa_{c,c+1}]^\top$ with $\kappa_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$. However, here the goal is to accomplish this without resorting to any computationally demanding EVD algorithm.

Also, $\tilde{\mathbf{C}}$ in (6.1) is given by $\tilde{\mathbf{C}} = [\tilde{\mathbf{W}}^\top, \tilde{\mathbf{A}}^\top]^\top \in \mathbb{R}^{(N+1)\times(c+1)}$, where $\tilde{\mathbf{A}} = [\mathbf{A}, \boldsymbol{\zeta}] \in \mathbb{R}^{(N-c)\times(c+1)}$, with $\boldsymbol{\zeta} = \mathbf{k}(\mathbf{Z}_U, \mathbf{z}_{c+1}) = \begin{bmatrix}\kappa(\mathbf{z}_{U,1}, \mathbf{z}_{c+1}) & \dots & \kappa(\mathbf{z}_{U,N-c}, \mathbf{z}_{c+1})\end{bmatrix}^\top$ and $\mathbf{z}_{U,i} \in \mathbf{Z}_U = \mathbf{Z} \setminus \mathbf{Z}_R$ i.e. sample set containing $N - c$ unimportant samples discarded as a result of the sampling.

The first approach for the incremental embedding update presented in this section uses the information about the current $\mathbf{W}$ matrix, and its eigenpairs, to compute the new eigenpairs of $\tilde{\mathbf{W}}$. This is accomplished by utilizing the rank-one update procedure in [53, 56].

Let us define $\boldsymbol{\nu}_1 = [\boldsymbol{\rho}^\top \ \frac{1}{2}\kappa_{c+1,c+1}]^\top$, $\boldsymbol{\nu}_2 = [\boldsymbol{\rho}^\top \ \frac{1}{4}\kappa_{c+1,c+1}]^\top$, and $\sigma = 4/\kappa_{c+1,c+1}$. Then, it can be shown [53] that $\tilde{\mathbf{W}}$ can be expressed as

$$\begin{aligned}\tilde{\mathbf{W}} &= \begin{bmatrix}\mathbf{W} & \mathbf{0} \\ \mathbf{0}^\top & \frac{1}{4}\kappa_{c+1,c+1}\end{bmatrix} + \sigma\boldsymbol{\nu}_1\boldsymbol{\nu}_1^\top - \sigma\boldsymbol{\nu}_2\boldsymbol{\nu}_2^\top, \\ &= \mathbf{W}^0 + \sigma\boldsymbol{\nu}_1\boldsymbol{\nu}_1^\top - \sigma\boldsymbol{\nu}_2\boldsymbol{\nu}_2^\top \end{aligned} \tag{6.3}$$

corresponding to a diagonal expansion of $\mathbf{W}$ and two rank one updates, where $\mathbf{0}$ is a column vector of zeros of appropriate dimension. As can be seen from (6.3) the matrix $\mathbf{W}^0$ has an additional eigenvalue $\frac{1}{4}\kappa_{c+1,c+1}$ and corresponding eigenvector

Given the eigendecomposition of the symmetric matrix $\mathbf{W}^0 = \mathbf{V}^0\boldsymbol{\Sigma}^0{\mathbf{V}^0}^\top$ where $\mathbf{V}^0$ is an orthogonal matrix, we define the *perturbed* matrix

$$\begin{aligned}\mathbf{B} &= \mathbf{V}^0\boldsymbol{\Sigma}^0{\mathbf{V}^0}^\top + \sigma\boldsymbol{\nu}\boldsymbol{\nu}^\top \\ &= \mathbf{V}^0(\boldsymbol{\Sigma}^0 + \sigma\boldsymbol{\tau}\boldsymbol{\tau}^\top){\mathbf{V}^0}^\top\end{aligned}$$

where $\boldsymbol{\tau} = {\mathbf{V}^0}^\top\boldsymbol{\nu}$. Using the eigendecomposition of $\boldsymbol{\Sigma}^0 + \sigma\boldsymbol{\tau}\boldsymbol{\tau}^\top = \boldsymbol{\mathcal{U}}\boldsymbol{\Sigma}^1\boldsymbol{\mathcal{U}}^\top$ [57], the eigendecomposition of $\mathbf{B}$ can be found by $\mathbf{B} = \mathbf{V}^1\boldsymbol{\Sigma}^1{\mathbf{V}^1}^\top$ where $\mathbf{V}^1 := \mathbf{V}^0\boldsymbol{\mathcal{U}}$. Since both $\mathbf{V}^0$ and $\boldsymbol{\mathcal{U}}$ are orthogonal matrices, $\mathbf{V}^1$ will also be an orthogonal matrix. The eigenvalues of $\mathbf{B}$ can be computed by finding the roots of the secular equation [58]. The eigenvectors of the perturbed matrix $\mathbf{B}$ are given by [57]

$$\mathbf{v}_i^1 = \frac{\mathbf{V}^0\mathbf{D}_i^{-1}\boldsymbol{\tau}}{||\mathbf{D}_i^{-1}\boldsymbol{\tau}||} \tag{6.4}$$

where $\mathbf{D}_i := \boldsymbol{\Sigma}^0 - \lambda_i'\mathbf{I}$ where λ_i' is the i^{th} eigenvalue of $\mathbf{B}$. This method exhibits some numerical stability issues for invertible, but poorly conditioned matrices. Alternatively, a stabilized approach, e.g., the one in [56], can be utilized for applications where many rank one iterations are necessary. This approach is summarized in Algorithm 1 below.

Algorithm 1 Incremental Eigendecomposition of kernel matrix [53]

Require: The augmented data matrix $\tilde{\mathbf{Z}}_R$, matrices $\mathbf{V}$ and $\mathbf{\Sigma}$ for kernel matrix $\mathbf{W}$ and the kernel function $\kappa(\cdot,\cdot)$.

Ensure: Eigenvalue and eigenvector matrices $\tilde{\mathbf{\Sigma}}$ and $\tilde{\mathbf{V}}$ of $\tilde{\mathbf{W}}$

$\mathbf{\Sigma}^0 \leftarrow \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0}^\top & \kappa_{c+1,c+1}/4 \end{bmatrix}$

$\mathbf{V}^0 \leftarrow \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$

$\sigma \leftarrow 4/\kappa_{c+1,c+1}$

$\boldsymbol{\nu}_1 \leftarrow [\kappa_{1,c+1}, \kappa_{2,c+1}, \ldots, \kappa_{c+1,c+1}/2]$

$\boldsymbol{\nu}_2 \leftarrow [\kappa_{1,c+1}, \kappa_{2,c+1}, \ldots, \kappa_{c+1,c+1}/4]$

$\mathbf{\Sigma}^1, \mathbf{V}^1 \leftarrow \text{rankoneupdate}(\sigma, \boldsymbol{\nu}_1, \mathbf{\Sigma}^0, \mathbf{V}^0)$

$\tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}} \leftarrow \text{rankoneupdate}(-\sigma, \boldsymbol{\nu}_2, \mathbf{\Sigma}^1, \mathbf{V}^1)$

6.2.2 An Alternate Approach using Arrowhead Updates

In this section we present a novel approach to solving the expanding eigendecomposition problem using arrowhead matrix eigendecompositions [54]. In this approach, the update from the eigendecomposition of $\mathbf{W}$ to that of $\tilde{\mathbf{W}}$ can be done by noting that the addition made to $\mathbf{W}^0$ to form $\tilde{\mathbf{W}}$ yields an addition to the eigenvalue matrix of $\mathbf{W}^0$ resulting in an arrowhead matrix. This arrowhead matrix's eigendecomposition can be obtained more efficiently compared to the method in the previous section which uses two consecutive rank-one updates. The eigendecomposition of this arrowhead matrix is directly related to the eigendecomposition of $\tilde{\mathbf{W}}$ through $\mathbf{W}^0$'s eigenvectors matrix.

As before, we represent $\mathbf{W} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^\top$ and

$\tilde{\mathbf{W}} = \mathbf{W}^0 + \mathbf{\Gamma}$, where $\mathbf{W}^0$ was defined before and

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{0} & \boldsymbol{\rho} \\ \boldsymbol{\rho}^\top & \frac{3}{4}\kappa_{c+1,c+1} \end{bmatrix}$$

We also noted that $\mathbf{W}^0 = \mathbf{V}^0\mathbf{\Sigma}^0{\mathbf{V}^0}^\top$ where

$$\mathbf{V}^0 = \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

and

$$\boldsymbol{\Sigma}^0 = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0}^\top & \frac{1}{4}\kappa_{c+1,c+1} \end{bmatrix}$$

Now, it is easy to see that ${\mathbf{V}^0}^\top \boldsymbol{\Gamma} \mathbf{V}^0$ has the following form

$${\mathbf{V}^0}^\top \boldsymbol{\Gamma} \mathbf{V}^0 = \begin{bmatrix} \mathbf{0} & \mathbf{V}^\top \boldsymbol{\rho} \\ \boldsymbol{\rho}^\top \mathbf{V} & \frac{3}{4}\kappa_{c+1,c+1} \end{bmatrix}$$

Note that this is nearly an arrowhead matrix, all it is missing is the diagonal. Therefore, using a similar approach to a typical rank one perturbation problem, we can factor $\tilde{\mathbf{W}}$ as

$$\begin{aligned} \tilde{\mathbf{W}} &= \mathbf{W}^0 + \boldsymbol{\Gamma} \\ &= \mathbf{V}^0 \boldsymbol{\Sigma}^0 {\mathbf{V}^0}^\top + \boldsymbol{\Gamma} \\ &= \mathbf{V}^0 (\boldsymbol{\Sigma}^0 + {\mathbf{V}^0}^\top \boldsymbol{\Gamma} \mathbf{V}^0) {\mathbf{V}^0}^\top \end{aligned} \tag{6.5}$$

The factor in parentheses in (6.5) is an arrowhead matrix. Thus, using the method in [54], we can solve the eigendecomposition $\boldsymbol{\Sigma}^0 + {\mathbf{V}^0}^\top \boldsymbol{\Gamma} \mathbf{V}^0 = \boldsymbol{\mathcal{U}} \tilde{\boldsymbol{\Sigma}} \boldsymbol{\mathcal{U}}^\top$. Then, this eigendecomposition can be used to yield $\tilde{\mathbf{W}} = \mathbf{V}^0 (\boldsymbol{\mathcal{U}} \tilde{\boldsymbol{\Sigma}} \boldsymbol{\mathcal{U}}^\top) {\mathbf{V}^0}^\top$ which gives the eigenvector matrix $\tilde{\mathbf{V}} = \mathbf{V}^0 \boldsymbol{\mathcal{U}}$. The steps in this algorithm are outlined below in Algorithm 2. It should be noted that this symmetric arrowhead update (SAU) procedure can be used on any growing real symmetric full rank matrix $\mathbf{W}$ and is not limited to Gram matrices. Step 5 of this algorithm utilizes the arrowhead eigenvalue (aheig) decomposition (Algorithm 5) presented in [54].

Algorithm 2 SAU Eigendecomposition of growing kernel matrix

Require: Matrices $\mathbf{V}$, $\mathbf{\Sigma}$, vector $\boldsymbol{\rho}$, and scalar $\kappa_{c+1,c+1}$.
Ensure: Eigenvalue and eigenvector matrices $\tilde{\mathbf{\Sigma}}$ and $\tilde{\mathbf{V}}$ of $\tilde{\mathbf{W}}$

1: $\mathbf{\Sigma}^0 \leftarrow \begin{bmatrix} \mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0}^\top & \kappa_{c+1,c+1}/4 \end{bmatrix}$
2: $\mathbf{V}^0 \leftarrow \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
3: $\boldsymbol{\pi} \leftarrow \mathbf{V}^\top \boldsymbol{\rho}$
4: $\mathbf{\Gamma} \leftarrow \begin{bmatrix} \mathbf{0} & \boldsymbol{\pi} \\ \boldsymbol{\pi}^\top & \frac{3}{4}\kappa_{c+1,c+1} \end{bmatrix}$
5: $\boldsymbol{\mathcal{U}}, \tilde{\mathbf{\Sigma}} \leftarrow \text{aheig}(\mathbf{\Sigma}^0 + \mathbf{\Gamma})$ [54]
6: $\tilde{\mathbf{V}} \leftarrow \mathbf{V}^0 \boldsymbol{\mathcal{U}}$

6.2.3 Computational Complexity Analysis

A major motivation for the creation of the SAU updates method is the desire to reduce the computational cost of repeated model updates when varying data is arriving continuously. The proposed SAU algorithm provides a competitive approach to solving the incremental kernel PCA [36] and incremental empirical kernel map problems. A comparison of the computational cost of the SAU algorithm with that of [53] is presented in Table 6.1 which breaks down the leading terms for time and memory complexity for both methods.

As far as the SAU algorithm is concerned, step 5 relies on the arrowhead eigendecomposition of [54] which has $\mathcal{O}(c^2)$ time complexity and $\mathcal{O}(c)$ memory complexity. Step 6 is computationally the most expensive step of our procedure which involves matrix multiplication of two $c \times c$ matrices hence requiring approximately $2c^3$ operations. This cubic factor is difficult, if not impossible, to reduce, since the update procedure will always require rotating c eigenvectors. This brings our time complexity to $2c^3 + \mathcal{O}(c^2)$ and the memory complexity to $5c^2 + \mathcal{O}(c)$ due to the allocations for $\mathbf{\Sigma}^0$, $\mathbf{V}^0$, $\mathbf{\Gamma}$, $\boldsymbol{\mathcal{U}}$, and $\tilde{\mathbf{V}}$, respectively. In contrast, the method in [53] requires $4c^3 + \mathcal{O}(c^2)$ flops and also admits $\mathcal{O}(c^2)$ memory complexity. Therefore, our algorithm, like that of [53], admits $\mathcal{O}(c^3)$ time complexity and $\mathcal{O}(c^2)$ memory complexity but is approximately twice as efficient in the cubic factor. The experimental results in the next section attest to this fact.

Table 6.1: Complexity Comparison (Leading Terms)

Hallgren et. al.	**Time Complexity**	**Memory Complexity**
Step 6	$2c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
Step 7	$2c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
Total	$4c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
SAU Eig. (Ours)	**Time Complexity**	**Memory Complexity**
Step 5	$\mathcal{O}(c^2)$	$\mathcal{O}(c^2)$
Step 6	$2c^3$	$\mathcal{O}(c^2)$
Total	$2c^3 + \mathcal{O}(c^2)$	$\mathcal{O}(c^2)$

6.2.4 Computational Time and Consistency Comparison

Here, we compare the computational requirements to carry out several consecutive eigendecompositions of a growing kernel matrix $\mathbf{W}$ using the methods in Sections 6.2 and 6.2.2. More specifically, an experiment was conducted to determine which approach provides less computational time to perform a series of eigendecomposition of an expanding matrix. In this set up, all methods start with 1 sample and the corresponding Gram matrix $\mathbf{W}$ matrix is created. This matrix is then expanded by a single sample at a time, R times.

Figure 6.1 displays the time to fully compute all R for $R \in [300, 310, ..., 490, 500]$ eigendecompositions via consecutive EVD (red plot) [59], consecutive SVD (gray) [59], two Rank-One Updates (orange) [53], and the proposed symmetric arrowhead update (blue) in Section 6.2 B. As evident from this figure, for the task of incrementally updating the eigendecomposition, the SAU approach provides the quickest solutions by harnessing the information of the previous eigendecomposition steps.

Lastly, the consistency of eigenvalues obtained via the SAU updates and the standard batch SVD of the important samples are studied. Figure 6.2 gives the plot of the Frobenius norm of the difference between $\mathbf{\Sigma}_i = \mathbf{S}_i$, the (diagonal) eigenvalues matrix computed sequentially via the incremental method in Algorithm 2, and $\mathbf{\Sigma}_{svd} = \mathbf{S}_{svd}$, the eigenvalues matrix computed from decomposing all c samples simultaneously via the batch SVD. As can be seen, even after adding hundreds of important samples, the eigenvalues computed via our incremental approach are nearly identical to those obtained by performing more expensive

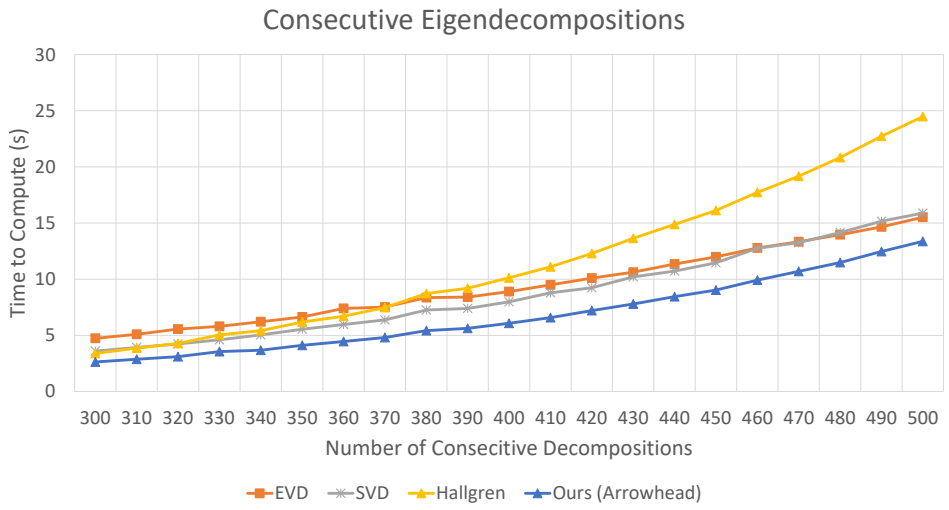


Figure 6.1: Runtime Comparison for R consecutive Eigendecompostion.

batch SVD, with the differences being on the order of 10^{-13}.

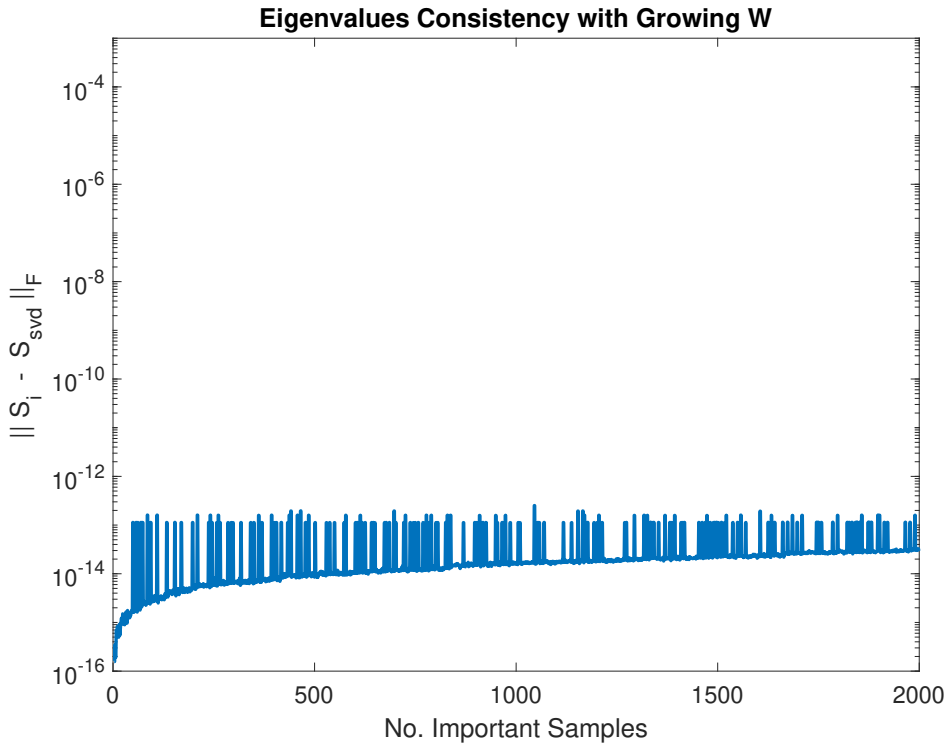


Figure 6.2: Incremental vs. Batch SVD eigenvalues for growing $\mathbf{W}$

6.2.5 Decremental update using approach from Algorithm 1

Having seen two methods for performing the incremental update to the kernel matrix approximation and linearized kernel embedding, a natural question arises: is it possible to perform a downdate (i.e. remove an important sample that has become unnecessary) using a similar procedure? While there is no apparent way to use the faster SAU updates procedure from Algorithm 2 to accomplish this, it is indeed possible to perform a downdate using two consecutive rank-1 updates as is done in Algorithm 1 at the beginning of Section 6.2. In order to perform such a downdate, the procedure would be essentially identical but performed in reverse order. Beginning with $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{\Sigma}}$, we can get the intermediate decomposition $\mathbf{V}^1, \mathbf{\Sigma}^1$ by running rankoneupdate$(\sigma, \boldsymbol{\nu}_2, \tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}})$ then to get $\mathbf{\Sigma}^0$ and $\mathbf{V}^0$ we would run rankoneupdate$(-\sigma, \boldsymbol{\nu}_1, \mathbf{\Sigma}^1, \mathbf{V}^1)$. All that remains is to then remove the eigenvalue equal to $\kappa_{c+1,c+1}/4$ from $\mathbf{\Sigma}^0$ and the corresponding eigenvector from $\mathbf{V}^0$.

6.3 Conclusion

In this chapter, we proposed a novel method for incrementally updating the linearized kernel embedding in the LKDL method from Chapter 4. We presented a new mechanism to incrementally perform eigendecomposition of a growing kernel matrix using arrowhead updates.

The computational time and consistency of the methods for eigendecomposition of a growing kernel matrix were also studied and compared. The results showed that when a model is expected to undergo many embedding updates during its lifetime, the arrowhead approach presented in this paper solves the new embedding in the most competitive time; furthermore this method is forward stable with deviations from the standard EVD or SVD computation of the eigendecomposition differing by only a small factor of the machines precision.

CHAPTER 7

OPTIMIZING KERNELS FOR DISCRIMINATION TASKS

7.1 Introduction

So far in our discussions we have presented a mechanism for kernelizing several linear learning algorithms. Chapter 4 demonstrated that a linearized kernel embedding (LKE) can be used to provide an efficient kernel learning algorithm that relies on only a handful of representative samples. In Chapter 5 we discussed mechanisms to pick out highly representative samples for the LKE method [5, 44]. The most appealing of which seeks out those samples which are the most unique (in linear independence sense) when viewed in the feature space, choosing such samples provides strong reconstruction guarantees for the Nyström approximation associated with the LKE. Later, in Chapter 6, we showed an efficient mechanism to expand the LKE by leveraging fast arrowhead eigendecompositions.

These tools provide excellent foundations for efficient lifelong learning systems but only partly address our ultimate goal of developing an efficient lifelong learning for *pattern classification* tasks. Without a feature space that, in some way, encodes the class-membership information into the relative geometry of the embedded samples, there is no guarantee that low-error reconstructions will correspond to good class-separability. With this in mind, in this chapter we turn our attention to the design of kernel functions themselves with the objective of formulating kernel functions to be utilized with the LKE which promote class-separability by minimizing a discrimination-based objective function [15]. We demonstrate three techniques in total which can be utilized for achieving this goal, two of which are

suitable for M-ary classifications problems.

This chapter is organized as follows. In Sections 7.2-7.5 we first overview the method presented in [15] which allows for optimizing a kernel function in order to provide better separability in the kernel feature space for two-class (binary) classification problems. Next in Section 7.6 we extend this method to the M-Ary classification case and formalize the learning algorithm for this scenario. Then, in Section 7.7 a novel multi-kernel discriminative kernel learning is introduced that utilizes the same objective function as the one used the in previous algorithms but utilizes a different assumption on how the kernel function is represented. Lastly, we provide numerical examples using several toy datasets to demonstrate the final kernel learning method.

7.2 Feature Space vs. Empirical Feature Space

The map from the input data space to an c-dimensional Euclidean space $\mathbf{\Phi}_c^e : \mathcal{X} \to \mathbb{R}^c$

$$\mathbf{x} \to \mathbf{\Sigma}^{-1/2}\mathbf{V}^\top [k(\mathbf{x}, \mathbf{x}_1), ..., k(\mathbf{x}, \mathbf{x}_c)]^\top = \mathbf{f},$$

is referred to as an empirical kernel map in the literature [60–62]. For the remainder of this chapter, we call the embedding space $\Phi_c^e(\mathcal{X}) \subset \mathbb{R}^c$ the "empirical feature space".

It is easy to verify that the empirical feature space preserves the geometrical structure in the feature space as we showed earlier in Section 4.2.3 (let $c = N$ and then $||\mathbf{f}_i - \mathbf{f}_j||^2 = ||\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)||^2 \ \ \forall\, i, j$). Since the training data has the same geometrical structure, it has the same class separability, in both domains.

Given the training data set, all kernel-based algorithms, such as the SVM [63], Kernel Fisher Discriminant Analysis (KFDA) [64–66], and Kernelized PCA (KPCA) [34], perform their learning in the subspace $\langle \phi(\mathbf{x}_i) \rangle \subset \mathcal{F}$, which is isomorphic and isometric with the empirical feature space according to the previous discussion.

From both the theoretical and practical points of view, it is easier to access the empirical feature space than the feature space. Since the geometrical structure of the training data in the empirical feature space is the same as that in the high-dimensional feature space, the

former provides a tractable framework to study the spatial distribution of $\{\phi(\mathbf{x}_i)\}$, to measure the class separability of $\{\phi(\mathbf{x}_i)\}$, and more importantly, to optimize the kernel in order to increase the separability and, hence, improve the performance of the kernel machines.

7.3 Data-Dependent Kernel Function

Different kernels create different data geometry in the feature space and hence different class discrimination. Since there is no general kernel function suitable to all data sets, the authors in [15] advocate the use of a data-dependent kernel function. Consider training data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m) \in \mathbb{R}^d \times \{\pm 1\}$ where each $\mathbf{x}_i$ is an input space feature with a corresponding binary class label y_i. We can use the so-called "conformal transformation of a kernel" [67] as our data-dependent kernel function

$$k(\mathbf{x}, \mathbf{y}) = q(\mathbf{x})q(\mathbf{y})k_0(\mathbf{x}, \mathbf{y}), \tag{7.1}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $k_0(\mathbf{x}, \mathbf{y})$ is called the basic kernel (ordinary kernel), e.g. Gaussian or a polynormial, and $q(\cdot)$ is the factor function of the form

$$q(\mathbf{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i k_1(\mathbf{x}, \mathbf{a}_i), \tag{7.2}$$

in which $k_1(\mathbf{x}, \mathbf{a}_i) = \exp(-\gamma||\mathbf{x} - \mathbf{a}_i||^2)$, $\mathbf{a}_i \in \mathbb{R}^d$, and α_i's are the combination coefficients. The set $\{\mathbf{a}_i |\ i = 1, 2, ..., n\}$, called the "empirical cores," can be chosen from the training data or determined according to the distribution of the training data. It is easy to see that the data-dependent kernel satisfies the Mercer condition for a kernel function via the argument in Appendix D.

The kernel data matrices corresponding to $k(\mathbf{x}, \mathbf{y})$ and $k_0(\mathbf{x}, \mathbf{y})$ are denoted by $\mathbf{K}$ and $\mathbf{K}_0$, respectively. That is, $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^m$ and $\mathbf{K}_0 = [k_0(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^m$. It is easy to see that

$$\mathbf{K} = [q(\mathbf{x}_i)q(\mathbf{x}_j)k_0(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^m = \mathbf{Q}\mathbf{K}_0\mathbf{Q}, \tag{7.3}$$

where $\mathbf{Q}$ is a diagonal matrix, whose diagonal elements are $\{q(\mathbf{x}_1), ..., q(\mathbf{x}_m)\}$. We denote

the vectors $\mathbf{q} = [q(\mathbf{x}_1), ..., q(\mathbf{x}_m)]^\top$ and $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, ..., \alpha_n]^\top$, respectively. Then, we have

$$\mathbf{q} = \begin{bmatrix} 1 & k_1(\mathbf{x}_1, \mathbf{a}_1) & \dots & k_1(\mathbf{x}_1, \mathbf{a}_n) \\ 1 & k_1(\mathbf{x}_2, \mathbf{a}_1) & \dots & k_1(\mathbf{x}_2, \mathbf{a}_n) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & k_1(\mathbf{x}_m, \mathbf{a}_1) & \dots & k_1(\mathbf{x}_m, \mathbf{a}_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \triangleq \mathbf{K}_1 \boldsymbol{\alpha}, \tag{7.4}$$

where $\mathbf{K}_1$ is an $m \times (n+1)$ matrix.

7.4 Separability in Empirical Feature Space

Class separability of the training data in the empirical feature space is measured using the Fisher scalar [26]:

$$J = \frac{\operatorname{tr} \mathbf{S}_b}{\operatorname{tr} \mathbf{S}_w} \tag{7.5}$$

where $\mathbf{S}_b$ is the "between-class scatter matrix," $\mathbf{S}_w$ the "within class scatter matrix," and "tr" denotes the trace of a matrix. In Fisher discriminant analysis (FDA) [26, 65], the Rayleigh quotient is used to measure the class separability,

$$R(\Omega) = \frac{|\Omega^\top \mathbf{S}_b \Omega|}{|\Omega^\top \mathbf{S}_w \Omega|}$$

Where Ω is the projection matrix to be determined in the optimization algorithm. Compared with $R(\Omega)$, J measures the class separability in the feature space rather than in the projection subspace. Since J is independent of the projections, it is better suited for use in kernel optimization. Optimizing the data-dependent kernel through J means increasing the linear separability of the training data in the feature space.

Let the number of samples in class C_1 be m_1, and the number of samples in class C_2 be m_2. Let $\{\mathbf{f}_i\}_{i=1}^m$ be the images of the training data in the empirical feature space, where $m = m_1 + m_2$. Let $\bar{\mathbf{f}}$, $\bar{\mathbf{f}}_1$, and $\bar{\mathbf{f}}_2$ denote the centeroids of the entire training data and those

of classes C_1 and C_2 in the empirical feature space, respectively. Then, we have

$$\mathbf{S}_b = \frac{1}{m}\sum_{i=1}^{2} m_i(\bar{\mathbf{f}}_i - \bar{\mathbf{f}})(\bar{\mathbf{f}}_i - \bar{\mathbf{f}})^\top,$$
$$\mathbf{S}_w = \frac{1}{m}\sum_{i=1}^{2}\sum_{j=1}^{m_i}(\mathbf{f}_j^i - \bar{\mathbf{f}}_i)(\mathbf{f}_j^i - \bar{\mathbf{f}}_i)^\top,$$

where the vector $\mathbf{f}_j^i$ denotes the jth data in the ith class.

Assuming that the first m_1 data belong to class C_1, and the remaining m_2 samples belong to C_2. The kernel matrices can be written as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \in \mathbb{R}^{m\times m},$$

where $\mathbf{K}_{11}$, $\mathbf{K}_{12}$, $\mathbf{K}_{21}$, and $\mathbf{K}_{22}$ represent the sub-matrices of $\mathbf{K}$ of dimensions $m_1 \times m_1$, $m_1 \times m_2$, $m_2 \times m_1$, and $m_2 \times m_2$, respectively. Let us call the following matrices "between-class" and "within-class" kernel scatter matrices, and denote them by $\mathbf{B}$ and $\mathbf{W}$, respectively

$$\mathbf{B} = \begin{bmatrix} \frac{1}{m_1}\mathbf{K}_{11} & 0 \\ 0 & \frac{1}{m_2}\mathbf{K}_{22} \end{bmatrix} - \frac{1}{m}\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix},$$
$$\mathbf{W} = \begin{bmatrix} k_{11} & 0 & \dots & 0 \\ 0 & k_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & k_{mm} \end{bmatrix} - \begin{bmatrix} \frac{1}{m_1}\mathbf{K}_{11} & 0 \\ 0 & \frac{1}{m_2}\mathbf{K}_{22} \end{bmatrix}. \tag{7.6}$$

We also denote by $\mathbf{B}_0$ and $\mathbf{W}_0$, the between-class and the within class kernel scatter matrices corresponding to the basic kernel $\mathbf{K}_0$. As we will show next, $\mathbf{B}$ and $\mathbf{W}$ defined above are closely related to $\mathbf{S}_b$ and $\mathbf{S}_w$

7.5 Kernel Optimization in Empirical Feature Space

Now, we turn our attention to optimizing the conformal kernel in the empirical feature space. To do so we first prove a useful result which relates the between and within class

scatter terms in J to the between and within class scatter matrices of the conformal kernel as well as those of the base kernel, utilized by the conformal kernel. We establish a relation between J and the kernel scatter matrices by the following theorem [15].

Theorem 7.1: Let $\mathbf{1}_k$ be the k-dimensional one vector whose entries are all equal to 1. Then, we have

$$J = \frac{\mathbf{1}_m^\top \mathbf{B} \mathbf{1}_m}{\mathbf{1}_m^\top \mathbf{W} \mathbf{1}_m} = \frac{\mathbf{q}^\top \mathbf{B}_0 \mathbf{q}}{\mathbf{q}^\top \mathbf{W}_0 \mathbf{q}} \tag{7.7}$$

Proof: Suppose the dimension of the empirical feature space be c ($c \leq m$), that is, the dot product matrix $\mathbf{K}$ has exactly c positive eigenvalues. Let $\mathbf{Y}$ be the $m \times c$ matrix whose *rows* are the vectors $\{\mathbf{f}_i^\top\}$ (Note $\mathbf{Y} = \mathbf{F}^\top$ in Chapter 4), $\mathbf{Y}_1$ be the $m_1 \times c$ matrix whose rows are $\{\mathbf{f}_i^\top\}$ ($i \leq m_1$), and $\mathbf{Y}_2$ be the $m_2 \times c$ matrix whose rows are the vectors $\{\mathbf{f}_i^\top\}$ ($i > m_1$). First, we have

$$\begin{aligned}
\bar{\mathbf{f}} &= \frac{1}{m} \sum_{i=1}^{m} \mathbf{f}_i = \frac{1}{m} \mathbf{Y}^\top \mathbf{1}_m, \\
\bar{\mathbf{f}}_1 &= \frac{1}{m_1} \sum_{i=1}^{m_1} \mathbf{f}_i = \frac{1}{m_1} \mathbf{Y}_1^\top \mathbf{1}_{m_1}, \\
\bar{\mathbf{f}}_2 &= \frac{1}{m_2} \sum_{i=m_1+1}^{m} \mathbf{f}_i = \frac{1}{m_2} \mathbf{Y}_1^\top \mathbf{1}_{m_2}.
\end{aligned}$$

Since the empirical feature space preserves the dot product

$$\begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix} (\mathbf{Y}_1^\top \mathbf{Y}_2^\top) = \mathbf{Y}\mathbf{Y}^\top = \mathbf{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix},$$

Hence, we have $\mathbf{Y}_1\mathbf{Y}_1^\top = \mathbf{K}_{11},\ \mathbf{Y}_2\mathbf{Y}_2^\top = \mathbf{K}_{22},\ \mathbf{Y}_1\mathbf{Y}_2^\top = \mathbf{K}_{12},$ and $\mathbf{Y}_2\mathbf{Y}_1^\top = \mathbf{K}_{21}$.

Therefore

$$
\begin{aligned}
\operatorname{tr}\mathbf{S}_b &= \frac{1}{m}\sum_{i=1}^{2} m_i(\bar{\mathbf{f}}_i - \bar{\mathbf{f}})^\top(\bar{\mathbf{f}}_i - \bar{\mathbf{f}}) \\
&= \left(\frac{1}{m}\sum_{i=1}^{2} m_i\bar{\mathbf{f}}_i^\top\bar{\mathbf{f}}_i\right) - \bar{\mathbf{f}}^\top\bar{\mathbf{f}} \\
&= \frac{1}{m}\sum_{i=1}^{2}\frac{1}{m_i}\mathbf{1}_{m_i}^\top\mathbf{Y}_i\mathbf{Y}_i^\top\mathbf{1}_{m_i} - \frac{1}{m^2}\mathbf{1}_m^\top\mathbf{Y}\mathbf{Y}^\top\mathbf{1}_m \\
&= \frac{1}{m}\sum_{i=1}^{2}\frac{1}{m_i}\mathbf{1}_{m_i}^\top\mathbf{K}_{ii}\mathbf{1}_{m_i} - \frac{1}{m^2}\mathbf{1}_m^\top\mathbf{K}\mathbf{1}_m \\
&= \frac{1}{m}(\mathbf{1}_{m_1}^\top\mathbf{1}_{m_2}^\top)\begin{pmatrix} \frac{1}{m_1}\mathbf{K}_{11} & 0 \\ 0 & \frac{1}{m_2}\mathbf{K}_{22}\end{pmatrix}\begin{pmatrix}\mathbf{1}_{m_1}\\ \mathbf{1}_{m_2}\end{pmatrix} - \frac{1}{m^2}\mathbf{1}_m^\top\begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12}\\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix}\mathbf{1}_m \\
&= \frac{1}{m}\mathbf{1}_m^\top\mathbf{B}\mathbf{1}_m
\end{aligned}
$$

and

$$
\begin{aligned}
\operatorname{tr}\mathbf{S}_w &= \frac{1}{m}\sum_{i=1}^{2}\sum_{j=1}^{m_i}(\mathbf{f}_j^i - \bar{\mathbf{f}}_i)^\top(\mathbf{f}_j^i - \bar{\mathbf{f}}_i) \\
&= \frac{1}{m}\sum_{i=1}^{2}\left(\sum_{j=1}^{m_i}(\mathbf{f}_j^i)^\top\mathbf{f}_j^i - m_i\bar{\mathbf{f}}_i^\top\bar{\mathbf{f}}_i\right) \\
&= \frac{1}{m}\left(\sum_{i=1}^{m}\mathbf{f}_i^\top\mathbf{f}_i - \sum_{i=1}^{2}\frac{1}{m_i}\mathbf{1}_{m_i}^\top\mathbf{K}_{ii}\mathbf{1}_{m_i}\right) \\
&= \frac{1}{m}\left[\sum_{i=1}^{m}k_{ii} - \mathbf{1}_m^\top\begin{pmatrix}\frac{1}{m_1}\mathbf{K}_{11} & 0\\ 0 & \frac{1}{m_2}\mathbf{K}_{22}\end{pmatrix}\mathbf{1}_m\right] \\
&= \frac{1}{m}\mathbf{1}_m^\top\mathbf{W}\mathbf{1}_m
\end{aligned}
$$

These two equalities prove the first part of the theorem. Furthermore, using (7.3), we can easily see that $\mathbf{B} = \mathbf{Q}\mathbf{B}_0\mathbf{Q}$ and $\mathbf{W} = \mathbf{Q}\mathbf{W}_0\mathbf{Q}$. Considering that $\mathbf{Q}\mathbf{1}_m = \mathbf{q}$, the proof of this theorem is established. □

Now, the goal of kernel design (optimization) is to find a set of $\boldsymbol{\alpha}$ that maximizes $J(\boldsymbol{\alpha})$

[15, 67]. To maximize $J(\boldsymbol{\alpha})$, we follow the standard gradient approach. Let us first define

$$J_1 = J_1(\mathbf{q}(\boldsymbol{\alpha})) = \mathbf{1}_m^\top \mathbf{B}\mathbf{1}_m = \mathbf{q}^\top \mathbf{B}_0 \mathbf{q}, \tag{7.8}$$

$$J_2 = J_2(\mathbf{q}(\boldsymbol{\alpha})) = \mathbf{1}_m^\top \mathbf{W}\mathbf{1}_m = \mathbf{q}^\top \mathbf{W}_0 \mathbf{q}. \tag{7.9}$$

The gradients of which can be computed via chain rule

$$\frac{\partial J_i}{\partial \boldsymbol{\alpha}} = \left(\frac{\partial \mathbf{q}}{\partial \boldsymbol{\alpha}}\right)^\top \frac{\partial J_i}{\partial \mathbf{q}}, \quad i \in \{1, 2\}$$

since $J_i = J_i(\mathbf{q}(\boldsymbol{\alpha}))$ $(i = 1, 2)$. From (7.8) and (7.9), we can see

$$\frac{\partial J_1}{\partial \mathbf{q}} = 2\mathbf{B}_0\mathbf{q}, \quad \frac{\partial J_2}{\partial \mathbf{q}} = 2\mathbf{W}_0\mathbf{q}$$

Considering $\mathbf{q} = \mathbf{K}_1\boldsymbol{\alpha}$, we have

$$\frac{\partial \mathbf{q}}{\partial \boldsymbol{\alpha}} = \mathbf{K}_1$$

Thus, we get

$$\frac{\partial J_1}{\partial \mathbf{q}} = 2\mathbf{B}_0\mathbf{K}_1\boldsymbol{\alpha}, \quad \frac{\partial J_2}{\partial \mathbf{q}} = 2\mathbf{W}_0\mathbf{K}_1\boldsymbol{\alpha}$$

and

$$\frac{\partial J_1}{\partial \boldsymbol{\alpha}} = 2\mathbf{K}_1^\top\mathbf{B}_0\mathbf{K}_1\boldsymbol{\alpha}, \quad \frac{\partial J_2}{\partial \boldsymbol{\alpha}} = 2\mathbf{K}_1^\top\mathbf{W}_0\mathbf{K}_1\boldsymbol{\alpha}$$

Theorem 7.2 The optimal $\boldsymbol{\alpha}$ is the eigenvector corresponding to the largest eigenvalue of matrix $\mathbf{N}_0^{-1}\mathbf{M}_0$ where $\mathbf{M}_0 = \mathbf{K}_1^\top\mathbf{B}_0\mathbf{K}_1$ and $\mathbf{N}_0 = \mathbf{K}_1^\top\mathbf{W}_0\mathbf{K}_1$ and the corresponding J is the maximum eigenvalue.

Proof. Using the above result and quotient rule we get

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = \frac{2}{J_2^2}(J_2\mathbf{M}_0 - J_1\mathbf{N}_0)\boldsymbol{\alpha}$$

To maximize J, let $\partial J/\partial\boldsymbol{\alpha} = 0$, we obtain the generalized eigenvalue problem

$$J_1\mathbf{N}_0\boldsymbol{\alpha} = J_2\mathbf{M}_0\boldsymbol{\alpha}$$

If $\mathbf{N}_0^{-1}$ exists, we get

$$J\boldsymbol{\alpha} = \mathbf{N}_0^{-1}\mathbf{M}_0\boldsymbol{\alpha}$$

which means that the maximum value of J is the largest eigenvalue of the matrix $\mathbf{N}_0^{-1}\mathbf{M}_0$, and the eigenvector corresponding to that is the optimal $\boldsymbol{\alpha}$.

7.5.1 Kernel Optimization via the Stochastic Gradient Ascent

Unfortunately, matrix $\mathbf{N}_0^{-1}\mathbf{M}_0$ is generally not symmetric, and even worse, $\mathbf{N}_0$ may be a singular matrix. To circumvent these problems, we can employ a gradient-based updating algorithm to get a sub-optimal solution for $\boldsymbol{\alpha}$. According to the stochastic gradient ascent method [68], the updating rule for maximizing the class separability is given by

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \eta \left(\frac{1}{J_2}\mathbf{M}_0 - \frac{J}{J_2}\mathbf{N}_0 \right) \boldsymbol{\alpha}^{(t)}$$

where J and J_2 are functions of $\boldsymbol{\alpha}^{(t)}$, $\mathbf{M}_0$ and $\mathbf{N}_0$ are two constant matrices defined before, and η is the learning rate. To ensure the convergence of the algorithm, a gradually decreasing learning rate is adopted

$$\eta(t) = \eta_0 \left(1 - \frac{t}{T} \right)$$

where η_0 is the initial learning rate, T denotes a prespecified number of iterations, and t represents the current iteration number. The two-class form of this kernel design algorithm can be viewed in Algorithm 1.

Algorithm 1 Data-dependent Kernel Optimization for Two-Class Problems

Require: Dataset $\{\mathbf{x}_i\}_{i=1}^{m+1}$; base kernel $k_0(\mathbf{x}, \mathbf{y})$
Ensure: Conformal kernel $k(\mathbf{x}, \mathbf{y}) = q(\mathbf{x})q(\mathbf{y})k_0(\mathbf{x}, \mathbf{y})$

1: Group the data according to their class labels. Calculate $\mathbf{K}_0$ and $\mathbf{K}_1$ first, then $\mathbf{B}_0$ and $\mathbf{W}_0$, and then $\mathbf{M}_0,\ \mathbf{N}_0$
2: Initialize $\boldsymbol{\alpha}^{(0)}$ by a vector $(1, 0, ..., 0)^\top$, and set $t = 0$
3: Calculate $\mathbf{q} = \mathbf{K}_1\boldsymbol{\alpha}^{(t)}$
4: Calculate $J_1^{(t)} = \mathbf{q}^\top\mathbf{B}_0\mathbf{q}$, $J_2^{(t)} = \mathbf{q}^\top\mathbf{W}_0\mathbf{q}$, and $J^{(t)}$
5: Update $\boldsymbol{\alpha}^{(t)}$

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \eta\left(\frac{1}{J_2^{(t)}}\mathbf{M}_0 - \frac{J^{(t)}}{J_2^{(t)}}\mathbf{N}_0\right)\boldsymbol{\alpha}^{(t)}$$

and normalize $\boldsymbol{\alpha}^{(t+1)}$ so that $||\boldsymbol{\alpha}^{(t+1)}|| = 1$
6: If n reaches a pre-specified number T, stop. Otherwise set $t = t + 1$, go to step 3.

7.6 Extension to Multi-class Case

A general form of Theorem 7.1 can be proven using a similar approach for the multi-class (or M-ary) case. We assume that the first m_1 data samples belong to class C_1, the next m_2 samples belong to class C_2 and so on with total number of training samples $m = m_1 + m_2 + ... + m_M$. Then, the kernel matrices can be written as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \dots & \mathbf{K}_{1M} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_{M1} & \dots & \mathbf{K}_{MM} \end{bmatrix} \in \mathbb{R}^{m\times m}$$

where $\mathbf{K}_{11}$, $\mathbf{K}_{22}$, ..., $\mathbf{K}_{MM}$ represent the class-specific kernel sub-matrices of $\mathbf{K}$ of dimensions $m_1 \times m_1$, $m_2 \times m_2$, ... and $m_M \times m_M$ respectively. The general "between-class" and "within-class" kernel scatter matrices to M classes are:

$$\mathbf{B} = \begin{bmatrix} \frac{1}{m_1}\mathbf{K}_{11} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{m_M}\mathbf{K}_{MM} \end{bmatrix} - \frac{1}{m} \begin{bmatrix} \mathbf{K}_{11} & \dots & \mathbf{K}_{1M} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_{M1} & \dots & \mathbf{K}_{MM} \end{bmatrix}, \tag{7.10}$$

$$\mathbf{W} = \begin{bmatrix} k_{11} & 0 & \dots & 0 \\ 0 & k_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & k_{mm} \end{bmatrix} - \begin{bmatrix} \frac{1}{m_1}\mathbf{K}_{11} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{m_M}\mathbf{K}_{MM} \end{bmatrix}, \tag{7.11}$$

respectively.

We also denote by $\mathbf{B}_0$ and $\mathbf{W}_0$, the between-class and the within class kernel scatter matrices corresponding to the basic kernel $\mathbf{K}_0$. With these defined, we can now state Theorem 7.3 for the M-Ary case. Again the goal is to establish a relation between J and the kernel scatter matrices but now for M classes.

Theorem 7.3: The Fisher Scalar [26] for M-ary case can be written as

$$J = \frac{\mathbf{1}_m^\top \mathbf{B} \mathbf{1}_m}{\mathbf{1}_m^\top \mathbf{W} \mathbf{1}_m} = \frac{\mathbf{q}^\top \mathbf{B}_0 \mathbf{q}}{\mathbf{q}^\top \mathbf{W}_0 \mathbf{q}} \tag{7.12}$$

Proof: Again suppose the dimension of the empirical feature space is c ($c \leq m$), i.e. kernel matrix $\mathbf{K}$ has exactly c positive eigenvalues. Let $\mathbf{Y}$ denote the $m \times c$ matrix whose rows are the vectors $\{\mathbf{f}_i^\top\}$, $\mathbf{Y}_1$ the $m_1 \times c$ matrix whose rows are $\{\mathbf{f}_i^\top\}$ ($i \leq m_1$), $\mathbf{Y}_2$ denote the $m_2 \times c$ matrix whose rows are the vectors $\{\mathbf{f}_i^\top\}$ ($m_1 < i \leq m_1 + m_2$), and so on. As

before, we can express the centroid vectors

$$\bar{\mathbf{f}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{f}_i = \frac{1}{m}\mathbf{Y}^\top \mathbf{1}_m$$

$$\bar{\mathbf{f}}_1 = \frac{1}{m_1}\sum_{j\in C_1}\mathbf{f}_j = \frac{1}{m_1}\mathbf{Y}_1^\top \mathbf{1}_{m_1}$$

$$\vdots \qquad\qquad \vdots$$

$$\bar{\mathbf{f}}_M = \frac{1}{m_M}\sum_{j\in C_M}\mathbf{f}_j = \frac{1}{m_M}\mathbf{Y}_M^\top \mathbf{1}_{m_M}$$

Since the empirical feature space preserves the inner product

$$\begin{pmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\\ \vdots\\ \mathbf{Y}_M\end{pmatrix}(\mathbf{Y}_1^\top\mathbf{Y}_2^\top\dots\mathbf{Y}_M^\top) = \mathbf{Y}\mathbf{Y}^\top = \mathbf{K} = \begin{bmatrix}\mathbf{K}_{11} & \dots & \mathbf{K}_{1M}\\ \vdots & \ddots & \vdots\\ \mathbf{K}_{M1} & \dots & \mathbf{K}_{MM}\end{bmatrix}$$

Where $\mathbf{Y}_1\mathbf{Y}_1^\top = \mathbf{K}_{11},\ \mathbf{Y}_2\mathbf{Y}_2^\top = \mathbf{K}_{22},\ \mathbf{Y}_1\mathbf{Y}_2^\top = \mathbf{K}_{12},\ \mathbf{Y}_2\mathbf{Y}_1^\top = \mathbf{K}_{21}\ \dots$ and $\mathbf{Y}_M\mathbf{Y}_M^\top = \mathbf{K}_{MM}$.

Therefore

$$\begin{aligned}
\operatorname{tr}\mathbf{S}_b &= \frac{1}{m}\sum_{i=1}^{M} m_i(\bar{\mathbf{f}}_i - \bar{\mathbf{f}})^\top(\bar{\mathbf{f}}_i - \bar{\mathbf{f}})\\
&= \frac{1}{m}\sum_{i=1}^{M} m_i\bar{\mathbf{f}}_i^\top\bar{\mathbf{f}}_i + \frac{1}{m}\sum_{i=1}^{M} m_i\bar{\mathbf{f}}^\top\bar{\mathbf{f}} - \frac{2}{m}\sum_{i=1}^{M} m_i\bar{\mathbf{f}}_i^\top\bar{\mathbf{f}}\\
&= \frac{1}{m}\sum_{i=1}^{M} m_i\bar{\mathbf{f}}_i^\top\bar{\mathbf{f}}_i - \bar{\mathbf{f}}^\top\bar{\mathbf{f}}\\
&= \frac{1}{m}\sum_{i=1}^{M}\frac{1}{m_i}\mathbf{1}_{m_i}^\top\mathbf{Y}_i\mathbf{Y}_i^\top\mathbf{1}_{m_i} - \frac{1}{m^2}\mathbf{1}_m^\top\mathbf{Y}\mathbf{Y}^\top\mathbf{1}_m\\
&= \frac{1}{m}\sum_{i=1}^{M}\frac{1}{m_i}\mathbf{1}_{m_i}^\top\mathbf{K}_{ii}\mathbf{1}_{m_i} - \frac{1}{m^2}\mathbf{1}_m^\top\mathbf{K}\mathbf{1}_m\\
&= \frac{1}{m}(\mathbf{1}_{m_1}^\top\mathbf{1}_{m_2}^\top\dots\mathbf{1}_{m_M}^\top)\begin{bmatrix}\frac{1}{m_1}\mathbf{K}_{11} & & 0\\ & \ddots & \\ 0 & & \frac{1}{m_M}\mathbf{K}_{MM}\end{bmatrix}\begin{pmatrix}\mathbf{1}_{m_1}\\ \mathbf{1}_{m_2}\\ \vdots\\ \mathbf{1}_m\end{pmatrix} - \frac{1}{m^2}\mathbf{1}_m^\top\begin{bmatrix}\mathbf{K}_{11} & \dots & \mathbf{K}_{1M}\\ \vdots & \ddots & \vdots\\ \mathbf{K}_{M1} & \dots & \mathbf{K}_{MM}\end{bmatrix}\mathbf{1}_m\\
&= \frac{1}{m}\mathbf{1}_m^\top\mathbf{B}\mathbf{1}_m \qquad (7.13)
\end{aligned}$$

Also,

$$\begin{aligned}
\operatorname{tr} \mathbf{S}_w &= \frac{1}{m} \sum_{i=1}^{M} \sum_{j=1}^{m_i} (\mathbf{f}_j^i - \bar{\mathbf{f}}_i)^\top (\mathbf{f}_j^i - \bar{\mathbf{f}}_i) \\
&= \frac{1}{m} \left(\sum_{i=1}^{M} \sum_{j=1}^{m_i} (\mathbf{f}_j^i)^\top \mathbf{f}_j^i - m_i \bar{\mathbf{f}}_i^\top \bar{\mathbf{f}}_i \right) \\
&= \frac{1}{m} \left(\sum_{i=1}^{m} \mathbf{f}_i^\top \mathbf{f}_i - \sum_{i=1}^{M} \frac{1}{m_i} \mathbf{1}_{m_i}^\top K_{ii} \mathbf{1}_{m_i} \right) \\
&= \frac{1}{m} \left[\sum_{i=1}^{m} k_{ii} - \mathbf{1}_m^\top \begin{bmatrix} \frac{1}{m_1} \mathbf{K}_{11} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{m_M} \mathbf{K}_{MM} \end{bmatrix} \mathbf{1}_m \right] \\
&= \frac{1}{m} \mathbf{1}_m^\top \mathbf{W} \mathbf{1}_m
\end{aligned} \tag{7.14}$$

These two equalities prove the first part of the theorem. Furthermore, using (7.3), we can again see that $\mathbf{B} = \mathbf{Q}\mathbf{B}_0\mathbf{Q}$ and $\mathbf{W} = \mathbf{Q}\mathbf{W}_0\mathbf{Q}$. Considering that $\mathbf{Q}\mathbf{1}_m = \mathbf{q}$, the theorem is established. ■

7.7 A Novel Discriminative Multi-kernel Learning

Building off the ideas from the previous section, we entertain a different model assumption which features a kernel function that is a convex combination of other kernel functions rather than a conformal kernel with a single base kernel. That is, we assume

$$k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} a_i k_i(\mathbf{x}, \mathbf{y}) \quad s.t. \quad \mathbf{a}^\top \mathbf{1} = 1$$

where $\mathbf{a} = [a_1, a_2, ..., a_m]^\top$ is a vector of the convex combination coefficients $a_i \geq 0$ for each of the kernel functions $k_i(\mathbf{x}, \mathbf{y})$. Learning a kernel function that is a linear combination of other kernel functions has been explored by works such as [62, 69, 70]. So long as each kernel function that we use as our basis in this convex combination each satisfies Mercer's condition, then the convex combination will also satisfy the condition. This can be seen by the following argument:

A necessary and sufficient condition for a function $k(\mathbf{x}, \mathbf{y})$ to be expressible as an inner product in some feature space $\mathcal{F}$ is a weak form of Mercer's condition, namely that:

$$\int_{\mathbf{x}} \int_{\mathbf{y}} k(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

for all square, integrable functions $g(\cdot)$ [38]. In this case above, where $k(\mathbf{x}, \mathbf{y})$ is defined by a convex combination of a finite number of Mercer kernels we have:

$$\begin{aligned} &\int_{\mathbf{x}} \int_{\mathbf{y}} \left(\sum_{i=1}^{m} a_i k_i(\mathbf{x}, \mathbf{y}) \right) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \\ &= \sum_{i=1}^{m} a_i \int_{\mathbf{x}} \int_{\mathbf{y}} k_i(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \end{aligned}$$

where the final inequality is established by the fact that $a_i \geq 0 \; \forall \; i \in [1, m]$.

Now, using this type of kernel function, and in light of the previous results, it should be obvious that the resulting kernel matrix would be of the form

$$\mathbf{K} = \mathbf{Q} \tilde{\mathbf{K}} \mathbf{Q}^{\top} \in \mathbb{R}^{n \times n}$$

where $\mathbf{Q} = [\sqrt{a_1} \mathbf{I}_{n \times n}, ..., \sqrt{a_m} \mathbf{I}_{n \times n}] = \boldsymbol{\alpha}^{\top} \otimes \mathbf{I}_{n \times n} \in \mathbb{R}^{n \times mn}$, symbol $\otimes$ denotes the Kronecker product, $\boldsymbol{\alpha} = [\sqrt{a_1}, ..., \sqrt{a_m}]^{\top}$ and $\tilde{\mathbf{K}}$ is a block diagonal matrix of the following form:

$$\tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{K}_m \end{bmatrix}$$

where $\mathbf{K}_1, ..., \mathbf{K}_m$ are the full kernel matrices associated with kernel functions $k_1(\mathbf{x}, \mathbf{y}), ..., k_m(\mathbf{x}, \mathbf{y})$, respectively. Using similar notations, we represent $\mathbf{W}$ and $\mathbf{B}$ as

$$\mathbf{W} = \mathbf{Q} \tilde{\mathbf{W}} \mathbf{Q}^{\top}, \qquad \mathbf{B} = \mathbf{Q} \tilde{\mathbf{B}} \mathbf{Q}^{\top}$$

where $\tilde{\mathbf{W}}$ and $\tilde{\mathbf{B}}$ are block diagonal matrices of the form

$$\tilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{W}_m \end{bmatrix},$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{B}_m \end{bmatrix},$$

and $\mathbf{W}_1, ..., \mathbf{W}_m$ and $\mathbf{B}_1, ..., \mathbf{B}_m$ are the $\mathbf{W}$ and $\mathbf{B}$ matrices defined in 7.11 and 7.10, for kernel functions $k_1(\mathbf{x}, \mathbf{y}), ..., k_m(\mathbf{x}, \mathbf{y})$, respectively. Using the first equality from (7.12) of Theorem 7.3, we have

$$\begin{aligned} n \operatorname{tr}(\mathbf{S}_w) &= \mathbf{1}_n^\top \mathbf{W} \mathbf{1}_n, \\ &= \operatorname{tr}(\mathbf{1}_n^\top \mathbf{W} \mathbf{1}_n), \\ &= \operatorname{tr}(\mathbf{1}_n^\top \mathbf{Q} \tilde{\mathbf{W}} \mathbf{Q}^\top \mathbf{1}_n) \end{aligned}$$

But $\mathbf{1}_n^\top \mathbf{Q} = \boldsymbol{\alpha}^\top \otimes \mathbf{1}_n^\top \Leftrightarrow \mathbf{Q}^\top \mathbf{1}_n = \boldsymbol{\alpha} \otimes \mathbf{1}_n$. Hence, we get

$$\mathbf{1}_n^\top \mathbf{W} \mathbf{1}_n = \operatorname{tr}((\boldsymbol{\alpha}^\top \otimes \mathbf{1}_n^\top) \tilde{\mathbf{W}} (\boldsymbol{\alpha} \otimes \mathbf{1}_n))$$

Letting $\mathbf{v} = \boldsymbol{\alpha} \otimes \mathbf{1}_n$ the denominator term of J becomes

$$J_2 = \mathbf{1}_n^\top \mathbf{W} \mathbf{1}_n = \mathbf{v}^\top \tilde{\mathbf{W}} \mathbf{v}$$

Using a similar approach, it is easy to see that the numerator term of J is as follows:

$$J_1 = \mathbf{1}_n^\top \mathbf{B} \mathbf{1}_n = \mathbf{v}^\top \tilde{\mathbf{B}} \mathbf{v}$$

From here it is easy to compute the derivative with respect to $\mathbf{v}$ and then relate these derivatives back to $\boldsymbol{\alpha}$ via the chain rule. Taking derivatives with respect to $\mathbf{v}$ yields

$$\frac{\partial J_2}{\partial \mathbf{v}} = 2\tilde{\mathbf{W}}\mathbf{v}, \quad \frac{\partial J_1}{\partial \mathbf{v}} = 2\tilde{\mathbf{B}}\mathbf{v}$$

Now, the chain rule gives us

$$\frac{\partial J_i}{\partial \boldsymbol{\alpha}} = \left(\frac{\partial \mathbf{v}}{\partial \boldsymbol{\alpha}}\right)^\top \frac{\partial J_i}{\partial \mathbf{v}}$$

Now using $\mathbf{v}^\top = \boldsymbol{\alpha}^\top \otimes \mathbf{1}_n^\top$ and the fact that $(\mathbf{AC}) \otimes (\mathbf{BD}) = (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D})$ for matrices $\mathbf{A}, \mathbf{C}$ and $\mathbf{B}, \mathbf{D}$ of appropriate dimensions, we can see

$$\begin{aligned}
\mathbf{v}^\top &= (\boldsymbol{\alpha}^\top \mathbf{I}_{m,m}) \otimes (1 \cdot \mathbf{1}_n^\top) \\
&= \boldsymbol{\alpha}^\top (\mathbf{I}_{m \times m} \otimes \mathbf{1}_n^\top) \\
\Rightarrow \frac{\partial \mathbf{v}}{\partial \boldsymbol{\alpha}}^\top &= \mathbf{I}_{m \times m} \otimes \mathbf{1}_n^\top \in \mathbb{R}^{m \times mn}
\end{aligned}$$

Combining these facts we can see that for $J = \frac{J_1}{J_2} = \frac{\mathbf{1}^\top \mathbf{B1}}{\mathbf{1}^\top \mathbf{W1}}$, the derivative takes the following form:

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = \frac{\frac{\partial J_1}{\partial \boldsymbol{\alpha}} J_2 - \frac{\partial J_2}{\partial \boldsymbol{\alpha}} J_1}{J_2^2}$$

Where $\frac{\partial J_1}{\partial \boldsymbol{\alpha}} = 2(\mathbf{I}_{m \times m} \otimes \mathbf{1}_n^\top)\tilde{\mathbf{B}}\mathbf{v}$ and $\frac{\partial J_2}{\partial \boldsymbol{\alpha}} = 2(\mathbf{I}_{m \times m} \otimes \mathbf{1}_n^\top)\tilde{\mathbf{W}}\mathbf{v}$. Now, letting $\mathbf{S} = (\mathbf{I}_{m \times m} \otimes \mathbf{1}_n^\top)$ we can rewrite $\frac{\partial J_1}{\partial \boldsymbol{\alpha}} = 2\mathbf{S}\tilde{\mathbf{B}}\mathbf{S}^\top\boldsymbol{\alpha}$ and $\frac{\partial J_2}{\partial \boldsymbol{\alpha}} = 2\mathbf{S}\tilde{\mathbf{W}}\mathbf{S}^\top\boldsymbol{\alpha}$. Similar to the previous result, a direct solution would lead to a generalized eigenvalue problem of the following sort $J(\mathbf{S}\tilde{\mathbf{W}}\mathbf{S}^\top)\boldsymbol{\alpha} = (\mathbf{S}\tilde{\mathbf{B}}\mathbf{S}^\top)\boldsymbol{\alpha}$, whose solution is the principal eigenvector of $(\mathbf{S}\tilde{\mathbf{W}}\mathbf{S}^\top)^{-1}(\mathbf{S}\tilde{\mathbf{B}}\mathbf{S}^\top)$ with maximal J as the largest eigenvalue. Using the gradient in a similar manner to the previous method, we must again ensure $\boldsymbol{\alpha}^\top\boldsymbol{\alpha} = 1$ since $\boldsymbol{\alpha} = [\sqrt{a_1}, ..., \sqrt{a_m}]$ and we desire $\sum_{i=1}^m a_i = 1$. Furthermore, to give every kernel function a fighting chance, we initialize $\boldsymbol{\alpha}_0 = \frac{1}{\sqrt{m}}\mathbf{1}_m$

In Algorithm 2 below, we describe the gradient ascent based learning algorithm to maximize J with respect to the $\boldsymbol{\alpha}$ vector that describes our convex combination of kernel functions.

Algorithm 2 Discriminative Multi-Kernel Optimization

Require: Dataset $\{\mathbf{x}_i\}_{i=1}^{n+1}$; base kernels $k_1(\mathbf{x},\mathbf{y}),...,k_m(\mathbf{x},\mathbf{y})$, training iterations N, base learning rate η_0

Ensure: Combination Kernel $k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} a_i k_i(\mathbf{x},\mathbf{y})$

1: Group the data according to their class labels. Calculate $\mathbf{K}_1$, ... $\mathbf{K}_m$ with samples grouped by class first, then $\mathbf{B}_1,...,\mathbf{B}_m$ and $\mathbf{W}_1,...,\mathbf{W}_m$, and then $\tilde{\mathbf{B}}, \tilde{\mathbf{W}}, \tilde{\mathbf{K}}$

2: Initialize $\boldsymbol{\alpha}^{(0)}$ by a vector $\frac{1}{\sqrt{m}}(1,1,...,1)^\top$, $\mathbf{S} = (\mathbf{I}_{m\times m} \otimes \mathbf{1}_n^\top)$ and set $t = 0$

3: Calculate $\mathbf{v}^{(t)} = \mathbf{S}^\top \boldsymbol{\alpha}^{(t)}$

4: Calculate $J_1^{(t)} = \mathbf{v}^\top \tilde{\mathbf{B}} \mathbf{v}$, $J_2^{(t)} = \mathbf{v}^\top \tilde{\mathbf{W}} \mathbf{v}$, and $J^{(t)}$

5: Update $\boldsymbol{\alpha}^{(t)}$

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \eta \left(\frac{1}{J_2^{(t)}} \mathbf{S}\tilde{\mathbf{B}} - \frac{J^{(t)}}{J_2^{(t)}} \mathbf{S}\tilde{\mathbf{W}} \right) \mathbf{v}^{(t)}$$

and normalize $\boldsymbol{\alpha}^{(t+1)}$ so that $||\boldsymbol{\alpha}^{(t+1)}|| = 1$

6: set $\eta = (1 - \frac{t}{T})\eta_0$

7: If t reaches a pre-specified number T, stop. Otherwise set $t = t + 1$, go to step 3.

Remark 7.1

Extending this method to include a bias term $a_0 \geq 0$ to the kernel function can easily be accomplished by making the following changes

$$k(\mathbf{x},\mathbf{y}) = a_0 + \sum_{i=1}^{m} a_i k_i(\mathbf{x},\mathbf{y})$$

The corresponding kernel matrix takes the form

$$\mathbf{K} = \mathbf{Q}\tilde{\mathbf{K}}\mathbf{Q}^\top$$

where $\mathbf{Q} = [\sqrt{a_0}\mathbf{I}_{n,n}, \sqrt{a_1}\mathbf{I}_{n,n}, ..., \sqrt{a_m}\mathbf{I}_{n,n}] \in \mathbb{R}^{n\times(m+1)n}$,

$$\tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{1}_{n,n} & & & \mathbf{0} \\ & \mathbf{K}_1 & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{K}_m \end{bmatrix}$$

and $\mathbf{1}_{n,n}$ denotes a $n \times n$ matrix where all entries are equal to 1. Similarly, $\mathbf{B}$ and $\mathbf{W}$ can be represented as

$$\mathbf{B} = \mathbf{Q}\tilde{\mathbf{B}}\mathbf{Q}^\top \quad \mathbf{W} = \mathbf{Q}\tilde{\mathbf{W}}\mathbf{Q}^\top$$

with

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{B}_0 & & & \mathbf{0} \\ & \mathbf{B}_1 & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{B}_m \end{bmatrix}$$

and

$$\tilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W}_0 & & & \mathbf{0} \\ & \mathbf{W}_1 & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{W}_m \end{bmatrix}$$

Where

$$\mathbf{B}_0 = \text{blkdiag}(\frac{1}{n_1}\mathbf{1}_{n_1,n_1}, ..., \frac{1}{n_C}\mathbf{1}_{n_C,n_C}) - \frac{1}{n}\mathbf{1}_{n,n}$$
$$\mathbf{W}_0 = \mathbf{I}_{n,n} - \text{blkdiag}(\frac{1}{n_1}\mathbf{1}_{n_1,n_1}, ..., \frac{1}{n_C}\mathbf{1}_{n_C,n_C}).$$

And $\mathbf{W}_1, \cdots, \mathbf{W}_m$ and $\mathbf{B}_1, \cdots, \mathbf{B}_m$ are as before.

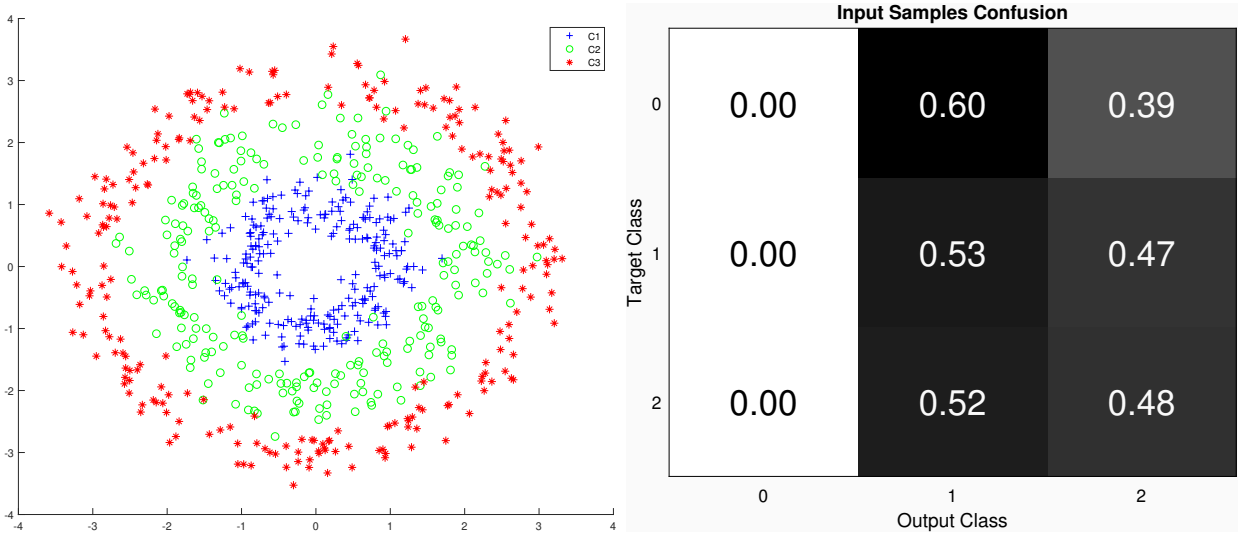
Numerical examples utilizing the method presented in this section can be found in the next section.

7.8 Numerical Examples using DMKL

Several experiments were run using the proposed method comparing a linear kernel, standard Gaussian kernel (i.e. $\sigma = 1.0$) and a learned kernel generated via the Discriminative multi-kernel learning proposed in the previous section. For these experiments, the DMKL from Algorithm 2 was trained for 500 iterations and candidate kernels included a linear kernel, the standard Gaussian kernel, and gaussian kernels with $\sigma = \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 10\}$. The initial learning rate η_0 was set to 0.005 for all datasets. For all datasets, 1/4 of the samples were randomly sampled for training and the remaining 3/4 were used for testing, with the exception of the human activity recognition (HAR) [71] dataset where 1.557% of samples were used for training and the remaining samples were used in testing.

Dataset	# of Samples	# Classes	feature dimension
Rings	1000	3	2
Swiss Roll	1000	4	3
Wine	178	3	13
HAR	24075	5	60
Swirl	1000	2	2
XOR	1000	2	2

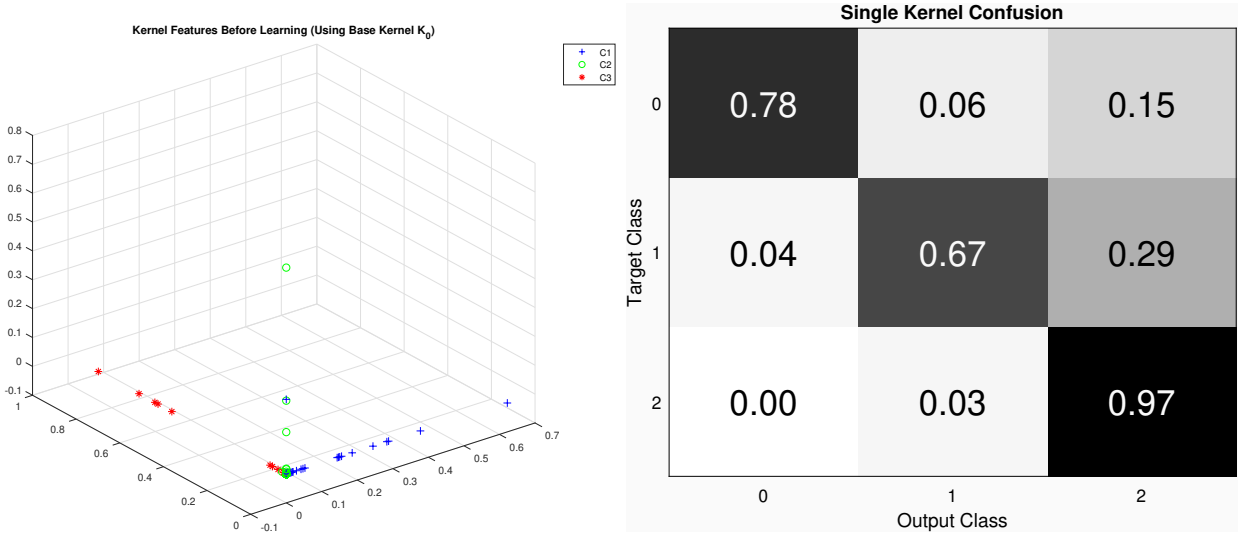
Table 7.1: Datasets Used in Testing DMKL



(a) Input Space Features

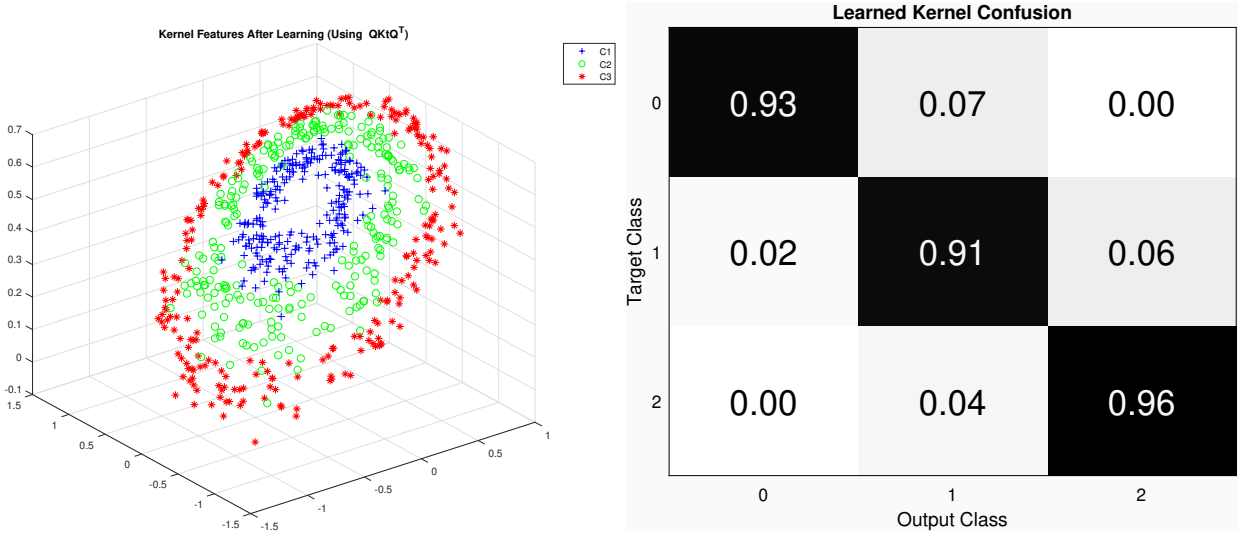
(b) Confusion

Figure 7.1: Rings Dataset



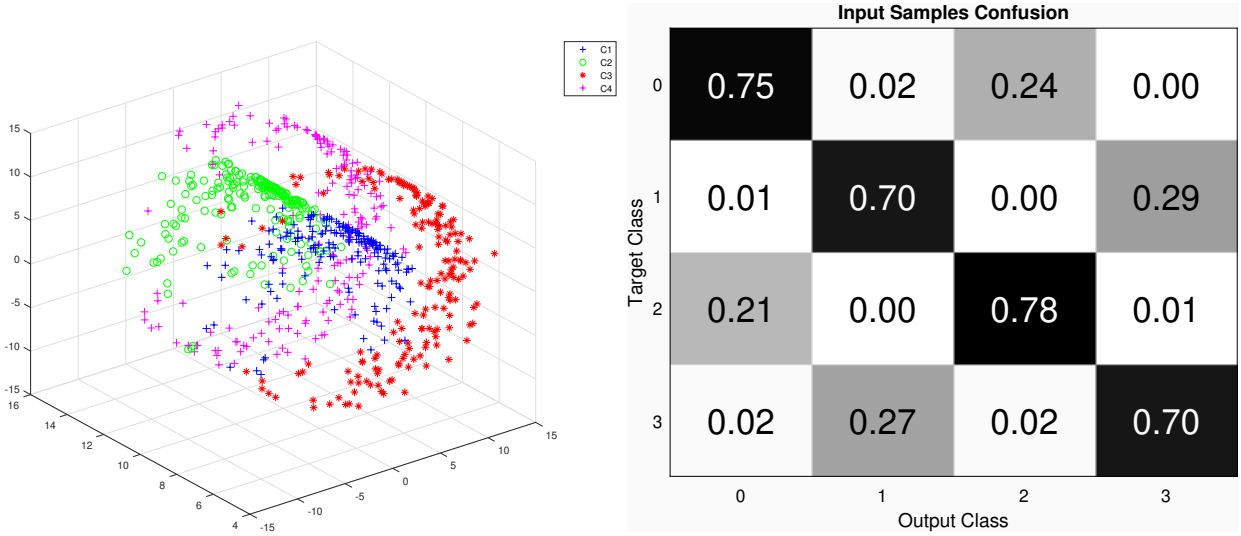
(a) Single Kernel Features (b) Confusion

Figure 7.2: Rings Dataset Single Kernel



(a) Learned Kernel Features (b) Confusion

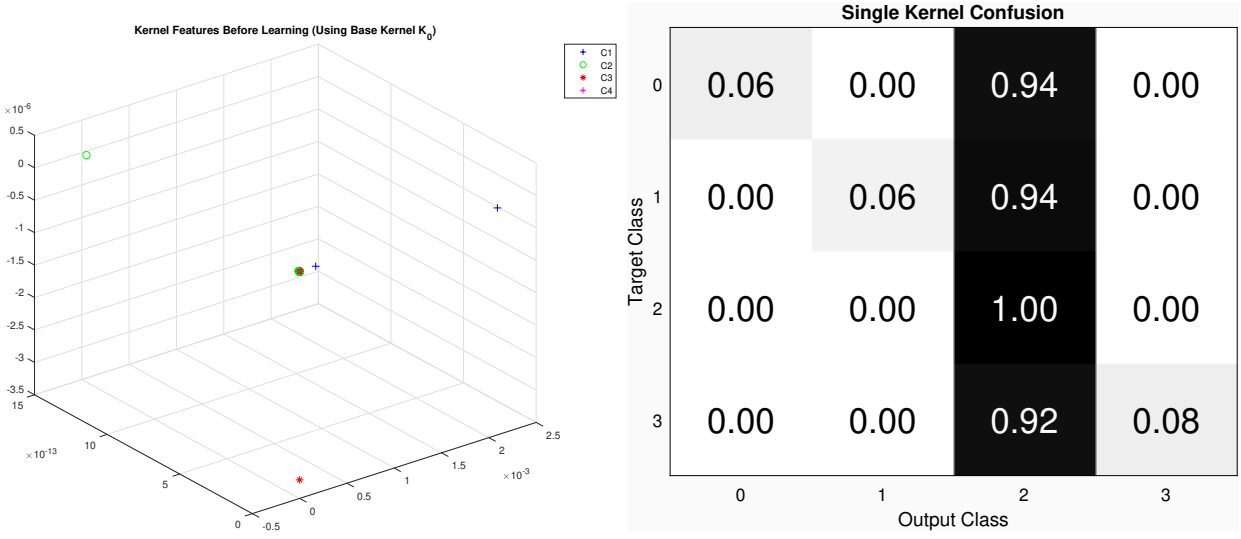
Figure 7.3: Rings Dataset Learned Kernel



(a) Input Space Features

(b) Confusion

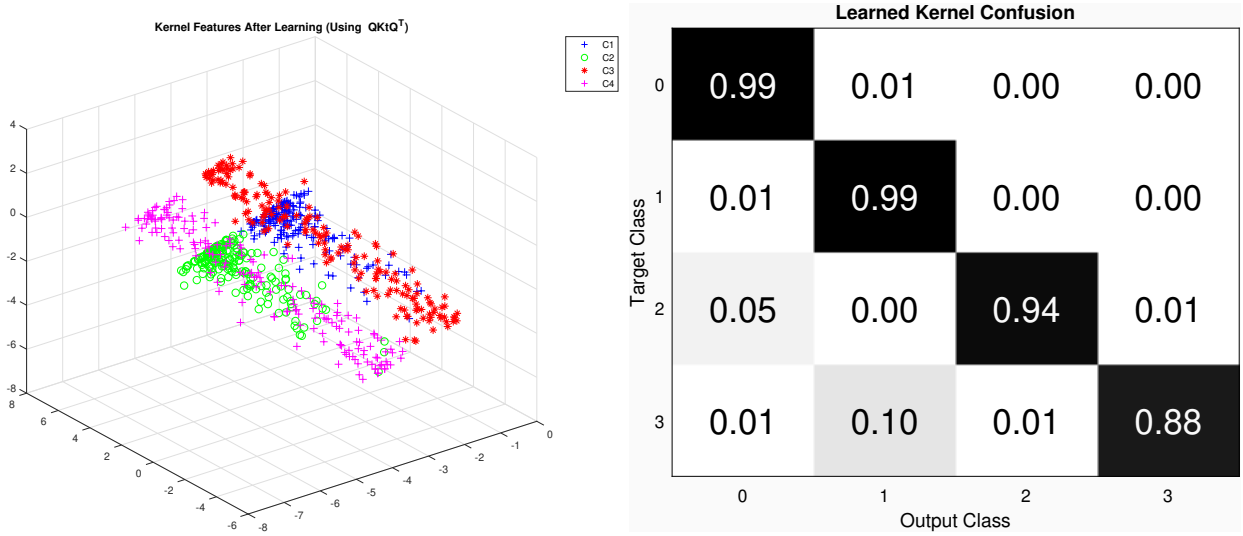
Figure 7.4: 4-Class Swissroll Dataset



(a) Single Kernel Features

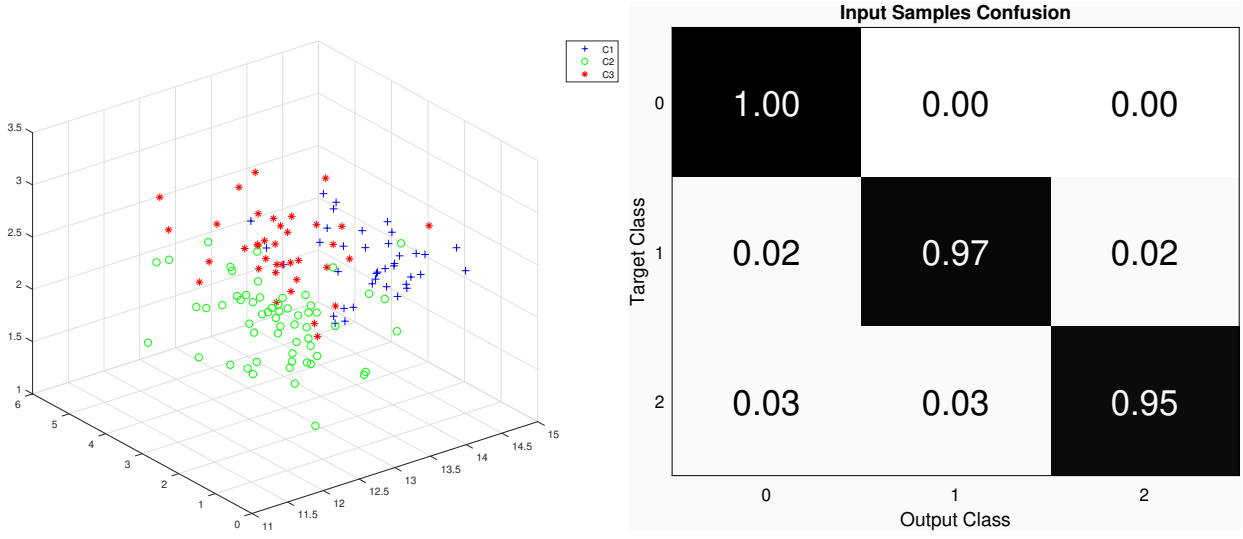
(b) Confusion

Figure 7.5: 4-Class Swissroll Dataset Single Kernel



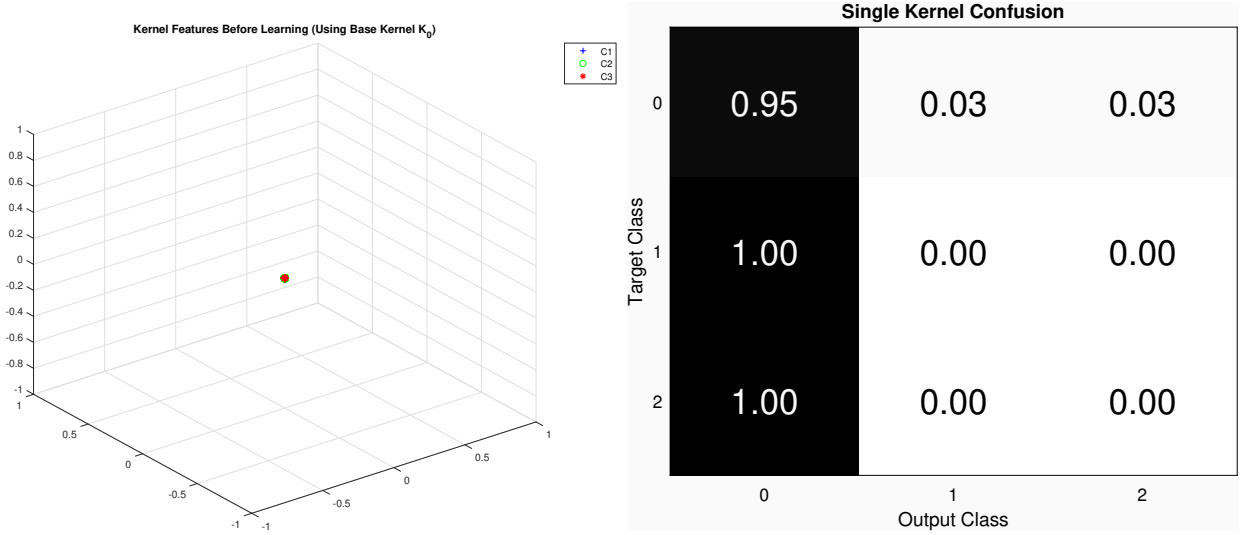
(a) Learned Kernel Features (b) Confusion

Figure 7.6: 4-Class Swissroll Dataset Learned Kernel



(a) Input Space Features (b) Confusion

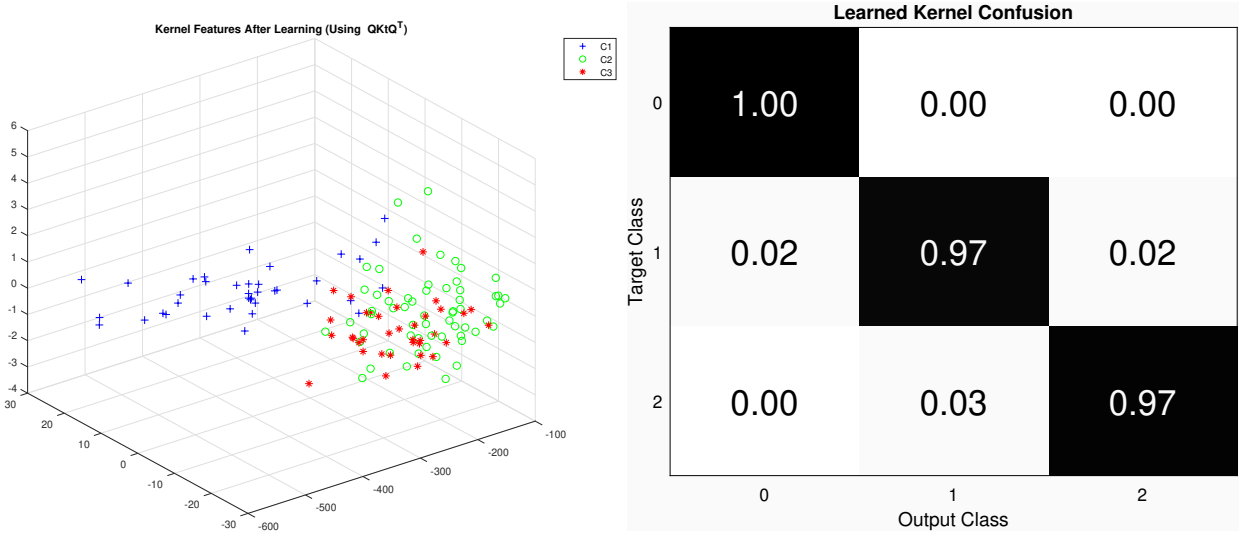
Figure 7.7: Wine Dataset



(a) Single Kernel Features

(b) Confusion

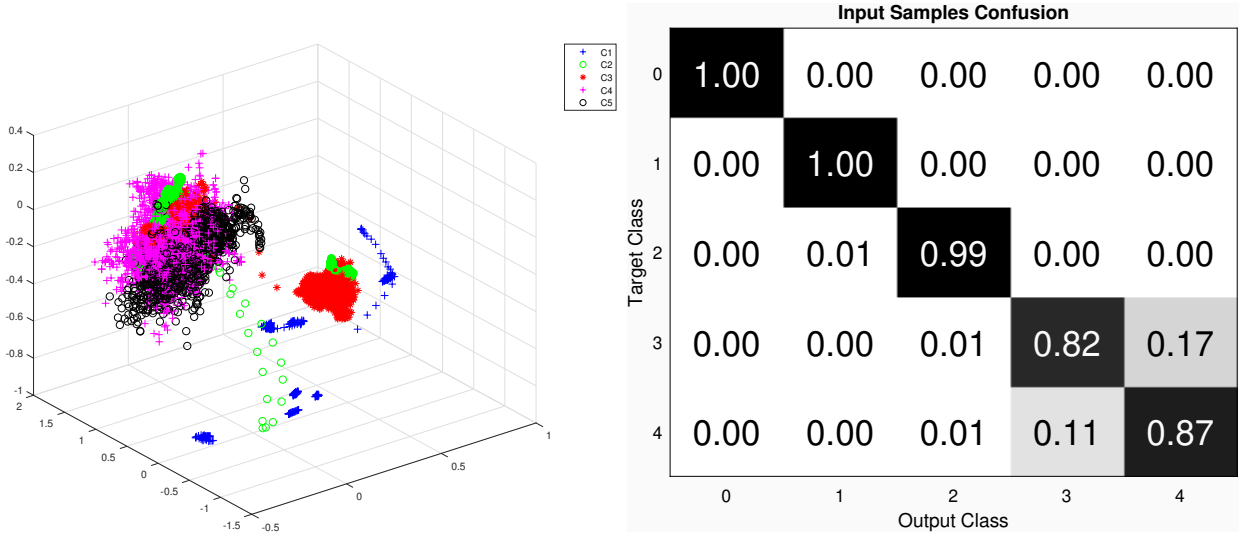
Figure 7.8: Wine Dataset Single Kernel



(a) Learned Kernel Features

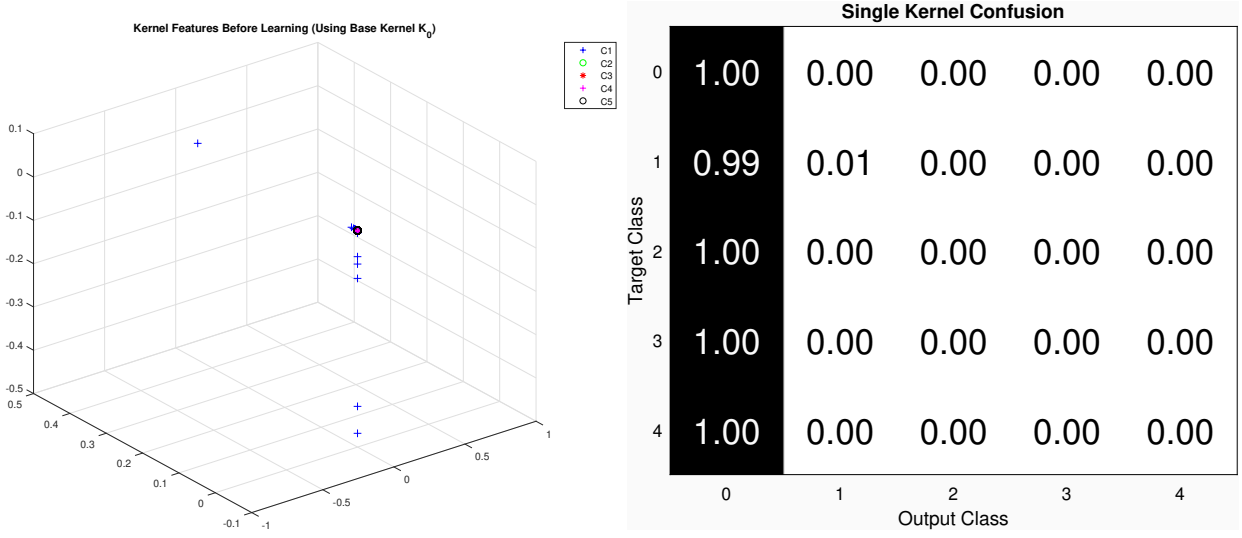
(b) Confusion

Figure 7.9: Wine Dataset Learned Kernel



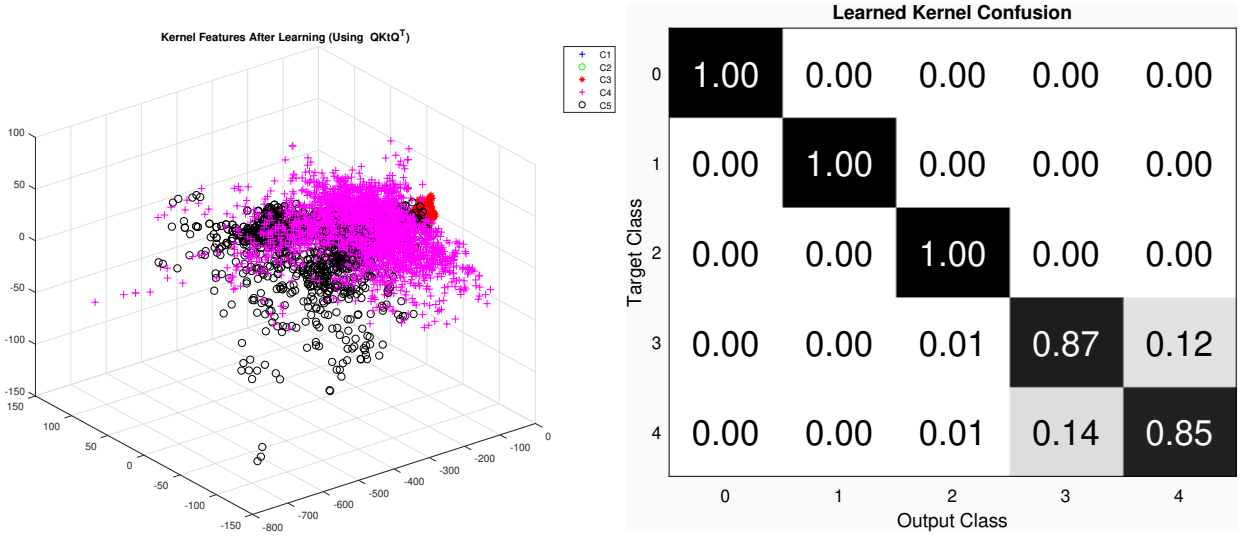
(a) Input Space Features (b) Confusion

Figure 7.10: HAR Dataset



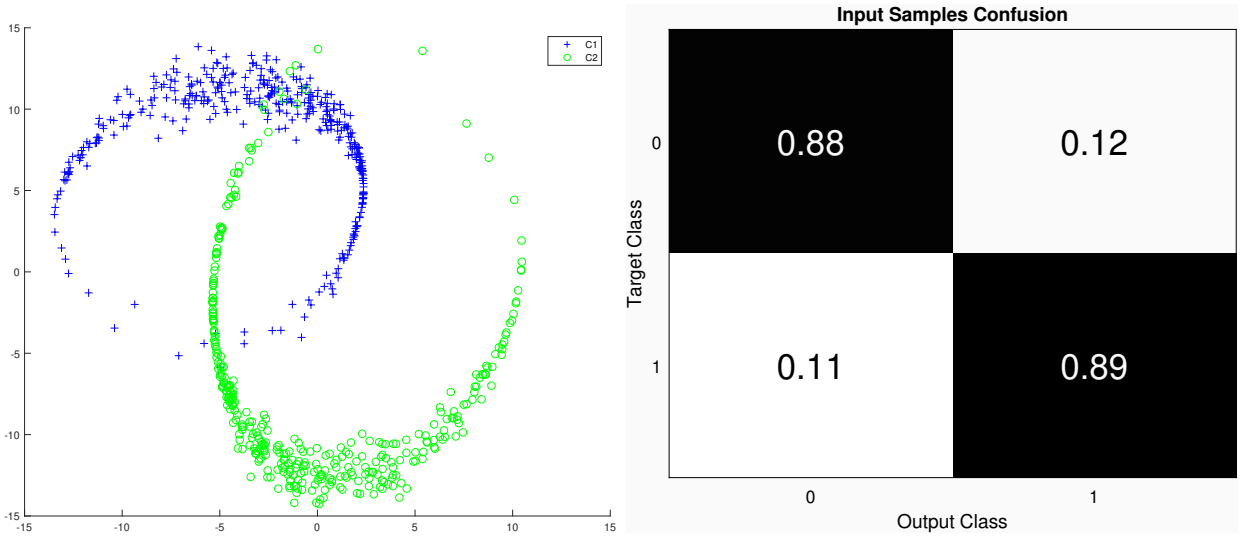
(a) Single Kernel Features (b) Confusion

Figure 7.11: HAR Dataset Single Kernel



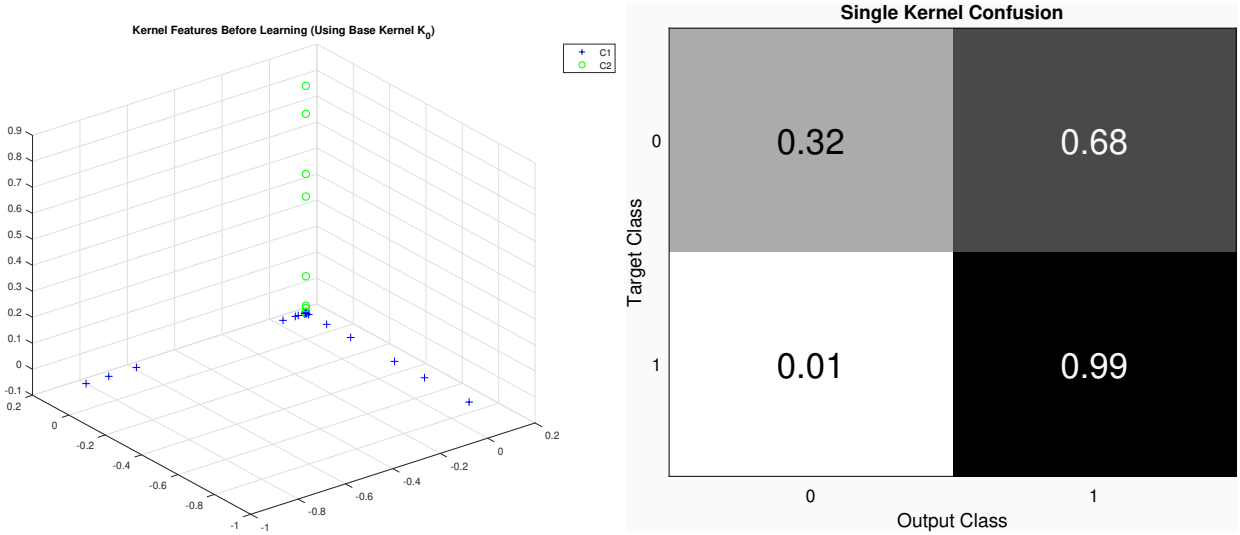
(a) Learned Kernel Features (b) Confusion

Figure 7.12: HAR Dataset Learned Kernel



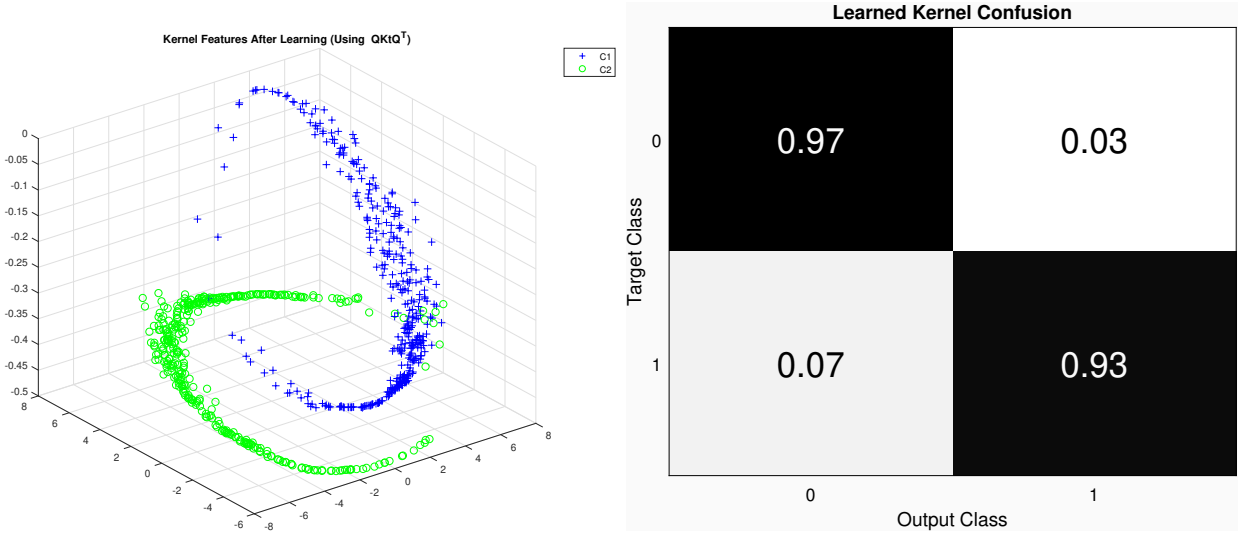
(a) Input Space Features (b) Confusion

Figure 7.13: Swirled Points Dataset



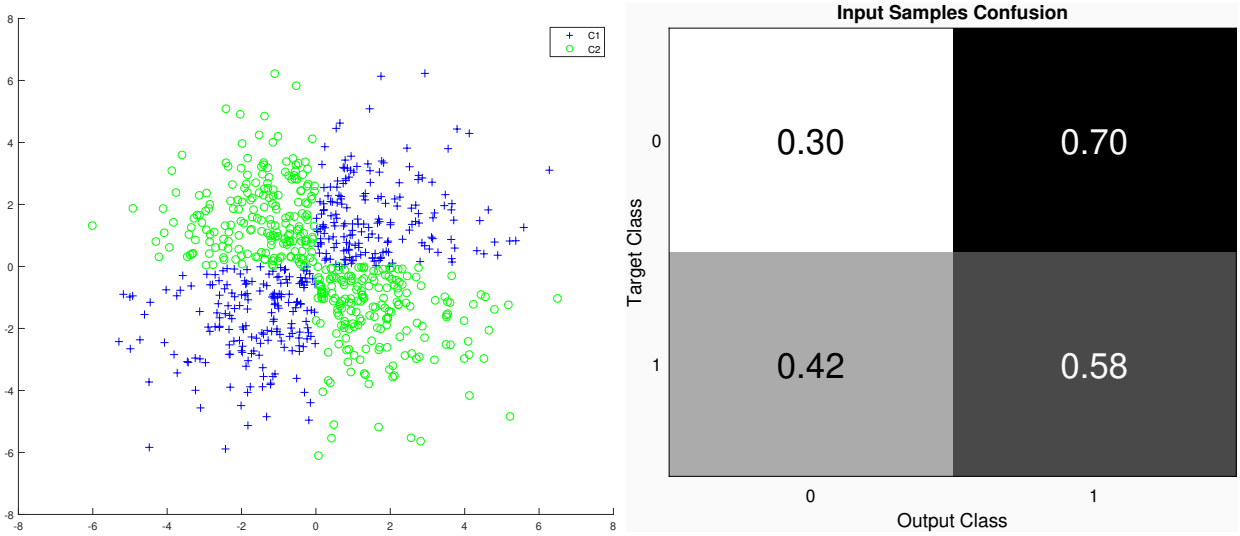
(a) Single Kernel Features (b) Confusion

Figure 7.14: Swirled Points Dataset Single Kernel



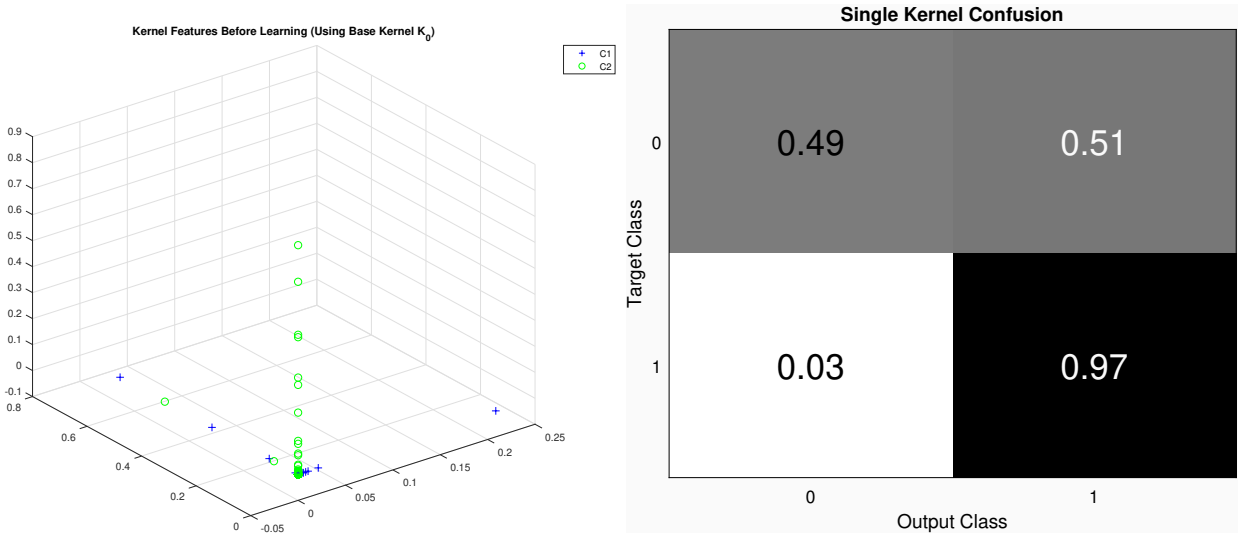
(a) Learned Kernel Features (b) Confusion

Figure 7.15: Swirled Points Dataset Learned Kernel



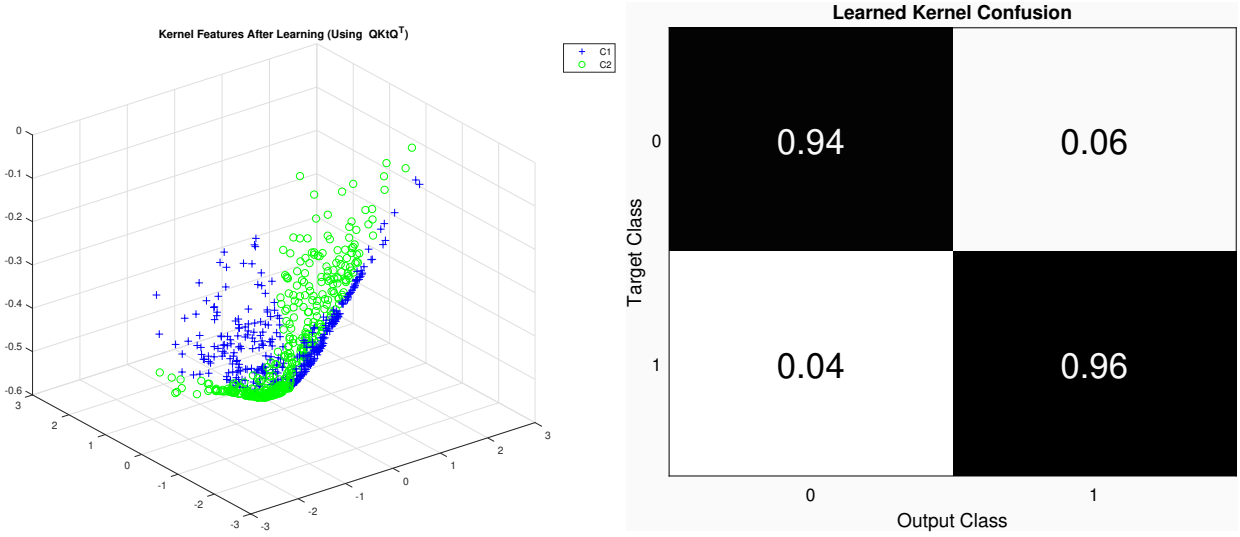
(a) Input Space Features (b) Confusion

Figure 7.16: XOR Dataset



(a) Single Kernel Features (b) Confusion

Figure 7.17: XOR Dataset Single Kernel



(a) Learned Kernel Features

(b) Confusion

Figure 7.18: XOR Dataset Learned Kernel

7.9 Conclusion

In this chapter we presented three different algorithms using two new methods for optimizing a kernel function based on the geometry of data samples in the empirical feature space. These methods serve as an essential tool to ensuring that our linearized kernel embeddings are suitable for classification tasks. With these algorithms, the difficulty of choosing an appropriate kernel function for a given classification task is reduced. Ambiguity over the best choice of kernel mapping is reduced by developing a procedure for generating the kernel function itself. After developing the theory for these methods, numerical examples were provided for several datasets that are known to be difficult, or impossible, to linearly separate and which necessitate the usage of non-linear mappings such as those provided by kernel methods.

CHAPTER 8

CLASSIFICATION RESULTS ON REAL SONAR DATASETS

8.1 Introduction

In this Chapter we provide extensive testing of the proposed methods and sampling approaches discussed in Chapters 4-7. To demonstrate the performance of these methods, one synthetic and two real sonar datasets are utilized. The synthetic dataset is the Fast Ray Model (FRM model) described in Chapter 2 and [8]. The two real sonar datasets are those described in Chapter 2, namely, the TREX and the PondEX datasets. In Section 8.2.1 we provide results using the standard modified MSC with incremental model updates via the IK-SVD approach [17]. In Section 8.2.2 we provide results using the incremental LKE modified MSC with the incremental model updates made via the IK-SVD. In these experiments a uniform sampling strategy is used for the Nyström approximation. In Section 8.2.3 we provide results using the incremental LKE modified MSC with the incremental model updates made via the IK-SVD, in which the RLS sampling strategy from Chapter 5 is adopted instead of uniform sampling. In Section 8.2.4 we provide results using the incremental LKE modified MSC with RLS sampling, however, in this method we use a kernel function learned via the discriminative multi-kernel learning (Algorithm 2) in Chapter 7. In each of these sections, results are provided for a "No-holdout" and "holdout" scenario, the latter of which performs incremental updates without prior training or incremental updating from UXO objects 4 and 9. Then in Section 8.3 we compare the baseline, incremental 1 and incremental 2 models of each method side by side for both of the real datasets in the no holdout case. Lastly, we provide some concluding remarks on the results.

8.2 Experimental Results

The results in these subsections are presented on 3 sonar datasets for 4 different approaches, referred to as Cases 1 through 4, to incremental model adaptation which address the lifelong learning problem for sonar munitions classification. In the following we present Receiver Operating Characteristic (ROC) curves which have been generated for 3 different models generated via the 4 different incremental methods. Case 1 presents results using the linear version of the modified MSC, as with the other methods, here the incremental updates are made via IK-SVD. Case 2 presents results using the LKE variant of the modified MSC with uniform sampling. Case 3 presents results using the LKE variant of modified MSC with ridge leverage score (RLS) sampling. Lastly in Case 4, we present the same method as in Case 3 but with a discriminative kernel function learned via Algorithm 2 from Chapter 7. Table 8.1 displays the objects that were present in each of the datasets used for training and testing of the models.

Table 8.1: Object Types in Training and Testing Sets

Class No.	Type	Description	**FRM***	**TREX**	**PondEX**
Cl.1	**Non-UXO**	**2 ft Alum. Cyl.**	✓	✓	✓
Cl.2	**Non-UXO**	**3 ft Alum. Cyl.**	✓	✓	
Cl.3	**Non-UXO**	**Alum. Pipe**	✓	✓	✓
Cl.4	**UXO**	**Alum. UXO**	✓	✓	✓
Cl.5	**UXO**	**105 mm Bullet (Air)**	✓	✓	
Cl.6	**UXO**	**105 mm Bullet (H2O)**	✓	✓	
Cl.7	**UXO**	**Howitzer w/ Cap (Air)**	✓	✓	✓
Cl.8	**UXO**	**Howitzer w/ Cap (H2O)**	✓	✓	
Cl.9	**UXO**	**Howitzer w/o Cap**	✓	✓	
Cl.10	**UXO**	**St. Steel UXO**	✓	✓	✓

In this chapter, all reported results are provided using 7 aspects per decision. Multi-aspect classification is more amenable to actual operational situations where several views from an underwater objects are received. In the TREX, and PondEX experiments, aspect separation is uniform due to the rail system used in collection. To account for platform instability in a realistic data collection scenario, aspect separation in testing sets is modeled by a uniformly distributed random variable $s \sim unif\ \{8, 16\}$ which shuffles data, beginning with the first

aspect in a given run. Note that the procedure does not shuffle between rotations of objects, only the order in which the aspects of a single linear run are encountered is changed.

Across all methods, the same essential training strategy was used. First, the FRM dataset was utilized in training of a Baseline Model. Then, the Incremental 1 (Inc. 1) model is generated by updating the Baseline model, via IK-SVD, using $\approx 5\%$ of *all* TREX samples. The Incremental 2 (Inc. 2) Model is then generated by updating the Inc. 1 model using $\approx 5\%$ of all PondEX samples.

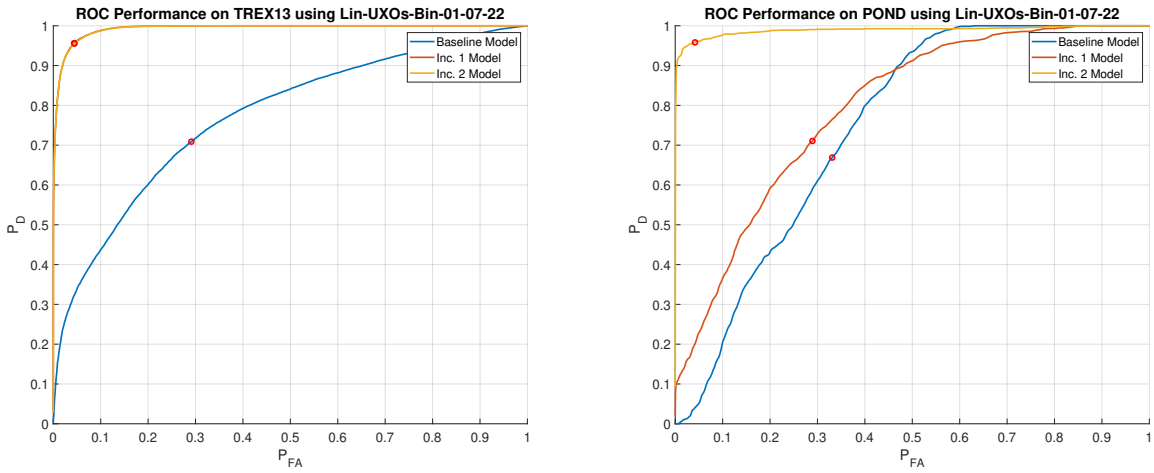
For Cases 2 through 4, where LKE based methods are used, the chosen parameters are, $c = 2000$, $k = 290$, $k_2 = k_3 = 315$ and $b = 50$. The methods in Cases 2 and 3 used a radial basis function (RBF) kernel $\kappa(\mathbf{x}, \mathbf{y}) = \exp(\frac{-||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})$ with $\sigma = 1.0$ In order to allow IK-SVD updates after an embedding update is made, the mapping $\mathbf{T} = \text{argmin}_{\mathbf{T}} ||\mathbf{T}\mathbf{F}_R - \tilde{\mathbf{F}}_R||_F^2$ (where $\mathbf{F}_R, \tilde{\mathbf{F}}_R$ represent the old and new virtual feature representations of the important samples, respectively) is utilized to transport all dictionaries learned in the old embedding up to the new embedding. Then IK-SVD training can be performed using the updated virtual samples corresponding to the incremental samples [17].

During baseline training, K-SVD was utilized [19]. Dictionaries were trained to meet an mean absolute error (MAE) of 10^{-7} or a maximum of 30 training iterations per dictionary. The sparsity factor $\tau = 15$, and $K = 300$ atoms were trained per dictionary. When performing IK-SVD updates to the dictionaries, the number of incrementally trained atoms $K_1 = 40$ for the Inc. 1 model and $K_1 = 10$ for the Inc. 2 model. All other IK-SVD parameters were the same as those for the baseline K-SVD training. For Case 4 where a custom kernel is learned via discriminative multi-kernel learning (DMKL), only the baseline training samples, i.e. the FRM samples, are used to learn the custom kernel function. The chosen parameters for the DMKL are $\eta_0 = 0.0001$, and $T = 300$ with three base kernels utilized: two RBF kernels with $\sigma \in \{1.0, 2.0\}$ and a cosine kernel function.

The ROCs were then generated for 3 different models: Baseline, Incremental 1, and

Incremental 2, for each of the datasets in both holdout and no holdout scenarios. The knee-point performance for each of the ROC curves, i.e. the point where $P_{CC} + P_{FA} = 1$, are recorded in tables.

8.2.1 Case 1 – Linear Modified MSC

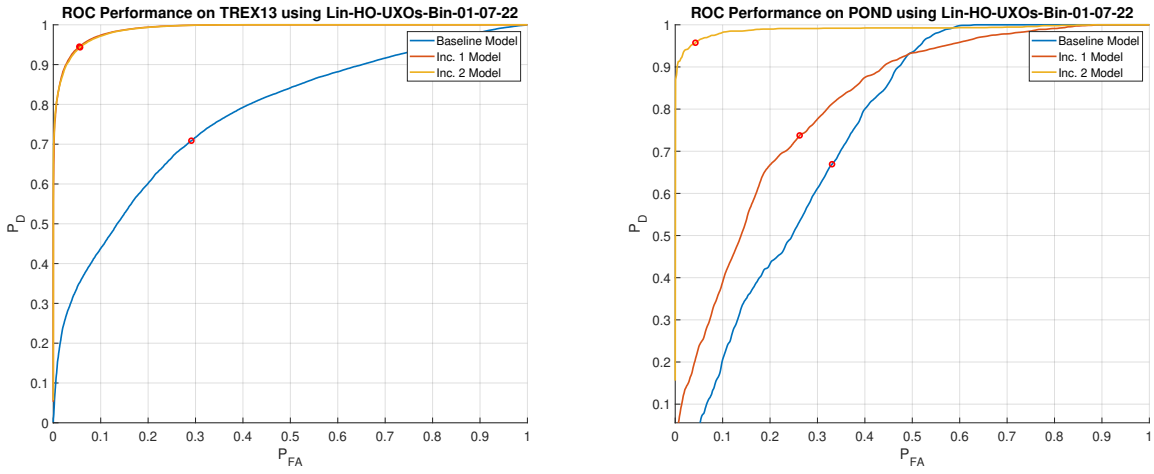


(a) TREX Dataset

(b) PondEX Dataset

Figure 8.1: Linear modified MSC (no holdout)

To begin, we note that the final incremental model (denoted Inc. 2) for the linear form of the incremental modified MSC achieved $P_{CC} = 95.5\%$, $P_{FA} = 4.5\%$, and $P_{CC} = 95.8\%$, $P_{FA} = 4.2\%$ for TREX, and PondEX respectively. Furthermore, the TREX and FRM performance only saw minor degradations in performance in successive incremental trainings. In the holdout scenario, the linear modified MSC still managed to attain acceptable results but the lacking representation of the heldout objects impacted the performance of on the TREX dataset. The linear form of the incremental modified MSC when objects were held out achieved $P_{CC} = 94.3\%$, $P_{FA} = 5.7\%$, and $P_{CC} = 95.7\%$, $P_{FA} = 4.3\%$ for TREX, and PondEX respectively. Another interesting thing to note is that for the holdout and no-holdout scenarios, the performance of PondEX increased by $\approx 4\%$ and $\approx 7\%$, respectively,



(a) TREX Dataset

(b) PondEX Dataset

Figure 8.2: Linear modified MSC (w/ holdout)

when the model was incrementally trained on the TREX samples, i.e. when the models transitioned from baseline to Incremental 1. While the final TREX performance of the other methods (Cases 2-4) were comparable, the linear model managed to hold the lead on this dataset alone.

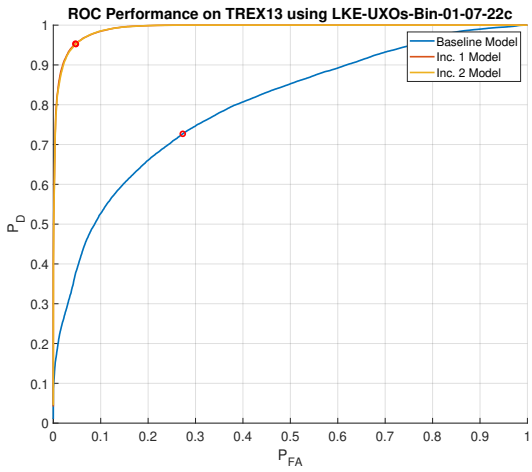
Table 8.2: Linear modified MSC no Holdout Knee-Points

	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.0010575	0.99894	0.29125	0.70875	0.33075	0.66925
Inc. 1	0.0010575	0.99894	0.043819	0.95618	0.28985	0.71015
Inc. 2	0.0010805	0.99892	0.04458	0.95542	0.041734	0.95827

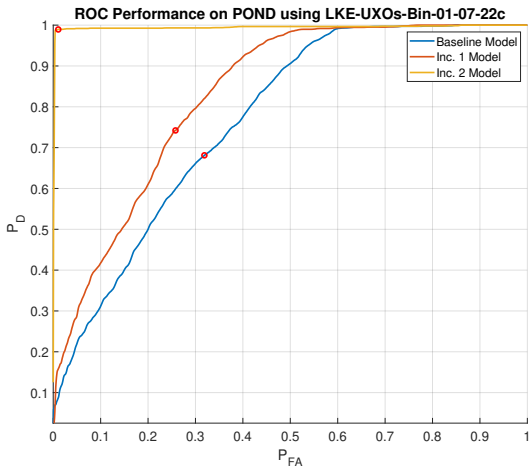
Table 8.3: Linear modified MSC w/ Holdout Knee-Points

	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.00093103	0.99907	0.29117	0.70883	0.33141	0.66859
Inc. 1	0.00095402	0.99905	0.055042	0.94496	0.26177	0.73823
Inc. 2	0.001023	0.99898	0.056559	0.94344	0.042552	0.95745

8.2.2 Case 2 – Linearized Kernel Embedding modified MSC (Uniform Sampling)

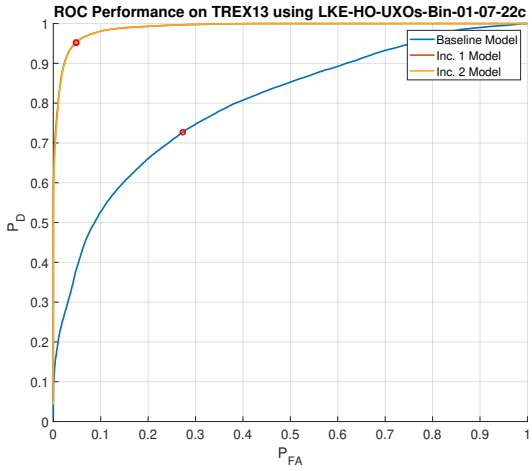


(a) TREX Dataset

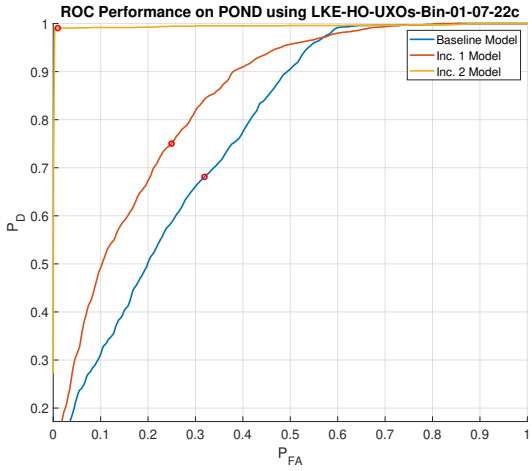


(b) PondEX Dataset

Figure 8.3: LKE modified MSC (no holdout)



(a) TREX Dataset



(b) PondEX Dataset

Figure 8.4: LKE modified MSC (w/ holdout)

In the next method tested, the linearized kernel embedding (LKE) modified MSC with uniform sampling was tasked with incrementally learning the three real sonar datasets. The

final incremental model using LKE modified MSC with uniform sampling achieved $P_{CC} = 95.2\%$, $P_{FA} = 4.8\%$, and $P_{CC} = 98.9\%$, $P_{FA} = 1.1\%$ for TREX, and PondEX, respectively. When compared to the linear case, the PondEX datasets saw considerable improvements over the linear classifier and the TREX performance was only slightly degraded in the no holdout case and improved in the holdout case. Surprisingly, in the holdout case, this method improved on its PondEX performance when compared to the no holdout case! The LKE modified MSC with uniform sampling in the objects-heldout case achieved $P_{CC} = 95.2\%$, $P_{FA} = 4.8\%$, and $P_{CC} = 99.0\%$, $P_{FA} = 1.0\%$ for TREX and PondEX, respectively. As with the linear version of modified MSC, it is interesting to note that for both the holdout and no-holdout scenarios, the performance of PondEX increased by nearly 7% when the model was incrementally trained on the TREX samples.

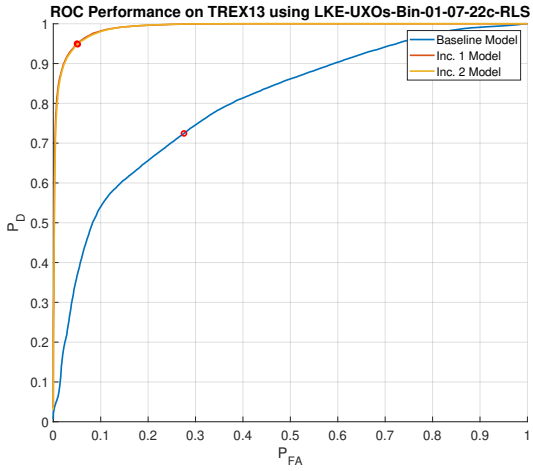
Table 8.4: LKE modified MSC no Holdout Knee-Points

	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.0011138	0.99889	0.2728	0.7272	0.31895	0.68105
Inc. 1	0.0012274	0.99877	0.046685	0.95331	0.25757	0.74243
Inc. 2	0.0012729	0.99873	0.047641	0.95236	0.010575	0.98943

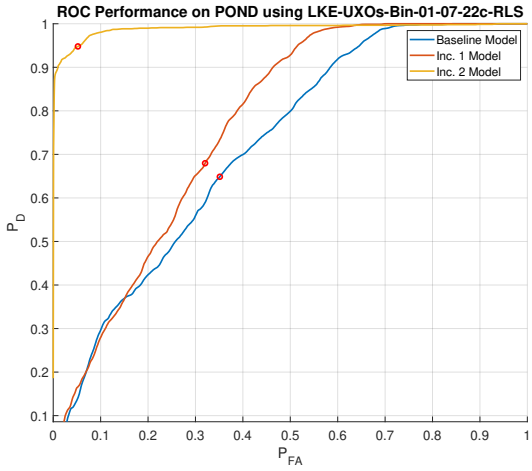
Table 8.5: LKE modified MSC w/ Holdout Knee-Points

	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.0010115	0.99899	0.27281	0.72719	0.31906	0.68094
Inc. 1	0.0011592	0.99884	0.04788	0.95212	0.24951	0.75049
Inc. 2	0.0010438	0.99896	0.048203	0.9518	0.009518	0.99048

8.2.3 Case 3 – Linearized Kernel Embedding modified MSC (RLS Sampling)

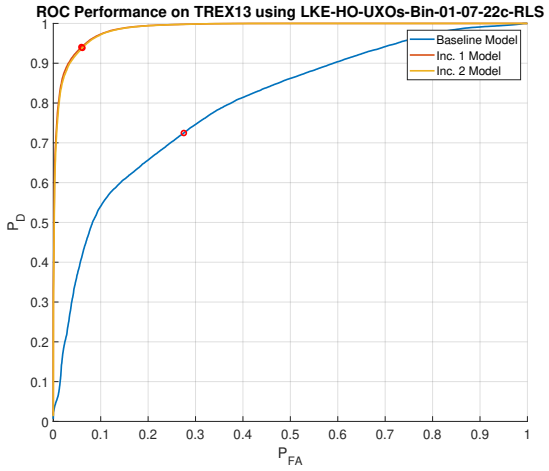


(a) TREX Dataset

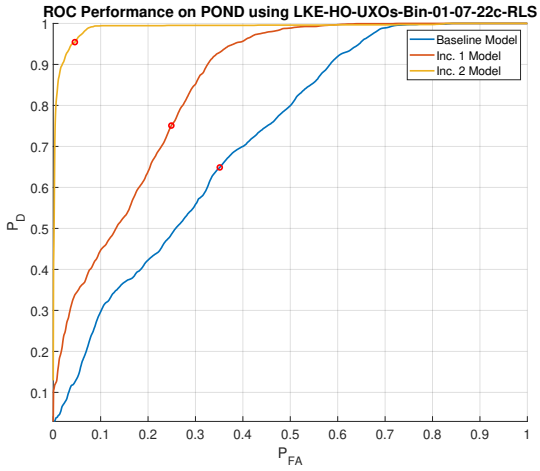


(b) PondEX Dataset

Figure 8.5: LKE modified MSC (RLS Sampling) (no holdout)



(a) TREX Dataset



(b) PondEX Dataset

Figure 8.6: LKE modified MSC (RLS Sampling) (w/ holdout)

The third method that was tested once again used the LKE modified MSC classifier, but this time, RLS sampling was utilized for selecting the new important samples, rather

than uniform sampling. The final incremental model for this method, somewhat counter-intuitively, provided the lowest overall performance among the methods tested. The LKE modified MSC with RLS sampling achieved $P_{CC} = 94.8\%$, $P_{FA} = 5.2\%$, and $P_{CC} = 94.8\%$, $P_{FA} = 5.2\%$ for TREX and PondEX, respectively. In the holdout case, this method managed to achieve better results than the linear method for the PondEX dataset. The LKE modified MSC with RLS sampling in the objects-heldout case achieved $P_{CC} = 93.9\%$, $P_{FA} = 6.1\%$, and $P_{CC} = 95.4\%$, $P_{FA} = 4.6\%$ for TREX and PondEX, respectively.

As we discussed in Chapter 7, having strong reconstruction guarantees alone may not be enough to ensure good discrimination performance for the MSC. In the next section we try to improve on this result by coupling the RLS sampling with a custom kernel learned for discrimination tasks. As we will see in Case 4, not only does this coupling allow us to capitalize on the RLS sampling, this approach seems to allow for better generalization on unseen data.

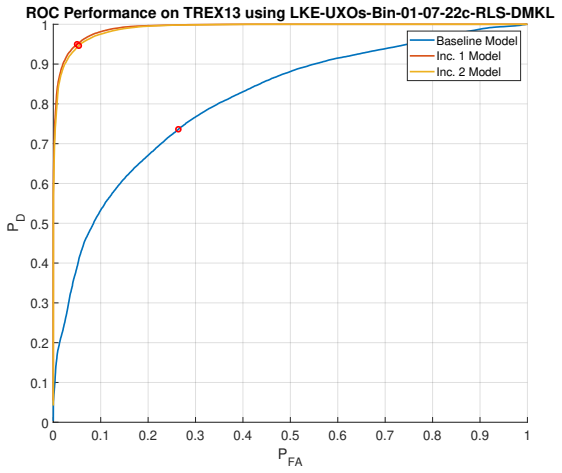
Table 8.6: LKE modified MSC (RLS Sampling) no Holdout Knee-Points

	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.0010115	0.99899	0.27559	0.72441	0.35112	0.64888
Inc. 1	0.00098875	0.99901	0.050028	0.94997	0.32086	0.67914
Inc. 2	0.0011365	0.99886	0.051528	0.94847	0.052217	0.94778

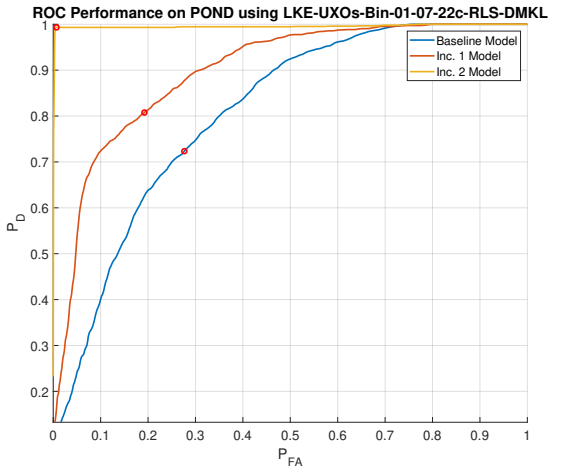
Table 8.7: LKE modified MSC (RLS Sampling) w/ Holdout Knee-Points

	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.00093192	0.99907	0.27559	0.72441	0.35184	0.64816
Inc. 1	0.0011056	0.99889	0.059583	0.94042	0.25039	0.74961
Inc. 2	0.0010115	0.99899	0.061408	0.93859	0.046064	0.95394

8.2.4 Case 4 – Linearized Kernel Embedding modified MSC (RLS Sampling + DMKL)

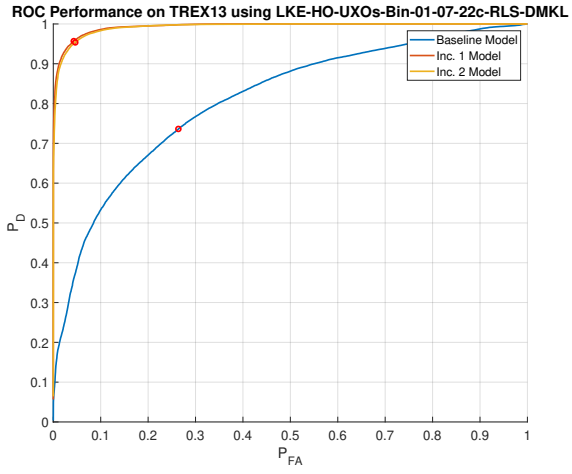


(a) TREX Dataset

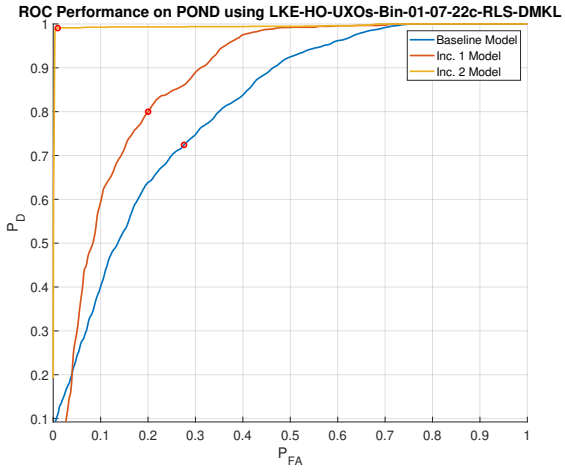


(b) PondEX Dataset

Figure 8.7: LKE modified MSC (RLS Sampling + DMKL) (no holdout)



(a) TREX Dataset



(b) PondEX Dataset

Figure 8.8: LKE modified MSC (RLS Sampling + DMKL) (w/ holdout)

The fourth and final method that was tested once again used the LKE modified MSC

classifier, but this time, RLS sampling was utilized for selecting the new important samples *and* the discriminative multi-kernel learning was used to help ensure that the strong approximation guarantees of RLS Nyström correspond to better discrimination. The final incremental model for this method provided the best overall performance among the methods tested. While the final TREX performance was slightly lower than the other tested methods, it was still comparable. The LKE modified MSC with RLS sampling and a learned kernel achieved $P_{CC} = 94.6\%$, $P_{FA} = 5.4\%$, and $P_{CC} = 99.3\%$, $P_{FA} = 0.7\%$ for TREX and PondEX, respectively. In the holdout case, this method managed to achieve better results than the linear method for both the PondEX and TREX datasets. The LKE modified MSC with RLS sampling and learned kernel in the objects-heldout case achieved $P_{CC} = 95.4\%$, $P_{FA} = 4.6\%$, and $P_{CC} = 99.0\%$, $P_{FA} = 1.0\%$ for TREX and PondEX, respectively. Perhaps the most impressive result for this method is the improvements that were made to PondEX testing when incremental TREX data was used to generate the Incremental 1 models. For both holdout and no holdout cases, the PondEX performance increased by $\approx 8\%$, providing the best Incremental 1 performance across all methods.

Table 8.8: LKE modified MSC (RLS Sampling + DMKL) no Holdout Knee-Points

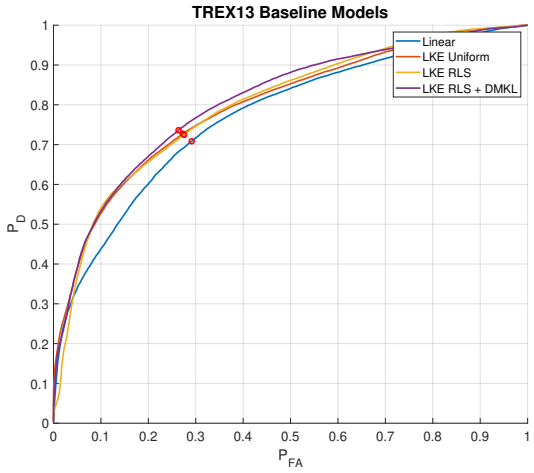
	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.00079999	0.9992	0.26375	0.73625	0.27637	0.72363
Inc. 1	0.0011251	0.99887	0.050251	0.94975	0.19319	0.80681
Inc. 2	0.0008751	0.99912	0.053879	0.94612	0.0067981	0.9932

Table 8.9: LKE modified MSC (RLS Sampling + DMKL) w/ Holdout Knee-Points

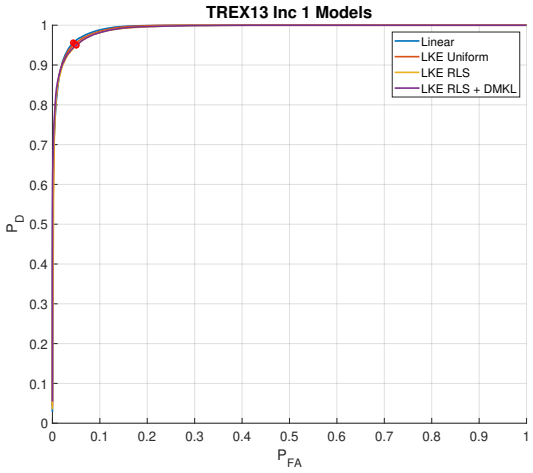
	FRM		**TREX**		**PondEX**	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Baseline	0.00082964	0.99917	0.26387	0.73613	0.27659	0.72341
Inc. 1	0.0011024	0.9989	0.043147	0.95685	0.19932	0.80068
Inc. 2	0.00094329	0.99906	0.046224	0.95378	0.0094629	0.99054

8.3 Comparing by Model

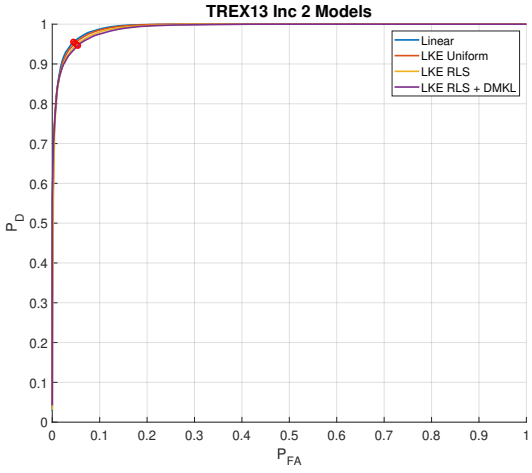
In this section we display the no-holdout scenario results in a different format in order to more clearly compare the performance of the 4 methods as model updates are made. Each figure displays 4 ROC curves, each corresponding to one of the cases presented above, for a fixed model (i.e. Baseline, Inc. 1, etc.) and dataset.



(a) Baseline



(b) Inc. 1



(c) Inc. 2

Figure 8.9: TREX Performance, 4 Methods

From this first set of figures we look at the performance on TREX data using the 4 methods. We can see that all four methods provide comparable Inc. 1 and Inc. 2 performance

for the TREX dataset, all close to 95%. In baseline, the LKE methods manage to edge out the linear method, with the RLS + DMKL method (Case 4) performing the best. In the first incremental update, the Linear method performs the best with the LKE methods close behind. The same is true for the second incremental update.

Table 8.10: TREX Performance, 4 Methods, 3 Models

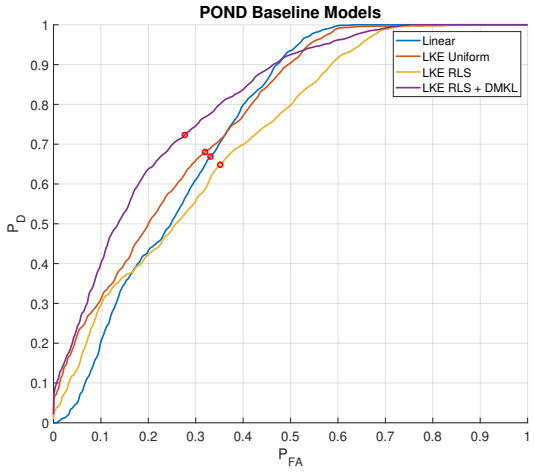
	Baseline		Inc. 1		Inc. 2	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Linear	0.29125	0.70875	0.043819	0.95618	0.04458	0.95542
LKE + Uniform	0.2728	0.7272	0.046685	0.95331	0.047641	0.95236
LKE + RLS	0.27559	0.72441	0.050028	0.94997	0.051528	0.94847
LKE + RLS + DMKL	0.26375	0.73625	0.050251	0.94975	0.053879	0.94612

For the PondEX dataset, we observe that among the baseline models for each method, the LKE + RLS method (Case 3) performs the worst being beaten even by the linear case. The best baseline model is the one using the learned kernel function (Case 4). A similar story holds for the incremental 1 models for each method, while all methods improve by seeing TREX samples, the RLS + DMKL method performs the best managing to obtain over 80% correct classification on PondEX samples using a model which has never encountered samples from the PondEX dataset! Finally in the Incremental 2 models, the Uniform LKE and RLS + DMKL version of LKE modified MSC have a clear advantage over the linear method.

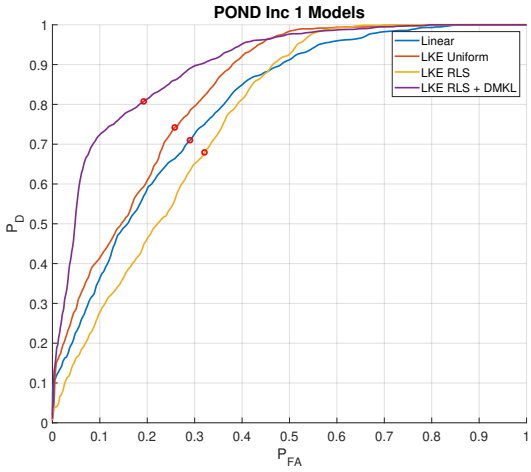
Table 8.11: PondEX Performance, 4 Methods, 3 Models

	Baseline		Inc. 1		Inc. 2	
	P_{FA}	P_{CC}	P_{FA}	P_{CC}	P_{FA}	P_{CC}
Linear	0.33075	0.66925	0.28985	0.71015	0.041734	0.95827
LKE + Uniform	0.31895	0.68105	0.25757	0.74243	0.010575	0.98943
LKE + RLS	0.35112	0.64888	0.32086	0.67914	0.052217	0.94778
LKE + RLS + DMKL	0.27637	0.72363	0.19319	0.80681	0.0067981	0.9932

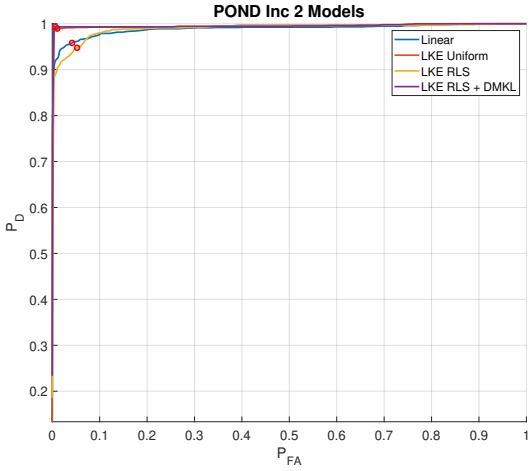
The results presented above provide a number of interesting points of discussion. All four methods perform quite well on the tested datasets and provide great motivation for the utilization of incremental model adaptation strategies for underwater target classifiers in the lifelong learning setting.



(a) Baseline



(b) Inc. 1



(c) Inc. 2

Figure 8.10: PondEX Performance, 4 Methods

One possible reason for the inferior performance of the RLS sampling strategy to the Uniform sampling strategy is that the RLS sampling looks for more linearly independent samples and since the kernel function utilized in these tests is not tailored to enforce class separability, the reconstruction guarantees of the RLS-Nyström could potentially do more harm than help. The uniform sampling method makes no such guarantees and sometimes may get "lucky" in finding an important sample that isn't necessarily unique in the linear dependence sense but that acts as an excellent template in the proceeding environments.

Utilizing a kernel learned via one of the techniques in Chapter 7, which uses a discriminative objective to enforce class structure to the latent space representations, could perhaps allow the RLS sampling strategy to win in all scenarios. This was the motivation for the experiments in Case 4.

Surprisingly, the linear method (Case 1) did quite well with the TREX13 dataset. managing to achieve the best Incremental 1 and Incremental 2 performance among the methods. Despite this, due to the its inferior performance in baseline testing and on the pond data set, the average performance over the systems lifetime lacked when compared to the LKE methods. Table 8.12 list the Performance over the lifetime of each method, for holdout and no holdout scenarios. This lifetime performance is computed by taking

$$P_{CC,Life} = \frac{N_{FRM}}{N_{tot}} P_{CC,FRM} + \frac{N_{TREX}}{N_{tot}} P_{CC,TREX} + \frac{N_{POND}}{N_{tot}} P_{CC,POND},$$
$$P_{FA,Life} = \frac{N_{FRM}}{N_{tot}} P_{FA,FRM} + \frac{N_{TREX}}{N_{tot}} P_{FA,TREX} + \frac{N_{POND}}{N_{tot}} P_{FA,POND}$$

where N_{data} is the number of testing samples from dataset 'data', $N_{tot} = N_{FRM} + N_{TREX} + N_{POND}$, and $P_{CC,data}, P_{FA,data}$ is the average knee-point performance of a given method for dataset 'data'.

Table 8.12: Overall Lifetime Performance by Method

	Lifetime P_{CC}	Lifetime P_{FA}	Lifetime P_{CC} HO	Lifetime P_{FA} HO
Linear	0.8820	0.1180	0.8754	0.1246
LKE + Uniform	0.8862	0.1138	0.8860	0.1140
LKE + RLS	0.8821	0.1179	0.8771	0.1229
LKE + RLS + DMKL	0.8871	0.1129	0.8913	0.1087

This table provides the expected average performance of a given method across all datasets over its lifetime. While the performance increase does not seem as notable in the no holdout scenario, the kernel methods appear to have a clear advantage in the, more realistic, holdout scenario. In light of these results, the coupling of efficient incremental kernel maps [4, 5], incremental model adaptation [17, 18], and discriminative kernels [15], provide a promising framework for the development of efficient lifelong learning systems for

pattern recognition that are resilient to "catastrophic forgetting".

8.4 Conclusion

In this chapter we provided extensive results comparing four different strategies to incremental learning for underwater target classification in three different environments. The models presented each had their baseline training performed on purely model-generated data and then were progressively adapted using only a small fraction of samples from newly encountered environments. In both the object holdout and non-holdout scenarios, the incremental linearized kernel embedding (LKE) variants provided the best overall performance. When using the LKE modified MSC with RLS sampling and custom kernel function, the we achieve $P_{CC} = 94.6\%$, $P_{FA} = 5.4\%$ and $P_{CC} = 99.3\%$, $P_{FA} = 0.7\%$ for TREX and Pond Datasets respectively. The variant utilizing RLS sampling provided superior performance to the linear case both in the object holdout and no holdout scenarios. For these experiments, the RLS sampling + DMKL approach provided the best overall performance including when objects were held out. The testing in this chapter provides great motivation for the utilization of the incremental model updating strategies for underwater target classification or similar problems. Lastly, it should be noted that while the LKE methods provided superior performance to the linear modified MSC, they also incurred extra cost, not only in the necessary kernel evaluations but in the computation of a least squares transform for transporting old dictionaries into updated embeddings. Each of these factors must be taken into consideration in the design of a lifelong learning system.

CHAPTER 9

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

9.1 Conclusions and Discussions

The desire for repeatable and reliable Automatic Target Recognition (ATR) in ever-changing shallow underwater environments was the primary motivation for the methods we have developed and presented. Over a systems lifetime, we would like the system to perform well on old and newly encountered environments despite variations in operating and environmental conditions, variations in the targets, and the presence of clutter of varying size and composition. In this work we investigated lifelong learning solutions for underwater ATR problems and introduced several new techniques for developing classification models that can adapt over time to accommodate for and circumvent the aforementioned variations and difficulties. In particular, we investigated solutions which coupled the benefits of sparse reconstruction classifiers and kernel embeddings and converted these solutions to an incremental form.

The main objective of the work in this dissertation was the investigation of incremental training and embedding strategies to allow a classifier to adapt to new environments for use in a lifelong learning setting. Throughout the body of this work we introduced several new techniques: 1) a method for fully kernelizing the modified Matched Subspace Classifier (MSC) of [2] was developed which utilized the Kernel K-SVD learning method of [21]. This method allows dictionary atoms to be represented through a linear combination of the samples in the kernel feature space but requires the utilization of the full kernel matrix for both dictionary learning and decision making via the modified MSC. 2) We introduced a more

efficient technique for kernelizing the modified MSC using a linearized kernel embedding [4]. This embedding has several appealing properties that were discussed in Chapter 4 and is closely related to the kernel PCA solution [34]. 3) Next, in Chapter 5 we introduced several techniques for performing importance sampling for the important representative samples used in said embedding. We discussed a new sampling technique with strong reconstruction guarantees known as ridge leverage score (RLS) sampling [5]. 4) Then, in Chapter 6 we discussed an incremental strategy for updating this linearized kernel embedding when new important samples have been discovered. We provided two techniques for performing the incremental update to the embedding including a new method that leverages fast arrowhead eigendecompositions [54] to perform the update in the most competitive time.

These methods were complimented by several other minor accomplishments such as the development of two techniques for the design of kernel functions themselves which promote class separability, thereby making a reconstructive importance sampling even more effective at identifying the most critical samples for constructing a linearized kernel embedding. In our results chapter we picked the most promising among these techniques in order to construct incremental variants of a MSC-based UXO vs. non-UXO classifier whose baseline models were trained exclusively using model-generated AC data. We evaluated these methods using two real low-frequency sonar datasets, namely TREX13 and PondEX09-10. These results provide great motivation for the incorporation of incremental feature embeddings and model adaptation for underwater object classifiers in the lifelong learning setting.

9.2 Future Work

The need to accomplish several new tasks has been revealed by this work. This research has suggested a path towards more consistent UXO identification that can provide high accuracy decisions over the lifetime of a systems deployment. In future endeavors, we hope to accomplish the following two major tasks and several smaller tasks related to the problem of underwater UXO classification in changing environments. These endeavors include

information gain based sampling and multi-aspect decision making strategies.

1 - Information Gain Informed Active Learning Throughout this work, the focus of the importance sampling for the linearized kernel embedding was to pick samples which bring the most representative capabilities for reconstructing the full kernel matrix solution. The RLS Nyström sampling scheme in [5] implements this idea by finding those samples which are the "most" linearly independent among the training ensemble. However, for the purposes of classification, excellent reconstructions of kernelized samples alone can not always guarantee excellent classification performance. Another approach that could be considered is a sampling strategy that is based on the information gain (IG) of observed samples [72]. Like the RLS sampling strategy, this technique could be utilized with or without the availability of label information for incremental samples. By utilizing a metric like the information gain, the uniqueness of samples could be described in a manner that is not completely dependent on their reconstructive uniqueness and can leverage the class-conditional distributions that have accumulated in order to pick out the samples that have more ambiguous class membership.

In a similar vein, this approach may be able to provide a holistic mechanism for not only sample selection, but also sample elimination or pruning. Since many samples will become redundant over the lifetime of a system, the desire for a mechanism which removes the easily classified samples which carry little unique information with respect to the rest of the important sample set.

2 - IG Informed Multi-Aspect Decision Making The usage of the information gain is not limited to identifying the most important samples for model building or for kernel embeddings. The same metric can be utilized in the sensing process in order to pick the most informative subset of aspect angles, on an object, among our observations so as not to dilute the classification accuracy by presenting redundant or uninformative aspects to the inference system. Such an approach would allow for more efficient decision making and would provide the IG for encountered samples in the process which would directly serve the

importance sampling strategy discussed previously.

3 - Robustness Testing The usage of real and synthesized sonar data to determine the effectiveness of the developed adaptive dictionary learning and sparse representation methods in different situations. One should investigate how the model and dictionaries can be designed to guarantee robustness to partial burial or occlusion of the targets, and separation of similar class types.

4 - Decremental Updating The development of a decremental embedding "downdate" strategy with similar complexity to the update strategy using arrowhead eigendecompositions. As was discussed briefly at the end of Chapter 6, it can easily be shown that the method of Hallgren et. al. [53] can be utilized to perform downdates by essentially reversing the steps of the update procedure. Such a downdate is not so apparent for the arrowhead updates approach but the appealing complexity decrease of the arrowhead updates approach begs for further investigation into a similar method for downdating the embedding.

5 - Field Testing with UUV or Towbody In order to further verify the utility of incremental embedding and classification strategies, and to determine the optimal hyper parameters for continual adaptation (e.g. how many more kernel principal components to add and "the best" importance sampling approach), field testing of the strategy using a real sonar platform is of the utmost importance. Controlled experiments could be performed which allow the sonar platform to encounter several environments, each with objects of known location and type, in sequence.

Underwater UXO classification presents one of the most difficult problems in the realms of pattern recognition and remote sensing. In order to overcome the inherent environmental dynamics associated with this problem, models too must continuously adapt. With these objectives in place, a clearer path to a true lifelong learning machine for separating 'UXO' and 'non-UXO' objects will be achieved. By further developing incremental models, incremental

embeddings, and discriminative kernel maps, along with mechanisms for maintaining these components we will approach ever closer to a complete solution to the problem of classifying munitions for reclamation.

REFERENCES

[1] J. J. Hall, "Underwater UXO Classification using Matched Subspace Classifier with Synthetic Sparse Dictionaries," *Master's Thesis, Colorado State University*, 2016.

[2] J. J. Hall, M. R. Azimi-Sadjadi, S. G. Kargl, Y. Zhao, and K. L. Williams, "Underwater unexploded ordnance (UXO) classification using a matched subspace classifier with adaptive dictionaries," *IEEE Journal of Oceanic Engineering*, vol. 44, no. 3, pp. 739–752, 2018.

[3] C. K. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, 2001, pp. 682–688.

[4] A. Golts and M. Elad, "Linearized kernel dictionary learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 726–739, 2016.

[5] C. Musco and C. Musco, "Recursive sampling for the nystrom method," in *Advances in Neural Information Processing Systems*, 2017, pp. 3833–3845.

[6] A. Salberg, A. Hanssen, and L. Scharf, "Robust multidimensional matched subspace classifiers based on weighted least-squares," *IEEE Transactions Signal Processing*, vol. 55, pp. 873–880, March 2007.

[7] M. R. Azimi-Sadjadi and S. G. Kargl, "Multichannel detection and acoustic color-based classification of underwater uxo in sonar," *Final Report, SERDP Project MR-2416*, pp. 1–28, June 2015.

[8] S. Kargl, A. España, K. Williams, J. Kennedy, and J. Lopes, "Scattering from objects at a water-sediment interface: Experiment, high-speed and high-fidelity models, and physical insight," *IEEE Journal of Oceanic Engineering*, vol. 40, no. 3, pp. 632–642, 2014.

[9] S. Kargl, "Acoustic response of underwater munitions near a sediment interface: Measurement-model comparisons and classification schemes," *Final Report, SERDP Project MR-2231*, pp. 1–104, April 2015.

[10] S. Kargl and K. Williams, "Full scale measurement and modeling of the acoustic response of proud and buried munitions at frequencies from 1-30khz," *Final Report, SERDP Project MR-1665*, pp. 18–19, May 2012.

[11] S. Kargl, "Bistatic acoustic scattering from munitions: Interactions with nearby clutter and the water-sediment interface," *Interim Report, SERDP Project MR19-1234*, 2019.

[12] S. Kargl, "Acoustic response of underwater munitions near a water-sediment boundary," *Interim Report, SERDP Project MR-2505*, 2019.

[13] P. Ruvolo and E. Eaton, "Ella: An efficient lifelong learning algorithm," in *International Conference on Machine Learning*, 2013, pp. 507–515.

[14] A. Kumar and H. Daume III, "Learning task grouping and overlap in multi-task learning," *arXiv preprint arXiv:1206.6417*, 2012.

[15] H. Xiong, M. Swamy, and M. O. Ahmad, "Optimizing the kernel in the empirical feature space," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 460–474, 2005.

[16] D. P. Williams, "Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks," in *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016, pp. 2497–2502.

[17] L. Wang, K. Lu, P. Liu, R. Ranjan, and L. Chen, "IK-SVD: dictionary learning for spatial big data via incremental atom update," *Computing in Science & Engineering*, vol. 16, no. 4, pp. 41–52, 2014.

[18] M. R. Azimi-Sadjadi, Y. Zhao, and S. Sheedvash, "Incremental dictionary learning with sparsity," *Proceedings of 2018 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2018.

[19] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, November 2006.

[20] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, vol. 5. IEEE, 1999, pp. 2443–2446.

[21] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5123–5135, 2013.

[22] J. Yang, D. Tang, B. Hefner, K. Williams, and J. Preston, "Overview of reverberation measurements in target and reverberation experiment 2013," *TREX13 Workshop*, pp. 784–794, November 2014.

[23] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *Proceedings of 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 2021–2024.

[24] M. Azimi-Sadjadi, J. Kopacz, and N. Klausner, "K-SVD dictionary learning using a fast OMP with applications," *Proc. of IEEE Intern. Conference on Image Processing (ICIP)*, pp. 1599–1603, October 2014.

[25] M. B. Cohen, C. Musco, and C. Musco, "Input sparsity time low-rank approximation via ridge leverage score sampling," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 1758–1777.

[26] K. Fukunaga, *Introduction to statistical pattern recognition.* Elsevier, 2013.

[27] S. Kargl, A. España, K. Williams, J. Kennedy, and J. Lopes, "Scattering from objects at a water-sediment interface: Experiment, high-speed and high-fidelity models, and physical insight," *IEEE Journal of Oceanic Engineering*, vol. 40, no. 3, pp. 632–642, August 2015.

[28] K. Williams, S. Kargl, E. Thorsos, D. Burnett, J. Lopes, M. Zampolli, and P. Marston, "Acoustic scattering from a solid aluminum cylinder in contact with a sand sediment: Measurements, modeling, and interpretation," *J. Acoust. Soc. Am.*, vol. 127, pp. 3356–3371, 2010.

[29] A. L. España, K. L. Williams, D. Plotnick, and P. Marston, "Acoustic scattering from a water-filled cylindrical shell: Measurements, modeling, and interpretation," *Acoustical Society of America*, vol. 136, pp. 109–121, July 2014.

[30] K. Williams, 2015, private communication on TREX13 Experimental Setup.

[31] T. Marston, P. Marston, and K. Williams, "Scattering resonances, filtering with reversible sas processing, and applications of quantitative ray theory," *Proc. 2010 Oceans MTS/IEEE Seattle*, pp. 1–9, September 2010.

[32] H. Kwon and N. M. Nasrabadi, "Matched subspace detectors for hyperspectral target detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 178–194, 2006.

[33] Y. Zhou, J. Gao, and K. Barner, "Locality preserving KSVD for nonlinear manifold learning," *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3372–3376, May 2013.

[34] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International conference on artificial Neural Networks.* Springer, 1997, pp. 583–588.

[35] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, 1993, pp. 40–44.

[36] B. Schölkopf and A. J. Somola, *Learning with Kernels*. The MIT press, 2002.

[37] B. Schölkopf, C. J. C. Burgues, and A. J. Somola, *Advances in Kernel Methods*. The MIT press, 1999.

[38] V. Vapnik, "Statistical learning theory new york," *NY: Wiley*, vol. 1, p. 2, 1998.

[39] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the Nyström method," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 981–1006, 2012.

[40] J. Shawe-Taylor, C. Williams, N. Cristianini, and J. Kandola, "On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2510–2522, 2005.

[41] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[42] P. Drineas, M. W. Mahoney, and N. Cristianini, "On the Nyström method for approximating a gram matrix for improved kernel-based learning." *Journal of Machine Learning Research*, vol. 6, no. 12, 2005.

[43] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix," *SIAM Journal on computing*, vol. 36, no. 1, pp. 158–183, 2006.

[44] A. E. Alaoui and M. W. Mahoney, "Fast randomized kernel methods with statistical guarantees," *arXiv preprint arXiv:1411.0306*, 2014.

[45] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proceedings of the National Academy of Sciences*, vol. 106, no. 3, pp. 697–702, 2009.

[46] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2002.

[47] T. Zhang, "Learning bounds for kernel regression using effective data dimensionality," *Neural Computation*, vol. 17, no. 9, pp. 2077–2098, 2005.

[48] R. Bhatia, *Matrix analysis*. Springer Science & Business Media, 2013, vol. 169.

[49] M. B. Cohen, Y. T. Lee, C. Musco, C. Musco, R. Peng, and A. Sidford, "Uniform sampling for matrix approximation," in *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, 2015, pp. 181–190.

[50] J. Bucaro, Z. Waters, B. Houston, H. Simpson, A. Sarkissian, S. Dey, and T. Yoder, "Acoustic identification of buried underwater exploded ordnance using a numerically trained classifier," *Journal of the Acoustical Society of America*, vol. 132, pp. 3614–3617, December 2012.

[51] S. Dey, A. Sarkissian, H. Simpson, B. Houston, F. Bulat, L. Kraus, M. Saniga, and J. Bucaro, "Structural-acoustic modeling for three-dimensional freefield and littoral environments with verification and validation," *Journal of the Acoustical Society of America*, vol. 129, pp. 2979–2990, May 2011.

[52] D. P. Williams, "On the use of tiny convolutional neural networks for human-expert-level classification performance in sonar imagery," *IEEE Journal of Oceanic Engineering*, vol. 46, no. 1, pp. 236–260, 2020.

[53] F. Hallgren and P. Northrop, "Incremental kernel PCA and the Nyström method," *arXiv preprint arXiv:1802.00043*, 2018.

[54] N. J. Stor, I. Slapnicar, and J. L. Barlow, "Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications," *Linear Algebra and its Applications*, vol. 464, pp. 62–89, 2015.

[55] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[56] R. D. DeGroat and R. A. Roberts, "Efficient, numerically stabilized rank-one eigenstructure updating (signal processing)," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 301–316, 1990.

[57] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.

[58] G. H. Golub, "Some modified matrix eigenvalue problems," *Siam Review*, vol. 15, no. 2, pp. 318–334, 1973.

[59] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013, vol. 3.

[60] B. Scholkopf, S. Mika, C. J. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1000–1017, 1999.

[61] K. Tsuda, "Support vector classifier with asymmetric kernel functions," in *in European Symposium on Artificial Neural Networks (ESANN*. Citeseer, 1999.

[62] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine learning research*, vol. 5, no. Jan, pp. 27–72, 2004.

[63] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[64] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural computation*, vol. 12, no. 10, pp. 2385–2404, 2000.

[65] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*. IEEE, 1999, pp. 41–48.

[66] V. Roth and V. Steinhage, "Nonlinear discriminant analysis using kernel functions," in *NIPS*, vol. 12, 1999, pp. 568–574.

[67] S.-I. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[68] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. Pearson Education Upper Saddle River, 2009.

[69] A. Pezeshki, M. R. Azimi-Sadjadi, and C. Robbiano, "A multiple kernel machine with incremental learning using sparse representation," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[70] C. S. Ong, A. Smola, R. Williamson *et al.*, "Learning the kernel with hyperkernels," 2005.

[71] A. El Helou, "Sensor HAR recognition app," *MathWorks File Exchange*, pp. https://www.mathworks.com/matlabcentral/fileexchange/54 138–sensor–har–recognition–app.

[72] C. Robbiano, E. K. Chong, and M. R. Azimi-Sadjadi, "Information-theoretic approach to navigation for efficient detection and classification of underwater objects," *arXiv preprint arXiv:2007.05072*, 2020.

[73] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *Information Theory, IEEE Transactions on*, vol. 49, no. 12, pp. 3320–3325, 2003.

[74] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley, 1991.

[75] J. M. Cioffi and T. Kailath, "Fast, recursive-least-squares transversal filters for adaptive filtering," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 32, pp. 304–337, 1984.

[76] S. Alexander, *Adaptive Signal Processing: Theory and Applications*. Springer-Verlag, 1986.

APPENDIX A

K-SVD DICTIONARY LEARNING ALGORITHM

A.1 Introduction

An overcomplete dictionary that leads to sparse representations [19, 33, 73] can either be chosen as a pre-specified set of functions or designed by adapting its content to fit a given set of signal examples. In this appendix, we overview the K-SVD dictionary learning method [19], the dictionary is designed specifically for a signal type in order to meet an imposed sparsity model. The goal is to find the dictionary that yields signal-dependent sparse representations for a given training set. Such dictionaries have the potential to outperform [24] commonly used predetermined dictionaries for the problem of sonar classification owing to the fact that for different object classes specifically trained dictionaries are utilized to perform classification. The K-SVD 's strength lies in its adherence to strict minimization of squared-error reconstruction w.r.t. a specified sparse signal model. The outline of this chapter is as follows. In Section A.2, an overview of the K-SVD signal subspace construction method is given. We begin by discussing the reconstruction problem considered by K-SVD. Then we introduce a method for performing sparse coding for the sparse coding phase of the K-SVD learning. Lastly we discuss how to perform the dictionary update phase of the K-SVD dictionary learning.

A.2 K-SVD Dictionary Learning Review

The purpose of K-SVD is to create an optimal signal-dependent dictionary that reduces the dimension of a high dimensional signal vector by representing it as a sparse linear combination of relatively few atoms. More specifically, K-SVD aims to solve a constrained minimization problem to reduce the reconstruction error in a set of training vectors. Let

$\mathbf{Z}_m \in \mathbb{R}^{N \times Q}$ be a matrix consisting of class m training data vectors with $\mathbf{z}_q$ for $q \in [1 \cdots Q]$ as its columns, $\mathbf{H}_m \in \mathbb{R}^{N \times K}$ be the dictionary matrix to be found, and $\mathbf{\Theta}_m \in \mathbb{R}^{K \times Q}$ be the sparse representation of $\mathbf{Z}_m$ in terms of dictionary atoms in $\mathbf{H}_m$. Note it is desired that the number of non-zero elements of each $\boldsymbol{\theta}_q$ be substantially less than N as the dimension should be reduced in this process. The constrained optimization problem [19] is given by,

$$\min_{\mathbf{H}_m, \mathbf{\Theta}_m} \{\|\mathbf{Z}_m - \mathbf{H}_m \mathbf{\Theta}_m\|_F^2\} \qquad \text{subject to }, \quad \|\boldsymbol{\theta}_q\|_0 \leq \tau, \quad \forall q \tag{A.1}$$

where $||\cdot||_F^2$ is the Frobenius norm of a matrix [74], and $||\cdot||_0$ is the ℓ_0 norm which counts the non-zero elements of a vector. Put simply, this optimization seeks $\mathbf{H}_m$ and $\mathbf{\Theta}_m$ which simultaneously (1) minimizes squared error of reconstructing training samples $\mathbf{Z}_m$, and (2) enforces reconstructions that depend on at most $\tau \ll N$ basis elements.

During the training, the K-SVD algorithm is composed of two-phases. First, a sparse representation phase is applied where for each $\mathbf{z}_q$ the corresponding $\boldsymbol{\theta}_q$ is computed based on a given subspace matrix $\mathbf{H}_m$. These sparse codes are found using a pursuit method such as Basis Pursuit (BP) [41] or Orthogonal Matching Pursuit (OMP) [24]. Second, a dictionary update phase where each column $\mathbf{h}_k$ of matrix $\mathbf{H}_m$ is updated one at a time based on minimizing the reconstruction error using the Singular Value Decomposition (SVD) [74] of a restricted error matrix $\mathbf{E}_k^R$, a matrix representing the reconstruction error incurred when leaving out the k^{th} atom $\mathbf{h}_k$. These two coupled phases are repeated until convergence through monotonic MSE reduction [19]. These steps are briefly reviewed in the following subsections.

A.2.1 Sparse Coding Phase Using a Fast OMP Method

In the sparse representation phase, for each $\mathbf{z}_q$ in the training data matrix $\mathbf{Z}_m$, a corresponding $\boldsymbol{\theta}_q$ is computed based on a given $\mathbf{H}_m$ using a pursuit method, namely the OMP. An implementation of OMP that does not require matrix inversion for the sparse coding is briefly reviewed here [24].

To begin, let $\mathbf{H}_m = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_K] \in \mathbb{R}^{N \times K}$ be an overcomplete dictionary matrix representing the signal subspace for class m samples. Now, given an training vector $\mathbf{z}_k \in \mathbb{R}^N$, we would like to find its sparse representation $\boldsymbol{\theta}_k \in \mathbb{R}^K$, $K \ll N$ iteratively using the smallest possible number of basis vectors $\mathbf{h}_j$'s such that the norm of the reconstruction error is less than a pre-selected tolerance ϵ, i.e. $||\mathbf{e}_k|| \leq \epsilon$, where $\mathbf{e}_k = \mathbf{z}_k - \mathbf{H}_m\boldsymbol{\theta}_k$.

For a given observation vector $\mathbf{z}_k$, each iteration of standard OMP involves finding the inner product of the current residual error vector, calculated at iteration $t-1$, with each of the remaining (unselected) dictionary atoms and selecting that with the largest inner product. More specifically, at iteration t if the current residual error vector is $\mathbf{r}_{t-1}$, the next dictionary atom is chosen using,

$$k_t = \text{argmax}_j \, |\mathbf{r}_{t-1}^T \mathbf{h}_j| \tag{A.2}$$

Then, the augmented dictionary matrix (column-wise) and index set respectively become $\mathbf{H}_{m,t} = [\mathbf{H}_{m,t-1}\mathbf{h}_{k_t}]$ and $S_t = S_{t-1} \bigcup k_t$ with initial values $\mathbf{H}_{m_0}$ being a matrix with randomly selected training samples, and $S_0 = \emptyset$. Using this matrix $\hat{\boldsymbol{\theta}}_k(t)$ is estimated by the minimization,

$$\hat{\boldsymbol{\theta}}_k(t) = \text{argmin}_{\boldsymbol{\theta}_k} \, ||\mathbf{z}_k - \mathbf{H}_{m,t}\boldsymbol{\theta}_k|| \tag{A.3}$$

which leads to the least squares (LS) solution for $\hat{\boldsymbol{\theta}}_k(t)$,

$$\hat{\boldsymbol{\theta}}_k(t) = (\mathbf{H}_{m,t}^T\mathbf{H}_{m,t})^{-1}\mathbf{H}_{m,t}^T\mathbf{z}_k = \mathbf{Q}_{\mathbf{H}_{m,t}}\mathbf{z}_k \tag{A.4}$$

where $\mathbf{Q}_{\mathbf{H}_{m,t}} = (\mathbf{H}_{m,t}^T\mathbf{H}_{m,t})^{-1}\mathbf{H}_{m,t}^T$ represents the LS filter based upon the augmented matrix $\mathbf{H}_{m,t}$. The new residual error term then becomes

$$\mathbf{r}_t = \mathbf{z}_k - \mathbf{H}_{m,t}\hat{\boldsymbol{\theta}}_k(t) = \mathbf{P}_{\mathbf{H}_{m,t}}^{\perp}\mathbf{z}_k \tag{A.5}$$

where $\mathbf{P}_{\mathbf{H}_{m,t}}^{\perp} = \mathbf{I} - \mathbf{P}_{\mathbf{H}_{m,t}}$and $\mathbf{P}_{\mathbf{H}_{m,t}} = \mathbf{H}_{m,t}(\mathbf{H}_{m,t}^T\mathbf{H}_{m,t})^{-1}\mathbf{H}_{m,t}^T$ is the projection matrix onto subspace $\langle\mathbf{H}_{m,t}\rangle$ spanned by columns of $\mathbf{H}_{m,t}$. The process is repeated until the error tolerance is met.

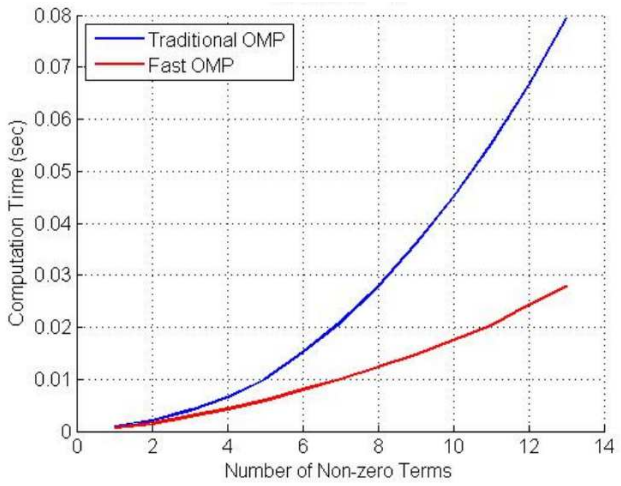


Figure A.1: Timing Analysis of Pursuit Methods

As can be seen in (A.4) and (A.5), the computational cost of the original OMP algorithm grows very quickly as the number of chosen atoms increases. This is caused by the rapid increase in cost of matrix inversion as dimensionality increases, the relationship is demonstrated in Figure A.1. Additionally, to relate this sparse coding to iterative K-SVD an interface between these two processes is needed. To develop a fast OMP algorithm that avoids matrix inversion operation, the authors in [24] used the orthogonal projection updating [75, 76] for $\mathbf{P}_{\mathbf{H}_{m,t}}$ and $\mathbf{Q}_{\mathbf{H}_{m,t}}$ to get

$$\mathbf{P}^{\perp}_{\mathbf{H}_{m,t}} = \mathbf{P}^{\perp}_{\mathbf{H}_{m,t-1}} - \mathbf{P}_{\tilde{\mathbf{h}}_{k_t}} \tag{A.6}$$

and

$$\mathbf{Q}_{\mathbf{H}_{m,t}} = \begin{bmatrix} \mathbf{Q}_{\mathbf{H}_{m,t-1}} \\ 0 \end{bmatrix} + \begin{bmatrix} -\mathbf{b}_{t-1} \\ 1 \end{bmatrix} \mathbf{q}_t^T \tag{A.7}$$

where $\mathbf{P}_{\tilde{\mathbf{h}}_{k_t}} = \frac{\tilde{\mathbf{h}}_{k_t}\tilde{\mathbf{h}}_{k_t}^T}{||\tilde{\mathbf{h}}_{k_t}||^2}$ is the projection matrix for $\tilde{\mathbf{h}}_{k_t} = \mathbf{P}^{\perp}_{\mathbf{H}_{m,t-1}}\mathbf{h}_{k_t}$ which is the projection of $\mathbf{h}_{k_t}$ onto the orthogonal subspace of $\langle \mathbf{H}_{m,t-1} \rangle$ (or innovation); $\mathbf{b}_{t-1} = \mathbf{Q}_{\mathbf{H}_{m,t-1}}\mathbf{h}_{k_t}$, the filtered version of $\mathbf{h}_{k_t}$ based upon LS filter $\mathbf{Q}_{\mathbf{H}_{m,t-1}}$; and $\mathbf{q}_t^T = \frac{\tilde{\mathbf{h}}_{k_t}^T}{||\tilde{\mathbf{h}}_{k_t}||^2}$ is the LS filter operator using $\tilde{\mathbf{h}}_{k_t}$. Premultiplying (A.6) and (A.7) by $\mathbf{z}_k$ yields recursive update equations for $\mathbf{r}_t$ and $\hat{\boldsymbol{\theta}}_k(t)$, respectively, i.e.

$$\mathbf{r}_t = \mathbf{r}_{t-1} - \alpha_t \tilde{\mathbf{h}}_{k_t} \tag{A.8}$$

and

$$\hat{\boldsymbol{\theta}}_k(t) = \begin{bmatrix} \hat{\boldsymbol{\theta}}_k(t-1) \\ 0 \end{bmatrix} + \alpha_t \begin{bmatrix} -\mathbf{b}_{t-1} \\ 1 \end{bmatrix} \tag{A.9}$$

where $\alpha_t = \mathbf{q}_t^T \mathbf{z}_k = \frac{\mathbf{h}_{k_t}^T \mathbf{r}_{t-1}}{||\mathbf{h}_{k_t}||^2}$ i.e. filtered version of $\mathbf{z}_k$ based upon LS filter operator $\mathbf{q}_t^T$. Thus, the adjustment term in (A.9) corresponding to the previous coefficients is equal to the products of two filtered outputs, namely $\mathbf{b}_{t-1}$ and α_t; whereas the coefficient associated with the newly added atom is α_t. These equations allow for "time-order" updates after adding a new dictionary atom. Note that in this algorithm, computing the projection matrix $\mathbf{P}_{\mathbf{H}_{m,t}}$ and LS filter operator $\mathbf{Q}_{\mathbf{H}_{m,t}}$ are completely avoided and hence no matrix inversion is required. To calculate the new part of $\mathbf{h}_{k_t}$ i.e. $\tilde{\mathbf{h}}_{k_t} = \mathbf{P}^{\perp}_{\mathbf{H}_{m,t-1}} \mathbf{h}_{k_t}$ we only need the filtered output $\mathbf{b}_{t-1}$ since, $\tilde{\mathbf{h}}_{k_t} = \mathbf{h}_{k_t} - \mathbf{H}_{m,t-1}\mathbf{b}_{t-1}$.

A.2.2 Dictionary Update Phase

The dictionary update phase involves writing the cost function in (A.1) in terms of columns of the dictionary matrix, $\mathbf{h}_j$'s and rows of matrix $\boldsymbol{\Theta}_m$, $\boldsymbol{\theta}_T^j$'s. More specifically,

$$\left|\left|\mathbf{Z}_m - \sum_{j=1}^{K} \mathbf{h}_j \boldsymbol{\theta}_T^j \right|\right|_F^2 \tag{A.10}$$

Then, separating the effects of the k^{th} dictionary atom ($k \in [1, K]$) from the other terms yields,

$$\left|\left|\left(\mathbf{Z}_m - \sum_{j \neq k} \mathbf{h}_j \boldsymbol{\theta}_T^j\right) - \mathbf{h}_k \boldsymbol{\theta}_T^k \right|\right|_F^2 = ||\mathbf{E}_k - \mathbf{h}_k \boldsymbol{\theta}_T^k||_F^2 \tag{A.11}$$

where $\mathbf{E}_k$ is the error matrix that represents the reconstruction error when neglecting the k^{th} atom. Now, to consider the sparsity of $\boldsymbol{\theta}_T^k$, we define a subset of indices $\omega_k = \{i, |x_T^k(i) \neq 0\}$ which correspond to the training samples in $\mathbf{Z}_m$ that use column dictionary atom $\mathbf{h}_k$ for which $x_T^k(i) \neq 0$. Consequently, A.11 is reduced to

$$||\mathbf{E}_k \Omega_k - \mathbf{h}_k \boldsymbol{\theta}_T^k \Omega_k||_F^2 = ||\mathbf{E}_k^R - \mathbf{h}_k \boldsymbol{\theta}_R^k||_F^2 \tag{A.12}$$

where Ω_k is a $Q \times P$ matrix whose entries $(\omega_k(i), i)$ are ones and the rest are zeros. Note that $P = |\omega_k| \leq K$ where $|\cdot|$ represents cardinality of set ω_k. When post-multiplying a

vector by Ω_k, all the zero elements are discarded and the dimension of $\boldsymbol{\theta}_T^k$ is reduced to $\boldsymbol{\theta}_R^k = \boldsymbol{\theta}_T^k \Omega_k \in \mathbb{R}^P$ and also $\mathbf{E}_k^R = \mathbf{E}_k \Omega_k \in \mathbb{R}^{N \times P}$ is the restricted error matrix. Note that this is not needed if the proposed fast OMP method is used. When decomposing $\mathbf{E}_k^R = \mathbf{U}\Delta\mathbf{V}^T$ using SVD, the solution for the updated dictionary atom $\hat{\mathbf{h}}_k = \mathbf{u}_1$, or the first column of $\mathbf{U}$. Now, since $\hat{\mathbf{h}}_k$ has changed, $\boldsymbol{\theta}_R^k$ must also be updated. This restricted vector is updated using $\hat{\boldsymbol{\theta}}_R^k = \mathbf{v}_1 \Delta_{1,1}$ where $\mathbf{v}_1$ is the first row of $\mathbf{V}^T$ and $\Delta_{1,1}$ is the first entry in the Δ matrix. After updating the k^{th} dictionary atom, the same procedure is followed for the $(k+1)^{th}$ atom until all atoms have been individually updated. Then, using this updated dictionary the sparse matrix $\boldsymbol{\Theta}_m$ can be recomputed. The dictionary update and sparse coding steps are repeated until the stopping condition is met i.e. either maximum number of iterations or a reconstruction error below some threshold. The final K-SVD procedure is described below in Algorithm A.1

Table A.1: K-SVD Algorithm

K-SVD Optimal Dictionary Construction Algorithm:

a) Initialization: Set the dictionary matrix $\mathbf{H}_{m,0} \in \mathbb{R}^{N \times K}$ with K randomly selected l_2 normalized columns of $\mathbf{Z}_m$. Set $t = 1$. Repeat following steps until a stopping rule is met.

b) Sparse Coding Stage: Generate $\boldsymbol{\Theta}_m$ by computing the sparse representation $\boldsymbol{\theta}_{q,t}$ for each $\mathbf{z}_q$ based on $\mathbf{H}_{m,t-1}$ using the Fast-OMP method [24].

c) Dictionary Update Stage: Each column $\mathbf{h}_{k,t-1},\ k \in [1, ..., K]$, in $\mathbf{H}_{m,t-1}$ is updated separately by:

1. Compute $k-$exclusive error matrix $\mathbf{E}_k = \mathbf{Z}_m - \sum_{j \neq k} \mathbf{h}_j \boldsymbol{\theta}_T^j$, where $\boldsymbol{\theta}_T^j$ is the j^{th} row of $\boldsymbol{\Theta}_m$.
2. Define column indices of training data $\mathbf{Z}_m$ that use the k^{th} atom in their reconstruction via $\mathbf{H}_{m,t-1}$: $\omega_k = \{i \mid x_T^k(i) \neq 0\}$.
3. Compute $\mathbf{E}_k^R$ and $\boldsymbol{\theta}_R^k$, the restricted error matrix and coefficient vector respectively, by selecting only columns of $\mathbf{E}_k$ corresponding to ω_k indices and likewise for entries of $\boldsymbol{\theta}_T^k$ (i.e. discard zero entries in the row vector).
4. Apply SVD: $\mathbf{E}_k^R = \mathbf{U}\Delta\mathbf{V}^T$. The updated dictionary column $\hat{\mathbf{h}}_{k,t} = \mathbf{u}_1$, the first column of $\mathbf{U}$ and the updated coefficient vector $\hat{\boldsymbol{\theta}}_R^k = \mathbf{v}_1 \Delta_{1,1}$, where $\Delta_{1,1}$ is the first and largest singular value in the SVD of $\mathbf{E}_k^R$, and $\mathbf{v}_1$ is the first row of $\mathbf{V}^T$.

Set $t = t + 1$ and repeat until repeat b) and c) until stopping criterion is satisfied.

APPENDIX B

INCREMENTAL K-SVD (IK-SVD) METHOD

When a baseline dictionary is no longer adequate to represent the data in new environments, the dictionary matrix, $\mathbf{H}$, need to be adapted to maintain the classification performance. To accomplish this dictionary updating, a common choice is the incremental K-SVD algorithm in [17], which can introduce new atoms into the existing dictionary using the collected in-situ data set, $\tilde{\mathbf{Z}} = [\mathbf{z}_{Q+1} \cdots \mathbf{z}_{Q+\tilde{Q}}]$ from the new environment. The objective function for this incremental learning model is given by,

$$\min_{\bar{\mathbf{H}},\tilde{\mathbf{\Theta}}} ||\tilde{\mathbf{Z}} - \bar{\mathbf{H}}\tilde{\mathbf{\Theta}}||_F^2 \qquad \text{s.t.} \qquad \|\tilde{\boldsymbol{\theta}}_q\|_0 \leq \tau \quad \forall q \in [Q+1, Q+\tilde{Q}] \tag{B.1}$$

where $\tilde{\mathbf{\Theta}}$ is the associated sparse representation of $\tilde{\mathbf{Z}}$, and $\bar{\mathbf{H}} = [\mathbf{H}\ \tilde{\mathbf{H}}]$ represents the new dictionary matrix consisting of the old dictionary matrix $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_K]$ and the incremental one $\tilde{\mathbf{H}} = [\mathbf{h}_{K+1}, \cdots, \mathbf{h}_{K+K_1}]$ with $\mathbf{h}_{K+i}, i = 1, \cdots, K_1$ being the newly added atoms. In this algorithm, the old dictionary $\mathbf{H}$ remains unchanged while only the new atoms are updated by the K-SVD algorithm. Hence, the error matrix for $\mathbf{h}_i,\ i \in [K+1, K+K_1]$ is,

$$\tilde{\mathbf{E}}_i = \tilde{\mathbf{Z}} - \sum_{j=1}^{K} \mathbf{h}_j \hat{\boldsymbol{\theta}}_T^j - \sum_{j=K+1, j\neq i}^{K+K_1} \mathbf{h}_j \hat{\boldsymbol{\theta}}_T^j. \tag{B.2}$$

This error matrix is used in the original K-SVD algorithm of Appendix A to find the new dictionary atoms.

In [17], an entropy-based criterion is employed to select the initial values of the new atoms by first sparsely coding $\tilde{\mathbf{Z}}$ using the old dictionary matrix $\mathbf{H}$. Then, the entropy of each sparse coefficient vector in $\tilde{\mathbf{\Theta}}$ is calculated, and the samples with the largest entropy (maximal disagreement) are used to initialize the new atoms. These samples correspond to those that are least sparse and cannot be accurately represented by the old dictionary matrix.

APPENDIX C

MATCHED SUBSPACE CLASSIFICATION AND THE MODIFIED MSC

C.0.1 Modified MSC and SRC Framework

Let us consider a general $M-$ary classification problem in which the observations can come from $m = 1, \ldots, M$ possible classes. The MSC [6] assumes a signal model of the form,

$$\mathbf{Z} = \mathbf{H}_m \mathbf{\Theta}_m + \mathbf{N} \qquad m \in [1, M] \tag{C.1}$$

where $\mathbf{Z}$ is the data matrix containing observation vectors $\mathbf{z}_i$ as its column, $\mathbf{H}_m$ is the dictionary matrix whose columns are the basis vectors that span the subspace associated with the m^{th} class, $\mathbf{\Theta}_m$ is an unknown matrix with the columns being the parameter vector associated with data vectors $\mathbf{z}_i$, and $\mathbf{N}$ denotes an additive zero-mean noise matrix.

The decision-making in MSC is carried out by determining the class that satisfies

$$m^* = \text{argmin}_{m \in [1,M]} \{ ||\mathbf{Z} - \mathbf{H}_m \hat{\mathbf{\Theta}}_m||_F^2 \} \tag{C.2}$$

where, traditionally, $\hat{\mathbf{\Theta}}_m$ is the least squares (LS) estimate of $\mathbf{\Theta}_m$ for signal subspace matrix $\mathbf{H}_m$ and observation $\mathbf{Z}$ [6]. In the *modified* MSC, $\hat{\mathbf{\Theta}}_m$ is generated using the Orthogonal Matching Pursuit (OMP) [2,35] algorithm when dictionary matrix $\mathbf{H}_m$ is used to approximate observation $\mathbf{Z}$. That is, the correct class m^* is the one which makes the magnitude of the $\tau-$sparse reconstruction error in C.2 the smallest. Training of the MSC amounts to constructing class-dependent dictionary matrices $\mathbf{H}_m, m = 1, \ldots, M$ from representative training data sets in each class m which can be done using the K-SVD algorithm [19].

APPENDIX D

PRODUCTS OF MERCER KERNELS RESULT IN MERCER KERNELS

Given two Mercer kernels $k_0(\mathbf{x}, \mathbf{y})$ and $k_1(\mathbf{x}, \mathbf{y})$, it is relatively straightforward to show that their product is also a Mercer kernel via Mercer's Theorem.

Proof:

Assume $k_0(\mathbf{x}, \mathbf{y})$ and $k_1(\mathbf{x}, \mathbf{y})$ are Mercer Kernels. Then, by Mercer's theorem they must admit an inner product representation. Let $\mathbf{p}$ denote the feature vector for k_0 and $\mathbf{q}$ denote the feature vector for k_1.

$$k_0(\mathbf{x}, \mathbf{y}) = \mathbf{p}(\mathbf{x})^\top \mathbf{p}(\mathbf{y}), \quad \mathbf{p}(\mathbf{z}) = [p_1(\mathbf{z}), p_2(\mathbf{z}), ..., p_M(\mathbf{z})]^\top$$
$$k_1(\mathbf{x}, \mathbf{y}) = \mathbf{q}(\mathbf{x})^\top \mathbf{q}(\mathbf{y}), \quad \mathbf{q}(\mathbf{z}) = [q_1(\mathbf{z}), q_2(\mathbf{z}), ..., q_N(\mathbf{z})]^\top$$

So $\mathbf{p}$ and $\mathbf{q}$ are mappings that produce M and N dimensional vectors from input space $\mathbb{R}^d$ where $\mathbf{x}$ and $\mathbf{y}$ live.

Now we will construct a feature vector by looking carefully at the product of k_0 and k_1.

$$\begin{aligned}
k_p(\mathbf{x}, \mathbf{y}) &= k_0(\mathbf{x}, \mathbf{y}) k_1(\mathbf{x}, \mathbf{y}) \\
&= (\mathbf{p}(\mathbf{x})^\top \mathbf{p}(\mathbf{y}))(\mathbf{q}(\mathbf{x})^\top \mathbf{q}(\mathbf{y})) \\
&= \left(\sum_{m=1}^{M} p_m(\mathbf{x}) p_m(\mathbf{y}) \right) \left(\sum_{n=1}^{N} q_n(\mathbf{x}) q_n(\mathbf{y}) \right) \\
&= \sum_{m=1}^{M} \sum_{n=1}^{N} [p_m(\mathbf{x}) q_n(\mathbf{x})][p_m(\mathbf{y}) \mathbf{q}_n(\mathbf{y})] \\
&= \sum_{m=1}^{M} \sum_{n=1}^{N} r_{mn}(\mathbf{x}) r_{mn}(\mathbf{y}) \\
&= \mathbf{r}^\top(\mathbf{x}) \mathbf{r}(\mathbf{y}) \qquad \text{(D.1)}
\end{aligned}$$

Where $\mathbf{r}(\mathbf{z})$ is an MN-dimensional vector s.t. $r_{mn}(\mathbf{z}) = p_m(\mathbf{z}) q_n(\mathbf{z})$. Since we can write $k_p(\mathbf{x}, \mathbf{y})$ as an inner product using the feature vector $\mathbf{r}$, that means that properties of the inner product transfer to the kernel k_p. The inner product is by definition symmetric and positive semi-definite, hence so is k_p, hence it is also a Mercer kernel.