DISSERTATION

SCHUBERT VARIETY OF BEST FIT WITH APPLICATIONS

AND

ACROSS DOMAINS SPARSE FEATURE EXTRACTION

Submitted by

Karim Karimov

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2024

Doctoral Committee:

Advisor: Michael Kirby

Chris Peterson
Charles Anderson
Henry Adams

ABSTRACT

SCHUBERT VARIETY OF BEST FIT WITH APPLICATIONS
AND
ACROSS DOMAINS SPARSE FEATURE EXTRACTION

This dissertation presents two novel approaches in applied mathematics for data analysis and feature selection, addressing challenges in both geometric data representation and multi-domain biological data interpretation. The first part introduces the Schubert Variety of Best Fit (SVBF) as a new geometric framework for analyzing sets of datasets. Leveraging the structure of Grassmann manifolds and Schubert varieties, we develop the SVBF-Node, a computational unit for solving related optimization problems. We demonstrate the efficacy of this approach through three classification algorithms and a new clustering method, SVBF-LBG. These techniques are evaluated on various datasets, including synthetic data, image sets, video sequences, and hyperspectral remote sensing data, showing improved performance over existing similar methods, particularly for complex, high-dimensional data. The second part proposes a multi-domain, multi-task (MDMT) architecture for feature selection in biological data. This method integrates multi-domain learning with masked feature selection, specifically applied to gene expression data from multiple tissues. We demonstrate its ability to identify novel biomarkers in host immune responses to infection, which are not detectable through single-domain analyses. The approach is validated using bulk RNA sequences from different tissues, revealing its potential to uncover cross-domain biological insights. Both contributions offer interpretable, mathematically grounded approaches to data analysis, providing new tools for researchers in applied mathematics, machine learning, and bioinformatics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This doctoral thesis in applied mathematics explores two distinct yet complementary areas of research: the development of novel geometric methods for analyzing sets of datasets and the creation of a multi-domain, multi-task approach for feature selection in biological data. Through these two main focus areas, this work contributes to geometric data analysis and multi-domain learning, providing new tools and methodologies for tackling complex data analysis problems in applied mathematics.

1.1 Schubert Variety of Best Fit

The first principal component of this work introduces and develops the concept of Schubert Variety of Best Fit (SVBF) as a powerful tool for characterizing and analyzing sets of datasets. This innovative approach leverages the rich geometric structure of Grassmann manifolds and Schubert varieties to develop new data representation, classification, and clustering methods. Recently, a growing interest has been in exploiting geometric frameworks to analyze big data sets. The Stiefel, Grassmann, and Flag manifolds have shown robust characterizations for highly variable data in various applications, including subspace discriminant analysis, illumination variations, face recognition, motion recognition, and image set analysis. These methods share the common feature of encoding data sets by orthonormal matrices but differ in how distances between representatives are computed. This dissertation proposes a methodology based on an interpretable mathematical framework - the geometric setting of the Schubert Variety. The goal is to begin with a mathematically motivated, explainable approach and then optimize its performance through numerical algorithms. This approach aims to lead to results of comparable accuracy to traditional ML/AI toolkits, with the added advantage of explainability and trustworthiness. The SVBF approach is motivated by the idea that a Schubert variety is to a Flag or Grassmann manifold what a subspace

is to a vector space. It provides a novel way to represent and analyze sets of linear spaces, offering a more nuanced and geometrically informed approach to data analysis.

Chapter 2 lays the theoretical foundations of the SVBF approach. It begins by introducing the mathematical background necessary for understanding Grassmann manifolds and Schubert varieties. The chapter then formulates two key optimization problems that form the basis of the SVBF method. A significant contribution of this chapter is the development of the SVBF-Node, a novel computational unit designed to solve these optimization problems efficiently. The chapter concludes with a detailed analysis of the SVBF-Node's computational complexity and performance characteristics.

Chapter 3 explores the application of SVBF to classification tasks. It presents three distinct algorithms that utilize the SVBF framework in different ways. The first algorithm uses a single SVBF solution for classification, while the second employs multiple SVBFs to model different classes. The third algorithm introduces an innovative approach that uses auxiliary features derived from the SVBF for classification. These algorithms are rigorously tested and compared using various datasets, providing insights into their relative strengths and potential applications.

Chapter 4 extends the SVBF framework to address clustering problems. This chapter introduces the SVBF-LBG method, which combines the geometric insights of SVBF with the principles of vector quantization. The effectiveness of this new clustering approach is demonstrated through extensive experiments on diverse datasets, including synthetic data, image datasets, video action sequences, and hyperspectral remote sensing data. The results are compared with state-of-the-art clustering methods, highlighting the advantages of the SVBF-LBG approach.

1.2 A Multi-Domain Multi-Task Approach for Feature Selection

The second primary focus of this dissertation is the development of a novel architecture for multi-domain, multi-task feature selection with a specific application to the analysis of biological data.

Researchers often use microarray or next-generation sequencing techniques in bioinformatics to study gene expression levels. Each sample typically has tens of thousands of features. The large number of features often necessitates feature selection algorithms to improve the performance of machine learning tasks and to determine the processes related to biological mechanisms.

This dissertation addresses multi-domain learning (MDL) that involves classifying data related to the host's immune response to infection. The proposed machine learning approach has the potential to identify novel biomarkers whose signals are too weak to be captured by analyzing domains individually. While various feature selection methods exist, most focus on single-domain feature selection. This work suggests a new multi-domain, multi-task (MDMT) method, which is less common in analyzing biological datasets. The proposed approach leverages prior art in multi-domain learning while adding a masked feature selection approach that identifies biologically relevant aspects of the host's immune response to infection.

Chapter 5 will detail the innovative MDMT architecture, including the methodology, network design, and optimization techniques. It will be applied to the analysis of gene expression data from multiple tissues, showcasing its ability to uncover novel biomarkers amplified by the multi-domain approach. The chapter will conclude with a discussion of the biological implications of the findings and potential future directions for research.

Chapter 2

SVBF-Node

2.1 Introduction

There has been a growing interest in exploiting geometric frameworks for the analysis of big data sets. In particular, the Stiefel, Grassmann, and Flag manifolds have been shown to provide robust characterizations for highly variable data, e.g., subspace discriminant analysis [1], variations in illumination [2], video-based face recognition [3], motion recognition [4], subspace tracking [5], graph embeddings [6] and image set analysis [7]. These subspace methods share the common feature of encoding a set of data by an orthonormal matrix but differ in how distances between the representatives are computed. The Grassmann and Flag manifolds use principal angles for distance computation, while the metric for the Stiefel is the usual trace operation for the Euclidean metric [8]. In all these distinct geometric frameworks, data is associated with points on these manifolds and the distances computed according to the rules of their geometry. They share the common feature that each single point on one of these manifolds arises from an appropriately chosen set of observations. As such, these methods are natural tools for data-driven discovery in sets of data sets [9].

A variety of powerful tools have been developed in Machine Learning and Artificial Intelligence, leading to remarkable applications. A damper on the success of these tools is the fact that the resulting models are frequently difficult to explain, and the predictions may not be trustworthy in high-stakes scenarios, e.g., those related to medical diagnoses, battlefield scenarios or intelligence gathering [10–12]. One reason the success of ML/AI tools is challenging to explain or trust is that these models were designed, first and foremost, to make accurate predictions; attempts to interpret or explain the effectiveness of the models are only an afterthought.

Here, we propose a methodology based on an interpretable mathematical framework, i.e., the geometric setting of the Schubert Variety, as the starting point and explore variations on the theme

to determine optimal architectures for predictive modeling. This research aims to begin with a mathematically motivated explainable approach and then optimize its performance through numerical algorithms. Optimistically, this approach will lead to results of comparable accuracy to the traditional ML/AI toolkit but with the advantage of explainability and trustworthiness. To this end, we propose an approach for characterizing sets of linear spaces, i.e., fitting/approximating subspaces with one or more representative spaces. Additionally, we demonstrate how the mathematical framework and resulting algorithms can be integrated into current tools, including deep-feedforward neural networks.

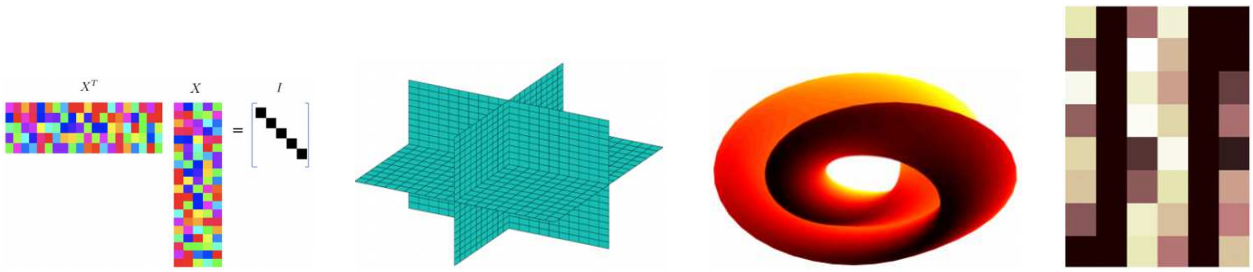
The initial development of ML/AI is focused more on thinking machines than interpretability. Human intelligence and the associated architecture of the human brain have been a driving force in biomimetic approaches. For example, the McCullough-Pitts node [13] and its associated weights were proposed as a mathematical model of the cell and its associated neurons, respectively. The first transfer function at a computational node was a simple step function replicating the firing or quiescence of a neuron. Impressively, arrays of such networks were shown to be able to serve as models of associative memory and even to recall patterns that were partially occluded [14]. Hebbian learning [15] was proposed as a model for memory and convincingly analyzed as a dynamical system where the patterns were stored as fixed points [14], an early appearance of the application of mathematical analysis for the explainability of artificial neural networks.

Geometric ideas emerged with Rosenblatt's simple perceptron [16], still loosely based on neurons firing, where the dot product operation between a pattern and a weight vector gave rise to classification via the interpretation of the models as splitting a space into two half-spaces. The multilayer perceptron extended these ideas in natural ways, however, at the expense of geometric interpretability. Nonlinear data reduction was made possible by autoencoder neural networks [17, 18]. The encoder-decoder architecture has been widely exploited by Variational Autoencoders [19], Centroid-Encoders [20] and Transformers [21]. While these developments provide powerful tools, they widely lack mathematical underpinnings that provide insight into their utility.

In this research, we illustrate how mathematical theory and geometric frameworks can be used as a design philosophy in the construction of novel interpretable neural network architectures. This general idea can be found in previous work, e.g., geometric, or topological nodes such as circular [22], or spherical computational units [23]. Whitney's theorem has also been invoked to provide a basis to understand the power of autoencoders from a geometric perspective [24] and to provide insights into novel architectures [25] and dimension estimation [26–28]. These ideas are central to the computation of homeomorphisms between *data sets* residing in spaces of differing dimensions. Using these ideas as motivation, we can envision extending the concept of an abstract node more generally to algebraic varieties related to Generalized Principal Component Analysis [29], Klein bottles, Grassmannians, and Schubert Varieties; see Figure 2.1.

We focus our attention on the mathematical framework of the Schubert Variety described in what follows. We are motivated by the idea that a Schubert variety is to a Flag or Grassmann manifold what a subspace is to a vector space. Flag manifolds and their special case, the Grassmann manifolds, are examples of homogeneous manifolds particularly relevant to and amenable to subspace methods in Data Science. They have been observed to be particularly robust to data collected under variations in pattern state (and indeed exploit structure in such data sets). For example, digital images of an object collected under variations in illumination are known to sweep out a convex cone. If the object is Lambertian, then this cone has been shown to lie close to a low dimensional linear space, which can, in turn, be represented by a point on a Grassmannian [2]. The Flag manifold comes equipped with geometric features capable of representing sets of data where the number of points is larger than the number of dimensions in the ambient space [9]. We note that the Flag mean proposed in [30, 31], and its various extensions, are a special case of the work proposed here.

In this chapter, we propose several optimization problems that are used to produce a geometric object, e.g., a Schubert Variety of Best Fit (SVBF), that optimally represents a set of linear subspaces of a fixed vector space $\mathbb{R}^n$. In several applications, the set of linear spaces is obtained from sets of sets of data. The optimization problem is determined by a real-valued objective function



(a) Unit with matrix values X required to satisfy $X^T X = I$

(b) Unit with data residing in a union of subspaces/planes

(c) Klein Bottle - example of unit manifold constraint

(d) A Schubert Variety constraint

Figure 2.1: Various examples of abstract nodes. These *computational units* can be integrated into data fitting problems, e.g., multilayer neural networks, to extract topological or geometric structure. This work focuses on the particular case of the Schubert variety constraint (d).

on a manifold (typically a Grassmann or Flag manifold) whose points parameterize a family of potential Schubert varieties of best fit. Further, we show how this framework can be viewed as a component of a machine learning architecture integrated, e.g., into the broader framework of feed-forward neural networks.

2.2 Background

Consider the set, S, of $n \times n$ invertible matrices whose inverse is equal to its transpose, i.e., $S = \{A \in \mathbb{R}^{n \times n} \mid A^T A = I_n\}$. S contains the identity matrix, I_n, and is closed under matrix inversion and multiplication operations. In other words, if A and B are in S then both A^{-1} and AB are in S. The set S together with the binary operation of matrix multiplication is known as the *orthogonal group* $O(n)$. Alternatively, $O(n)$ is the group of distance-preserving transformations of an n-dimensional Euclidean space that preserve a fixed point. A distinguished subgroup of $O(n)$ is the special orthogonal group $SO(n)$ consisting of elements of S with a determinant equal to one (orientation preserving transformations). From a geometric perspective, the orthogonal group $O(n)$ can be considered a manifold whose points parameterize ordered orthonormal bases of $\mathbb{R}^n$. As a manifold, it consists of two connected components corresponding to the square orthogonal matrices with determinant $+1$ and those with determinant -1. Its dimension as a real manifold is $\binom{n}{2}$. Several interesting manifolds can be built through a "quotient" operation by considering the

action of subgroups of $O(n)$ (resp. $SO(n)$) on $O(n)$ (resp. $SO(n)$) through multiplication. Some particularly relevant examples are the following:

- Grassmann Manifolds $GR(l, n)$
- Oriented Grassmann Manifolds $\widetilde{GR}(l, n)$
- Flag Manifolds $FL(l_1, l_2, ..., l_m; n)$
- (Partially) Oriented Flag Manifolds
- Steifel Manifolds $ST(l, n)$

Grassmann Manifold - The Grassmannian of l-dimensional subspaces of R^n is denoted by $GR(l, n)$. Points on $GR(l, n)$ correspond to l-dimensional subspaces of $\mathbb{R}^n$. It can be built as a coset space $O(n)/O(l) \times O(n-l)$ where $O(l) \times O(n-l)$ denotes $n \times n$ matrices consist of an $l \times l$ orthogonal block and an $n-l \times n-l$ orthogonal block. Through this identification, points on $GR(l, n)$ correspond to equivalences classes of $n \times n$ orthogonal matrices where two such matrices are identified if the span of their first l columns agrees. We can also think of points on $GR(l, n)$ as corresponding to equivalences classes of $n \times l$ orthogonal matrices where two such matrices are identified if they have the same column space. $GR(l, n)$ can be considered a homogeneous space and a differentiable manifold. As a real manifold, the dimension of $GR(l, n)$ is $l(n-l) = dim(O(n)) - dim(O(l)) - dim(O(n-l))$.

Oriented Grassmann Manifold - The oriented Grassmannian of all *oriented* l-dimensional subspaces of $\mathbb{R}^n$ is denoted $\widetilde{GR}(l, n)$. It can be built as a coset space $SO(n)/SO(l) \times SO(n-l)$. There is a natural 2:1 covering map from $\widetilde{GR}(l, n)$ to $GR(l, n)$. A special case is $\widetilde{GR}(1, n)$ whose cosets correspond to points on the $n-1$ dimensional sphere S^{n-1} in R^n. $GR(1, n)$ corresponds to the real projective space RP^{n-1}. RP^{n-1} can also be built from S^{n-1} by identifying antipodal points on the sphere. In the map from $\widetilde{GR}(1, n)$ to $GR(1, n)$, a pair of antipodal points on the

sphere get mapped to a single point in projective space. As a real manifold, the dimension of $\widetilde{GR}(l,n)$ is the same as the dimension of $GR(l,n)$.

Flag Manifold - $FL(l_1, l_2, ..., l_m; n)$ = collection of all flags of the form $V_1 \subset V_2 \subset \cdots \subset V_m \subset \mathbb{R}^n$ such that $\dim V_i = l_i$. The Flag manifolds can be built by considering quotients of $O(n)$ by a direct product of smaller orthogonal groups. More precisely by the quotient of $O(n)$ by $O(l_1) \times O(l_2 - l_1) \times O(l_m - l_{m-1}) \times O(n - l_m)$. As a real manifold, the dimension of $FL(l_1, l_2, ..., l_m; n)$ is $dim(O(n)) - dim(O(l_1)) - dim(O(l_2 - l_1)) - dim(O(l_3 - l_2)) - \cdots - dim(O(l_m - l_{m-1})) - dim(O(n - l_m))$.

Oriented Flag Manifold - $FL^\circ(l_1, l_2, ..., l_m; n)$ = collection of all oriented flags of the form $V_1 \subset V_2 \subset \cdots \subset V_m \subset \mathbb{R}^n$ such that $\dim V_i = l_i$. The oriented Flag manifolds can be built by considering quotients of $SO(n)$ by a direct product of smaller special orthogonal groups. There is a natural $2^m : 1$ covering map from $FL^\circ(l_1, l_2, ..., l_m; n)$ to $FL(l_1, l_2, ..., l_m; n)$. As a real manifold, the dimension of $FL^\circ(l_1, l_2, ..., l_m; n)$ is the same as the dimension of $FL(l_1, l_2, ..., l_m; n)$

Partially Oriented Flag Manifold - The partially oriented Flag manifolds can be built by considering quotients of $SO(n)$ by a direct product of a mixture of smaller special orthogonal groups and smaller orthogonal groups (with the additional constraint that the direct product is a subgroup of $SO(n)$). There are many different types of partially oriented Flag manifolds.

Steifel Manifold - Points on the Steifel manifold $ST(l,n)$ correspond to ordered orthonormal sets of l vectors in $\mathbb{R}^n$. Alternatively, points on $ST(l,n)$ correspond to elements in the set $\{A \in R^{n\times l} | A^T A = I_l\}$. The Steifel manifold $ST(l,n)$ can also be considered as the oriented Flag manifold $FL^\circ(1, 2, 3, \ldots, l; n)$. The dimension of $ST(l,n)$ is $(n-1) + (n-2) + \cdots + (n-l)$. There is a natural $2^l : 1$ covering map from $ST(l,n)$ to $FL(1, 2, \ldots, l; n)$.

The homogeneous spaces listed above are all compact differentiable manifolds whose points parameterize flags of (oriented) subspaces of R^n with a common signature $l_1, \ldots, l_m$. Many problems of interest can be formulated in terms of optimizing some function on one or several of these or related parameter spaces. The formulation and solution is often driven by geometric considerations. Problems with an efficient numerical solution are particularly appealing. Some additional parameter spaces of interest include Affine space, Euclidean space, Hyperbolic space, Anti-di Sitter space, and product spaces built out of any combination of these spaces, their oriented versions, or any previously described homogeneous spaces.

2.3 Schubert Varieties

A Schubert variety in a Grassmann or Flag manifold is a certain kind of subvariety (typically with singularities) that can be defined by a collection of linear algebraic incidence constraints with respect to a fixed flag, F, drawn from some Flag variety $FL(k_1, k_2, \ldots, k_m; n)$. If you vary the flag then you vary the Schubert variety. The Schubert variety can be viewed as a kind of moduli space, while points on the Flag variety can be seen as parameterizing a family of Schubert varieties of a particular type. Schubert varieties also exist for the more general class of manifolds described in the previous section, but we will postpone their formal description to a follow-up paper.

2.3.1 Definition of Schubert variety

We first consider an example of a kind of Schubert variety that will be referred to several times in this paper. If $W \in GR(k, n)$ then W is a rank k subspace of $\mathbb{R}^n$. Given a pair of non-negative integers (c, l), we can define a collection of points in $GR(l, n)$ by

$$\Omega_{c,k,l}(W) = \{V \in GR(l, n) \mid \dim(V \cap W) \geq c\}$$

For this set of points to be nonempty, we will need that $c \leq l$ and that $c \leq k$. For each $W \in GR(k, n)$, $\Omega_{c,k,l}(W)$ is a subvariety of $GR(l, n)$. Consequently, $GR(k, n)$ can be seen as parameterizing a family of such subvarieties of $GR(l, n)$. The subvariety $\Omega_{c,k,l}(W)$ is an example

of a particular kind of Schubert variety on $GR(l, n)$. As mentioned in the previous paragraph, Schubert varieties are typically singular.

The example in the previous paragraph can be extended in several directions. The following is an example where W is drawn from a more general Flag manifold. Recall that $FL(k_1, k_2, ..., k_m; n)$ is the collection of all flags of the form $W_1 \subset W_2 \subset \cdots \subset W_m \subset \mathbb{R}^n$ such that $\dim W_i = k_i$. To emphasize that W is a flag of vector spaces, we will write W as $\mathbf{W}$. We call $FL(k_1, k_2, ..., k_m; n)$ an m step Flag manifold. Points on this manifold are m step flags of signature $(k_1, k_2, \ldots, k_m)$. A Grassmann manifold is a one-step Flag manifold. For instance, $GR(k, n) = FL(k; n)$. A large collection of Schubert subvarieties of $GR(l, n)$ can be built as follows: Pick a point $\mathbf{W}$ on a Flag manifold $FL(k_1, k_2, ..., k_m; n)$ and an m-tuple $\vec{c} = (c_1, \ldots, c_m)$ (thus $\mathbf{W}$ corresponds to a specific flag $W_1 \subset W_2 \subset \cdots \subset W_m$ where $\dim W_i = k_i$). Let $\vec{k} = (k_1, k_2, ..., k_m)$. An associated subvariety of $GR(l, n)$ is given by

$$\Omega_{\vec{c},\vec{k},l}(\mathbf{W}) = \{V \in GR(l, n) \mid \dim(V \cap W_i) \geq c_i \text{ for } 1 \leq i \leq m\}$$

Points on $FL(k_1, k_2, ..., k_m; n)$ parameterize a family of such subvarieties.

Similarly, one can build subvarieties of a more general Flag manifold $FL(l_1, \ldots, l_s; n)$. The subvarieties will be written $\Omega_{C,\vec{k},\vec{l}}(\mathbf{W})$ The data that is determining the subvariety, $\Omega_{C,\vec{k},\vec{l}}(\mathbf{W})$, is the space from which we draw our fixed flags (e.g. $\mathbf{W} \in FL(k_1, \ldots, k_m; n)$), a space on which the subvariety lives (e.g. $FL(l_1, \ldots, l_s; n)$), and incidence constraints, $c_{i,j}$, stored in an $m \times s$ array C. We have

$$\Omega_{C,\vec{k},\vec{l}}(\mathbf{W}) = \{\mathbf{V} \in FL(\vec{l}; n) \mid \dim(V_j \cap W_i) \geq C_{i,j} \text{ for } 1 \leq i \leq m,\ 1 \leq j \leq s\}.$$

2.3.2 Schubert Varieties of Best Fit

Suppose we are given a collection of l-dimensional subspaces $D = \{V_1, \ldots, V_r\}$ of $\mathbb{R}^n$. Each element in D can be thought of as a point in the Grassmannian $GR(l, n)$; thus, we have r points on $GR(l, n)$. We seek to determine a Schubert variety that best fits r points. For this problem to

make sense, we need to answer two questions: the first is "What class of Schubert varieties are you going to use to best fit the data?" and the second question is "What is the objective function you are trying to optimize when searching for a Schubert Variety of Best Fit?". Intuitively, we are searching for a Schubert variety that comes as "close as possible" to the set of points determined by D. This should remind you of finding a "linear space of best fit" to a set of points in $\mathbb{R}^n$. For our purposes, given a point or collection of points on $Gr(l,n)$ and a Schubert variety S, we further seek to have a measurement of closeness that is orthogonally invariant, i.e., is invariant to the action of the orthogonal group $O(n)$. Points on a Grassmannian correspond to subspaces, and Schubert varieties are defined in terms of incidence conditions with respect to a fixed flag of subspaces. With this in mind, in order to achieve measurements that are orthogonally invariant, it is natural to write the measurement of closeness in terms of principle angles between the subspaces involved. There are many different ways in which this can be carried out, leading to many different answers to the problem.

In what follows, let $D = \{V_1, V_2, \ldots, V_r\}$ be a collection of l dimensional subspaces considered as points on $GR(l,n)$. Given positive integers k and c, our goal is to find a point $W \in GR(k,n)$ such that the Schubert variety

$$\Omega_{c,k,l}(W) = \{V \in GR(l,n) \mid \dim(V \cap W) \geq c\}$$

comes as close as possible to the points in D. We will break this up into three parts. The first part will be to define a measurement of closeness, $d(V_i, \Omega_{c,k,l}(W))$, between a single point $V_i \in GR(l,n)$ and the Schubert variety $\Omega_{c,k,l}(W)$ (with $W \in GR(k,n)$). The second part will be to combine these single point measurements into a measurement of closeness between a set of points $D = \{V_1, V_2, \ldots, V_r\} \subset GR(l,n)$ and the Schubert variety $\Omega_{c,k,l}(W)$. The third part will be to find a $W^* \in GR(k,n)$ that optimizes this measure of closeness.

With respect to the third part, suppose we have fixed a real-valued measurement of closeness between a set of points, $D \subset GR(l,n)$ and a Schubert variety $\Omega_{c,k,l}(W)$. For each point $W \in GR(k,n)$, we have effectively assigned a real number (the closeness measure). Thus, we have a

function $F : GR(k,n) \to \mathbb{R}$. Since $GR(k,n)$ is a compact manifold, this real-valued function will attain both its maximum and minimum values. Our goal is to describe an algorithm implemented in a neural network to find a point in $GR(k,n)$ that achieves either a maximum or a minimum of this function.

2.3.3 Variations of Distance/Closeness measures

Let V_1, V_2 be subspaces of $\mathbb{R}^n$ of dimension l. Recall that the principal angles between V_1 and V_2 satisfy $0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_l \leq \pi/2$. If $\Theta(V_1, V_2) = [\theta_1, \theta_2, \ldots, \theta_l]$ then $d_g(V_1, V_2) = \|\Theta(V_1, V_2)\|_2$ is known as the geodesic norm between V_1 and V_2. If

$$\sin\Theta(V_1, V_2) = [\sin\theta_1, \sin\theta_2, \ldots, \sin\theta_l]$$

then $d_c(V_1, V_2) = \|\sin\Theta(V_1, V_2)\|_2$ is known as the chordal norm between V_1 and V_2.

We can generalize to the setting where V_1 and V_2 have potentially differing dimensions and modify the measure of the length of the principal angle vector between these subspaces. We will rename V_1 as V and V_2 as W to emphasize this flexibility in dimensional differences. Let V, W be subspaces of $\mathbb{R}^n$ of dimensions l and k and let $m = \min(l, k)$. Let c be a positive integer less than or equal to m and define $\Theta_c(V, W) = [\theta_1, \theta_2, \ldots, \theta_c]$ and $\sin\Theta_c(V, W) = [\sin\theta_1, \sin\theta_2, \ldots, \sin\theta_c]$. Given a vector norm, $\|\cdot\|_\alpha$ on $\mathbb{R}^c$, we can measure the "size" of the vector $\Theta_c(V, W)$ or of $\sin\Theta_c(V, W)$. We claim that in either case, this norm gives a measure of closeness between a point $V \in GR(l,n)$ and the Schubert variety $\Omega_{c,k,l}(W)$ by defining $d(V, \Omega_{c,k,l}(W)) = \|\Theta_c(V, W)\|_\alpha$ or by defining $d(V, \Omega_{c,k,l}(W)) = \|\sin\Theta_c(V, W)\|_\alpha$. To see that this is a measure of closeness, note that $\theta_1, \ldots, \theta_c$ are the c smallest principal angles between the subspaces. They correspond to the c smallest possible principal angles between c-dimensional subspaces of V and c-dimensional subspaces of W.

If we now pick a norm $\|\cdot\|_\beta$ on $\mathbb{R}^r$, once we have chosen a norm for measuring the size of $\Theta_c(V, W)$ or of $\sin\Theta_c(V, W)$, we can measure a Schubert variety's fit to a collection of l-

dimensional subspaces $D = \{V_1, \ldots, V_r\}$ of R^n by

$$FIT(D, \Omega_{c,k,l}(W)) = \|d(V_1, \Omega_{c,k,l}(W)), d(V_2, \Omega_{c,k,l}(W)), \ldots, d(V_r, \Omega_{c,k,l}(W))\|_\beta$$

and we can define the Schubert Variety of Best Fit to D as $BEST(D, \Omega_{c,k,l}) = \Omega_{c,k,l}(W^*)$ where

$$W^* = \arg \min_{W \in GR(k,n)} FIT(D, \Omega_{c,k,l}(W))$$

For programming advantages, in the next section, an optimization problem is described in terms of maximizing the norm of $\cos \Theta_c(V, W) = [\cos \theta_1, \cos \theta_2, \ldots, \cos \theta_c]$ instead of minimizing the norm of $\sin \Theta_c(V, W)$. The goal of the optimization problem is to find $BEST(D, \Omega_{c,k,l})$.

2.4 Optimization Problem for SVBF

Consider a set of matrices $\{X_i\}_{i=1}^r$ with each $X_i \in \mathbb{R}^{n \times l}$ having orthonormal columns. Let $K \in \mathbb{R}^{n \times k}$ be an unknown matrix with orthonormal columns. Let $\mathcal{R}(X_i)$ denote the column space of X_i. Define θ_{ij} to be the jth smallest principal angle between $\mathcal{R}(X_i)$ and $\mathcal{R}(K)$.

Problem Statement: Given the set $\{X_i\}_{i=1}^r$ *and an integer* c *with* $1 \leq c \leq min(l, k)$*, find* K *that comes as close as possible to satisfying* $dim(\mathcal{R}(X_i) \cap \mathcal{R}(K)) \geq c$ *for each of the subspaces* $\mathcal{R}(X_i)$*.*

One approach is to find a matrix $K^* \in \mathbb{R}^{n \times k}$ such that

$$K^* = \arg \max_{K \in \mathbb{R}^{n \times l}} \frac{1}{rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \cos^2(\theta_{ij}) \tag{2.1}$$

subject to $K^T K = I$ and $1 \leq c \leq \min\{l, k\}$. We have normalized by the factor $1/rc$ so that the optimal value of the solution will be one. Note that for $c = \min\{l, k\}$ this is the Flag mean [30].

Alternatively, we can solve

$$K^* = \arg \max_{K \in \mathbb{R}^{n \times k}} \frac{1}{rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \cos(\theta_{ij}) \tag{2.2}$$

subject to $K^T K = I$ and $1 \leq c \leq \min\{l, k\}$. Note that for $c = \min\{l, k\}$ this is the Flag median [32].

2.4.1 SVBF Optimization Problem Formulation

For simplicity, in this paper, we will focus on the case $c = 1$, i.e., we are looking for the column space of each X_i to be close to intersecting the column space of K in at least one dimension.

Suppose we are given a set of matrices $\{X_i\}_{i=1}^r$, each $X_i \in \mathbb{R}^{n \times l}$ satisfying $X_i^T X_i = I_l$. We would like to find a matrix K whose column space intersects the column space of each X_i in at least a one-dimensional space. This is achieved if and only if the first principle angle between the column space of K and the column space of X_i is equal to zero for each i. Let $\theta_1(K, X_i)$ denote the first principle angle between the column space of K and the column space of X_i. In the optimization below we seek to find a matrix K that maximizes the function $\sum_{i=1}^r \cos^2(\theta_1(K, X_i))$. This amounts to finding a a matrix $K^* \in \mathbb{R}^{n \times k}$ such that

$$K^* = \arg \max_{b_i, K} \sum_{i=1}^{r} b_i^T K^T X_i X_i^T K b_i \tag{2.3}$$

subject to $b_i \in \mathbb{R}^{k \times 1}$, $\|b_i\| = 1$, $K \in \mathbb{R}^{n \times k}$, and $K^T K = I_k$.

In the second optimization problem, given below, we seek to find a matrix K that maximizes the function $\sum_{i=1}^r \cos(\theta_1(K, X_i))$. This amounts to finding a matrix $K^* \in \mathbb{R}^{n \times k}$ such that

$$K^* = \arg \max_{a_i, b_i, K} \sum_{i=1}^{r} a_i^T X_i^T K b_i \tag{2.4}$$

subject to $a_i \in \mathbb{R}^{l \times 1}$, $\|a\| = 1$, $b_i \in \mathbb{R}^{k \times 1}$, $\|b_i\| = 1$, $K \in \mathbb{R}^{n \times k}$, and $K^T K = I_k$.

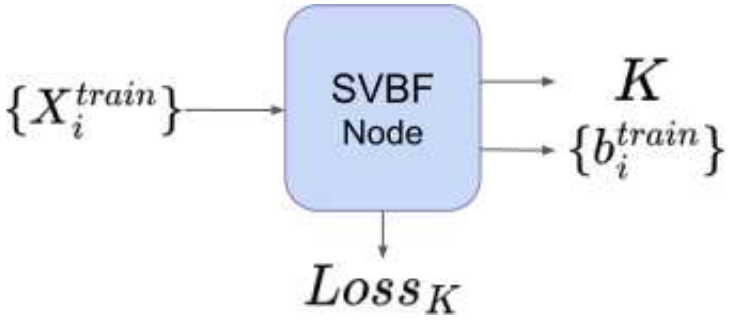


Figure 2.2: SVBF-Node diagram

2.4.2 SVBF-Node: Optimization with PyTorch

All of the experiments in this paper are conducted for the special case where $c = 1$. This constraint and the relative simplicity of our dataset allow us to initialize the problem directly with the PyTorch class. In this case, the unknown matrix K and vectors $\{b_i\}$ are initialized in the PyTorch class as two sets of parameters (Figure 2.2): $\{K_{ij}\}$, $1 \leq i \leq n, 1 \leq j \leq k$, and $\{b_{ij}\}$, $1 \leq i \leq r$, $1 \leq j \leq k$. The Lagrangian, associated with the problem given by equation (2.3), is considered via the Adam optimizer [33] and provides a path to the approximate solution:

$$Loss_K(K, b) = -\sum_{i=1}^{r} b_i^T K^T X_i X_i^T K b_i + \lambda_0 \left(K^T K - I_k\right) + \sum_{i=1}^{r} \lambda_i \left(\|b_i\| - 1\right) \tag{2.5}$$

2.4.3 Complexity

Let's focus on the first part of this expression, containing the majority of complexity. In our design, for each sample it is calculated in three steps:

1. $L_i^1 = K^T X_i$
2. $L_i^2 = L_i^1 L_i^{1T}$
3. $L_i^3 = b_i^T L_i^2 b_i$

Based on this chain of matrix multiplications and assuming that $n \gg k$, we can calculate the time complexity of the forward pass as $\mathcal{O}\left(2k(nl + kl + 2k)\right) = \mathcal{O}\left(nkl\right)$. The backward pass

Table 2.1: Tables of processing times for one iteration of optimization loop in milliseconds. The number of samples is fixed to 100, while the dimensionality of inputs l, dimensionality k of K, and dimensionality n of ambient space vary.

(a) $k = 8 \times 1$

$l \setminus n$	8^2	8^3	8^4	8^5
1	1.18	1.23	1.58	2.77
4^1	1.27	1.70	5.00	49.31
4^2	1.26	1.69	5.00	45.27
4^3	1.30	1.82	5.20	48.80

(b) $k = 8 \times 4^1$

$l \setminus n$	8^2	8^3	8^4	8^5
1	1.30	1.49	2.45	1
4^1	1.47	1.97	5.09	28.18
4^2	1.47	1.98	5.18	30.53
4^3	1.54	2.13	5.87	52.20

(c) $k = 8 \times 4^2$

$l \setminus n$	8^2	8^3	8^4	8^5
1	1.54	2.13	8.62	160.36
4^1	1.86	2.72	10.43	191.06
4^2	1.87	2.76	10.91	192.08
4^3	2.01	3.01	11.99	204.16

(d) $k = 8 \times 4^3$

$l \setminus n$	8^2	8^3	8^4	8^5
1	4.09	6.76	73.90	491.70
4^1	5.17	7.72	77.40	506.52
4^2	5.49	7.97	80.87	518.80
4^3	6.58	9.96	89.52	616.49

complexity, calculated based on the chain of multiplications of respective Jacobian matrices, is also equal to $\mathcal{O}(nkl)$. Hence, along with complexity of the Adam optimizer, equal to $\mathcal{O}(nk)$, the overall complexity of one iteration is equal to $\mathcal{O}(nkl)$ and the complexity of the entire optimization process is equal to $\mathcal{O}(tnklr)$, where t is the number of iterations and r is a number of l-dimensional inputs. Note that it is effectively equal to the complexity of training one fully connected layer with k nodes at the output, given that it is trained on $l \times r$ n-dimensional vectors with the same number of iterations t.

The given approximations of the complexity are derived following the traditional approach that was common before the era of GPUs. The GPUs benefit from parallelizing an immense number of simple operations and can accelerate matrix multiplications by orders of magnitude. We run all of our experiments on V-100 Tesla GPUs; hence, we present computational profiling in Table 2.1 to understand the performance in actual experiments better. The profiling was conducted for 100 randomly generated inputs. The dimensionalities of inputs vary as $n \in \{8^2, 8^3, 8^4, 8^5\}$, $l \in \{1, 4^1, 4^2, 4^3\}$, while the dimensionality of K varies as $k \in \{8 \times 1, 8 \times 4^1, 8 \times 4^2, 8 \times 4^3\}$. Given that SVBF optimization can also be run as training with batches for a larger number of samples,

this range of parameters supposedly covers several possible cases one may encounter in real experiments.

To approximate the empirical time complexity under realistic conditions, we focus on areas such as image and video recognition, remote sensing, and genomics or bioinformatics. In image and video recognition, spatial data often occupy ambient spaces with dimensionalities on the order of hundreds, as is typical for mid-resolution images. Similarly, in fields like remote sensing and bioinformatics, the number of points, i.e., the dimensionality of samples in our method, around the tens are frequently used to capture essential features while maintaining computational feasibility. Therefore, we have set the ambient space dimensionality to 8^3 and the sample dimensionality to 4^2, enabling effective modeling and analysis across various applications while maintaining a balance between data richness and practical applicability. Through experimentation, we found that, generally, increasing the dimensionality of the SVBF solution beyond 32 does not enhance performance and may even negatively impact it. With these settings, as shown in Table 2.1 the complexity measures approximately 1.98 milliseconds per iteration. In our trials, we observed that 10,000 to 40,000 iterations are typically sufficient for the SVBF solution to converge. Consequently, the total computation time would be 19.8 to 79.2 seconds. While the suitability of this time depends on the application, we believe that the substantial performance improvements demonstrated by our solution across various real-world tasks make it a viable option for high-risk, high-reward applications.

In fact, instead of including the unit length condition for each of the b_i's in the cost function, we use the normalizing transformations of b_i's suggested in [22], which doesn't introduce much complexity and, at this point, has already been implemented in PyTorch as a built-in routine. The resulting loss function is as follows:

$$Loss_K\left(K,b\right) = -\sum_{i=1}^{r} b_i^T K^T X_i X_i^T K b_i + \lambda_0 \left(K^T K - I_k\right) \tag{2.6}$$

where $\|b_i\| = 1 \forall i \in \{1, r\}$ by construction of the PyTorch class. As mentioned before, this function is minimized with Adam optimizer, and the number of iterations is equal to 40,000. The following is a list of some other details important for the reproduction of results:

- λ_0 = 10,000
- Initialization of parameters K and $\{b_i\}$:

  element-wise random numbers from a uniform distribution on the interval [0,1)
- Adam optimizer settings:

  lr =0.001, betas =(0.9,0.999), eps=1e−08, weight_decay=0, amsgrad=False
- Python version: 3.6.8
- Torch version: 1.10.1
- Cuda version: 11.0

All these settings are preserved exactly the same across all experiments, including SVBF problem-solving. For the faster processing in benchmarking experiments, we also leverage Ray 2.0.0 Python package to run several experiments in parallel.

2.5 Other prototypes on Grassmann Manifold

The SVBF solution investigated in this work can be considered a prototype for a set of subspaces. Effectively, in all of our implementations in Chapter 3 and Chapter 4, we use the SVBF solution as a prototype representative of a group of points on a Grassmanian. Depending on the specific pipeline, this prototype may be used to characterize the group or a cluster of unlabeled new points. Naturally, ML tasks in our experiments can also be performed with different definitions of prototypes and respective measures of closeness. Therefore, before we set off to more practical content of this dissertation, it is important to overview the most widely used and benchmarked prototypes on the Grassmann manifold, especially those used in our experiments for comparative analysis.

Table 2.2: Summary of other prototypes on Grassmann manifold that will be used in our future experiments

Name	Dimensions	Optimization Problem	Complexity	In
Karcher Mean	Fixed	$\min_{Y \in \mathrm{Gr}(k,n)} \sum_{i=1}^{p} \|\Theta(X_i, Y)\|_2^2$	$O(trl^2(n+l))$	[34]
Flag Mean	Adaptive	$\min_{Y \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \|\sin\Theta(X_i, Y)\|_2^2$	$O(nl^2)$	[35]
ℓ_2-median	Fixed	$\min_{Y \in \mathrm{Gr}(k,n)} \sum_{i=1}^{p} \|\Theta(X_i, Y)\|_1$	$O(trl^2(n+l))$	[36], [37]
Flag Median	Adaptive	$\min_{Y \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \|\sin\Theta(X_i, Y)\|_1$	$O(tnl^2)$	[32]

2.5.1 Karcher mean and l_2-median

The Karcher mean Y^* of a set of subspaces $\{X_i\} \in \mathrm{Gr}(k,n)$, introduced in [34], is a point on Grassmanian minimizing the sum of squared geodesic distances to all of the points on Grassmanian representing the set of subspaces, i.e., is the solution to the problem

$$Y^* = \arg\min_{Y \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \|\Theta([X_i], Y)\|_2^2 \tag{2.7}$$

In [36], Fletcher et al. suggested the Weiszfeld-type based algorithm to find the l_2-median, the subspace now solving

$$Y^* = \arg\min_{Y \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \|\Theta(X_i, Y)\|_2 \tag{2.8}$$

Karcher and l_2-median represent the so-called "intrinsic" means, fully considering the topology of the Grassmann manifold. The solution for both of them can be found as an iterative optimization with gradient descent, and the complexity for t iterations is equal to $O(tnrl^2) + O(trl^3))$.

2.5.2 Flag Mean

The chordal distance, as a measure of closeness, can also be used for subspace averaging. In [35], Draper et al. suggest the Flag mean, the solution for the problem of minimization of

aggregate squared chordal distances:

$$Y^* = \arg \min_{Y \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \left\| \sin \Theta \left(X_i, Y \right) \right\|_2^2 \tag{2.9}$$

where $\sin \Theta_c(X_i, Y) = [\sin \theta_1, \sin \theta_2, \ldots, \sin \theta_r]$ is a vector of sines of principal angles (see Section 2.3.3) and $r = \min\{rank(X_i), rank(Y^*)\}$. This optimization problem has a closed form solution that can be obtained with eigenvector decomposition of a sum of covariance matrices of all points, therefore, the complexity of solution is $O(nl^2)$. The chordal distance has also been used in [38]; therefore, we will be using it for comparative analysis.

2.5.3 Flag Median

Another way to average subspaces, slightly modifying the use of chordal distances, is referred to as the Flag median or Flag IRLS. It was suggested in [32] as a solution for the problem

$$Y^* = \arg \min_{Y \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \left\| \sin \Theta \left(X_i, Y \right) \right\|_2 \tag{2.10}$$

Here, the solution is based on an iterative implementation of a weighted Flag mean, resulting in a complexity being that of the Flag mean times the number of iterations, i.e., $O(trnlk)$. Clustering method developed in [32] has proven very effective, hinting towards its possible efficiency with other ML tasks on Grassmann manifold, therefore, it will be used for benchmarking as well.

Flag mean and Flag median optimization problems, as well as SVBF solution, allow different dimensionalities of subspaces and the final solution, while for "intrinsic" means, it is necessary that the dimensionality of subspaces is fixed and equal to that of a solution.

In addition to the prototypes discussed, the literature also suggests other prototypes on the Grassmann manifold, such as the Grassmannian Minimum Enclosing Ball (GMEB) [39] and the Max Length Unit Vector (also known as the Maximum Correlation Problem) [40], [41], [42]. However, they weren't included in our experiments, as either they did not provide the most meaningful or balanced metrics for comparative analysis or we couldn't find the stable code for a solution. It's

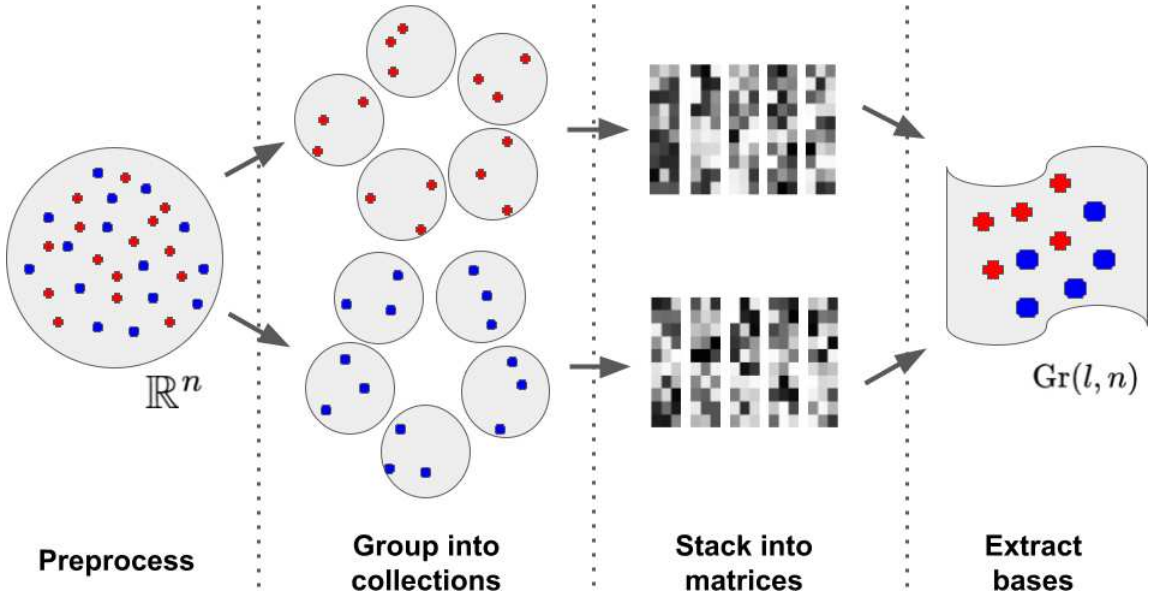


Figure 2.3: Data Preparation pipeline

important to note that the prototypes presented in this section, summarized in Table 2.2, do not encompass all possible variations of prototypes on the Grassmann manifold.

2.6 Data Preparation

In applications with different ML tasks, we usually have the categories combining the points in $\mathbb{R}^n$, and, typically, but not necessarily, we group these points into collections of equal size, such that every point falls into one collection. All points in one collection typically have the same category, allowing the entire collection to be labeled by any of its points. Finally, extracting the bases from all the collections generates points on Grassmaniann, Figure 2.3. Preprocessing for images includes normalizing and flattening into the vectors in $\mathbb{R}^n$. There are several options for video preprocessing. Each video is split into frames, and further, we can either form the collection from the frames of one video or different videos sharing the same category. For hyperspectral remote sensing data, the preprocessing is slightly different, we will expand on this later in Section 3.2.

Figure 2.4: Images from the Cat and Dog dataset. The top row corresponds to the data in a single cat sample with 3 elements. Similarly, the bottom row corresponds to a dog sample consisting of 3 elements.

2.7 Illustrative Example

We can now present the results for some implementations of the SVBF algorithm. Given that our objective at this point is exploring the algorithms, their possible implementations, and limitations, we focus on a modestly sized Cat and Dog dataset consisting of 99 images of cats and 99 images of dogs. All the images are 64×64 greyscale images; you can see some of the representatives of each class in Figure 2.4. We preprocess the data by flattening each image into a vector of length $n = 4096$ and scale the entries into the range from -1 to 1. Following the pipeline in Figure 2.3 we split the data into equally sized collections of l vectors and extract an orthonormal basis for each set. The resulting orthonormal bases are used as the inputs. In other words, the inputs, or samples, consist of tall orthonormal matrices of dimension $\mathbb{R}^{n\times q}$ where $q = l$ for training data and $q = m$ for test data. Importantly, no cat or dog vector is used in more than one sample. Further, we never mix classes within one sample; samples consist of sets of only dog vectors or only cat vectors. For example, the notations $\{X_i^{train}\}$, $l = 3$ describes the set of 3-dimensional bases of mono-class sets of scaled image vectors sampled from the training data.

In Figure 2.5, we show sample solutions of Schubert Varieties of Best Fit. In each case, the matrix K to be determined is chosen to have one column, and these solutions are displayed for cats and dogs individually. It is interesting to compare the solutions to equations (2.3) and (2.4). We see that for $\cos(\theta)$, the Schubert variety matrices K show higher resolution detail while for $\cos^2(\theta)$

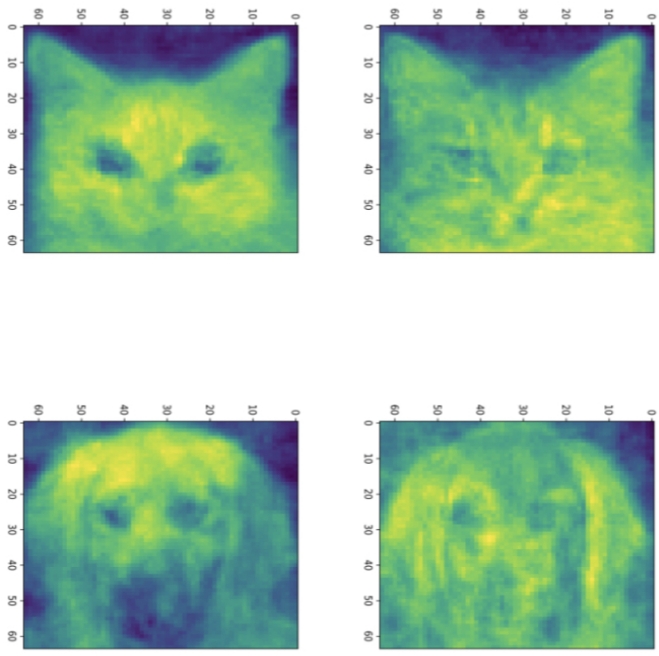


Figure 2.5: Solutions to the Schubert Variety of Best Fit problem for $a = 1$. The Left column consists of solutions to equation (2.3) while the right column consists of solutions to equation (2.4). In each case, there are three images in each sample. The top and bottom rows use cat and dog samples, respectively.

the features are larger scale. This example underscores the potential significance of the selection of distance measures in computing SVBFs.

The Figure 2.6 illustrates trained parameters for SVBF-Node for the entire Cat and Dog dataset with l = 2 and k=3. As we now describe, this picture captures many interesting features of the method. Figure 2.6 (a) depicts the three columns of K. Interestingly, one column is a dog, and one is clearly a cat, but all columns seem to capture salient features in the data. In Figure 2.6 (b), we show the directions Kb_i for three cat samples and three dog samples, where we recall that each sample is a 4096×2 matrix, so each of these six figures is a direction in the span of K that is closest to the data training sample. In Figure 2.6 (c) we show the set of all features in terms of b_i for cats (blue) and dogs (red). We note that their clear separation is a reflection of the fact that this method captures information capable of discriminating between cats and dogs.

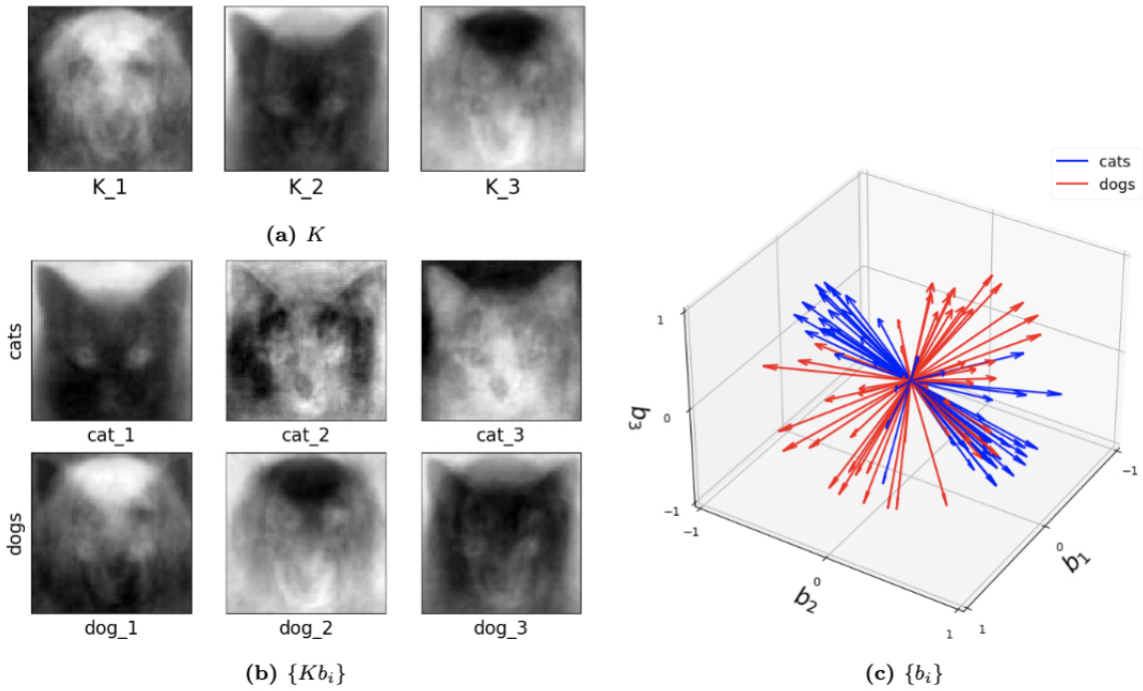


Figure 2.6: Visualisation of the outputs of the trained SVBF node and generated embeddings for the entire Cat and Dog dataset, k=3, and l=2: (a) learned basis K, (b) some samples reconstructed in ambient space from embeddings, (c) learned embeddings.

2.8 Optimal Dimension of K

We illustrate the empirical values of the resulting objectives of the SVBF optimization problems in Figure 2.7. Figure 2.7 (a) shows the value of the objective function for the optimization problem given by equation (2.3), i.e., the average sum of the squared cosines of the smallest principal angles for 10 2-dimensional subsets of cats sampled from both training and test datasets versus the dimension k of the solution subspace $\mathcal{R}(K)$. We can see that the objective function for test samples effectively flattens out after $k = 2$. Similar behavior can be observed for the dog dataset, presented in Figure 2.7 (b), except the flattening is smoother and starts approximately at $k = 4$. The results for the combined datasets, presented in Figure 2.7(c), show that the flattening also starts at around $k = 4$. These plots lead us to several observations. Firstly, the flattening of the curves itself suggests that more columns in K do not lead to improvements in the solution. Thus, the set of points $\{X_i\}$ has an intrinsic subspace dimension as captured by K. We see that the dimension of this subspace is smaller than the direct sum of the dimensions of the class-specific subspaces.

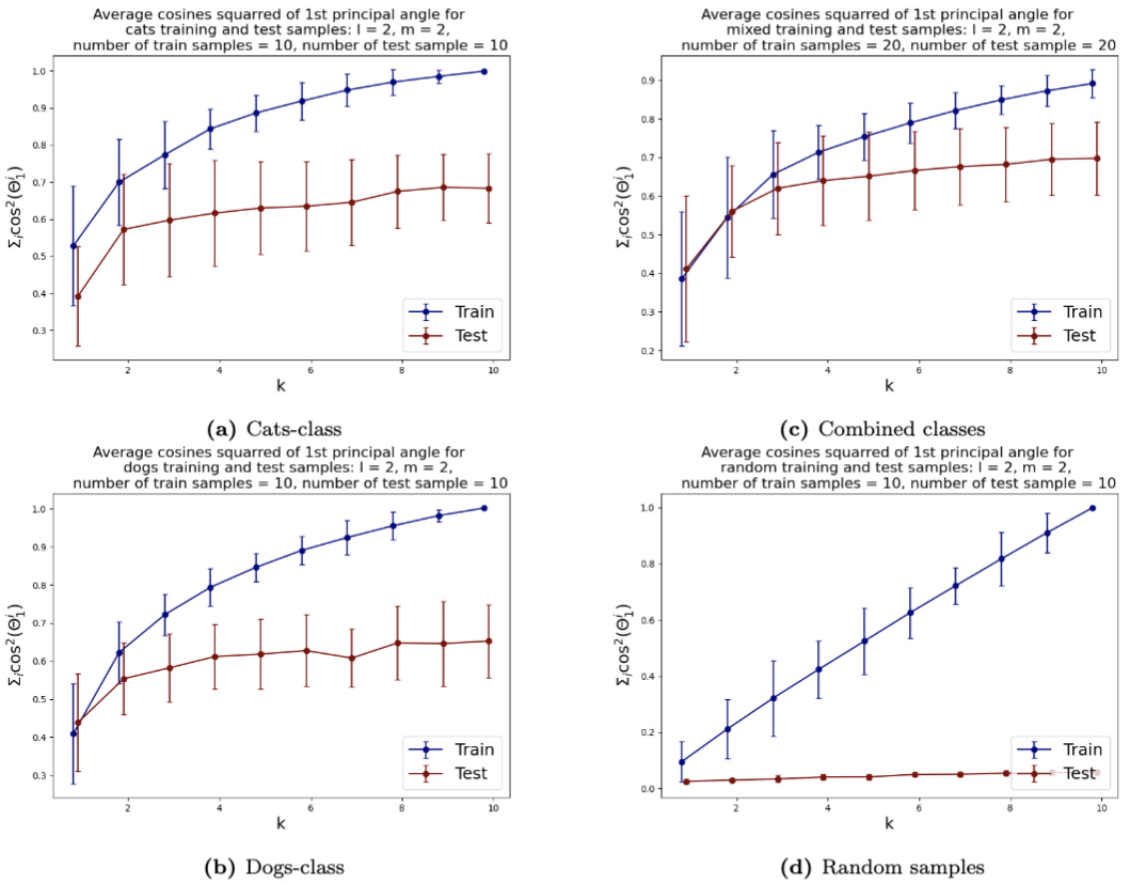


Figure 2.7: Average squared-cosines of the 1st principle angle for training and test data sampled from different classes, from both classes and sampled from the uniform distribution on the Grassmann manifold: (a) Cat class, (b) Dog class, (c) Cat and Dog classes combined (d) Random samples.

This is not a surprise when one considers the correlation between the images. Finally, the Figure 2.7 (d) illustrates how samples, randomly selected from the uniform distribution (with respect to Haar measure) on a Grassmann manifold, do not possess the same structure and the number of columns required in K does not converge. Experiments with other dimensions of samples have also been conducted, and they align with the results reported here. Other approaches to dimension optimization are also possible with the SVBF method, e.g., task-specific optimizations that will be considered in Chapter 3.

2.9 Conclusions

In this chapter, we developed a novel geometric approach for analyzing sets of datasets through the concept of a Schubert Variety of Best Fit. We formulated two optimization problems and proposed a specific solution design, the SVBF-Node, as an abstract computational unit that minimizes the respective objective function via the Adam optimizer, a gradient descent-based approach suited to the unique structure and parameters of our network. We demonstrated that SVBF-Node can generate new types of embeddings in $\mathrm{Gr}(l, n)$, where l and n denote the dimensionalities of the samples and the ambient space, respectively. These embeddings can be used directly or integrated further into larger networks that include SVBF-Node as a component. Several such implementations will be discussed in Chapter 3.

We derived the theoretical time complexity of our method and provided empirical benchmarks within a specific coding environment. This analysis shows that our method's complexity is comparable to training a single fully connected layer in a neural network with the number of nodes matching the dimensionality of the SVBF solutions over the entire dataset. Empirical results, specifically from implementations in PyTorch, suggest that our approach is efficient for most machine learning tasks, provided that the product of the SVBF solution's dimensionality, ambient space dimensionality, and sample dimensionality remains below a certain threshold - a condition typically met by most real-world datasets. Furthermore, our benchmarks indicate that the method can achieve practical runtimes, making it suitable even for complex datasets, provided certain dimensional constraints are observed.

Finally, this chapter introduces a very general, unsupervised method for approximating the intrinsic dimensionality of an entire dataset. Notably, the optimal dimensionality identified here for a specific dataset will be validated in Chapter 3 through a supervised learning approach, further reinforcing the robustness and versatility of our method.

Chapter 3

Classification with SVBF-Node

3.1 Classification Frameworks

Here we explore the applications of the SVBF with the classification task, including implementation of SVBF-Node as a computational unit in a feed-forward neural network. The Cat and Dog dataset used in the process and data preparation routine is described in Section 2.7. We will also consider the cases when dimensionality of samples in training and test data might differ, therefore, we allow for l and m to be distinct.

3.1.1 Algorithm I

Given a solution K and a data sample X_i, it is natural to consider the change of coordinates of the sample that can be implemented in the neural network, i.e.,

$$Y_i = K^T X_i$$

The design of the classification experiment for this case is presented in Figure 3.1. Step 1 involves training the SVBF Node, the outcome of which is the representative K of the SVBF. This K is then used in Step 2 to compute the change of coordinates to produce Y. This is done using training data. Further, in Step 2, the 1-nearest neighbor (1NN) classification is implemented using the Scikit-learn Python package. Step 3 involves the application of 1NN to the test data; the results are shown in Figure 3.2. Figure 3.2 (a) illustrates the average for 10 different samplings of a 1NN-classifier applied to the $\{Y_i\}$ embeddings of the Cat and Dog dataset. In each sampling, the data is split randomly into train and test datasets with proportions 0.78 and 0.22, respectively. Benchmarking is performed for different dimensions $l = m = 1, \ldots, 5$ of samples in training and tests subsets, and different dimensions $k = 1, \ldots, 10$ for K. The accuracy grows with the dimension of the samples but interestingly starts decreasing for $l \geq 3$. At the same time, increasing

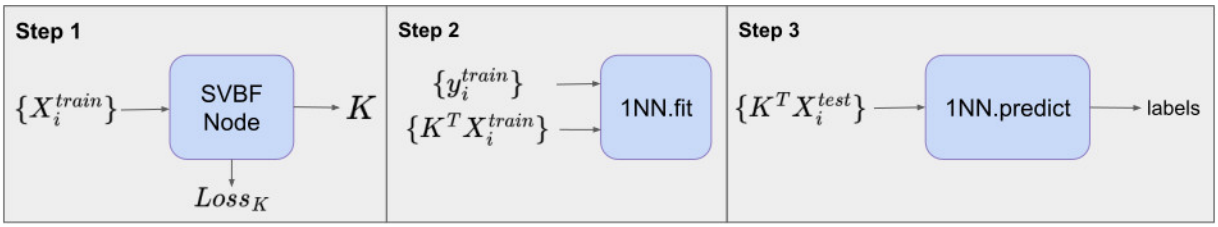


Figure 3.1: Workflow of Cat and Dog classification experiment using Algorithm I.

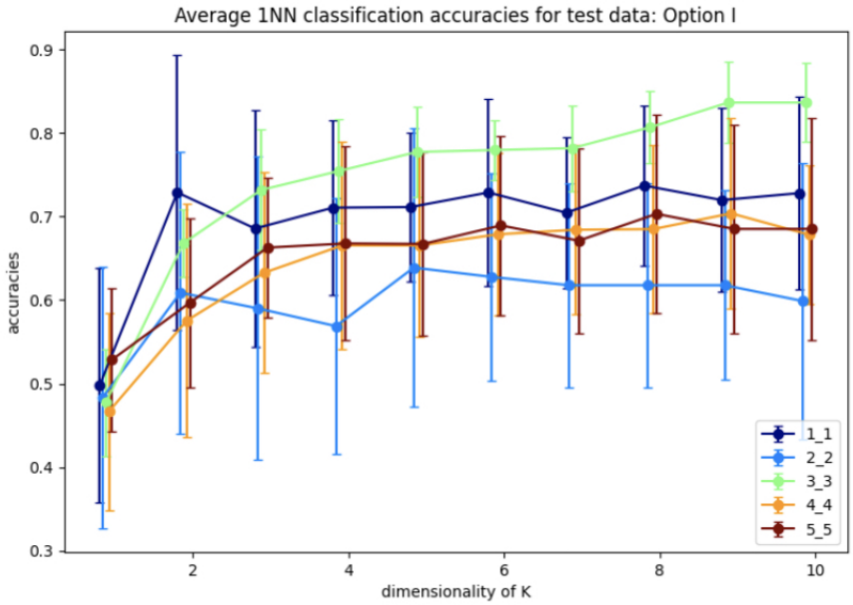
the dimension of K also significantly increases the accuracy up to dimension 3 and only slightly affects the accuracy above that level. This low accuracy is a result of using the Frobenius norm to compute distances between matrices. Later, we will see that using subspace distances, in general, produces superior results. For a better understanding of the benefits of the SVBF method, we provide side by side the results for classification based on PCA-embeddings as well, Figure 3.2. In this first experiment, the performance of the classification method based on learning K is slightly better than the one based on learning a representative subspace based on PCA analysis.

3.1.2 Algorithm II

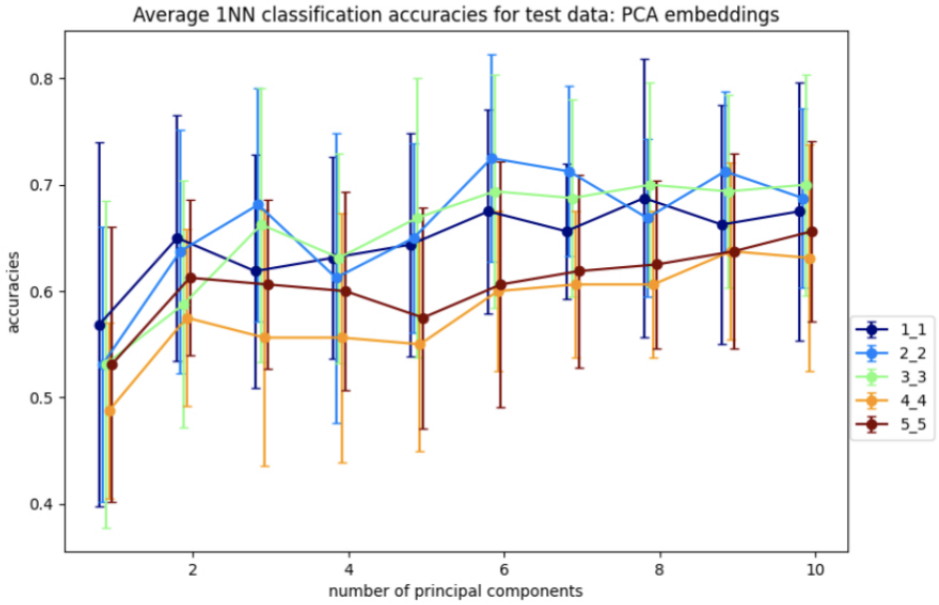
Another possible approach is to learn a representative K_i for each data class. In our example, in Step 1, we propose to learn K_1 as a solution to equation (2.3) for the cat class and K_2 as a solution to equation (2.3) for the dog class. Now, these matrices K_1, K_2 can be used to classify the data. To assign a class to an unknown sample X_i in Step 2, we compute the smallest principle angle between X_i and each of K_1 and K_2. The smallest of these two angles provides the classification. The diagram for such an experiment with the Cat and Dog dataset is presented in Figure 3.3. Labeling of the test data is performed by finding the nearest, in terms of the smallest principle angle to K_i, for each sample.

Figure 3.4 (a) shows the average classification accuracy across 10 different samplings of the test data. Again, this procedure is implemented for a range of l, m and k. The data was split into training and test sets with the same ratio as in the Algorithm I experiments.

The benchmarking plots indicate a significantly more accurate classification versus Algorithm I for the comparable cases. Due to the higher overall accuracy of this design, we also investigated



(a) Y embeddings - Algorithm I design



(b) PCA embeddings

Figure 3.2: Average accuracies with error bars for 1NN-classification of test data for Y embeddings from Algorithm I and for PCA embeddings: (a) Y embeddings, plots are labeled as l_m depending on the dimensions of training samples l and dimensions of test samples m, (b) PCA embeddings, plots are labeled as in (a).

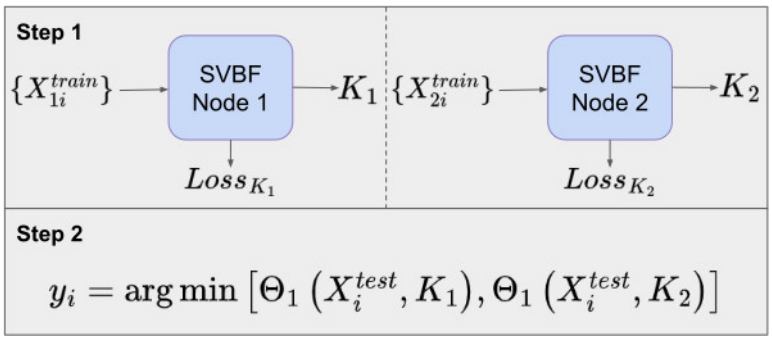


Figure 3.3: Workflow of Cat and Dog classification experiment using Algorithm II.

it for different dimensions of training and test samples, which can be useful for a wide variety of datasets, specifically when test samples cannot be grouped by labels. As before, increasing the dimension k of K leads to higher accuracies for all cases. However, in contrast to Algorithm I, now the accuracy increases monotonically with the dimensions l, m.

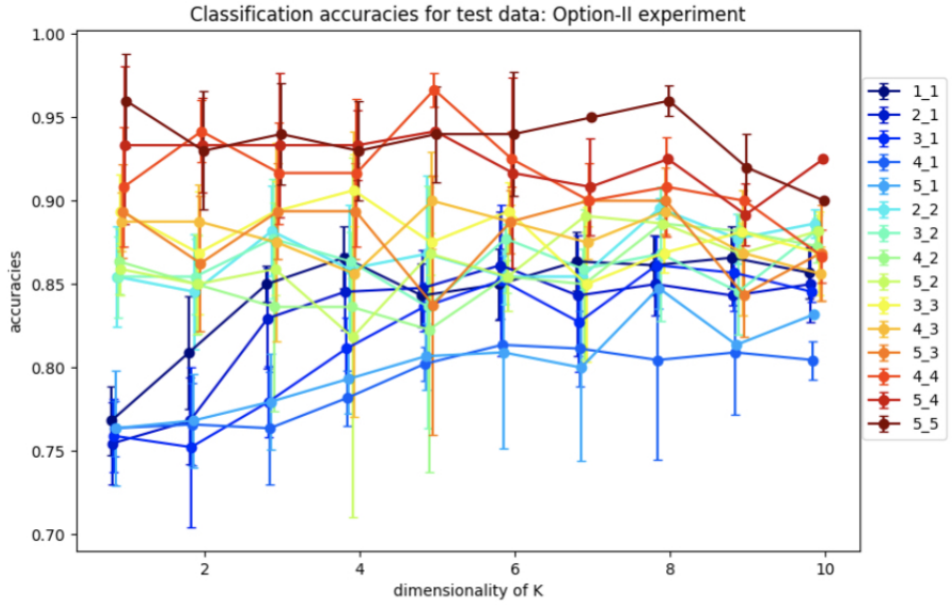
For further comparative analysis, we also calculate PCA embeddings for each class and label test samples based on the nearest, in terms of the smallest principle angle, subspace captured by PCA analysis. We repeat this experiment 10 times for different samplings similar to the Algorithm I experiment. The results of this experiment are shown in Figure 3.4 (b). The lower accuracies for the class-specific PCA experiment indicate that the SVBF optimization problem is capturing additional useful information that is helpful for classification.

3.1.3 Algorithm III

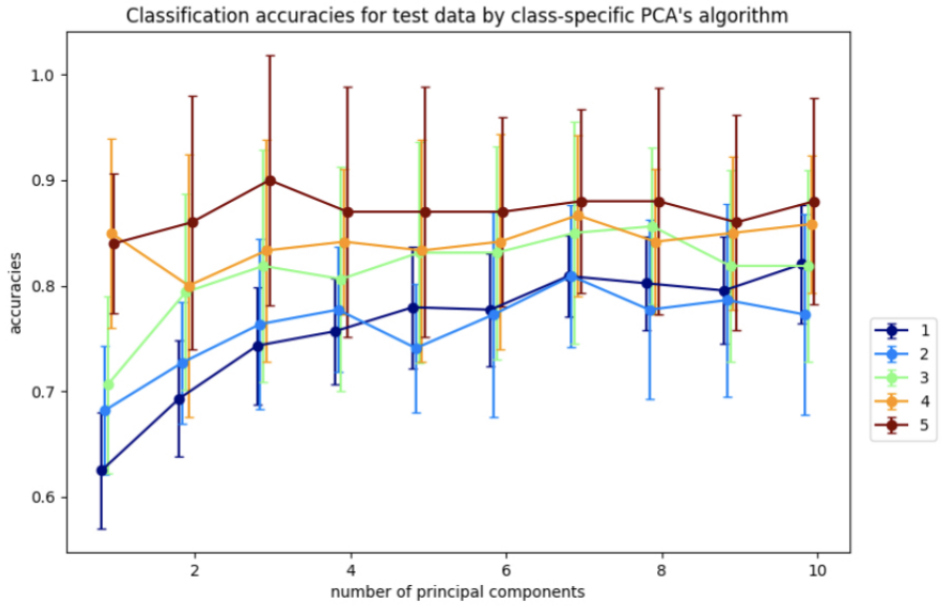
Note that the element $\hat{k} \in \mathcal{R}(K)$ given by

$$\hat{k}_i = Kb_i$$

is the vector in $\mathcal{R}(K)$ that is closest to the span of X_i. There is a natural association between X_i and b_i and this can be exploited for classification. Hence, a consequence of equation (2.3) is that once we approximate K we can use b_i, the coordinates of the closest to X_i vector in K, as a proxy for X_i in the classification. One possible experiment that exploits this option is classification with neural networks outlined in Figure 3.5.



(a) Algorithm II design



(b) Closest PCA design

Figure 3.4: Average accuracies of classification of test set by Algorithm II design and design based on the closest class-specific PCA subspaces: (a) Algorithm II design, plots are labeled as l_m depending on the dimensions of training samples l and dimensions of test samples m, (b) Closest PCA design, plots are labeled as in (a).

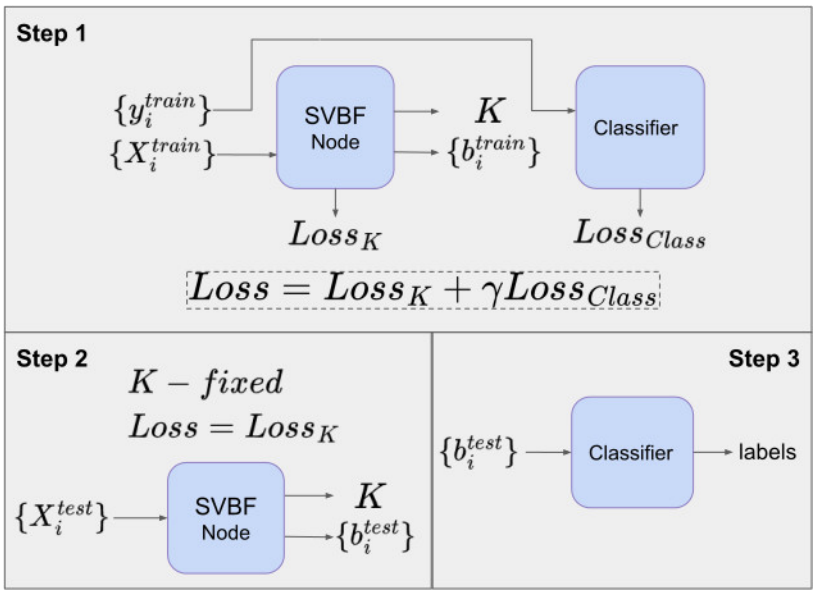


Figure 3.5: Workflow of Cat and Dog classification experiment using Algorithm III.

This experiment also illustrates how SVBF nodes can be stacked into a larger network and trained simultaneously with other network parts. In this case, we attach a classifier that takes outputs b_i of SVBF nodes as inputs, learns centroids in a training process, and generates soft labels at the inference step. In Step 1, we optimize K and b_i independently at each iteration. In other words, K is not adjusted by any contribution to the gradient from the classification error. Generally, the outline given in Figure 3.5 does not require detaching the classifier's error back-propagation from that of an SVBF node. However, detaching the training of K and b_i, as we have done here, seems to improve performance without loss of the benefits of parallelization. The loss function for Step 1 is defined as a weighted sum of three loss functions:

$$
\begin{aligned}
Loss_\theta &= -\sum_{i=1}^{M} b_i^T K^T X_i X_i^T K b_i \\
Loss_{orth} &= K^T K - I_{k\times k} \\
Loss_{Class} &= -\sum_{j=1}^{k} y_{ij} \log\left(\text{logit}_{ij}\right) \\
Loss &= \alpha Loss_\theta + \beta Loss_{orth} + \gamma Loss_{Class}
\end{aligned}
\tag{3.1}
$$

where $\text{logit}_i = Softmax((b_i^T W)^2, T)$ with W learnable centroids, and the hyperparameter T. The output of Step 1 is the matrix K, set $\{b_i^{train}\}$ and the classifier W. Now, given K, W, the prediction of the labels is accomplished in Steps 2 and 3. In Step 2, we initialize the SVBF node with the K computed in Step 1 to determine the coordinates b_i^{test}, associated with the test samples X_i^{test}, using the loss function $Loss_K$ similar to Step 1, but without the classification component and with K fixed. In Step 3, we use the classifier W trained in Step 1 to map the b_i^{test} to their class label.

This hybrid network was also tested with the Cat and Dog dataset following exactly the same preprocessing pipeline as in the Algorithm I and Algorithm II experiments. In Figure 3.6, you can find the classification accuracy for test data for different dimensions of training and test samples as well as different dimensions for K. It is very interesting that the accuracy of the classification problem levels off at 4 dimensions, which is consistent with the optimal size $k = 4$ of K as suggested in Figure 2.7. Hence, we view $k = 4$ as the apparent *working* dimension for K. We see the accuracy increases monotonically with the dimension of the samples. Importantly, the results in which the samples have dimensions greater than 5, which we are not reporting here, support this observation. However, given our limited data, we were not able to pursue this behavior further.

In the same Figure 3.6, we also report the average accuracies for classifying l_2-normalized vectors used as inputs. In this design, instead of an SVBF-Node, we use one fully connected layer with k nodes at the output and with the hyperbolic tangent as the activation function (all other settings are kept the same). In Section Section 2.4.2 it was shown that this network has the same computational complexity as the SVFB-Node. Along with the higher resulting average accuracies for some cases, this method also has the advantage of approximately 10 times faster convergence (in terms of the required number of iterations). On the other hand, we can see that for dimensionalities of test samples ≥ 3, the SVBF method is more accurate. It's important to note here that benchmarking against the regular networks processing vector inputs wasn't within the focus of this chapter. Our prime goal was to show that in the suggested framework, a big part of the progress made with the regular neural networks during the last decades can be directly leveraged in the case of sets of datasets.

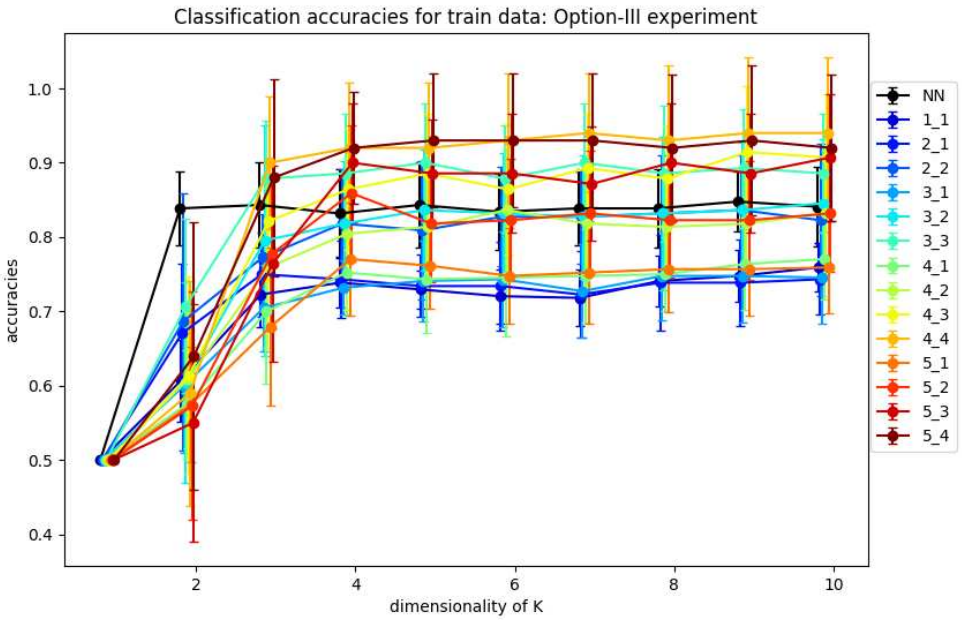
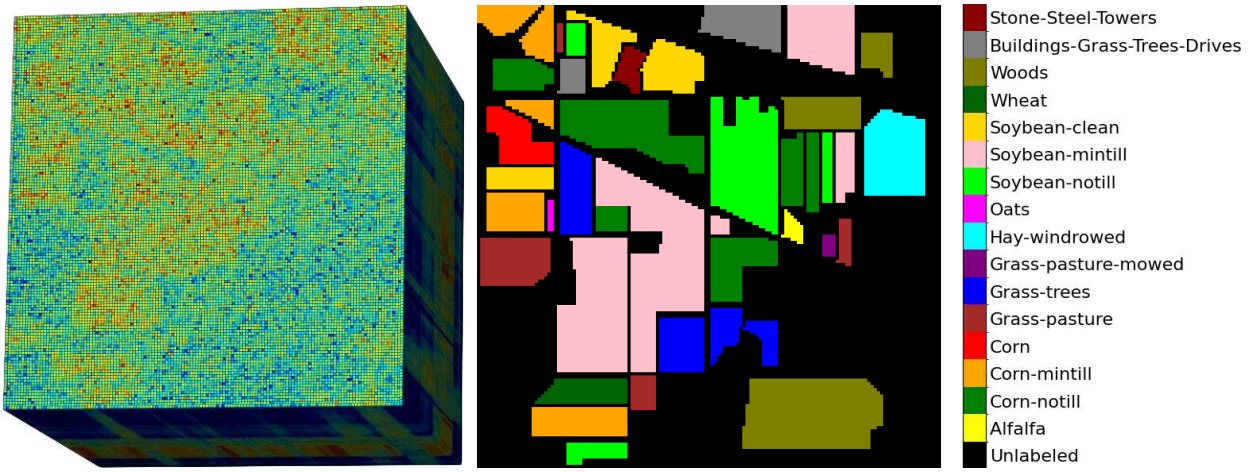


Figure 3.6: Average accuracies of classification of test data by Algorithm III design. Accuracies for one-layer NN design are labeled as 'NN'; accuracies for SVBF-Node design as 'l_m'; where l is the dimensionality of train samples and m is the dimensionality of test samples.

3.2 Indian Pines classification with Algorithm II

Recall that SVBF-Node is based on the cosine squared of the first principle angle. In [38, 43] Chepushtanova et al. coin this metric as a "pseudo" metric and develop the classification method training Sparse SVM classifier on Multi-Dimensional Scaling (MDS) embeddings generated with the use of "pseudo" metric as a similarity measure between subspaces. The benefits of the "pseudo" metric are clearly demonstrated in those papers, and this partially motivated us to conduct similar experiments with our most performant method, Algorithm II, modified for usage with different definitions of prototypes and measures of closeness. In [38], authors test their results versus geodesic and chordal metrics, while we will also use the measure used in the definition of Flag median since it has proven very effective when tested with clustering tasks in [32]. Chepushtanova et al. run the experiments on two 'best' and two 'worst', in terms of separability, classes of Indian Pines dataset.

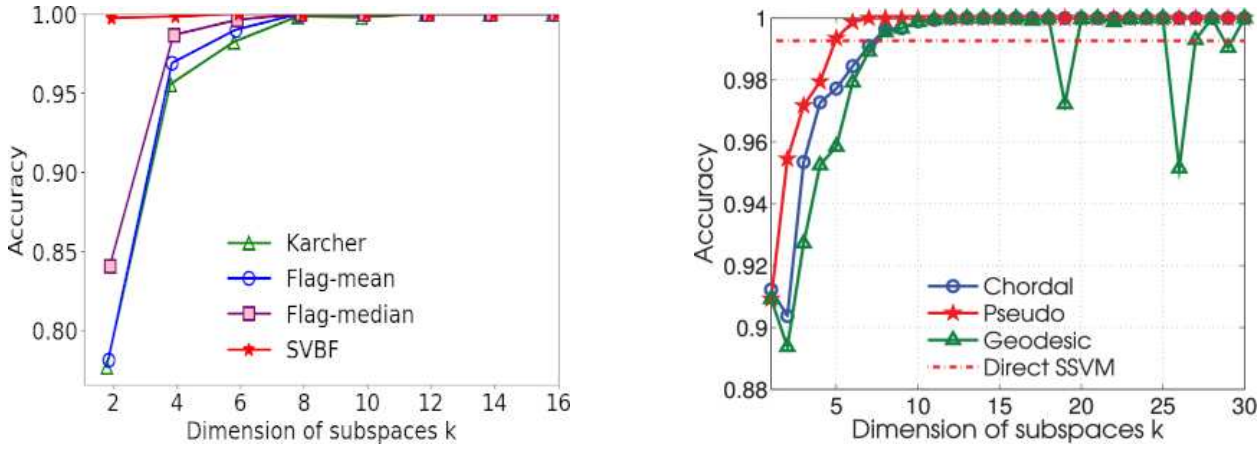


(a) **(b)** **(c)**

Figure 3.7: Indian Pines dataset: (a) Multiple bands stacked together, (b) Label Map, (c) Labels

The Indian Pines dataset is a hyperspectral landuse dataset with 16 distinct ground-truthed classes [44]. The data consists of hyperspectral bands collected over farmland in Indiana, US, with 145×145 pixels. For each pixel, the dataset contains 220 spectral reflectance bands representing different portions of the electromagnetic spectrum in the wavelength range $0.4 - 2.5 \times 10^{-6}$, Figure 3.7. The two most separable classes are *Corn-notill* and *Grass/Pasture* classes, and the two least separable classes are *Soybeans-notill* and *Soybeans-min*. We preprocess this dataset identically to the pipeline used in [38] and generally illustrated in Figure 2.3. All the pixels filtered for classes *Corn-notill*, *Grass/Pasture*, *Soybeans-notill* and *Soybeans-min* are then divided into equal-sized groups of l pixels, from which an orthonormal basis is then extracted for each group. These orthonormal bases serve as the inputs. Similar to the previous experiment, each pixel is included in only one group, ensuring that classes are not mixed within any single group.

The experiments are run 10 times, and the average results are reported. For all prototypes, the dimensionalities of samples l and dimensionalities of prototypes k vary in the range [2, 4, 6, 8, 10, 12, 14]. In the case of Karcher-mean $l = k$, while for all other experiments, the whole variety of k's is tested for each l. In Figure 3.8 (a), you can see the average over ten runs accuracies of classification of two "best" classes for different prototypes and varying dimensionalities of samples l and prototypes k with $k = l$. In Figure 3.8 (b), we also show the results from [38]. We added the

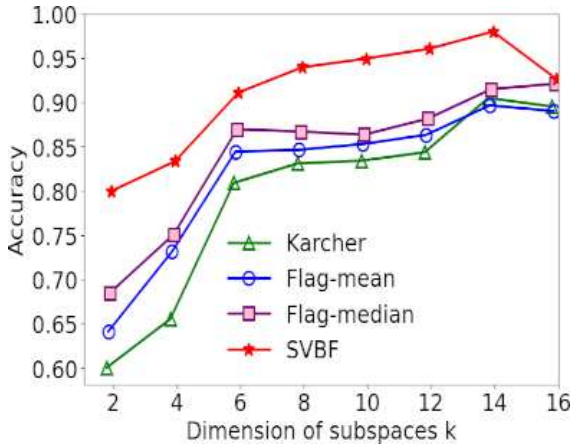


(a) SVBF based Algorithm II

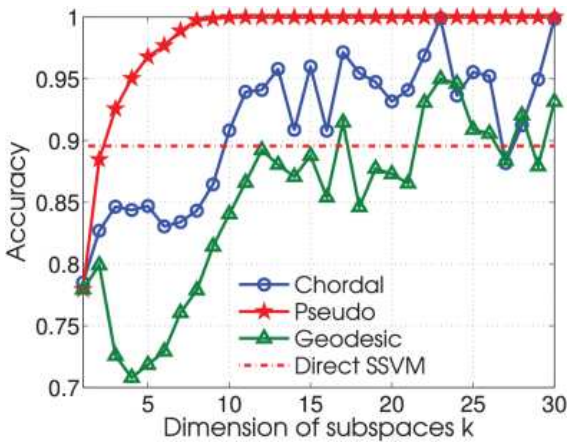
(b) SSVM classification of MDS embeddings

Figure 3.8: Average accuracies of classification of classes *Corn-notill* and *Grass/Pasture* with two different methods: (a) Algorithm II using SVBF versus Algorithm II using other prototypes, (b) SSVM classification of MDS embeddings from [38] for different metrics

figures here to facilitate the comparison of results in [38]. In Figure 3.8 (a), we can see that the performance of our method is significantly better for lower dimensionalities, starting with nearly perfect classification even for $k = 2$, and stays superior up until $k = 8$. Interestingly, the Flag median appeared as the second best, we will see similar order in efficiency with the results of clustering task in Chapter 4 as well. Similar behavior we can see in Figure 3.8 (b), with the accuracies for other metrics catching up with the "pseudo" metric, which is also used in our method, around $k = 12$. We can see that in contrast to our method, the robustness of the "pseudo" metric in [38] starts degrading for higher dimensionalities. In Figure 3.9 you can see the results of classification for the two "worst" classes. Note that apart from [38] and the methods based on neural networks, no other methods able to separate these two classes could be found in the literature. As you can see in Figure 3.9, with our method, the classification accuracies steadily increase with the increasing dimensionality up until $k = 14$, reaching nearly perfect classification at its peak. Here, we again can see that the classification with SVBF prototype performs consistently better than classification with other prototypes, similar to how in [38] "pseudo" metric consistently outperforms other metrics. Note that the accuracies for other prototypes in Figure 3.9 (a) catch up with accuracies for SVBF slower than in Figure 3.8 (a), signaling that it might be the case that SVBF method is



(a) SVBF based Algorithm II



(b) SSVM classification of MDS embeddings

Figure 3.9: Average accuracies of classification of classes *Soybeans-notill* and *Soybeans-min* with two different methods: (a) Algorithm II using SVBF versus Algorithm II using other prototypes, (b) SSVM classification of MDS embeddings from [38] for different metrics

especially superior with the less separable data, given that we explore the optimal or "working" dimensionality first. In both experiments, we can observe degradation of accuracies beyond a certain dimensionality, typically 12-14. Similar behavior was observed for different dimensionality, even though we cannot directly compare the optimal dimensionality because of the drastic difference between the two methods, [38]. This is another argument in favor of exploring the optimal dimensionality case by case, depending on the dataset and framework.

3.3 Consclusions

In this chapter, we proposed three distinct algorithms for data classification and explored and benchmarked them using the same dataset and preprocessing pipeline. Algorithm I uses a single solution K to characterize the data and a Frobenius norm to measure distances. Overall, this algorithm performed poorly when compared to Algorithms II and III. Based on learning two SVBFs Algorithm II provides the most accurate classifications. This algorithm explicitly builds a model for the cats and dogs through their respective SVBFs, and appears to be more capable of addressing complex structures of data, including classes with higher within-class variance. Algorithm III in-

troduces the idea of using auxiliary features to perform data classification, and is based on a single model K.

All three algorithms suggest that there is a best dimension for the representative subspace and that exceeding this doesn't improve classification accuracy. Hence, the SVBF approach appears to provide a measure of the complexity of a set of data sets through this *working* dimension. Interestingly, Algorithm III provides a very clear signal for this optimal dimension through a classification criterion. Algorithm III illustrates how the SVBF node can be used as an abstract unit of computation in a neural network. This enables researchers to process sets of datasets in the same spirit as sets of vectors are processed in a variety of ML libraries based on neural networks.

Algorithm II, as the most performant, was also tested with complex remote sensing data and benchmarked against algorithms using a similar pipeline but different from SVBF prototypes on the Grassmann manifold. Benchmarking with two extreme subsets, i.e., the least and the most separable classes of the dataset, revealed the optimal or *working* dimensionality for a given dataset, $k = 12$. Below the optimal dimensionality, our method significantly outperformed other methods, especially for the least separable classes. Overall, the results of this chapter indicate the promise of the SVBF approach for finding a representative subspace for the set of datasets in other tasks, such as the clustering task, that will be considered in Chapter 4.

Chapter 4

Subspace Clustering with SVBF-Node

4.1 Introduction

Subspace clustering extends the notion of data clustering to the setting of collections of linear or affine spaces. In each case, the goal is the same, i.e., to determine a set of representative points that can be used to characterize the data. One measure for understanding the complexity is a distortion error [45]. One can generally adapt clustering algorithms designed for Euclidean spaces to more general geometric frameworks. Typically, such algorithms rely on the ability to measure distances between pairs of points. For example, vector quantization relies on the ability to partition a set of points by computing the distance between each of the points and a set of representatives or cluster subspaces. At each iteration of the algorithm, each data subspace is viewed as a member of the closest cluster subspace. Thus, the partitioning is induced by the current collection of clusters. At this stage, the set of data subspaces that are the members of the same cluster can then be used to determine an updated estimate for that cluster. This leads to a dynamical system that can be used to evolve a set of initial clusters/representatives such that they gradually become closer to, and more representative of, the structure of the data. There are a range of approaches to update a cluster representative given a set of points closest to that representative, e.g., one may compute a new cluster representative as the centroid using the mean or the median of the members. This approach is typically referred to as LBG clustering [45], which is a batch modification of the k-means algorithm [46].

There have been a variety of approaches for extending the idea of LBG clustering to the setting of Stiefel, Grassmann, and Flag manifolds by computing various centroids or Frechet means. In this chapter, we propose to adapt the LBG clustering algorithm using the notion of a Schubert Variety of Best Fit. In this setting, a cluster for a set of subspaces, represented as points on a Grassmannian, chooses a point on a different Grassmannian (alternatively a Flag manifold) whose

corresponding subspace comes as close as possible to intersecting each of the cluster member subspaces in at least k dimensions, for some fixed k that is chosen by the user. We show that an iterative approach to vector quantization based on the SVBF produces state-of-the-art subspace cluster purity scores on a variety of benching marking data sets when compared to subspace mean algorithms in the literature.

The organization of this chapter is as follows: In Section 4.2 we discuss related work and describes several known clustering techniques on Grassmann manifold. Section 4.3 gives the approach to the SVBF problem and provides the pseudocode for the actual algorithm that was utilized. Section 4.4 describes four benchmarking experiments and compares SVBF clustering with subspace mean and median clustering. Section 4.5 outlines important experimental flow details and how we determined and set the experimental parameters and iteration counts for our algorithms in the absence of a universal convergence criterion. In Section 4.6, we then provide concluding remarks, limitations, a summary of results, and future work.

4.2 Related work

A wide variety of clustering methods have been suggested over the last decades [47,48], including subspace clustering methods [49, 50], however, the most related to our work are the k-means algorithms operating over the points on Grassmann manifold. Generally, they can be categorized into two groups: direct and kernel methods. Kernel methods typically require a pre-processing step, i.e., projecting the points to Reproducing Kernel Hilbert Space for the kernel-based methods [51, 52], or even projecting into the lower dimensional space via the kernelization as in [53]. The direct methods operate over the Grassmann manifold, and the diversity of direct k-means methods is typically provided by the differences in definitions of the distances on the manifold leading to different optimization problems for the means or centroids, see Table 2.2. In this context, some of the most related papers are the ones suggesting different ways to calculate the mean or centroid or other representative of the group of points on the Grassmann manifold. In our method the respresentative of the cluster cannot be righfully called mean or centroid, it is rather a custom

prototype optimizing certain geometric conditions. Exploiting custom prototypes on Grassmann manifold is not a completely novel idea, e.g. in [54] it was shown that with a proper design the prototypes can be very efficient for a variety of ML tasks on the manifold, including clustering. However, given that the points in our method are processed as they are and the optimization problem can be defined directly on the manifold, our method falls into the direct category.

Generally, direct methods include three types: intrinsic, extrinsic, and hybrid. Intrinsic methods completely consider the topology of the manifold. Specifically, Chendra Hadi et al., [55], develop k-means algorithm based on Karcher mean and geodesic distance as a measure of closeness, with updates performed along the geodesic. Extrinsic methods operate over the points in different space, e.g., points mapped into projection matrices in Euclidean space. The closeness, in this case, can be defined in different ways, as well as updating function. In [31] Tim Marrinan at al. use Flag means as the centers of clusters, with squared chordal distance measuring the closeness between supspaces and previous centers fully replaced by the new ones at each iteration. Flag mean and squared chordal distance have also been used in the hybrid method suggested in [56], with the centers updated by partially moving them towards the new centers along a geodesic. To the best of our knowledge, the LBG method developed in [32] is currently the state-of-the-art among these type of methods for subspace clustering, therefore, it will be used for benchmarking in all the experiments. This method is based on Flag median, see Table 2.2, the closeness is defined by chordal distance, and the centers are entirely replaced by new ones at each iteration. Several other prototypes have also been used in that paper for benchmarking, specifically, the Flag mean and l_2-median. The l_2-median based clustering appeared consistently underperforming, therefore, it wasn't considered for experimentation in out benchmarking.

4.3 SVBF-LBG algorithm

The LBG algorithm [45] was designed to quantize the distribution of samples in Euclidean space by labeling all samples by the closest centers. The central idea of the method is an iterative updating of the labels and local centers. At every step, the center is recalculated based on the

definition of representative of the group of samples assigned to this center. The representative was originally introduced simply as a mean of samples in Euclidean space. The convergence is defined as the minimization of the distortion error, i.e., aggregate distance from samples to the corresponding centers.

Here, we present the SVBF-LBG clustering method, a novel variation of the LBG algorithm designed to use the SVBF to approximate clusters for each subspace. The subspaces are represented by points on a Grassmann manifold. The clusters are computed by solving the SVBF problem (2.3) $\{K_i\}$ over the set of points closest to each cluster where the closeness between the subspaces X_i and K is taken to be $\sin^2\theta_1(X_i, K)$. This allows one to label subspaces as belonging to given clusters and calculate the distortions and purities of clustering.

The following is an outline of the SVBF-LBG algorithm:

Init: **Initialize Centers.** Initialize a number of random centers $\{K_i\}$, on $\mathrm{Gr}(k, n)$.

Step 1: **Label Samples.**

Label all samples by their closest centers using the distance measure $d_{ij} = \sin^2\theta_1(K_i, X_j)$.

Step 2: **Calculate New Centers.**

Calculate new centers by solving the equation (2.3) for each labeled group of samples.

Step 3: **Calculate Distortion.**

Calculate overall distortion by summing up the distances from all samples to corresponding centers.

Step 4: **Stopping Criterion.**

If the stopping criterion is not met, then go back to Step 1.

In practice, we initialize the algorithm by randomly sampling the required number of centers from the dataset. The stopping criterion should ideally indicate the flattening of the overall distortion. The numerical exploration of the convergence of the algorithm for many different datasets

indicated that it was more effective to start with a preliminary analysis and then proceed to approximate the number of iterations required for convergence case by case, i.e., we explicitly set the number of iterations for each experiment. This is a rather basic approach, which is non-optimal, more details in Section Section 4.5.

4.4 Numerical Experiments

We now present the results of the application of the SVBF-LBG algorithm for several benchmarking datasets. We start with synthetic data designed to be suitable for our algorithm. The MNIST dataset used for the second experiment is chosen given its widespread familiarity. The last two experiments provide more fundamental insights into the potential of the algorithm with closer to real-world datasets. For the benchmarking, we compare the proposed approach with the best performing subspace LBG methods, i.e., the methods based on the Flag mean and the Flag median, while omitting comparisons with the non-competitive methods based on Karcher-mean and l_2-median.

4.4.1 Synthetic data

The hypothesis is that a fundamental strength of the method presented in this work is related to the more granular geometric distribution of the subspace clusters. In order to test this hypothesis, we construct a dataset with samples grouped into clusters of samples indistinguishable in terms of the distance measure used in our method. We start by creating a 100×100 matrix with entries sampled from the uniform distribution on the interval [-0.5,0.5] and extracting an orthonormal basis through a QR decomposition. As a result, we now have an orthonormal 100×100 matrix Q. Each of the first 5 column vectors of this matrix will be used as one-dimensional prototypes of clusters. We proceed with creating groups with a fixed size of 10 elements, each of 10-dimensional subspaces clustering around prototypes and represented by 10 tall 100×10 matrices with orthonormal columns. For each prototype, such matrices are created by randomly sampling 9 vectors from the other 95 columns of matrix Q and stacking them together with the prototype into one matrix. As a

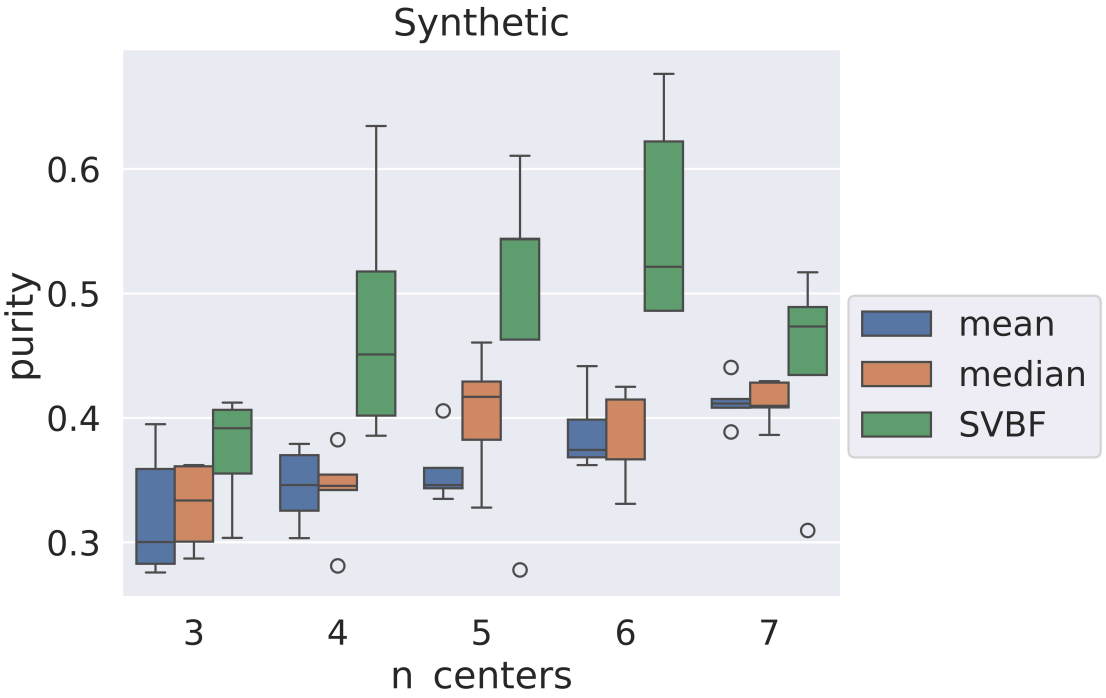
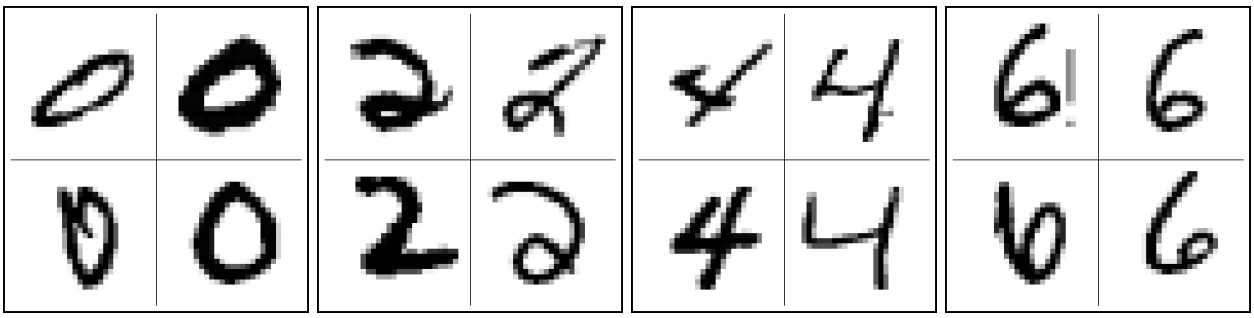


Figure 4.1: Median purities for a synthetic dataset for different methods with the first and third quartiles, whiskers, and outliers.

result, we generate 5 groups of subspaces with the size of each group equal to 10, summing up to overall give 50 subspaces with dimensionalities all equal to 10 and intersecting with any other subspace within their group at least in one direction. For the benchmarking experiment, we consider the number of centers around the true number of clusters, i.e., varying in the range [3,4,5,6,7]. The methods considered for comparison are the LBG clustering algorithms based on the Flag mean, Flag median, and SVBF reference subspace K, with all respective dimensionalities equal to 10. All experiments are run 5 times with the resulting purities of clustering averaged across all runs and present in Figure 4.1 in the form of comparative plots for different methods across different numbers of centers.

As you can see from the Figure 4.1, our method outperformed other methods for the whole range of numbers of centers. These results show that our method can be especially efficient when certain geometrical structures are present in a dataset, such as the one designed in this experiment.



digits ”0” digits ”2” digits ”4” digits ”6”

Figure 4.2: Random samples from the subset [”0”, ”2”, ”4”, ”6”] of MNIST dataset.

4.4.2 MNIST dataset

MNIST is a dataset [57] consisting of 28×28 grayscale images of handwritten digits from 0 to 9. We consider a small subset of digits representing the numbers [0,2,4,6], Figure 4.2, with the further vectorizing and scaling of all images to have values in the interval [-1,1]. The SVBF algorithm operates over sets of subspaces, therefore we split the data into equally sized collections of 10 vectors and extract an orthonormal basis for each collection. The resulting orthonormal bases are used as the inputs. In other words, the inputs, or samples, are the tall 784×10 matrices with orthonormal columns. Importantly, each image is used only in one collection, and we never mix digits within one collection, i.e., each sample is extracted from a mono-class collection of images. The resulting dataset includes 97 samples of ”0”, 103 samples of ”2”, 98 samples of ”4” and 95 samples of ”6”. The dimensionality of K in this experiment is set to 10. In Figure 4.3, you can see the average across 5 trials for the purities of clusters generated by our algorithm and benchmarked against the Flag mean and Flag median-based LBG algorithms for a number of centers in the range from 2 to 15. The distortions for the LBG algorithms flatten after 7 iterations, therefore, the number of iterations was set to 7 for all trials. The dimensionalities of Flag mean and Flag median are set to 10. Given that the subset used in this experiment includes only 4 digits and is known to be quite separable, it is not surprising that all the methods produce high-quality results, effectively performing perfect clustering for the number of centers greater than the number of label groups. Flag mean and Flag median methods are more accurate for a smaller number of centers in this case since the dataset is easily separable, i.e., the clusters are geometrically distant

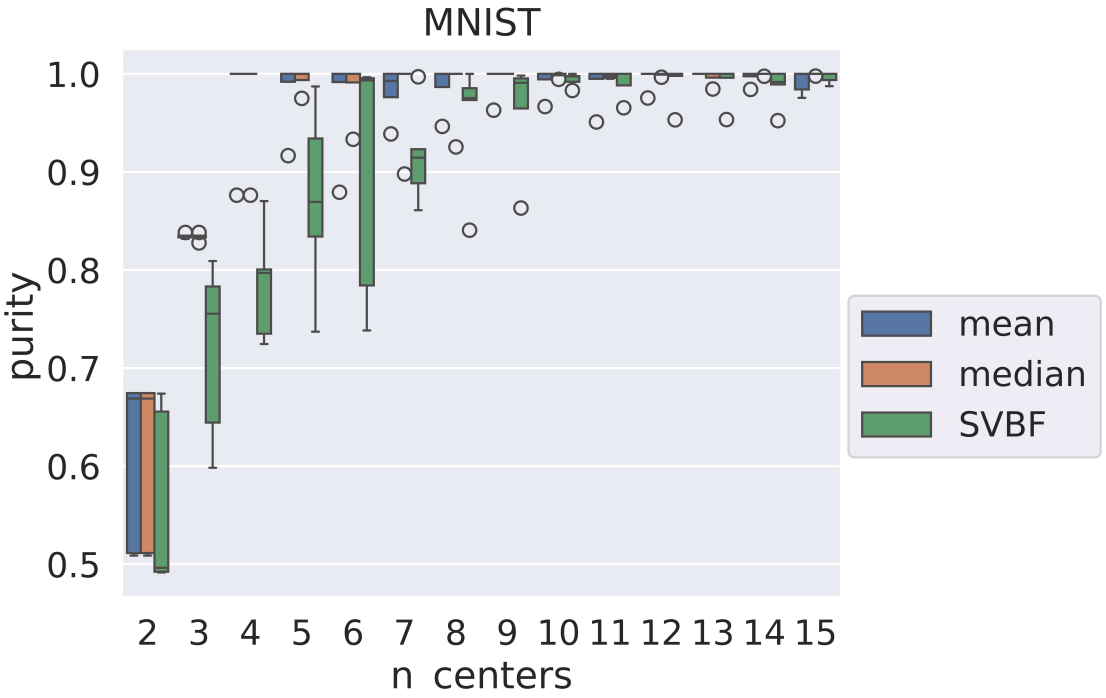


Figure 4.3: Median purities for MNIST dataset for different methods with the first and third quartiles, whiskers, and outliers.

from each other in many directions, all of which contribute to distance measures used in these two methods, while our method exploits only one direction and performs better when more intricate geometry is present in the data. However, for a larger number of centers, there is no difference in purities across all methods. Importantly, even though the variance of purities for our method is larger, hinting towards inferior robustness, the code for Flag median and Flag mean method shared in [32] was consistently returning errors related to SVD-implementation with the Python Numpy package not being able to generate a converging solution for the number of centers larger than 4. We had to rerun the experiments multiple times until convergence, while our method was consistently producing results.

4.4.3 Indian Pines dataset

We use the reduced version of the dataset described in Section 3.2 with 200 bands after removing bands associated with water absorption. The dataset includes 10776 unlabelled pixels and 16 classes of labeled pixels, but we sort only for the 4 largest and spectrally distinct classes,

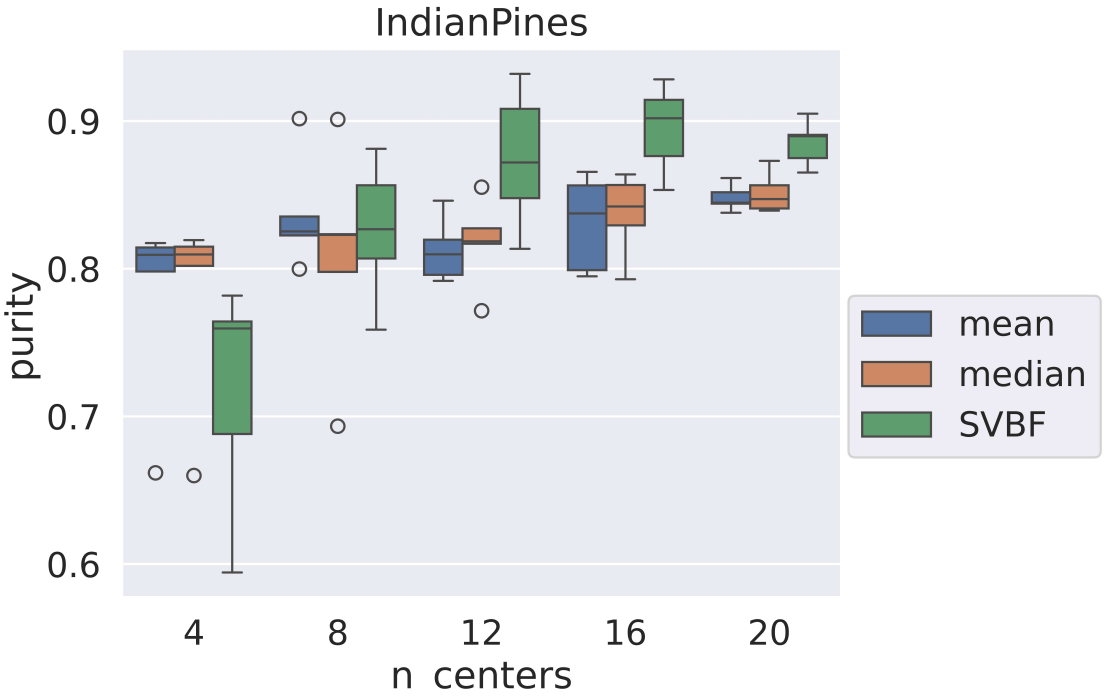


Figure 4.4: Median purities for Indian Pines dataset for different methods with the first and third quartiles, whiskers, and outliers.

specifically, *Corn-notill*, *Grass-trees*, *Soybean-mintill*, and *Woods*, each with 1428, 730, 2455, and 1265 pixels respectively. The preprocessing follows the pipeline used in the previous experiment. Specifically, the data is split into equally sized collections of 10 pixels with the further extraction of an orthonormal basis for each collection. The resulting orthonormal bases are used as the inputs. As in the previous experiment, each pixel is used only in one collection, and classes are never mixed within one collection. The resulting dataset consists of 10-dimensional bases, including 142 *Corn-notill* samples, 72 *Soybean-mintill* samples, 245 *Soybean-mintill* samples, and 126 *Woods* samples. Further we proceed with 5 trials of the SVBF-LBG algorithm implementation for different numbers of centers in the range [4,8,12,16,20], while using the labels of pixels used to generate bases as the true labels of samples in the calculation of cluster purities. Note that the choice of dimensionality equal to 10 was basically dictated by dimensionality chosen in [32] for this dataset, but it is fortunately very close to the optimal dimensionality approximated in Section 3.2 and equal to 12. The dimensionality of means and K are set to be equal to the dimensionality of the samples,

i.e., equal to 10. The distortions seem to flatten after 5 iterations, and the experiment is stopped at that point. In Figure 4.4 you can see the results averaged across all trials. Our method significantly outperforms other methods for all but one number of centers, highlighting the benefits of this granular geometric approach to clustering in this dataset.

4.4.4 UCF YouTube Action dataset

The dataset utilized for benchmarking in this section is derived from a specific subset of the UCF YouTube Action dataset [58] comprising 11 distinct categories of actions, Figure 4.5. Within each category, videos are organized into groups sharing common characteristics. For the purposes of our experiment, we selected one representative example from each group across all action categories. The resulting dataset includes 23 instances of *basketball shooting*, 23 of *biking*, 25 of *diving*, 25 of *golf swinging*, 24 of *horseback riding*, 25 of *soccer juggling*, 24 of *tennis swinging*, 24 of *trampoline jumping*, 24 of *volleyball spiking*, 24 of *walking*, and 22 of *swinging*. Given the substantial size of these RGB videos, we convert all frames extracted from videos into 25×18 grayscale images, followed by flattening the resulting images into vectors of length n = 450. The vectors are further grouped by source videos, with groups finally forming the matrices. As usual, we extract the basis for each matrix, i.e., generate tall orthonormal matrices of dimension $\mathbb{R}^{450\times 10}$ representing respective videos. The average results for different methods are presented in Figure 4.6. We compare average purities generated by 5 trials of Flag mean and Flag median LBG clusterings for a number of centers in a range [4,8,12,16,20] with the average purities generated by 5 trials of SVBF-LBG clustering, each including 5 iterations, for the same range of numbers of centers. The results clearly indicate considerable improvement provided by the SVBF-LGB method across all numbers of centers.

4.5 Experimental Pipeline Details

As mentioned before, for practical reasons, we chose to run all the experiments without a universal convergence criterion. Our design includes two nested loops: LBG iterations and SVBF

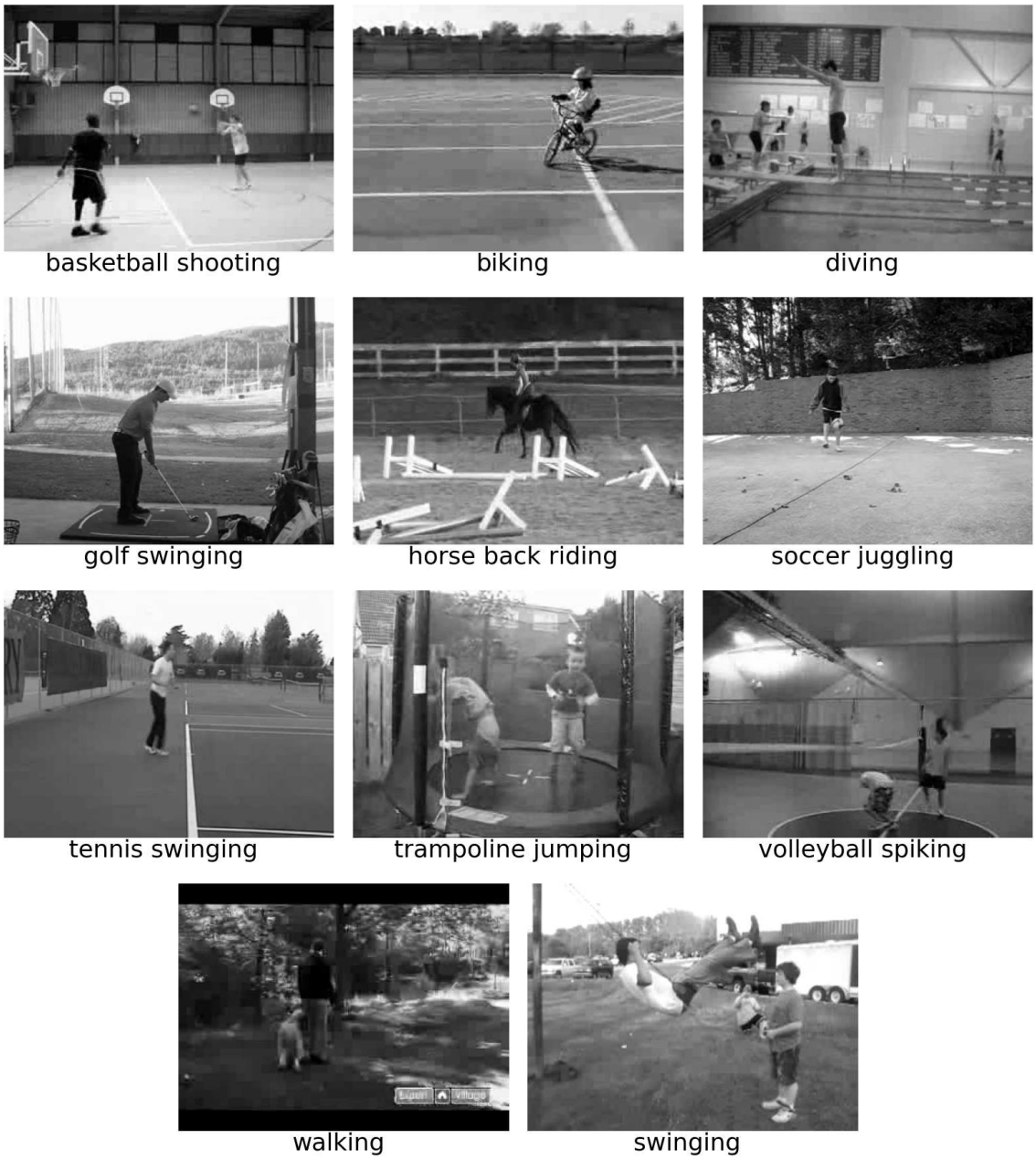


Figure 4.5: All 11 UCF YouTube Action categories: 1st frame of one video for different categories.

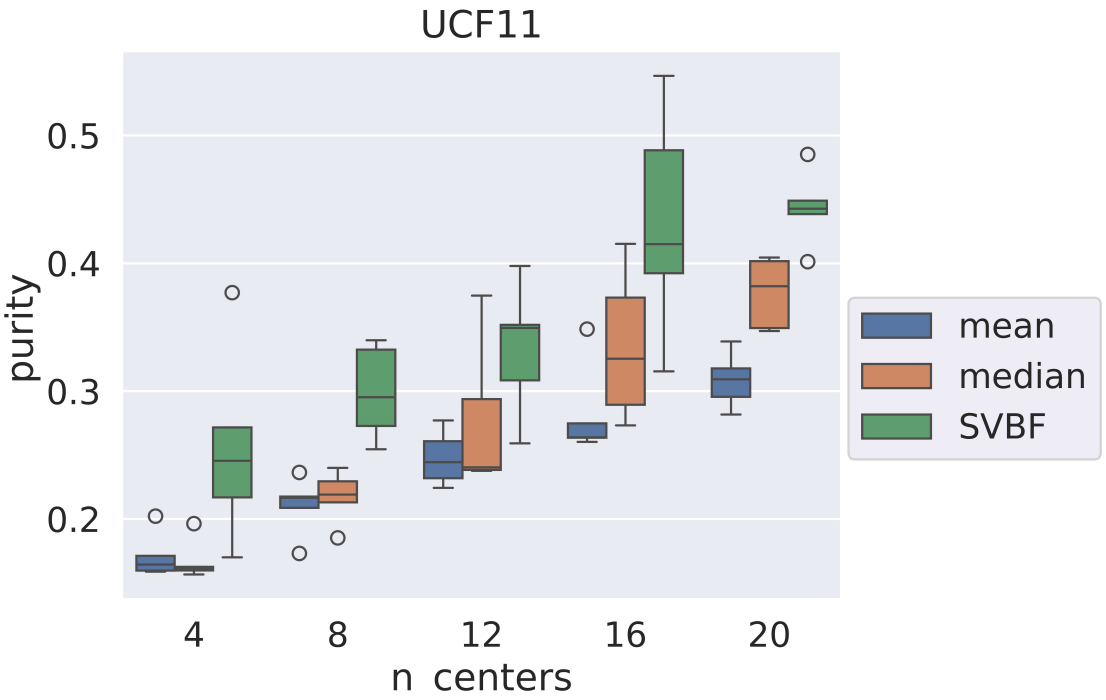


Figure 4.6: Median purities for the YouTube Action dataset for different methods with the first and third quartiles, whiskers, and outliers.

problem solutions at each iteration. The time complexity of SVBF solution was investigated in more detail in Section 2.4.3, including the complexity with and without consideration of the improvements provided by GPU parallelization. It was shown that the time complexity is proportional to the size and dimensionalities of the datasets. Running the experiments for several different datasets without convergence criterion allows us to approximate the processing time case by case. For the preliminary analysis, we first run the experiments with a fixed number of iterations equal to 1000 and the maximum number of centers that are going to be used in the final experiment. As a result, we can calculate the expected overall processing time as the time that it takes to run one iteration of the SVBF solution times a fixed number of iterations set for the SVBF solution and multiplied by the fixed number of iterations set for the LBG loop. These approximations appeared quite accurate in real experiments. Further, based on the convergence rate and approximations of the processing time, we decided on the optimal settings for each dataset and ran the final experiments. Based on the observed convergence rates, the number of iteration for LGB loop for the final

experiments was set to 5 for the Synthetic dataset and 10 for all other datasets, while the number of iterations for SVBF solution was set to 1000 for the synthetic dataset, and 3000 for all other datasets.

4.6 Conclusions

In this chapter, we exploit the geometry of the real Grassmann manifold in order to find the best representatives for collections of real vector spaces. We construct the Schubert Variety of Best Fit to characterize sets of data sets through numerical experiments with the Synthetic data, video action sequences, MNIST, and spectral data from the Indian Pines data set. We see that the approach provides an alternative to subspace means by forming a representative that is "close" to the cluster members in that every member comes close to intersecting the cluster in at least one dimension. We see that the resulting representatives capture attributes of the data. Given the nature of the optimization problem, we interpret the method as providing a more granular approach to subspace clustering.

The proposed SVBF-LBG method outperforms the leading methods in the literature on a variety of disparate data sets. Our synthetic example shows a case, by design, that illustrates the strengths of the algorithm. Interestingly, for the number pattern classification task using the MNIST data set, we find that for a sufficiently large number of centers, all the methods are essentially perfect. We conclude that the SVBF-LBG approach performs better as it has more degrees of freedom to match the data in contrast to subspace means.

The Remote Sensing data (Indian Pines dataset) appears to have an intricate geometry that is well captured by SVBF-LBG for 8 or more centers. Again, see that a more granular fit is obtained than for the subspace algorithms. The video action domain is an important subspace clustering task, and the SVBF-LBG method significantly outperforms state-of-the-art methods when applied to real-world data in this domain.

Our method, as presented, has several limitations. We only ran experiments with a fixed dimensionality of K, and we only looked at intersections in one dimension. Our convergence-stopping

criterion is currently simply a fixed number of iterations and has not been optimized. In its current implementation, the proposed method is expensive and requires GPU acceleration. We hope to improve upon these limitations in future work.

Chapter 5

A Multi-Domain Multi-Task Approach for Feature Selection from Bulk RNA Datasets

5.1 Introduction

In the field of bioinformatics, researchers often use microarray or next-generation sequencing techniques to study the expression levels of genes, with each sample typically having tens of thousands of features. The large number of features often necessitates the use of feature selection algorithms to improve the performance of machine learning tasks such as classification, given that many of the observed features may be unrelated to the biological phenomenon of interest. In this way, feature selection algorithms can be used to determine the processes related to a biological mechanism, e.g., the host's immune response to infection. Other downstream benefits of feature selection include data visualization and understanding, reduced storage requirements, and faster computations.

Multi-domain feature extraction addresses the problem of leveraging data from disparate sources and is related to the more general problem of multi-domain learning (MDL) [59]. In this work, we address a special case of MDL that involves the classification of data related to the host's immune response to infection. The host consists of multiple lines of the Collaborative Cross mouse; the pathogen under consideration is Salmonella. The two domains consist of gene expression data collected from liver and spleen tissues. The proposed machine-learning approach has the potential to identify novel biomarkers whose signals are are too weak to be captured by analyzing domains individually. The methodology will be demonstrated by selecting potentially important biomarkers that appear to be amplified in strength by the multi-domain data synthesis for characterizing biological processes that exist simultaneously in different tissues.

In a variety of existing methods, the designs vary from shallow to deep, with the networks optimized for regression, classification, dimensionality reduction, or a combination of multiple tasks, i.e., multi-task learning (MTL) [60]. Most of the feature selection methods fall into the category of MLT methods, with the objective function considering the combination of different goals resulting in better generalization of results. However, the majority of them focus on a single domain feature selection. In this work, we suggest a new MDL method with a multi-task objective (MDL/MTL) function, or, in terms of [61], a multi-domain multi-task (MDMT) method. The MDMT methods have been used widely in Natural Language Processing applications but much less for the analysis of biological data sets.

The feature selection task is frequently performed by the introduction of l_p-norms with $p = 0, 1, 2$, or a combination of norms of weights at intermediate layers, possibly stacked closer to the input. The l_0 is of course appealing but the most expensive to compute [62, 63]. However, in [64], it was shown that l_1-norm-based methods could be quite competitive when applied to biological data while not having the complications of dealing with l_0-norm. When applied to biological tasks, the l_0-norm based methods may require *a priori* knowledge of the size of a subset of biologically significant features, which is certainly not the case for many explorative tasks. Moreover, when the domains are very different, the subsets of important features can be very different as well. Hence it becomes even harder to approximate the number of selected features. At the same time, l_0-norm based methods haven't been battle-tested across domains as much as l_1-norm based methods, intuitively, the discrete distribution can negatively effect the alignment of disparate domains. Considering all the pros and cons, in this work we opt to employ ℓ_1-norm based sparsity promotion.

The contributions of this research include the following: we propose a new sparsity promoting MDMT architecture for feature selection; this approach uses a new masking term restricting the features that contribute to the cost function; we demonstrate the utility of the developed algorithm on gene expression dataset, including liver and spleen domains; we conclude that our MDMT approach allows us to find new features that are significantly discriminative only across two domains,

i.e., are identified when the data is restricted to a single domain. The promise of this approach is an enriched picture of the host immune response that has the potential to lead to a better understanding of the biological process *across tissues*.

The organization of this chapter is as follows: In Section 5.2, we present a review of the related articles, situating our study within the broader context of the field and highlighting key contributions from prior research. Section 5.3 details the method we employed, outlining the design of a neural network behind our chosen approach and the techniques utilized for design. In Section 5.4, we describe the data used in our investigation, elaborating on pre-processing steps and labeled groups. Section 5.5 delves into the computational experimental setup, discussing the training process and criteria for rough-tuning. In Section 5.6, we present the results of our experiments, providing an interpretation and discussion of our findings. Finally, in Section 5.7, we summarize the highlights of the research and contributions, the implications of our results, and potential avenues for future work in this domain.

5.2 Related Work

The majority of the feature selection methods study one domain. In what follows, we survey a variety of different techniques and algorithms, including linear and non-linear methods and the methods exploiting neural networks. In one of the most influential papers, [65], feature selection is cast as a regression task with l_1 regularization of the norm of a discriminative vector. This has become a common approach; see also, [66–69]. The Lasso-type methods fail to capture nonlinear interactions between the features. The non-linear methods developed as the kernelized modifications of the Lasso method showed decent efficiency when applied to biological data [70–72]. Note that there are also some other methods aiming to sparsify a signal in latent dimensions [73, 74].

Deep Feature Selection DFS [62] is one of the first deep neural network algorithms designed specifically for feature selection. DFS employs a one-to-one sparsity layer at the input. The weights on these single connections are penalized with a ℓ_1-norm minimization of the norm of weights of this layer in a spirit of [69]; the resulting non-zero weights in the sparsity layer cor-

respond to the selected features. In [75], autoencoders are proposed for feature selection with a sparsity layer used in a fashion similar to DFS. Now, the ℓ_1-norm of the weights of the sparsity layer is minimized jointly with the reconstruction error. Concrete Autoencoders (CAE) [76] use a concrete selector layer as the first layer in the autoencoder setting based on continuous relaxations of concrete random variables suggested in [77]. A supervised CAE method reported in [78] was apparently susceptible to overfitting with limited data. The FsNet paper [78] addressed this problem by introducing small weight-predictor networks. In terms of design, [78] is one of the closest to the design developed in this work, i.e., our method is based on two neural networks: autoencoder and classifier in latent space, but the approach to sparsification is different, and, most importantly, the method in [78] is designed for one domain, and the implication of extracted features is quite different.

Most of the feature selection methods can be grouped into three broad categories: filter, wrapper, and embedded methods. In filter methods, the features are typically scored, ranked, and thresholded with respect to some classification task using different measures such as correlation and mutual information [79]. Filter methods can be very fast, but the quality of extracted features is poor in terms of robustness and adaptability to different datasets. The wrapper methods [80] are universal methods used on top of any learning algorithm based on a practical heuristic search of a subset of d features in 2^d space, providing better performance for the underlying algorithm. They are universal and capable of obtaining great results, given the large number of samples. For small datasets in high-dimensional space, they tend to overfit, and the NP-hardness of the problem makes the computations prohibitively expensive. In embedded methods, the feature selection process is typically performed concurrently with some learning algorithm. For example, Iterative Feature Removal (IFR) uses the absolute weights of a sparse SVM model as a criterion for selecting features from a high-dimensional biological data set [81]. Our method and the most related works fall into the embedded method category.

5.3 Methodology

The methods described above only address data residing in one domain. The major question that motivated this study was what are the biological features in datasets sampled from different domains that appear to be related to the host immune response to infection only when studied across domains, naturally leading to consideration of the domain alignment task along with the feature selection. With that said and with a general design of the network in mind, we've been looking for the most capable in terms of domain alignment method in application to RNA data. This led us to [82], which is an MDMT method based on a pair of domain-specific variational autoencoders (VAEs) [83] generating aligned embeddings for datasets of very different modalities (singe-cell RNA and Chromatin images), and we adapt this method now enhanced with sparsity promoting optimization constraints for feature selection. In order to find a universal representation across tasks, the MTL methods in deep neural networks [84] either improve the architecture of neural networks or try to find a balance between concurrently trained objectives. Our method benefits from both since our network has shared subnets, and at the same time, we roughly fine-tune the coefficients used in [82] along with the contribution of sparsity promoting loss function. Utilizing both methods is also justified by the results of generalization of unbalanced optimization methods, e.g., [85], [86], [87], indicating that overall they don't outperform the naive approach when all loss functions are weighted with constant scalars.

Our proposed method is based on the network shown in Figure 5.1, implemented in PyTorch and trained with the AdamW optimizer. The cost function is a weighted combination of objective functions associated with the reconstruction, classification, and sparsification tasks. In our settings, we can observe that during the early stages of training, the algorithm learns the shared representations of different domains such that similar samples group together regardless of the domain of origin. In the later stages, the sparsification goal becomes more important, with a relatively higher contribution to the total objective, and this behavior continues until the conflicting tasks reach the balance and no further sparsification is possible without a significant loss in classification. We run the training process multiple times. In the spirit of embedded methods, we treat the resulting mag-

nitude of the weights of sparse layer as indicating the importance of different features. However, for the post-processing we employ the frequency of selected features across all runs as it appears to be a more robust metric for feature importance.

It was mentioned before that our design is developed based on the network suggested in [82] with modifications. The classification task is performed in latent space as before, but the inputs of domain-specific VAEs are sparsified by the shared Sparsification Layer (**SL**), and, naturally, the VAEs are trained to reconstruct only these sparsified inputs. Note that in [82], a primary goal is the alignment of the domains in the latent space coupled with the reconstruction of hyper-dimensional RNA data. In contrast to [82], our main goal is the across domains classification with the sparsification of inputs. The choice of variational modification of autoencoders was dictated by the distribution of latent space provided by this particular modification, allowing further indirect "easy" and relaxed alignment through the shared classifier. Hence, we not only train VAEs solely to reconstruct the sparsified signal, but we also omit the loss function, minimizing the KL divergence across domains from the original design. The resulting network consists of 4 subnets: shared between domains Sparse Layer (**SL**) and **Classifier** subnets, and two domain specific Variational AutoEncoders (**VAE1** and **VAE2**), as depicted in Figure 5.1.

The **SL** is a one-to-one mapping: $x \rightarrow \mathbf{W} \odot x$, with the ℓ_1-norm of weights $\|\mathbf{W}\|_1$ penalty used to promote sparsity.

The **VAE1** and **VAE2** subnets are deep fully-connected networks with the following specifications:

- **Encoder**: 2 linear layers with 1024 nodes with batch normalization and Relu-activations, followed by 1 linear mapping to μ and σ living in 128-dimensional space
- **Decoder**: 2 linear layers with 1024 nodes with batch normalization and Relu-activations, followed by 1 layer mapping to input space

Each **Encoder** performs the mapping of the sparsified input to $\mathbf{R}^{128}$: $\mathbf{W} \odot x \rightarrow \mu$, σ, while the **Decoder** maps distribution in the latent space into the input space: $\mu + \sigma \rightarrow \tilde{x} \in \mathbf{R}^{34861}$, and the

output is further masked by the frozen weights of the sparse layer $\mathbf{W}^* \odot \tilde{x}$ and fed to the MSE loss function of respective VAE.

The **Classifier** is the subnet consisting of 5 linear layers with 1024 nodes with Relu-activations, followed by 1 layer mapping to 2-dimensional space and 1 layer mapping to 1-dimensional space followed by a standard sigmoidal activation function. It is trained to classify embeddings of inputs from both domains in their respective latent spaces.

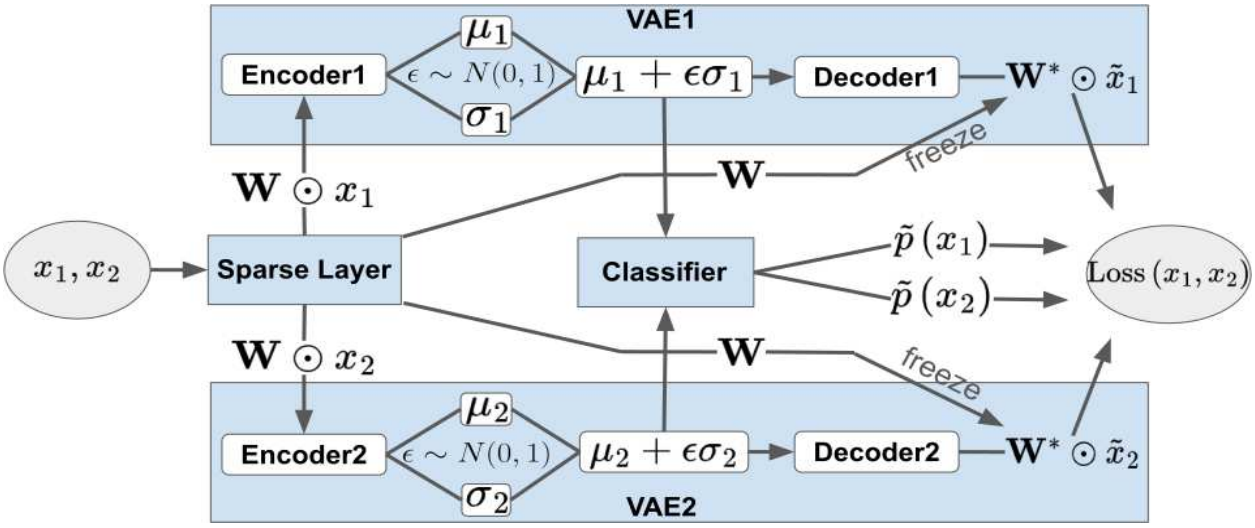


Figure 5.1: Network design.

The overall objective function is a weighted sum of objective functions for different tasks, including reconstruction, normalization, classification, and sparsification, namely,

$$\begin{aligned} \text{Loss}\,(x_1, x_2) =& \alpha \cdot (\text{Loss}_{\text{rec}}\ (x_1) + \text{Loss}_{\text{rec}}\ (x_2)) + \\ & \beta \cdot (\text{Loss}_{\text{var}}\ (x_1) + \text{Loss}_{\text{var}}\ (x_2)) + \\ & \gamma \cdot (\text{Loss}_{\text{class}}\ (x_1) + \text{Loss}_{\text{class}}\ (x_2)) + \\ & \theta \cdot \text{Loss}_{\text{sparse}} \end{aligned} \tag{5.1}$$

where

$$\begin{aligned}
\text{Loss}_{\text{rec}}(x) &= MSE(\mathbf{W} \odot x, \mathbf{W}^* \odot \tilde{x}) \\
\text{Loss}_{var}(x) &= D_{KL}\left(N\left(\mu, \sigma^2 I\right), N(0, I)\right) \\
\text{Loss}_{\text{class}}(x) &= \log \text{Loss}\left(p(x), \tilde{p}(x)\right) \\
\text{Loss}_{\text{sparse}} &= \|\mathbf{W}\|_1
\end{aligned}$$

The block scheme of one training process is given by Algorithm 1. After the rough-tuning of

Algorithm 1: Network Training Algorithm

Input : $\theta, \alpha, \beta, \gamma$ weight of loss functions
$argsOpt1$, $argsOpt2$ parameters of Adam optimizers for VAE's
$argsOptCls$ parameters of Adam optimizers for Classifier
$argsOptSL$ parameters of Adam optimizers for SL

Output : $\mathbf{W}$ - weights of SL

Data: $[x_1, x_2]$ list of pairs of equally seized batches sampled from both datasets
20 000 elements, i.e., epochs of training

Initialize: SL, VAE1, VAE2, Classifier (initialize subnets)
OptSL = AdamW($argsSL$)
Opt1 = AdamW($argsOpt1$)
Opt2 = AdamW($argsOpt2$)
OptCls = AdamW($argsCls$)

for $x_1, x_2 \in [x_1, x_2]$ **do**
  $sp(x_1), sp(x_2) = \mathbf{W} \odot x_1, \mathbf{W} \odot x_2$
  $\tilde{x}_1, \mu_1 + \epsilon\sigma_1 = \text{VAE1}(sp(x_1))$
  $\tilde{x}_2, \mu_2 + \epsilon\sigma_2 = \text{VAE2}(sp(x_2))$
  $\tilde{p}(x_1) = \text{Classifier}(\mu_1 + \epsilon\sigma_1)$
  $\tilde{p}(x_2) = \text{Classifier}(\mu_2 + \epsilon\sigma_2)$
  Calculate Loss as in equation (5.1)
  Backpropagate Loss
  Step all optimizers

the hyper-parameters from [82] along with the sparsity contribution and parameters for optimizers, we set the following hyper-parameters:

- α, β, γ, θ = 10, 10^{-4}, 1, 10^{-4}

- LR's for AdamW optimizer for VAE's, SL and Classifier = 10^{-4}
- all other parameters of AdamW optimizer are set to PyTorch default

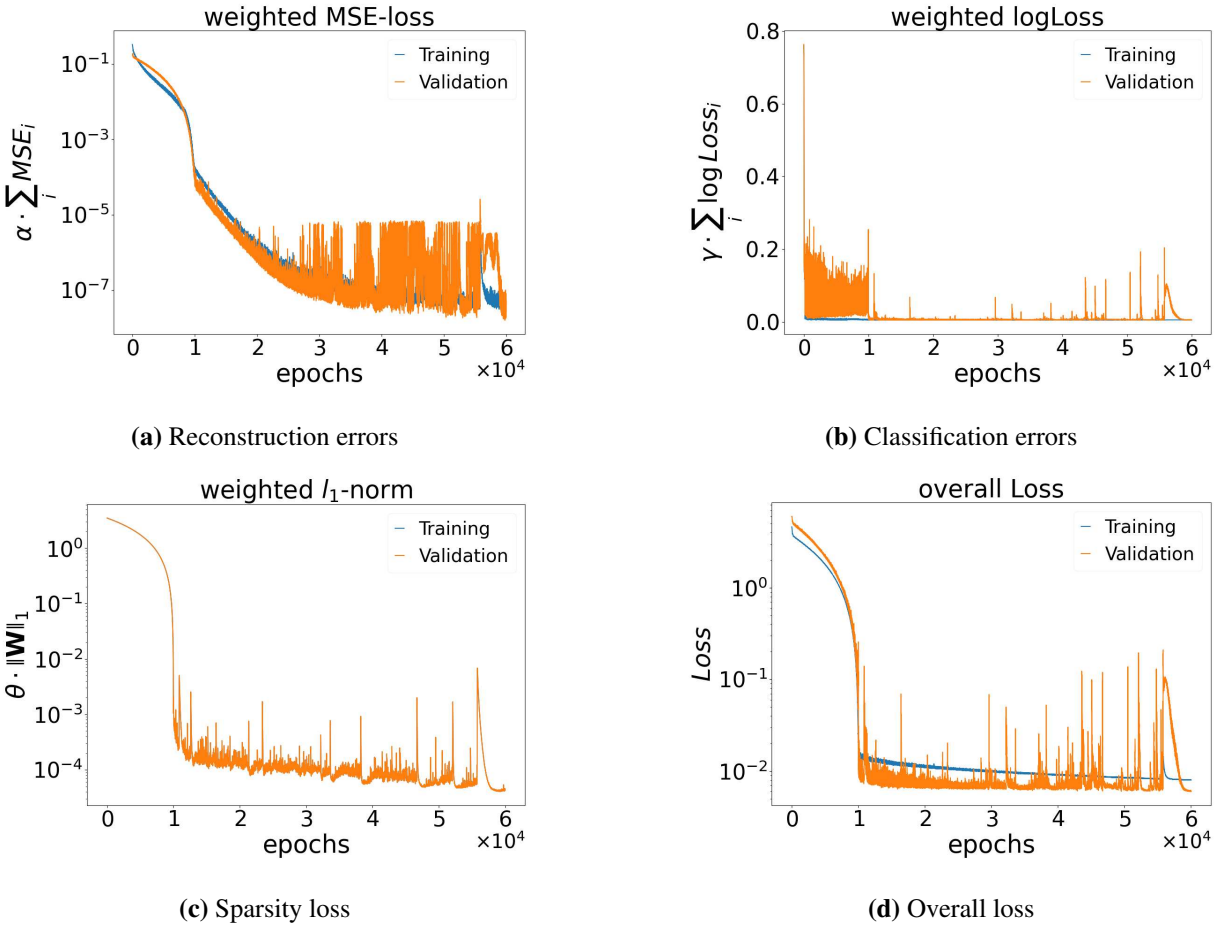
5.4 Data

The data consists of mice bulk RNA sequences extracted from two different tissues: spleen and liver, used as two different domains for the purposes of this research. The mice that were exposed to Salmonella infection were monitored and categorized by health status as tolerant, resistant, susceptible, or delayed susceptible, the latter two being related to strains unifying the mice who died within 1 or 3 weeks, respectively. In all our experiments we combine these latter two groups into one susceptible group. With this new labeling, we have 31 and 9 tolerant samples, 27 and 7 resistant samples, and 90 and 53 susceptible samples for spleen and liver domains, respectively, for all infected mice. Also, the data includes control samples representing the mice who had never been exposed to infection labeled as "never infected". This group accounts for 104 samples, with 93 samples from the spleen and 11 samples from the liver. The phenotypes of these samples are determined based on their genetic strains. Initial bulk RNA dataset was TMM-normalized, the outliers and duplicates have been detected and dropped out. Finally, the domain-specific data, i.e., combined RNA data for samples from the spleen and combined RNA data for samples from the liver, have been z-scored for each domain separately and filtered for common across tissues genes in all feature selection algorithms, resulting in data samples consisting of 34,861 genes, i.e., the dimension of the input space.

5.5 Experiment

We consider three distinct types of experiments. In the first type, the goal is to extract a small subset of features that discriminate among the phenotypes susceptible and tolerant. The second type extracts features discriminating among the phenotypes susceptible and resistant. The third type extracts the features discriminating between infected and never infected mice. The exact Python code with the training models and post-processing utilities is available at [88]. With the

use of Ray package [89], the experiment was run on 16 V100 GPUs in a multiprocessing mode for 10 different random samplings of 85% of data, with 90 different weights initializations for each, summing up to 900 runs. You can see the typical evolution of the training process, including 60,000 epochs for all three experiments in figures Figure 5.2, Figure 5.3, Figure 5.4.



(a) Reconstruction errors

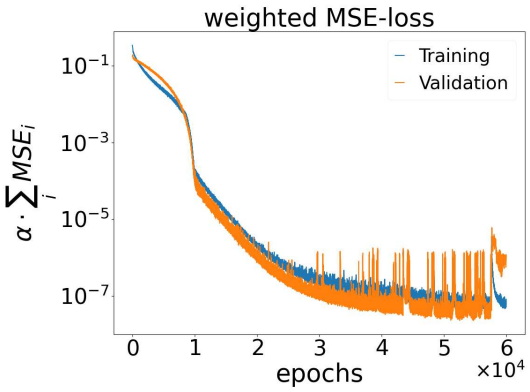
(b) Classification errors

(c) Sparsity loss

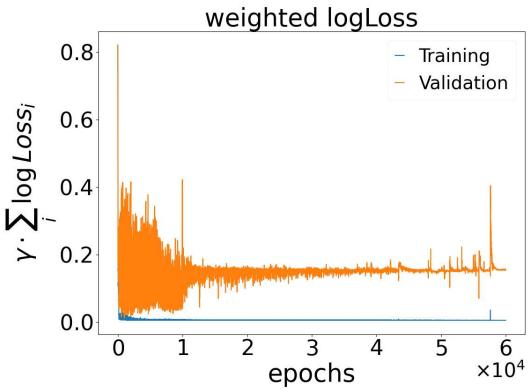
(d) Overall loss

Figure 5.2: Weighted components of overall losses for phenotypes tolerant versus susceptible across domains experiment: (a) reconstruction errors, (b) classification errors, (c) sparsity loss, (d) overall loss.

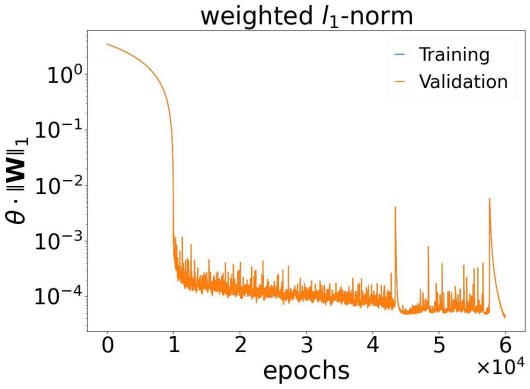
The number of epochs was fixed at 20,000 based on the indication of the flattening of the sparsity curves as long as it doesn't affect the accuracy of classification and the reconstruction loss is relatively small. Again, the reconstruction is not the primary focus of this method, i.e., it wasn't the primary task to consider when deciding on the number of epochs, especially since the required for the across domains classification alignment in the latent space was typically achieved even after



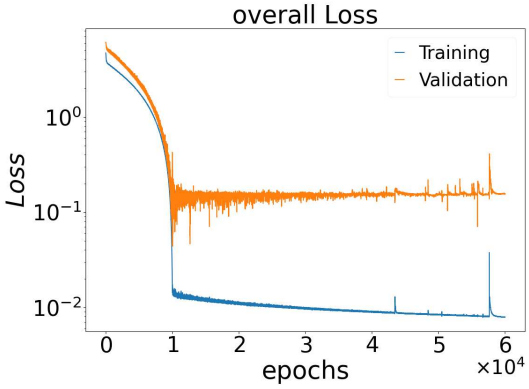
(a) Reconstruction errors



(b) Classification errors

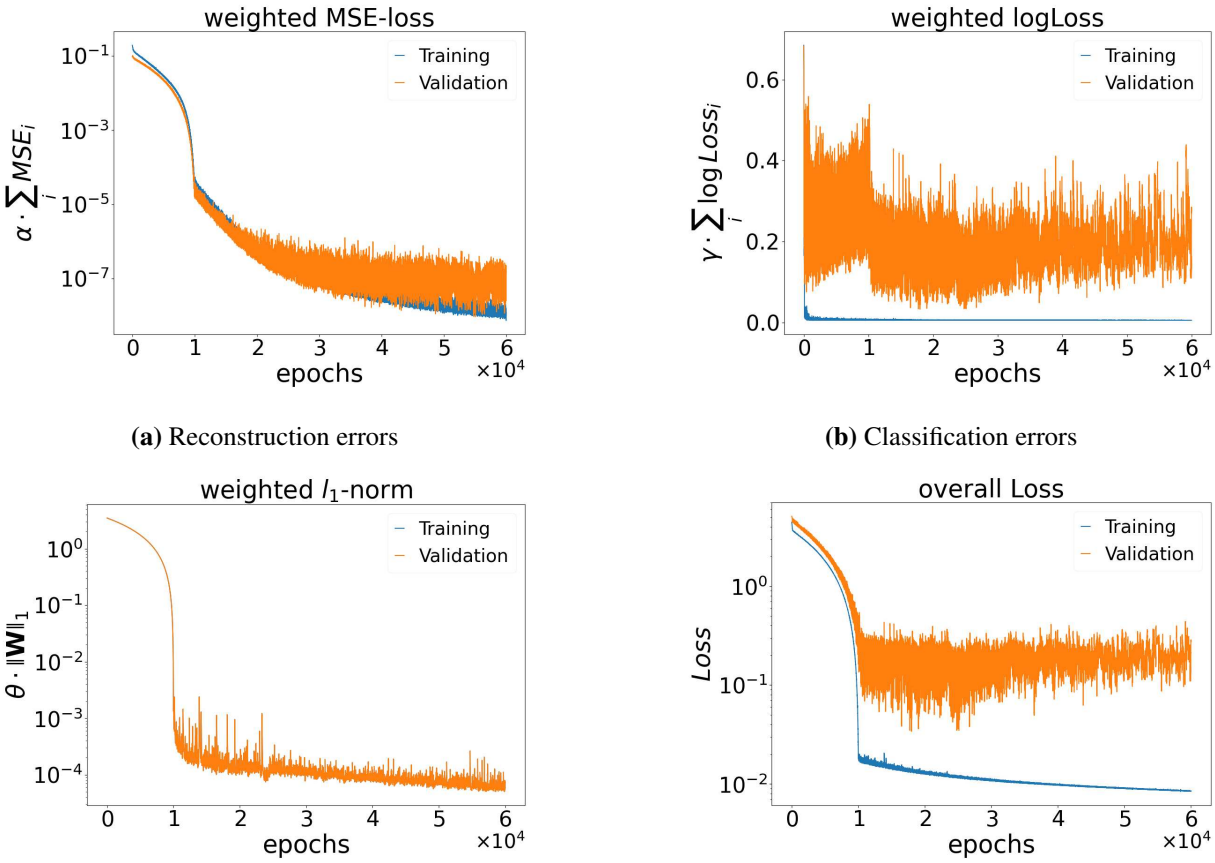


(c) Sparsity loss



(d) Overall loss

Figure 5.3: Weighted components of overall losses for phenotypes resistant versus susceptible across domains experiments: (a) reconstruction error, (b) classification error, (c) sparsity loss, (d) overall loss.



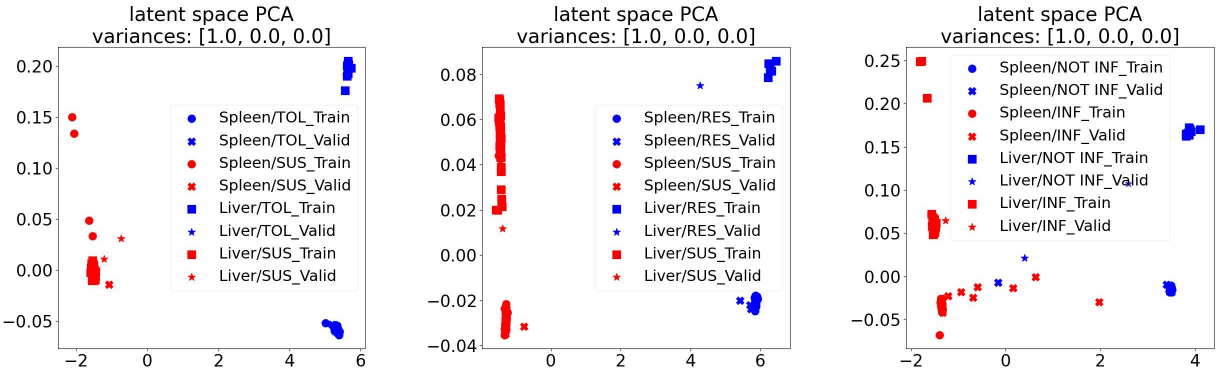
(a) Reconstruction errors

(b) Classification errors

(c) Sparsity loss

(d) Overall loss

Figure 5.4: Weighted components of overall losses for infected mice versus never infected mice across domains experiment: (a) reconstruction error, (b) classification error, (c) sparsity loss, (d) overall loss.



(a) TOL - SUS

(b) RES - SUS

(c) INF - NOT INF

Figure 5.5: Latent space PCA's for all three across domains experiments: (a) phenotypes tolerant (TOL) versus susceptible (SUS), (b) phenotypes resistant (RES) versus susceptible (SUS), (c) infected (INF) mice versus never infected mice (NOT INF).

20,000 epochs, see the Figure 5.5. The visualization of loss indicates that further sparsification slightly decreases the classification accuracy for two out of three experiments, by limiting the number of epochs we also prevent this long-run negative effect. The PCA images for all figures represent the PCA of combined representations of both domains in their bottleneck layers and indicate a proper clustering and separability across domains, i.e., good alignment in latent space.

5.6 Results

The post-processing of the sparsity layer weights was conducted in a same way for all 3 experiments. Firstly, all the weights across 900 runs were aggregated and normalized, and the Elbow Method was applied to find a threshold. Later, for each run, the weights below the threshold were set to zero. At the next step, we calculated the frequencies of features appearing in subsets of features with non-zero weights across all runs. The resulting distributions of frequencies are shown in Figure 5.6, along with the resulting number of features selected in two consecutive steps by the Elbow Method.

Importantly, in addition to the across domains learning, we also selected features for each domain separately for all three experiments. This allows us to evaluate the distinct characteristics of single domain and multi-domain alignment for feature extraction. In Figure 5.7 panels (a), (b), and (c), we can see the distribution of features for all three across domains experiments grouped by overlapping with features selected in one-domain experiments. "Both" features are a subset of these features that also appear in both the spleen and liver domains. "None" refers to the features that are the features that only appear in across domains results. "Spleen" are the features appearing in across domains and spleen domain results but not in the liver domain results. Finally, "Liver" features appear in across domains and liver domain results but not in the spleen domain results. We can see that for all the experiments, apart from the features captured from domain-specific experiments, some new highly weighted features were captured in across domains experiments (the blue line denoting "None"). These features that are only present in the across domains experiment re-

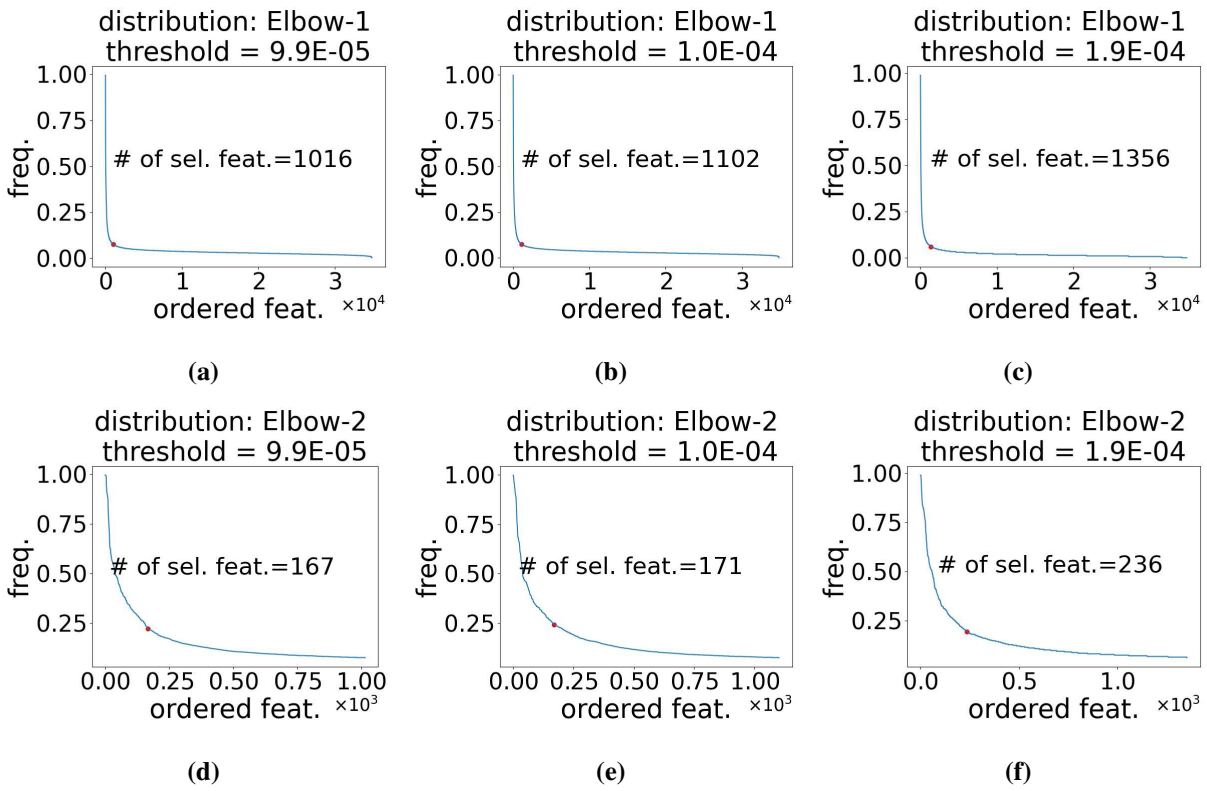


Figure 5.6: Results from the across domains (MDMT) experiments. The selected features are ordered by frequency for 3 experiments: panels (a) and (d) correspond to the phenotypes tolerant versus susceptible; panels (b) and (e) correspond to resistant versus susceptible; panels (c) and (f) correspond to infected versus never infected.

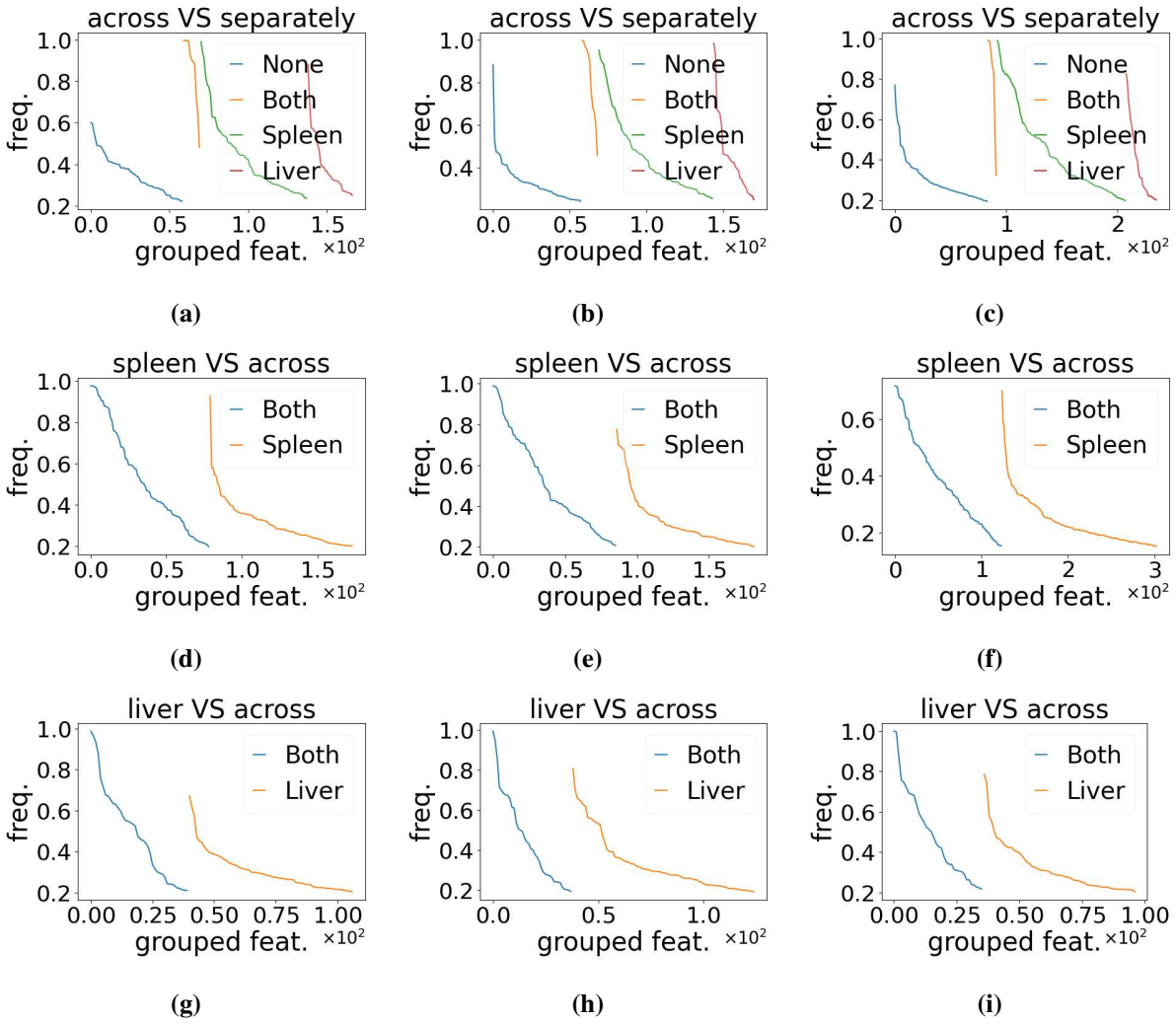


Figure 5.7: Grouped by overlapping distribution of features for all experiments. Columns are associated with experiments, i.e., 1st - tolerant versus susceptible, 2nd - resistant versus susceptible, 3rd - infected versus never infected. (a),(b),(c) distributions of features extracted in across domains experiments; (d),(e),(f) features extracted in separate spleen experiment, features are grouped by overlapping with features extracted in respective across domains experiment: "Both" - overlapping, "Spleen" - not overlapping; (g),(h),(i) features extracted in separate liver experiment, features are grouped by overlapping with features extracted in respective across domains experiment: "Both" - overlapping, "Liver" - not overlapping.

flect the new information being captured by the proposed method. These correspond to biomarkers that we suspect have a potentially unique role in the host response to infection.

In Figure 5.7, panels (d),(g),(e),(h),(f), and (i) show the distributions of features extracted in one domain experiments grouped by overlapping with respective across domains results. Even though the majority of the most discriminative features extracted in separate experiments are also extracted in across domains experiments, some highly-weighted features extracted from one domain experiments are apparently not captured in respective across domains experiments.These results might indicate that the improvement in robustness is needed, but at the same time they may contain biological insights about the across domains importance of some features that couldn't be obtained from studying only one domain for Salmonella infection or about the difference in manifestations of infection in different tissues. Based on the results from additional experiments with other datasets most likely the latter is true.

Returning to Figure 5.5, we see the results of the embedding of the sparsified input in a two-dimensional latent space using the features found in the across domains architecture for all three experiments. We observe that the tolerant and resistant versus susceptible experiments give excellent classification on test data in the latent space. In contrast, infected versus control mice are not as easily discriminated.

5.7 Conclusions

In this chapter, we propose a novel architecture for multi-domain, multi-task feature selection. This area of research has a relatively small literature, possibly because of the complexity associated with simultaneously exploring multiple incommensurate measurement domains. The proposed approach leverages prior art in multi-domain learning while adding a masked feature selection approach that serves to identify biologically relevant aspects of the host immune response to infection. We demonstrate that the demands of the MDMT problem formulation can be successfully addressed with the proposed architecture. Further, the application of the approach leads to the discovery of novel biomarkers whose signals appear to be amplified by the multi-domain approach;

indeed, a fraction of these biomarkers do not appear in either single experiment. Hence, this approach holds the promise of generating new biological insights that might go undetected using single domain methodologies. Additionally, we observed the MDMT features provide excellent classification results between susceptible and tolerant or resistant phenotypes.

There are several possible modifications to our method. In this method, the optimization problem descends in the direction of the gradient of weighted objectives. There is growing evidence that isolating descent directions to improve individual cost functions may lead to improved solutions. Additionally, alternative data reduction and reconstruction mappings could be explored reflecting recent developments in deep neural networks, including graph convolutional neural networks or transformers. This preliminary work is focused on algorithm development; we propose to explore the biological ramifications of the biomarkers in future work.

We believe further development in these directions could lead to even more efficient methods and provide biologists with a new perspective on understanding the evolution of infections in tissues.

Chapter 6

Conclusion and Future Work

This dissertation has explored two significant areas within applied mathematics: the development of the Schubert Variety of Best Fit (SVBF) approach for analyzing sets of datasets and the introduction of a novel architecture for multi-domain, multi-task (MDMT) feature selection.

6.1 Advancements in Schubert Variety of Best Fit

In Chapter 2, we introduced the SVBF-Node, an innovative computational unit designed to analyze sets of datasets using the geometric framework of Schubert varieties and Grassmann manifolds. By formulating two optimization problems and employing the Adam optimizer, a gradient descent-based method, we developed a network capable of minimizing a specific objective function tailored to our needs. The SVBF-Node generates new types of embeddings in the Grassmannian $\mathrm{Gr}(l, n)$, where l and n represent the dimensionalities of the samples and the ambient space, respectively. These embeddings are not only valuable on their own but can also be integrated into larger networks, enhancing the capability to process complex data structures.

Building upon this foundation, Chapter 3 presented three distinct algorithms for data classification, all leveraging the SVBF framework. Algorithm I utilized a single solution KK with a Frobenius norm for distance measurement but showed limited performance. In contrast, Algorithm II, which learned two SVBFs to model different classes, outperformed the others by accurately addressing complex data structures with high within-class variance. Algorithm III introduced auxiliary features for classification and demonstrated how the SVBF-Node can function as an abstract computational unit within neural networks. These algorithms revealed that an optimal dimension exists for the representative subspace beyond which classification accuracy does not improve. This finding underscores the SVBF approach's ability to determine the intrinsic complexity of datasets, offering a valuable tool for dimensionality reduction and feature selection in machine learning tasks.

In Chapter 4, we extend the application of SVBF methods to clustering tasks. By exploiting the geometry of the real Grassmann manifold, we developed the SVBF-LBG method to find optimal representatives or prototypes for collections of vector spaces. Through extensive experiments with synthetic data, the MNIST dataset, spectral data from the Indian Pines dataset, and video action sequences, we demonstrated that SVBF-LBG outperforms leading methods in the literature.

Overall, the results of Chapter 3 and Chapter 4 allow one to conclude that the classification and clustering methods based on SVBF-Node provide a more granular approach by forming representatives that closely intersect with cluster members in at least one dimension. This characteristic allows for capturing intricate geometries within data, leading to improved classification and clustering performance, especially in complex datasets like remote sensing data and video action recognition. In other words, SVBF shines when the datasets are difficult.

The collective contributions of this dissertation lie in advancing mathematical methodologies for data analysis and demonstrating their applicability across diverse domains. The development of the SVBF-Node as an abstract unit that can be integrated into other algorithms and frameworks, including neural networks, enriches the toolkit available for machine learning tasks, particularly in the classification and clustering of complex datasets. By harnessing the geometric properties of Grassmann manifolds, we have provided new perspectives on dimensionality reduction and data representation.

Despite the promising results, the current implementations of the SVBF methods have limitations. SVBF methods apparently have a *working* dimension that needs to be discovered on a case-by-case basis. The convergence criterion in current realization is based on a fixed number of iterations without optimization, and the computational demands require GPU acceleration. Future work aims to address these limitations by considering multiple intersections, optimizing convergence criteria, and improving computational efficiency.

6.2 Multi-Domain Multi-Task Feature Selection

In Chapter 5, we shifted focus to the second major contribution of this dissertation: a novel architecture for MDMT feature selection. Recognizing the complexity of simultaneously analyzing multiple incommensurate measurement domains, we proposed an approach that integrates prior art in multi-domain learning with a masked feature selection mechanism. This architecture is designed to identify biologically relevant aspects of host immune responses to infections.

By applying this method to biological data, we successfully discovered novel biomarkers exhibiting amplified signals in the multi-domain context. Remarkably, some of these biomarkers were not detectable in single domain analyses, highlighting the approach's ability to unveil insights that might otherwise remain hidden. The MDMT feature selection method also achieved excellent classification results between susceptible and tolerant or resistant phenotypes, demonstrating the method's practical utility in biomedical research.

The MDMT feature selection architecture effectively handles multi-domain, multi-task problems, addressing a gap in the literature. Its success in uncovering novel biomarkers underscores the importance of integrated analytical approaches in biomedical research and beyond.

The proposed MDMT architecture in its current stage primarily focuses on algorithm development rather than exhaustive biological validation. Additionally, incorporating recent developments in deep learning—such as graph convolutional neural networks or transformers—could enhance data reduction and reconstruction mappings. Future research will also delve deeper into the biological implications of the identified biomarkers. By collaborating with biologists, we aim to understand the roles these biomarkers play in infection progression and host immune responses, potentially contributing to the development of new therapeutic strategies.

Bibliography

[1] Jihun Hamm and Daniel D Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 376–383, 2008.

[2] J Ross Beveridge, Bruce A Draper, Jen-Mei Chang, Michael Kirby, Holger Kley, and Chris Peterson. Principal angles separate subject illumination spaces in ydb and cmu-pie. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):351–363, 2008.

[3] Zhiwu Huang, Ruiping Wang, Shiguang Shan, and Xilin Chen. Projection metric learning on grassmann manifold with application to video based face recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 140–149, 2015.

[4] Lincon S. Souza, Bernardo B. Gatto, Jing-Hao Xue, and Kazuhiro Fukui. Enhanced grassmann discriminant analysis with randomized time warping for motion recognition. *Pattern Recognition*, 97:107028, 2020.

[5] Anuj Srivastava and Eric Klassen. Bayesian and geometric subspace tracking. *Advances in Applied Probability*, 36(1):43–56, 2004.

[6] Mehrtash T Harandi, Conrad Sanderson, Sareh Shirazi, and Brian C Lovell. Graph embedding discriminant analysis on grassmannian manifolds for improved image set matching. In *IEEE conference on Computer Vision and Pattern Recognition 2011*, pages 2705–2712. IEEE, 2011.

[7] Dong Wei, Xiaobo Shen, Quansen Sun, Xizhan Gao, and Zhenwen Ren. Neighborhood preserving embedding on grassmann manifold for image-set analysis. *Pattern Recognition*, 122:108335, 2022.

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

[9] Xiaofeng Ma, Michael Kirby, and Chris Peterson. The flag manifold as a tool for analyzing and comparing sets of data sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 4185–4194, October 2021.

[10] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019.

[11] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), 2018.

[12] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *ArXiv: Machine Learning*, abs/1702.08608, 2017.

[13] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[14] John Hertz, Anders Krogh, Richard G Palmer, and Heinz Horner. *Introduction to the Theory of Neural Computation*. American Institute of Physics, 1991.

[15] Donald O Hebb. The first stage of perception: growth of the assembly. *The Organization of Behavior*, 4(60):78–60, 1949.

[16] D. Rosenblatt, A. Lelu, and A. Georgel. Learning in a single pass: A neural model for principal component analysis and linear regression. In *Proceedings of the IEE int. Conf. on Artificial Neural Networks*, pages 252–256, London, U.K., find year.

[17] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.*, 37(2):233–243, 1991.

[18] E. Oja. Principal components, minor components and linear neural networks. *Neural Networks*, 5:927–935, 1992.

[19] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[20] Tomojit Ghosh and Michael Kirby. Supervised dimensionality reduction and visualization using centroid-encoder. *J. Mach. Learn. Res.*, 23:20–1, 2022.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[22] M. Kirby and R. Miranda. Circular nodes in neural networks. *Neural Computation*, 8(2):390–402, 1996.

[23] D. Hundley, M. Kirby, and R. Miranda. Spherical nodes in neural networks with applications. In *Intelligent Engineering Through Artificial Neural Networks*, volume 5, pages 27–32, New York, 1995. The American Society of Mechanical Engineers.

[24] D.S. Broomhead and M. Kirby. A new approach for dimensionality reduction: Theory and algorithms. *SIAM J. of Applied Mathematics*, 60(6):2114–2142, 2000.

[25] D.S. Broomhead and M. Kirby. The Whitney reduction network: a method for computing autoassociative graphs. *Neural Computation*, 13:2595–2616, 2001.

[26] M. Anderle, D. Hundley, and M. Kirby. The bilipschitz criterion for mapping design in data analysis. *Intelligent Data Analysis*, 6(1):85–104, 2002.

[27] Henry Kvinge, Elin Farnell, Michael Kirby, and Chris Peterson. A gpu-oriented algorithm design for secant-based dimensionality reduction. In *2018 17th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 69–76. IEEE, 2018.

[28] Henry Kvinge, Elin Farnell, Michael Kirby, and Chris Peterson. Too many secants: a hierarchical approach to secant-based dimensionality reduction on large data sets. In *2018 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2018.

[29] Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). *IEEE transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, 2005.

[30] Tim Marrinan, J Ross Beveridge, Bruce Draper, Michael Kirby, and Chris Peterson. Flag manifolds for the characterization of geometric structure in large data sets. In *Numerical Mathematics and Advanced Applications-ENUMATH 2013*, pages 457–465. Springer, 2015.

[31] Tim Marrinan, Bruce Draper, J Ross Beveridge, Michael Kirby, and Chris Peterson. Finding the subspace mean or median to fit your need. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1082–1089. IEEE, 2014.

[32] Nathan Mankovich, Emily J King, Chris Peterson, and Michael Kirby. The flag median and flagirls. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10339–10347, 2022.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations*, Conference Track Proceedings, San Diego, CA, USA, May 2015.

[34] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541, 1977.

[35] Bruce Draper, Michael Kirby, Justin Marks, Tim Marrinan, and Chris Peterson. A flag representation for finite collections of subspaces of mixed dimensions. *Linear Algebra and its Applications*, 451:15–32, 2014.

[36] Thomas Fletcher, Suresh Venkatasubramanian, and Sarang Joshi. The geometric median on riemannian manifolds with application to robust atlas estimation. *NeuroImage*, 45(1, Supplement 1):S143–S152, 2009.

[37] Khurrum Aftab, Richard Hartley, and Jochen Trumpf. Generalized weiszfeld algorithms for lq optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(4):728–745, April 2015.

[38] Sofya Chepushtanova and Michael Kirby. Sparse grassmannian embeddings for hyperspectral data representation and classification. *IEEE Geoscience and Remote Sensing Letters*, 14(3):434–438, 2017.

[39] Tim Marrinan, P.-A. Absil, and Nicolas Gillis. On a minimum enclosing ball of a collection of linear subspaces. *Linear Algebra and its Applications*, 625:248–278, 2021.

[40] Daniel J. Bates, Brent R. Davis, Michael Kirby, Justin Marks, and Chris Peterson. The max-length-vector line of best fit to a set of vector subspaces and an optimization problem over a set of hyperellipsoids. *Numerical linear algebra with applications*, 22(3):453–464, 2015.

[41] Moody T. Chu and J. Loren Watterson. On a multivariate eigenvalue problem, part I: algebraic theory and a power method. *SIAM J. Sci. Comput.*, 14(5):1089–1106, 1993.

[42] Justin D Marks. *Mean Variants on Matrix Manifolds*. PhD thesis, Colorado State University, 2012.

[43] Sofya Chepushtanova and Michael Kirby. Classification of hyperspectral imagery on embedded grassmannians. In *2014 6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4, 2014.

[44] Marion Baumgardner, Larry Biehl, and David Landgrebe. 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3, 2015.

[45] Yoseph Linde, Andres Buzo, and Robert Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[46] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, 1967.

[47] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[48] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[49] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004.

[50] Maryam Abdolali and Nicolas Gillis. Beyond linear subspace clustering: A comparative study of nonlinear manifold clustering algorithms. *Computer Science Review*, 42:100435, 2021.

[51] Sadeep Jayasumana, Mathieu Salzmann, Hongdong li, and Mehrtash Harandi. A framework for shape analysis via hilbert space embedding. In *Proceedings of the IEEE International Conference on Computer Vision*, 12 2013.

[52] Mehrtash T. Harandi, Mathieu Salzmann, Sadeep Jayasumana, Richard Hartley, and Hongdong Li. Expanding the family of grassmannian kernels: An embedding perspective. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *European Conference Computer Vision (ECCV)*, pages 408–423, Cham, 2014. Springer International Publishing.

[53] Kun Zhao, Azadeh Alavi, Arnold Wiliem, and Brian C. Lovell. Efficient clustering on riemannian manifolds: A kernelised random projection approach. *Pattern Recognition*, 51:333–345, 2016.

[54] Dong Wei, Xiaobo Shen, Quansen Sun, Xizhan Gao, and Wenzhu Yan. Prototype learning and collaborative representation using grassmann manifolds for image set classification. *Pattern Recognition*, 100:107123, 2020.

[55] Chendra Hadi Suryanto, Hiroto Saigo, and Kazuhiro Fukui. Protein clustering on a grassmann manifold. In Tetsuo Shibuya, Hisashi Kashima, Jun Sese, and Shandar Ahmad, editors, *Pattern Recognition in Bioinformatics*, pages 71–81, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[56] Shannon Stiverson, Michael Kirby, and Chris Peterson. Subspace quantization on the grassmannian. In *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization*, pages 251–260, Cham, 2019. Springer International Publishing.

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

[58] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos "in the wild". In *IEEE conference on Computer Vision and Pattern Recognition*, pages 1996–2003, 2009.

[59] Mahesh Joshi, Mark Dredze, William W. Cohen, and Carolyn Rosé. Multi-domain learning: When do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1302–1312, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[60] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[61] Yongxin Yang and Timothy M. Hospedales. A unified perspective on multi-domain and multi-task learning. *ArXiv*, abs/1412.7489, 2014.

[62] Yifeng Li, Julie Chih-Yu Chen, and Wyeth Wasserman. Deep feature selection: Theory and application to identify enhancers and promoters. *Journal of Computational Biology*, pages 322–336, 05 2016.

[63] Jean Feng and Noah Simon. Sparse-input neural networks for high-dimensional nonparametric regression and classification, 2019.

[64] Tomojit Ghosh, Karim Salta, and Michael Kirby. Sparse linear centroid-encoder: A biomarker selection tool for high dimensional biological data. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 3012–3019, 12 2023.

[65] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[66] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

[67] Nicolai Meinshausen. Relaxed lasso. *Computational Statistics and Data Analysis*, 52(1):374–393, 2007.

[68] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, 67(1):91–108, 2005.

[69] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67(2):301–320, 2005.

[70] Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P. Xing, and Masashi Sugiyama. High-Dimensional Feature Selection by Feature-Wise Kernelized Lasso. *Neural Computation*, 26(1):185–207, 01 2014.

[71] Han Liu, Larry Wasserman, and John Lafferty. Nonparametric regression and classification with joint sparsity constraints. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

[72] S Shevade and S Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics (Oxford, England)*, 19:2246–53, 12 2003.

[73] Antoni B. Chan, Nuno Vasconcelos, and Gert R. G. Lanckriet. Direct convex relaxations of sparse svm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 145–153, New York, NY, USA, 2007. Association for Computing Machinery.

[74] Prudhvi Gurram and Heesung Kwon. Optimal sparse kernel learning in the empirical kernel feature space for hyperspectral classification. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 7:1217–1226, 04 2014.

[75] Kai Han, Yunhe Wang, Chao Zhang, C. Li, and Chao Xu. Autoencoder inspired unsupervised feature selection. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2941–2945, 2017.

[76] Muhammed Fatih Balın, Abubakar Abid, and James Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 444–453. PMLR, 09–15 Jun 2019.

[77] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *ArXiv*, abs/1611.00712, 2016.

[78] Dinesh Singh and Makoto Yamada. Fsnet: Feature selection network on high-dimensional biological data. *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020.

[79] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[80] George John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. *In International Conference on Machine Learning*, 129, 02 1998.

[81] Stephen O'Hara, Kun Wang, Richard Slayden, Alan Schenkel, Greg Huber, Corey O'Hern, Mark Shattuck, and Michael Kirby. Iterative feature removal yields highly discriminative pathways. *BMC genomics*, 14:832, 11 2013.

[82] Karren Yang, Anastasiya Belyaeva, Saradha Venkatachalapathy, Karthik Damodaran, Abigail Katcoff, Adityanarayanan Radhakrishnan, G. Shivashankar, and Caroline Uhler. Multi-domain translation between single-cell imaging and sequencing data using autoencoders. *Nature Communications*, 12, 01 2021.

[83] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ArXiv*, abs/1312.6114, 2013.

[84] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *ArXiv*, abs/1706.05098, 2017.

[85] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR, 10–15 Jul 2018.

[86] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[87] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[88] Karim Salta. *https://github.com/kkarimov/iccs2024*.

[89] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications, 2018.