DISSERTATION

SUBSPACE AND NETWORK AVERAGING FOR COMPUTER VISION AND BIOINFORMATICS

Submitted by

Nathan J. Mankovich

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2023

Doctoral Committee:

Advisor: Michael Kirby

Chris Peterson
Emily King
Charles Anderson

ABSTRACT

SUBSPACE AND NETWORK AVERAGING FOR COMPUTER VISION AND BIOINFORMATICS

Finding a central prototype (a.k.a. average) from a cluster of points in high dimensional space has broad applications to complex problems like action clustering in computer vision or gene co-expression module representation in bioinformatics. A central prototype of a set of points may be cast as the solution to an optimization problem that either minimizes distance or maximizes similarity between the prototype and each point in the cluster. In this dissertation we offer four novel prototypes for a cluster of points: the flag median, maximally correlated flag, cluster expression vector and eigengene subspace. We will formalize the flag median and the maximally correlated flag using subspace representations for data, specifically the Grassmann and flag manifolds. In addition to introducing these prototypes, we will derive a novel algorithm which can be used to calculate subspace prototypes: FlagIRLS. The third and fourth prototypes, the cluster expression vector and eigengene subspace, are inspired by problems involving gene cluster (e.g., pathway or module) representations. The cluster expression vector leverages connections within networks of genes whereas the eigengene subspace is computed using Principal Component Analysis (PCA). In this work we will explore the theoretical underpinnings of these prototypes, find algorithms to compute and them to computer vision and biological data sets.

ACKNOWLEDGEMENTS

I would like to acknowledge the following collaborators by the chapter they contributed to. Thank you to Michael Kirby for contributing to every chapter in this dissertation. Thanks to Chris Peterson and Emily King for their work on Subspace Prototypes: The Flag Median and the Maximally Correlated Flag and FlagIRLS: An Algorithm for Calculating Subspace Prototypes. Thanks to Erik Kehoe and Amy Peterson for their contributions to Pathway Expression Analysis. Finally, thanks to Helene Andrews-Polymenis and David Threadgill for contributing the Salmonella dataset in Module Representatives for Refining Gene Co-Expression Modules.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

Motivated by problems in computer vision and multi-omics biological datasets, we develop the mathematical foundation for calculating four different central prototypes of a cluster of points. Optimization problems provide us with a way of mathematically formalizing such prototypes.

We formulate the four novel central prototypes as minimizers of optimization problems over n-dimensional Euclidean space, $\mathbb{R}^n$, in Table 1.1. Suppose $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\} \in \mathbb{R}^n$ form a cluster of points. We define the following matrices and vectors:

- $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$
- $\hat{\mathbf{X}}$ is $\mathbf{X}$ with mean centered rows
- $\mathbf{u}_i = \mathbf{x}/\|\mathbf{x}_i\|_2$
- $c_i \in \mathbb{R}$ is a notion of the centrality of node i in a network of p nodes.

Central Prototype	Optimization Problem
Flag Median	$\min_{\mathbf{y}^T\mathbf{y}=1} \sum_{i=1}^{p} \sqrt{1-\mathbf{y}^T\mathbf{u}_i\mathbf{u}_i^T\mathbf{y}}$
Maximally Correlated Flag	$\max_{\mathbf{y}^T\mathbf{y}=1} \sum_{i=1}^{p} \lvert\mathbf{y}^T\mathbf{u}_i\rvert$
Eigengene (Subspace)	$\min_{\mathbf{y}^T\mathbf{y}=1} \mathbf{y}^T\hat{\mathbf{X}}\hat{\mathbf{X}}^T\mathbf{y}$
Cluster Expression Vector	$\min_{\mathbf{y}} \sum_{i=1}^{p} c_i\|\mathbf{y}-\mathbf{x}_i\|_2$

We will use the Grassmann manifold to generalize the flag median and maximally correlated flag to be central subspace prototypes of collections of subspaces in Chapter 2. We find that the flag median and other subspace prototypes can be approximated by the FlagIRLS algorithm which is explored in Chapter 3. Code for the examples in the first two chapters can be found at https://github.com/nmank/PhD_Code. The cluster expression vector is introduced in Chapter 4 as a method for re-envisioning biological transcriptomics analysis using pathways rather than genes. See https://github.com/nmank/PathwayAnalysis for the pathway expression analysis code. Known biological clusters of genes are called pathways and clusters of genes detected by clustering algorithms are called co-expression modules. So, depending on the context, the cluster expression vector will be referred to as either a pathway or module expression vector for the rest of this dissertation. Finally, in Chapter 5, we will use the eigengene subspace, a generalization of the eigengene, along with the other subspace prototypes and module expression to refine gene co-expression modules. Code for this project can be found at https://github.com/nmank/ModuleRefinement.

This chapter will provide a review of the notation used in the dissertation, some mathematical background for working on the Grassmann manifold, and a short introduction to frame theory.

1.2 Mathematical Background

We will now provide the mathematical notation and background for working with the Grassmann manifold and frame theory.

1.2.1 Notation

- $\mathbb{R}^n$ is the set of all n-dimensional real vectors
- $\mathbf{x} \in \mathbb{R}^n$ is a n-dimensional real vector with entries $x_1, x_2, \ldots, x_n$
- $\mathrm{Gr}(k,n)$ is the Grassmann manifold of k-dimensional subspaces of $\mathbb{R}^n$

- $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ is a matrix with columns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$
- $\mathbf{X}_{i,j}$ is the entry in the ith row and jth column of the matrix $\mathbf{X}$
- Orthogonal group $O(n) = \{\mathbf{X} \in \mathbb{R}^{n \times n} : \mathbf{X}^T\mathbf{X} = \mathbf{I}\}$
- $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ is the two-norm of $\mathbf{x}$

1.2.2 Introduction to the Grassmannian

The *Grassmann manifold* (a.k.a. "the Grassmannian"), denoted $\mathrm{Gr}(k, n)$, is the manifold whose points parameterize the k dimensional subspaces of $\mathbb{R}^n$ [1]. In this work will represent a point in $\mathrm{Gr}(k, n)$ using a tall $n \times k$ real matrix $\mathbf{X}$ with orthonormal columns. An equivalence class of points on $\mathrm{Gr}(k, n)$ determined by $\mathbf{X}$ is the column space of $\mathbf{X}$ and is denoted $[\mathbf{X}]$. Thus if $\mathbf{X}$ and $\mathbf{Y}$ have the same column space, then they determine the same point $[\mathbf{X}] = [\mathbf{Y}]$ on $\mathrm{Gr}(k, n)$.

In order to allow more flexibility in our generalization of central prototype optimization problems to subspaces, we work with points that are not all necessarily on the same Grassmann manifold but are in the same ambient space. Suppose we have a set of subspaces of n-dimensional space, $\{[\mathbf{X}_1], [\mathbf{X}_2], \ldots, [\mathbf{X}_p]\}$, where $[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)$. Note: each k_i corresponds to each $[\mathbf{X}_i]$. We want to find an r-dimensional subspace of $\mathbb{R}^n$, $[\mathbf{Y}^*] \in \mathrm{Gr}(r, n)$, that is the center of these points, i.e., that $[\mathbf{Y}^*]$ is a solution to

$$\arg \min_{[\mathbf{Y}] \in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} d([\mathbf{X}_i], [\mathbf{Y}]) \tag{1.1}$$

where $d : \mathrm{Gr}(k_i, n) \times \mathrm{Gr}(r, n) \to \mathbb{R}$ is a function which measures distance between its arguments.

Principal angles between subspaces are a common dissimilarity measure that is invariant to orthogonal transformations [2, 3]. Take $[\mathbf{X}], [\mathbf{Y}] \in \mathrm{Gr}(k, n)$. The ith smallest principal angle

between $[\mathbf{X}]$ and $[\mathbf{Y}]$, $\theta_i([\mathbf{X}],[\mathbf{Y}]) = \mathbf{x}_i^T\mathbf{y}_i \in [0,\pi/2]$ is defined as the solution to (1.2) [2].

$$\begin{aligned} \mathbf{x}_i^T\mathbf{y}_i &= \max_{\mathbf{x}\in[\mathbf{X}]} \max_{\mathbf{y}\in[\mathbf{Y}]} \mathbf{x}^T\mathbf{y} \\ \text{subject to } \mathbf{x}^T\mathbf{x} &= \mathbf{y}^T\mathbf{y} = 1 \\ \mathbf{x}^T\mathbf{x}_j &= \mathbf{y}^T\mathbf{y}_j = 0 \text{ for } j = 1,2,\ldots,i-1 \end{aligned} \tag{1.2}$$

Now let $\theta([\mathbf{X}],[\mathbf{Y}]) \in \mathbb{R}^k$ be the vector of principal angles between $[\mathbf{X}]$ and $[\mathbf{Y}]$. The *geodesic distance* on $\mathrm{Gr}(k,n)$ is $\|\theta([\mathbf{X}],[\mathbf{Y}])\|_2$ and the *chordal distance* on $\mathrm{Gr}(k,n)$ is $\|\sin(\theta([\mathbf{X}],[\mathbf{Y}]))\|_2$ [4]. Similarity between subspaces can be calculated by using the cosine of the principal angles $\|\cos(\theta([\mathbf{X}],[\mathbf{Y}]))\|_2$. We can calculate these quantities when $[\mathbf{X}] \in \mathrm{Gr}(k,n)$, $[\mathbf{Y}] \in \mathrm{Gr}(r,n)$ where $k \neq r$ by setting the last $\max\{k,r\} - \min\{k,r\}$ entries of $\theta([\mathbf{X}],[\mathbf{Y}]) \in \mathbb{R}^{\max\{k,r\}}$ to 0.

1.2.3 Introduction to Frame Theory

In this section we provide an introduction to frame theory which is used for some results in Chapter 2. See the lecture notes by King for the original references many of the following results [5].

A *real frame* is a set of vectors $\Phi = \{\boldsymbol{\varphi_k}\}_{k=1}^p \subset \mathbb{R}^n$ where there exist $0 < A \leq B$ so that for any $\mathbf{x} \in \mathbb{R}^n$ we have $A\|\mathbf{x}\|_2^2 \leq \sum_{k=1}^n |\langle \mathbf{x}, \boldsymbol{\varphi_k}\rangle|^2 \leq B\|\mathbf{x}\|_2^2$. We call $\boldsymbol{\Phi} = [\boldsymbol{\varphi_1}, \boldsymbol{\varphi_2}, \ldots, \boldsymbol{\varphi_n}]$ the *synthesis matrix* of the given frame. The *Gram matrix* of said frame is $\mathbf{G} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$.

A real frame is *equiangular* when there exists some $r \in \mathbb{R}$ so that $|\langle \boldsymbol{\varphi_j}, \boldsymbol{\varphi_k}\rangle| = r$ for all $j \neq k$ and $\langle \boldsymbol{\varphi_k}, \boldsymbol{\varphi_k}\rangle = 1$ for all k. A real frame is *tight* when $A = B$. This is equivalent to the rows of $\boldsymbol{\Phi}$ being orthogonal and all singular values of $\boldsymbol{\Phi}$ being $\sqrt{A}$ [5]. We call A the frame bound of this frame.

Frames are generalized to projection matrices using *fusion frames*. Let $\{\mathbf{P}_i\}_{i=1}^p \in \mathbb{R}^{n\times n}$ be a set of rank k projection matrices. A fusion frame formed by $\{\mathbf{P}_i\}_{i=1}^p$ with frame bounds $A, B \in \mathbb{R}$, $0 < A \leq B$ satisfies $A\mathbf{I} \leq \sum_{i=1}^p \mathbf{P}_i \leq B\mathbf{I}$. When $\sum_{i=1}^p \mathbf{P}_i = A\mathbf{I}$, we call $\{\mathbf{P}_i\}_{i=1}^p$ a *tight fusion frame* [6]. In fact, a set of subspaces $\{[\mathbf{X}_i]\}_{i=1}^p$ can form a set of orthogonal projections by $\mathbf{P}_i = \mathbf{X}_i\mathbf{X}_i^T$.

Now we return to more definitions involving real frames like $\{\boldsymbol{\varphi_k}\}_{k=1}^{p}$. We call an *equiangular tight frame (ETF)* a frame that is both equiangular and tight. A frame is *balanced* when $\sum_{k=1}^{n} \varphi_k = 0$. There exists a construction for balanced ETFs of size $n+1$ in $\mathbb{R}^n$. In this construction, we find the frame bound is $\frac{n+1}{n}$.

The *coherence* of a set of vectors $\Phi = \{\boldsymbol{\varphi_k}\}_{k=1}^{p} \subset \mathbb{R}^n$ is defined as $\mu(\Phi) := \max_{j\neq k} |\langle \boldsymbol{\varphi_j}, \boldsymbol{\varphi_k} \rangle|$. The Welch bound (Theorem 1.2.1) bounds the coherence of a set of unit vectors and gives us the coherence of these vectors if they form an ETF.

Theorem 1.2.1. *(Welch Bound) A set of unit vectors $\Phi = \{\boldsymbol{\varphi_k}\}_{k=1}^{p} \subset \mathbb{R}^n$ with $p \geq n$ satisfies*

$$\mu(\Phi)^2 \geq \frac{p-n}{n(p-1)}.$$

This bound is saturated when Φ is an ETF.

Suppose $\Phi = \{\boldsymbol{\varphi_k}\}_{k=1}^{n+1} \subset \mathbb{R}^n$ is an ETF, then the saturation of the Welch bound gives us

$$\min_{j\neq k} |\langle \boldsymbol{\varphi_j}, \boldsymbol{\varphi_k} \rangle| = \frac{1}{n}.$$

Since an ETF is equiangular we see $|\langle \boldsymbol{\varphi_j}, \boldsymbol{\varphi_k} \rangle| = \frac{1}{n}$ for all $j \neq k$.

We define *spark* of a set of vectors $\Phi = \{\varphi_k\}_{k=1}^{p} \subset \mathbb{R}^n$ as the size of the smallest linearly dependent subset of vectors. A set of vectors in $\mathbb{R}^n$ is said to be full spark when $\text{spark}(\Phi) = d+1$. Spark and coherence are related because $\text{spark}(\Phi) \geq \frac{1}{\mu(\Phi)} + 1$. We can use this to find the spark of an ETF of $n+1$ vectors in $\mathbb{R}^n$.

If Φ is an ETF of $n+1$ vectors in $\mathbb{R}^n$, then $\text{spark}(\Phi) \geq \frac{1}{\mu(\Phi)} + 1 = n+1$. But since we have $n+1$ vectors, $\text{spark}(\Phi) \leq n+1$. So we have $\text{spark}(\Phi) = n+1$.

One notion of equivalence between frames is switching equivalence. Suppose we have two frames with respective synthesis matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$. These two frames are *switching equivalent* when there exists some unitary matrix $\mathbf{U} \in O(n)$ and some diagonal matrix $\boldsymbol{\Lambda} \in O(p)$ with unimodular entries along the diagonal so that $\mathbf{U}\boldsymbol{\Phi}\boldsymbol{\Lambda} = \boldsymbol{\Psi}$.

Now we investigate what happens if we let $\mathbf{\Lambda} = \mathbf{V}$ to be unitary and not diagonal. Suppose we have two ETFs of size p with frame bound $A \in \mathbb{R}$ with respective synthesis matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ then there exitsts some $\mathbf{U}$ and $\mathbf{V}$ so that $\mathbf{U\Phi V} = \mathbf{\Psi}$. See the Lemma A.0.1 in the Appendix for the proof.

A example of an ETF of 4 points in $\mathbb{R}^3$ is

$$\{\boldsymbol{\varphi_k}\}_{k=1}^4 = \left\{ \begin{bmatrix} \sqrt{2/3} \\ 0 \\ \sqrt{1/3} \end{bmatrix}, \begin{bmatrix} 0 \\ \sqrt{2/3} \\ \sqrt{1/3} \end{bmatrix}, \begin{bmatrix} -\sqrt{2/3} \\ 0 \\ \sqrt{1/3} \end{bmatrix}, \begin{bmatrix} 0 \\ -\sqrt{2/3} \\ \sqrt{1/3} \end{bmatrix} \right\}.$$

This is an ETF because $|\langle \boldsymbol{\varphi_j}, \boldsymbol{\varphi_k} \rangle| = 1/3$ for all $j \neq k$. The Welch bound is $1/3$. Notice this frame is not balanced because the sum of the last coordinates of each vector is $\frac{4}{\sqrt{3}}$.

A balanced 4 vector ETF in $\mathbb{R}^3$ is

$$\{\boldsymbol{\psi_k}\}_{k=1}^4 = \left\{ \begin{bmatrix} -\sqrt{2}/3 \\ -\sqrt{6}/3 \\ -1/3 \end{bmatrix}, \begin{bmatrix} -\sqrt{2}/3 \\ \sqrt{6}/3 \\ -1/3 \end{bmatrix}, \begin{bmatrix} 2\sqrt{2}/3 \\ 0 \\ -1/3 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}.$$

The unitary $\mathbf{U}$ and $\mathbf{V}$ for the weaker version of equivalence (eg. $\mathbf{U\Phi V} = \mathbf{\Psi}$) are

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.894 & -0.447 \\ 0 & -0.447 & 0.894 \end{bmatrix} \text{ and } \mathbf{V} = \begin{bmatrix} -0.260 & -0.576 & 0.448 & 0.387 \\ 0.568 & -0.643 & -0.039 & 0.113 \\ 0.318 & 0.001 & -0.706 & 0.387 \\ -0.509 & 0.069 & -0.220 & 0.661 \end{bmatrix}.$$

Another interesting result comes from part of Lemma 4.2 in [7] and is reiterated in [5]. We will state this theorem for the field $\mathbb{R}$. Suppose $\Phi = \{\boldsymbol{\varphi_k}\}_{k=1}^p \subset \mathbb{R}^n$ is an ETF with Welch bound $\frac{1}{s}$. Then any $\{\boldsymbol{\varphi_j}\}_{j\in\beta}$ with $|\beta| = s+1$ is a *regular simplex ETF* if and only if the triple product

$$\langle \boldsymbol{\varphi_i}, \boldsymbol{\varphi_j} \rangle \langle \boldsymbol{\varphi_j}, \boldsymbol{\varphi_k} \rangle \langle \boldsymbol{\varphi_k}, \boldsymbol{\varphi_i} \rangle = \frac{-1}{s^3} \tag{1.3}$$

for all pairwise distinct $i, j, k \in \beta$. The proof of the result from [7] and [5] uses the fact that triple products (e.g., (1.3)) completely characterize switching equivalence of ETFs.

That is to say, two ETFs with the same triple products of $n+1$ vectors in $\mathbb{R}^n$ are switching equivalent to a n-simplex with Gram matrix $\mathbf{G} = \left(1 + \frac{1}{n}\right)\mathbf{I_{n+1}} - \frac{1}{n}\mathbf{J_{n+1}}$ where $\mathbf{I_{n+1}}$ is the $(n+1) \times (n+1)$ identity matrix and $\mathbf{J_{n+1}}$ is the $(n+1) \times (n+1)$ matrix of all ones [5].

Chapter 2

Subspace Prototypes: The Flag Median and the Maximally Correlated Flag

2.1 Introduction

An essential property of any probability distribution is its central prototype. The mean and median are two of the most basic methods for calculating central prototypes from a probability distribution. The median is commonly more robust to outliers than the mean. Generalizations of such prototypes to Euclidean space can be formulated as solutions to optimizations problems. Suppose we have a set of points in Euclidean space, $\mathscr{X} = \{\mathbf{x}_i\}_{i=1}^p \subset \mathbb{R}^n$. We can represent this set using a central prototype $\mathbf{y}$. We find $\mathbf{y}$ by solving

$$\arg\min_{\mathbf{y}\in A} \sum_{i=1}^{p} \|\mathbf{x}_i - \mathbf{y}\|_2^q. \tag{2.1}$$

The solution to (2.1) for $A = \mathbb{R}^n$ and $q = 2$ is called the centroid, which may be viewed as the generalization of the mean. In fact, the centroid is the component-wise mean of the vectors in $\mathscr{X}$, $\sum_{i=1}^p \mathbf{x}_i / p$. Generalizations of the median involve solving (2.1) when $q = 1$. When $A = \mathscr{X}$, the solution is called the medoid, while when $A = \mathbb{R}^n$, the solution is called the geometric median. The geometric median inherits the robustness to outliers from the median without being required to be a point in the data set. These Euclidean prototypes are used as a statistic for a data set and in common machine learning algorithms like LBG (Linde-Buzo-Gray) clustering and nearest centroid classification [8, 9].

Another method for finding a central prototype from a set of points is to maximize similarity rather than minimize distance between the prototype and the data. A ubiquitous measure of similarity between points in Euclidean space is correlation. Using correlation, we define the

maximally correlated vector of a set of vectors as

$$\arg\max_{\mathbf{y}\in\mathbb{R}^n} \sum_{i=1}^{p} \left| \frac{\langle \mathbf{x}_i, \mathbf{y} \rangle}{\|\mathbf{x}_i\|_2 \|\mathbf{y}\|_2} \right|. \tag{2.2}$$

This flavor of optimization problem was investigated by Bates et al. [10]. It also has ties to the multivariate eigenvalue problem which has been solved in numerous ways by a number of researchers [10–17].

Not all data sets are best represented using points in Euclidean space. Specifically, subspaces have been shown to provide useful, robust representations for data sets of images, videos and more. For example, the smallest principal angle between two subspaces has proven powerful for modeling illumination spaces [18]. Hyperspectral data has failed to be linearly separable in Euclidean space but separates linearly on the Grassmannian [19]. Subspaces are also used to formalize the separation of a data set into inliers and outliers which has applications to computer vision tasks like 3D reconstruction [20, 21].

Popular machine learning techniques, like dictionary learning, have been adapted to Riemannian manifolds [22]. Jayasumana et. al. consider learning on the Grassmannian (and Riemannian manifolds in general) with radial basis function kernels and advocate for chordal distance (sometimes referred to as the projection norm) kernels on the Grassmannian because chordal distance generates a positive definite Gaussian kernel [23]. More recently, Cherian et. al. use kernalized Grassmannian pooling for activity recognition [24]. Methods for Riemannian optimization like Riemannian Stochastic Variance Reduced Gradient (SVRG) have gained popularity alongside this surge of interest in Riemannian learning [25]. Even more uses for subspaces in computer vision and machine learning can be found in [3, 10, 17, 26–30].

Hence, it is useful to find versions of the Euclidean prototypes on the Grassmannian. A logical generalization of prototypes from Euclidean space to the Grassmannian is to replace the Euclidean 2-norm in the optimization problems for the centroid, medoid and geometric median with a distance between subspaces. The centroid is generalized using the geodesic distance in [31] and the chordal distance in [3, 32]. To the extent of our research, we have have

not found a generalization of the medoid. However, the geometric median has been generalized using the geodesic distance and is called the ℓ_2-median in [27, 33]. Prototypes like these have been used alone as a method to classify emotion in images [30], as a step in a k-means type algorithm [28] and in feature extraction [33].

Bates et. al. show that the problem of maximizing similarity (a generalization of (2.2) to subspaces) can be translated into the popular multivariate eigenvalue problem and maximum correlation problems [10]. Such problems have been studied extensively but haven't been seen much work in the past 10 years [10–17]. In addition, this problem has ties to the popular canonical correlation analysis and is just a maximization of the Dual Principal Component Pursuit (DPCP) objective function [20, 34].

Since subspaces correspond to points on a Grassmann manifold, one is led to consider the idea of a subspace prototype for a Grassmann-valued data set. While a number of different subspace prototypes have been described, the calculation of some of these prototypes has proven to be computationally expensive while other prototypes are affected by outliers and produce highly imperfect clusterings on noisy data.

This work proposes a two new subspace prototypes, the flag median and maximally correlated flag. The flag median is prototype which is a generalization of the geometric median to the Grassmannian using the chordal distance. The maximally correlated flag is a prototype which is essentially a generalization of max length unit vectors to higher dimensional subspaces [10]. We call these prototypes "flag" prototypes because the resulting subspace is really a collection of nested subspaces which make up a point on the flag manifold. After introducing these prototypes, we provide a unifying theory for subspace prototypes by considering matrix norms of the principal angle matrix. Along with introducing these novel prototypes, we mention a few new results the regarding the uniqueness of subspace prototypes for subspace configurations from special types of frames. Finally, we will use experiments on synthetic data to determine that the flag median is more robust to outliers than other popular subspace prototypes, the maximally correlated flag problem can be unstable, and a conjecture for the convexity of these prototypes.

Contributions:

- An introduction to two new subspace prototypes: the flag median and the maximally correlated flag.
- A unifying theory for the development of subspace prototypes using the principal angle matrix.
- Results regarding the flag median, flag mean, and ℓ_2-median of subspace configurations of specific types of frames.
- A comparison of these two new prototypes to each other and other common subspace prototypes using synthetic experiments.

2.2 A Taxonomy of Subspace Prototypes

In this section we will discuss a number of known subspace prototypes and two new subspace prototypes. The known prototypes are the Karcher mean, ℓ_2-median, flag mean, and max length unit vector. The Karcher mean and ℓ_2-median are geodesic distance prototypes and the flag mean is a chordal distance prototype. On the other hand, the max length unit vector is a prototype which comes from maximizing similarity between subspaces rather than minimizing distance. The two new subspace prototypes are the flag median and maximally correlated flag. The flag median is the chordal distance median of subspaces and the maximally correlated flag is the generalization of the max length unit vector problem to higher dimensional subspaces.

Before we set off on our prototype survey, let us establish some notation for our data and our prototypes.

- A data set of subspaces is $A = \{[\mathbf{X}_i]\}_{i=1}^{p}$ where for each i we have $[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)$ where $k_i < n$.
- $[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)$ is represented by $\mathbf{X}_i \in \mathbb{R}^{k_i, n}$ where $\mathbf{X}_i^T \mathbf{X}_i = \mathbf{I}$.
- $[\mathbf{Y}] \in \mathrm{Gr}(r, n)$ where $r < n$ denotes the a central subspace prototype.

2.2.1 ℓ_2-Median and Karcher Mean

The Euclidean mean and geometric median have been translated to the Grassmannian using geodesic distance. The mean on the Grassmannian using geodesic distance (the solution to (2.3) for $q = 2$) is called the Karcher mean and the geometric median using geodesic distance (the solution to (2.3) for $q = 1$) is called the ℓ_2-median.

$$\underset{[\mathbf{Y}]\in \mathrm{Gr}(r,n)}{\arg\min} \sum_{i=1}^{p} \|\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2^q. \tag{2.3}$$

The Karcher mean and the ℓ_2-median are only computable in the case where all subspaces are of equal dimensions (e.g., $r = k_1 = k_2 \cdots = k_p$) and Marrinan et al. use examples to show that the available algorithms to compute these prototypes are slow [28] . The most common algorithm for finding the solution to the Karcher mean was discovered by Karcher [31]. Fletcher et al. show we can find the ℓ_2-median using a Weiszfeld-type algorithm [33]. Marrinan et al. show that the Karcher mean is not only slow to compute, but also produces lower cluster purities than the ℓ_2-median in their LBG clustering example. So, we choose not to use the Karcher mean as a prototype in our experiments in this chapter and in Chapter 3.

Convexity of the Karcher mean was proven by Karcher [31]. Fletcher proved the convexity of the ℓ_2-median , [33]. These proofs are based on showing the squared geodesic distance and the geodesic distance are convex functions. Let Δ be the sectional curvature of the Grassmannian. Both optimization problems, the Karcher mean and ℓ_2-median, are convex on a set $U \subset \mathrm{Gr}(k,n)$ as long as $\mathrm{diam}(U) < \pi/2\sqrt{\Delta}$. If $\Delta > 0$, then these optimization problems are always convex.

2.2.2 The Flag Mean

Draper et al. [3] present the flag mean as an average of subspaces of different dimensions using the squared chordal distance. The optimization problem for the flag mean is

$$\arg \min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \|\sin(\theta([\mathbf{X}_i],[\mathbf{Y}]))\|_2^2. \tag{2.4}$$

This flag mean determines not only a point on $\mathrm{Gr}(r,n)$, it determines a point on various flag manifolds. A flag manifold is a manifold whose points represent a flag of subspaces $[\mathbf{S}_1] \subset [\mathbf{S}_2] \subset \cdots \subset [\mathbf{S}_r] = \mathbb{R}^n$. If we let $s_i = \dim([\mathbf{S}_i])$, then we say the flag is of type $s_1, s_2, \ldots, s_r$. For more details on flag manifolds, see [35].

Various additional formulations of this optimization problem have been solved. Santamaria et al. find the optimal dimension r of the flag mean. They find that if the eigenvalues of $\frac{1}{p}\sum_{i=1}^{p}\mathbf{X}_i\mathbf{X}_i^T$ are less than $1/2$, then the optimal flag mean will contain only the left singular vectors which are associated with singular values $\sigma_i \geq 1/\sqrt{2}$ [36]. The next year, Santamaria et al. solved this problem where we fix some subspace $[\mathbf{W}] \in \mathrm{Gr}(w,n)$, an integer $d > 0$, and restrict $\dim([\mathbf{Y}] \cap [\mathbf{W}]) = d$ [37].

Let $[\mathbf{Y}]$ be the flag mean of $\{[\mathbf{X}_i]\}_{i=1}^{k}$. Let $\mathbf{y}_i$ be the ith column of $\mathbf{Y}$, the orthonormal matrix representation of $[\mathbf{Y}]$. Then the r^{th} "real" flag mean is the point on the flag manifold of type $\{1,2,\ldots,r;n\}$ defined as

$$[\![\mathbf{Y}]\!] = \mathrm{span}\{\mathbf{y}_1\} \subset \mathrm{span}\{\mathbf{y}_1,\mathbf{y}_2\} \subset \cdots \subset \mathrm{span}\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_r\} \subset \mathbb{R}^n. \tag{2.5}$$

Draper et al. [3] show that we can calculate the flag mean by utilizing the singular value decomposition (SVD) of the matrix $[\mathbf{X}_1,\mathbf{X}_2,\ldots,\mathbf{X}_p]$. The flag mean, as a point on $\mathrm{Gr}(r,n)$, is the span of the r left singular vectors of the corresponding to the r largest singular values.

2.2.3 Grassmannian Minimum Enclosing Ball

The Grassmannian minimum enclosing ball (GMEB) is a newer chordal distance prototype by Marrinan et al. [38]. The optimization problem for this prototype is

$$\min_{[\mathbf{Y}]\in\mathrm{Gr}(r,n)} \max_{i=1,2,\ldots,p} \|\sin(\theta([\mathbf{X}_i],[\mathbf{Y}]))\|_2^2 \tag{2.6}$$

Like the flag mean, the GMEB is a prototype of a collection of mixed-dimensional subspaces. The GMEB can be found by using a sub-gradient method to solve the dual problem to (2.6). There is also a method to find the optimal r for the GMEB.

2.2.4 Max Length Unit Vectors

A way to measure similarity between subspaces uses the norms of the cosine of the principal angles between them. This problem has seen plenty of work with $[\mathbf{Y}] \in \mathrm{Gr}(1,n)$. One flavor of this optimization problem

$$\max_{[\mathbf{Y}]\in\mathrm{Gr}(1,n)} \sum_{i=1}^{p} \cos(\theta([\mathbf{X}_i],[\mathbf{Y}])) \tag{2.7}$$

is known as the Maximum Length Vector Problem [10]. Bates et al. use $[\boldsymbol{\alpha}_i] \in \mathrm{Gr}(1,k_i)$ to translate (2.7) from an angle maximization problem to a problem about inner products between $\mathbf{X}_i$ and $\boldsymbol{\alpha}_i$:

$$\max_{\substack{[\mathbf{Y}]\in\mathrm{Gr}(1,n)\\ [\boldsymbol{\alpha}_i]\in\mathrm{Gr}(1,k_i)}} \sum_{i=1}^{p} \mathbf{Y}^T\mathbf{X}_i\boldsymbol{\alpha}_i. \tag{2.8}$$

Let $\mathbf{v} = \sum_{i=1}^{p} \mathbf{X}_i\boldsymbol{\alpha}_i$. Notice $\mathbf{Y}^T\mathbf{v} = \|\mathbf{Y}\|_2\|\mathbf{v}\|_2\cos(\phi)$ where ϕ is the angle between $\mathbf{Y}$ and $\mathbf{v}$. The maximizer of (2.8) is the maximizer of $\|\mathbf{v}\|_2$ and $\mathbf{Y} = \mathbf{v}/\|\mathbf{v}\|_2$. Since maximizing $\|\mathbf{v}\|_2$ is the same as maximizing $\|\mathbf{v}\|_2^2$, the problem in (2.8) is equivalent to (2.9).

$$\max_{\boldsymbol{\alpha}_i\in\mathrm{Gr}(1,k_i)} \left\| \sum_{i=1}^{p} \mathbf{X}_i\boldsymbol{\alpha}_i \right\|_2^2. \tag{2.9}$$

We can use the matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{X}_1^T\mathbf{X}_2 & \cdots & \mathbf{X}_1^T\mathbf{X}_p \\ \mathbf{X}_2^T\mathbf{X}_1 & \mathbf{I} & \cdots & \mathbf{X}_2^T\mathbf{X}_p \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_n^T\mathbf{X}_1 & \mathbf{X}_n^T\mathbf{X}_2 & \cdots & \mathbf{I} \end{bmatrix}$$

to transform maximizing $\|\mathbf{v}\|_2^2$ in (2.9) to the Maximum Correlation Problem (MCP)

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \boldsymbol{\alpha}\mathbf{A}\boldsymbol{\alpha}, \\ &\text{subject to } \boldsymbol{\alpha}_i^T \boldsymbol{\alpha}_i = 1, \\ &\text{for } i = 1, 2, \ldots, p. \end{aligned} \tag{2.10}$$

The necessary conditions for a maximizer of the MCP in (2.10) are found in the Multivariate Eigenvalue Problem (MEP) in (2.11). Methods for solving the MEP can be found in [12], [39], and [13].

$$\begin{aligned} \mathbf{A}\boldsymbol{\alpha} &= \Lambda\boldsymbol{\alpha} \\ \boldsymbol{\alpha}_i^T \boldsymbol{\alpha}_i &= 1 \text{ for } i = 1, 2, \ldots, p \end{aligned} \tag{2.11}$$

We can recover $[\mathbf{Y}]$ from $\boldsymbol{\alpha}$ by setting $\mathbf{Y} = \frac{\mathbf{v}}{\|\mathbf{v}\|_2}$. We can also recover $\boldsymbol{\alpha}/\|\mathbf{v}\|_2$ from $\mathbf{Y}$ by solving $\mathbf{Y} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_p]\boldsymbol{\alpha}$ for $\boldsymbol{\alpha}$. In Marks's paper [17], he draws a connection to the Lagrangian of the prolific multiset canonical correlation analysis problem (see [40]) from the optimization problem in (2.11).

There are many algorithms for finding solutions to the max length vector problem, the MEP and the MCP. A non-exhaustive list of these algorithms and authors who have solved them are listed in Table 2.1

Table 2.1: Algorithms to solve (2.7) and its equivalent formulations.

Algorithm Name	Author
Horst-Jacobi	Horst et al. [11]
Flag Mean 1	Marks et al. [17]
Newton	Marks et al. [17]
Homotopy Continuation	Bates et al. [10]
Gauss Siedel	Zhang et al. [13] and [16]
SOR	Chu et al. [12]
Rayleigh Quotient	Zhang et al. [13]
Alternating Variable	Zhang et al. [15]
Riemannian Newton Method	Zhang et al. [14]

The general MEP tends to have many solutions and is therefore not convex. Most of these iterative methods to solve the MEP can be used to find only a single solution and must be re-run with different initialization to find more than one solution. Bates et al. find multiple solutions to the MEP by using a homotopy continuation method [10]. These methods have not been used in the case where $[\mathbf{Y}] \in \mathrm{Gr}(r,n)$ for $r > 1$ because the translation between Equations 2.7 and 2.10 only applies to $r = 1$.

2.2.5 Flag Median

The translation of the geometric median to the Grassmannian using chordal distance is called the flag median. The optimization problem for this novel prototype is in (2.12).

$$\arg \min_{[\mathbf{Y}] \in Gr(r,n)} \sum_{i=1}^{p} \| \sin\theta([\mathbf{X}_i],[\mathbf{Y}]) \|_2 \tag{2.12}$$

We call this the flag median since, using FlagIRLS (see Chapter 3), $[\mathbf{Y}]$ actually is a flag of subspaces rather than a single r dimensional subspace of n dimensional space. This flag median is indeed a median (similar to the geometric median) because it minimizes the chordal distance rather than the squared chordal distance problem in (2.4) that is solved by the flag mean.

2.2.6 Maximally Correlated Flag

The maximally correlated flag is a generalization of the max length unit vector problem by Bates et al. [10].The optimization problem for this novel prototype is in (2.13).

$$\arg \max_{[\mathbf{Y}] \in Gr(r,n)} \sum_{i=1}^{p} \| \cos \theta([\mathbf{X}_i], [\mathbf{Y}]) \|_2 \tag{2.13}$$

Using FlagIRLS, $[\mathbf{Y}]$ actually is a flag of subspaces rather than a single r dimensional subspace of n dimensional space. Notice that for $k_1 = k_2 = \cdots = k_p = r = 1$, this is essentially (2.2).

2.3 Unifying Subspace Prototypes

Let $m = \max\{k_1, k_2 \ldots, k_p, r\}$. The jth entry of the vector of principle angles between $[\mathbf{X}_i]$ and $[\mathbf{Y}]$ is denoted $\theta_j([\mathbf{X}_i], [\mathbf{Y}])$. We order the entries of this vector as $\theta_1([\mathbf{X}_i], [\mathbf{Y}]) \geq \theta_2([\mathbf{X}_i], [\mathbf{Y}]) \geq \cdots \geq \theta_m([\mathbf{X}_i], [\mathbf{Y}])$. Using this notation, we define the Θ, matrix of principle angles between $\{[\mathbf{X}_i]\}_{i=1}^{p}$ and $[\mathbf{Y}]$, in (2.14).

$$\Theta = \begin{bmatrix} \theta_1([\mathbf{X}_1], [\mathbf{Y}]) & \theta_1([\mathbf{X}_2], [\mathbf{Y}]) & \cdots & \theta_1([\mathbf{X}_p], [\mathbf{Y}]) \\ \theta_2([\mathbf{X}_1], [\mathbf{Y}]) & \theta_2([\mathbf{X}_2], [\mathbf{Y}]) & \cdots & \theta_2([\mathbf{X}_p], [\mathbf{Y}]) \\ \vdots & \vdots & \vdots & \vdots \\ \theta_m([\mathbf{X}_1], [\mathbf{Y}]) & \theta_m([\mathbf{X}_2], [\mathbf{Y}]) & \cdots & \theta_m([\mathbf{X}_p], [\mathbf{Y}]) \end{bmatrix}. \tag{2.14}$$

The ith column of the principle angle matrix contains the vector of principle angles between $[\mathbf{X}_i]$ and $[\mathbf{Y}]$. There are numerous options for stating optimization problems involving matrix norms of Θ, $\cos\Theta$ and $\sin\Theta$. For clarification, $\cos\Theta$ denotes the matrix with entries $\cos(\theta_i([\mathbf{X}_j], [\mathbf{Y}]))$. The matrix norm of Θ with $q, s > 0$ is defined in (2.15).

$$\|\Theta\|_{q,s} := \left(\sum_{j=1}^{p} \left(\sum_{i=1}^{m} |\Theta_{i,j}^{q}| \right)^{s/q} \right)^{1/s}. \tag{2.15}$$

Table 2.2 lists a non-exhaustive set of optimization problems involving matrix norms of Θ.

Table 2.2: The various optimization problems that can be phrased as optimizers of a matrix norm of the principal angle matrix.

Optimization Problem	Solution Name	Solved By
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(k,n)} \|\Theta\|_{2,2}^2$	Karcher Mean	Karcher [31]
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \|\sin\Theta\|_{2,2}^2$	Flag Mean	Draper et al. [3]
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(k,n)} \|\Theta\|_{2,1}$	Riemannian Geometric Median or ℓ_2-median ($k=1$)	Aftab et al. [27], Fletcher et al. [33], **dissertation** ($k=1$)
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \|\sin\Theta\|_{2,1}$	Flag Median	**This dissetation**
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \max_{j=1,2,\ldots,p} \|\sin\Theta_{:,j}\|_2^2$	GMEB	Marrinan et. al. [38]
$\max_{[\mathbf{Y}]\in \mathrm{Gr}(1,n)} \|\cos\Theta\|_1$	Max Length Unit Vector, or Maximum Correlation Problem	Bates et al. [10], Chu et al. [12], Marks et al. [17], **this dissertation**
$\max_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \|\cos\Theta\|_{2,1}$	Maximally Correlated Flag	**This dissertation**

Let $\{\omega_1, \omega_2, \ldots, \omega_p\} \subset \mathbb{R}$ be a set of weights where ω_i is the weight for $[\mathbf{X}_i]$. Now we can reformulate Table 2.2 with weights in Table 2.3.

Table 2.3: The weighted versions of the optimization problems in Table 2.2 with weights $\omega_i \in \mathbb{R}$.

Optimization Problem	Solution Name
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(k,n)} \sum_{i=1}^{p} \omega_i \|\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2^2$	Karcher Mean
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \omega_i \|\sin\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2^2$	Flag Mean
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(k,n)} \sum_{i=1}^{p} \omega_i \|\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2$	Riemannian Geometric Median or ℓ_2-median ($k=1$)
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \omega_i \|\sin\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2$	Flag Median
$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \max_{i=1,2,\ldots,p} \omega_i \|\sin\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2^2$	GMEB
$\max_{[\mathbf{Y}]\in \mathrm{Gr}(1,n)} \sum_{i=1}^{p} \omega_i \cos\theta([\mathbf{X}_i],[\mathbf{Y}])$	Max Length Unit Vector, or Maximal Correlation Problem
$\max_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} \omega_i \|\cos\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2$	Maximally Correlated Flag

2.4 Subspace Prototypes of Frames

We will investigate the behavior of central subspace prototypes (namely the flag mean, the ℓ_2-median and the flag median) of a set of evenly spaced subspaces. We find that, under certain point configurations, we are guaranteed that the flag mean is every subspace in $\mathrm{Gr}(r,n)$. Other 1-dimensional subspace configurations force the ℓ_2-median and the flag median to be one of the data subspaces, $[\mathbf{X}_i]$.

2.4.1 The Flag Mean

A tight fusion frame can be formed by a set of subspaces $\{[\mathbf{X}_i]\}_{i=1}^{p} \subset \mathrm{Gr}(k,n)$ where

$$\sum_{i=1}^{n} \|\mathbf{z}\mathbf{X}_i\mathbf{X}_i^T\| = A$$

for any unit vector $\mathbf{z} \in \mathbb{R}^n$ and some strictly positive $A \in \mathbb{R}$ [6].

Theorem 2.4.1. *The flag mean of subspaces that form a tight fusion frame in* $\mathrm{Gr}(k,n)$ *is every point in* $\mathrm{Gr}(k,n)$.

Proof. Suppose $\{[\mathbf{X}_i]\}_{i=1}^p \subset \mathrm{Gr}(k,n)$ is a tight fusion frame with frame bound $A \in \mathbb{R}$. Then the flag mean objective function for any $[\mathbf{Y}] \in \mathrm{Gr}(r,n)$ is

$$f([\mathbf{Y}]) = \sum_{i=1}^{p} \left(\min\{r,k\} - \mathrm{tr}\left(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}\right)\right). \tag{2.16}$$

Using the fact that $\{[\mathbf{X}_i]\}_{i=1}^p \subset \mathrm{Gr}(k,n)$ can be used to form a tight fusion frame we have

$$\begin{aligned}
\left(\min\{r,k\} - \mathrm{tr}\left(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}\right)\right) &= p\min\{r,k\} - \sum_{i=1}^{p}\mathrm{tr}\left(\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}\mathbf{Y}^T\right) && (2.17)\\
&= p\min\{r,k\} - \mathrm{tr}\left(\sum_{i=1}^{p}\left(\mathbf{X}_i\mathbf{X}_i^T\right)\mathbf{Y}\mathbf{Y}^T\right) && (2.18)\\
&= p\min\{r,k\} - \mathrm{tr}\left(A\mathbf{Y}\mathbf{Y}^T\right) && (2.19)\\
&= p\min\{r,k\} - A\mathrm{tr}\left(\mathbf{Y}^T\mathbf{Y}\right) && (2.20)\\
&= p\min\{r,k\} - Ar && (2.21)
\end{aligned}$$

This means that $f([\mathbf{Y}])$ is constant and therefore any $[\mathbf{Y}] \in \mathrm{Gr}(r,n)$ is a minimizer of f. □

2.4.2 ℓ_2-Median and Flag Median

We will simplify our problem by only considering points on $\mathrm{Gr}(1,n)$. Let us investigate the behavior of the ℓ_2-median and the flag median of $\{[\mathbf{x}_i]\}_{i=1}^{n+1} \subset \mathrm{Gr}(1,n)$ where $\{\mathbf{x}_i\}_{i=1}^{n+1}$ form an ETF. Lemma 2.4.2 and Lemma 2.4.3 study properties of ETFs and balanced unit norm frames. The objective functions for the ℓ_2-median and the flag median in $\mathrm{Gr}(1,n)$ are (2.22) and (2.23) respectively.

$$F_1(\mathbf{y}) = \sum_{i=1}^{p} \arccos\left(|\langle \mathbf{x}_i, \mathbf{y}\rangle|\right) \tag{2.22}$$

$$F_2(\mathbf{y}) = \sum_{i=1}^{p} \sqrt{1 - \langle \mathbf{x}_i, \mathbf{y} \rangle^2} \tag{2.23}$$

Lemma 2.4.2. If $\{\boldsymbol{\varphi_k}\}_{i=1}^{n+1}$ is an equiangular tight frame (ETF) in $\mathbb{R}^n$ for $n > 1$, then there exists no unit vector $\mathbf{x} \in \mathbb{R}^n$ so that $\{\boldsymbol{\varphi_k}\}_{i=1}^{n+1} \cup \{\mathbf{x}\}$ is a set of equiangular vectors.

Proof. Let $\{\boldsymbol{\varphi_k}\}_{i=1}^{n+1}$ be an ETF in $\mathbb{R}^n$. Equiangularity implies there exists some $a \in \mathbb{R}$ so that $|\langle \boldsymbol{\varphi_i}, \boldsymbol{\varphi_j} \rangle| = a$ for all $i \neq j$. Since these vectors form an ETF, we know they achieve equality for the Welch bound so $\mu(\Phi) = \frac{1}{n}$. Thusly, $a = 1/n$.

Suppose by way of contradiction that $\Psi = \{\boldsymbol{\varphi_i}\}_{i=1}^{n+1} \cup \{\mathbf{x}\}$ is an equiangular set of vectors. Then the Welch bound implies $\mu(\Psi) \geq \sqrt{\frac{2}{n(n+1)}}$.

Notice $\sqrt{\frac{2}{n(n+1)}} > \frac{1}{n}$ for $n > 1$ which means $\mu(\Psi) > \frac{1}{n} = \mu(\Phi) = |\langle \boldsymbol{\varphi_i}, \boldsymbol{\varphi_j} \rangle|$ for all $i \neq j$. But this means $|\langle \boldsymbol{\varphi_i}, \mathbf{x} \rangle| \neq \frac{1}{n}$ which contradicts our assumption that Ψ is a set of equiangular vectors with $|\langle \boldsymbol{\varphi_i}, \boldsymbol{\varphi_j} \rangle| = \frac{1}{n}$ for all $i \neq j$. Therefore no such $\mathbf{x}$ exists.

□

Lemma 2.4.3. If $\{\boldsymbol{\varphi_i}\}_{i=1}^{n+1}$ is a balanced set of unit vectors in $\mathbb{R}^n$ where n is even, then there exists no unit vector $\mathbf{x} \in \mathbb{R}^n$ so that $|\langle \boldsymbol{\varphi_i}, \mathbf{x} \rangle|$ is the same for all $i = 1, 2, \ldots, n+1$.

Proof. Let $\{\boldsymbol{\varphi_i}\}_{i=1}^{n+1}$ be a set of balanced vectors in $\mathbb{R}^n$. Any $\mathbf{x} \in \mathbb{R}^n$ that satisfies $|\langle \mathbf{x}, \boldsymbol{\varphi_i} \rangle| = a$ for all i also is a solution to $\Phi^T \mathbf{x} = \mathbf{a}$. Where $\mathbf{a} := [\epsilon_1 a, \epsilon_2 a, \ldots, \epsilon_{n+1} a]^T$ and $\epsilon_i = \pm 1$ for all k. Then we construct the augmented matrix $[\Phi^T \ \mathbf{a}]$ and do the row reduction $\boldsymbol{\varphi_{n+1}} \to \boldsymbol{\varphi_{n+1}} + \sum_{k=1}^{n} \boldsymbol{\varphi_i}$. Φ is balanced, so $\sum_{i=1}^{n+1} \boldsymbol{\varphi_i} = 0$. This gives us the new augmented matrix

$$\begin{bmatrix} \boldsymbol{\varphi}_1^T & \epsilon_1 a \\ \boldsymbol{\varphi}_2^T & \epsilon_2 a \\ \vdots & \vdots \\ \boldsymbol{\varphi}_n^T & \epsilon_n a \\ \mathbf{0}^T & a \sum_{i=1}^{n+1} \epsilon_i \end{bmatrix}$$

Notice for even n we have $a\sum_{i=1}^{n+1}\epsilon_i \neq 0$ which implies that this system does not have a solution and therefore no such $\mathbf{x}$ exists. □

Using the two Lemmas above, we can show a result regarding the flag median and the ℓ_2-median of balanced ETFs.

Theorem 2.4.4. *Suppose* $\{\mathbf{x_i}\}_{i=1}^{n+1} \subset \mathbb{R}^n$ *is a balanced ETF when* n *is even, then the* ℓ_2*-median and the flag median are all* $[\mathbf{y}] \in \{[\mathbf{x_i}]\}_{i=1}^{n+1}$.

Proof. Let $\{\mathbf{x_i}\}_{i=1}^{n+1}$ be a balanced ETF in $\mathbb{R}^n$, so they achieve equality for the Welch bound so $\mu(\Phi) = 1/n$.

$F_1(\mathbf{y})$ and $F_2(\mathbf{y})$ are defined in (2.22) and (2.23). We want to solve $\min_{\mathbf{y}^T\mathbf{y}=1} F_1(\mathbf{y})$ and $\min_{\mathbf{y}^T\mathbf{y}=1} F_2(\mathbf{y})$. Note: these are the 1-dimensional realizations of (2.3) and (2.12) respectively.

Now we will solve $\nabla F_j(\mathbf{y}) = 0$ or undefined for $j = 1,2$.

$$\nabla F_1(\mathbf{y}) = \sum_{i=1}^{n+1} \frac{-1}{|\langle \mathbf{x_i}, \mathbf{y}\rangle|\sqrt{1-\langle \mathbf{x_i}, \mathbf{y}\rangle^2}} \langle \mathbf{x_i}, \mathbf{y}\rangle \mathbf{x_i} \tag{2.24}$$

$$\nabla F_2(\mathbf{y}) = \sum_{i=1}^{n+1} \frac{-1}{\sqrt{1-\langle \mathbf{x_i}, \mathbf{y}\rangle^2}} \langle \mathbf{x_i}, \mathbf{y}\rangle \mathbf{x_i}. \tag{2.25}$$

We have an undefined gradient only when $\mathbf{y} = \mathbf{x_i}$ for any $i = 1,2,\ldots,n+1$, So these are all potential minimizers of $F_j(\mathbf{y})$ for $j = 1,2$.

Any other optimizers must satisfy $\nabla F_j(\mathbf{y}) = 0$. We will show any solution to $\nabla F_j(\mathbf{y}) = 0$ cannot be a minimizer. For $j = 1$ we have

$$\sum_{i=1}^{n+1} \alpha_i \mathbf{x_i} = 0 \text{ where } \alpha_i = \frac{-1}{|\langle \mathbf{x_i}, \mathbf{y}\rangle|\sqrt{1-\langle \mathbf{x_i}, \mathbf{y}\rangle^2}} \langle \mathbf{x_i}, \mathbf{y}\rangle. \tag{2.26}$$

and for $j = 2$ we have

$$\sum_{i=1}^{n+1} \beta_i \mathbf{x_i} = 0 \text{ where } \beta_i = \frac{-1}{\sqrt{1-\langle \mathbf{x_i}, \mathbf{y}\rangle^2}} \langle \mathbf{x_i}, \mathbf{y}\rangle. \tag{2.27}$$

Define the vectors $\boldsymbol{\alpha} = [\alpha_1 \cdots \alpha_{n+1}]^T$ and $\boldsymbol{\beta} = [\beta_1 \cdots \beta_{n+1}]^T$. Then we have $\boldsymbol{\Phi}\boldsymbol{\alpha} = 0$ and $\boldsymbol{\Phi}\boldsymbol{\beta} = 0$ where $\boldsymbol{\Phi}$ is the synthesis matrix for $\{\boldsymbol{x_i}\}_{i=1}^{n+1}$. Since Φ is full spark, the dimension of the null space

of $\mathbf{\Phi}$ is exactly 1. $\alpha_1 = \alpha_2 = \cdots = \alpha_{n+1} = 1$ is a solution to $\mathbf{\Phi\alpha} = 0$ because the columns of $\mathbf{\Phi}$ form a balanced frame. So the set of all solutions for minimizing F_1 and F_2 is $\{r\mathbf{1_n} : r \in \mathbb{R}\}$ where $\mathbf{1_n}$ is the n- dimensional vector of all ones.

Now suppose $\alpha_i = r$ for all i. Then

$$\frac{-1}{|\langle \mathbf{x_i}, \mathbf{y}\rangle|\sqrt{1-\langle \mathbf{x_i}, \mathbf{y}\rangle^2}}\langle \mathbf{x_i}, \mathbf{y}\rangle = r \iff \langle \mathbf{x_i}, \mathbf{y}\rangle^2 = \frac{r^2-1}{r^2}$$

Similarly, suppose $\beta_i = r$ for all i. Then

$$\frac{-1}{\sqrt{1-\langle \mathbf{x_i}, \mathbf{y}\rangle^2}}\langle \mathbf{x_i}, \mathbf{y}\rangle = r \iff \langle \mathbf{x_i}, \mathbf{y}\rangle^2 = \frac{r^2}{1+r^2}$$

This means $|\langle \mathbf{x_i}, \mathbf{y}\rangle|$ is constant for all i for both problems. Since these vectors are balanced, we can use Lemma 2.4.3. Therefore no such $\mathbf{y}$ exists. □

Corollary 2.4.5. *Suppose $\{\boldsymbol{\varphi_i}\}_{i=1}^{n+1} \subset \mathbb{R}^n$ is an ETF and n is even. Then the ℓ_2-median and the flag median are all $[\mathbf{y}] \in \{[\boldsymbol{\varphi_k}]\}_{k=1}^{n+1}$.*

Proof. Let $\{\mathbf{x_i}\}_{i=1}^{n+1} \subset \mathbb{R}^n$ be a balanced ETF. If we use the subspaces spanned by these frame vectors as our data, then Theorem 2.4.4 tells us that these subspaces are the minimizers of (2.22) and (2.23).

Let $\mathbf{X}$ and $\mathbf{\Phi}$ be the respective synthesis matrices for these frames. We know this balanced ETF is switching equivalent to $\{\boldsymbol{\varphi_k}\}_{k=1}^{n+1}$. So $\mathbf{\Phi} = \mathbf{UX\Lambda}$. Given the restriction that $\mathbf{\Lambda}$ is diagonal

with unimodular entries, it can only have entries of ± 1. So we have

$$\begin{aligned} G_1(\mathbf{y}) &:= \sum_{i=1}^{n+1} \arccos\left(\left|\langle \boldsymbol{\varphi_i}, \mathbf{y}\rangle\right|\right) \\ &= \sum_{i=1}^{n+1} \arccos\left(\left|\langle \mathbf{U}\mathbf{x_i}\lambda_i, \mathbf{y}\rangle\right|\right) \\ &= \sum_{i=1}^{n+1} \arccos\left(\left|\mathbf{y}^T\mathbf{U}\mathbf{x_i}\lambda_k\right|\right) \\ &= \sum_{i=1}^{n+1} \arccos\left(\left|\mathbf{y}^T\mathbf{U}\mathbf{x_i}\right|\right) \\ &= \sum_{i=1}^{n+1} \arccos\left(\left|\left(\mathbf{U}^T\mathbf{y}\right)^T\mathbf{x_i}\right|\right) \\ &= \sum_{i=1}^{n+1} \arccos\left(\left|\left\langle \mathbf{x_i}, \mathbf{U}^T\mathbf{y}\right\rangle\right|\right) \end{aligned}$$

Let $\mathbf{z} = \mathbf{U}^T\mathbf{y}$. Then we have

$$\min_{\mathbf{z}} \sum_{i=1}^{n+1} \arccos(|\langle \mathbf{x_i}, \mathbf{z}\rangle|).$$

The minimizers are $\{\mathbf{x_i}\}_{k=1}^{n+1}$. So the minimizers of $G_1(\mathbf{y})$ are $\{\mathbf{U}\mathbf{x_i}\}_{k=1}^{n+1}$ which are $\{\boldsymbol{\varphi_k}\}_{i=1}^{n+1}$ up to a change in sign.

A similar argument shows that the minimizers of

$$G_2(\mathbf{y}) = \min_{\mathbf{y}} \sum_{i=1}^{n+1} \sqrt{1 - \langle \boldsymbol{\varphi_i}, \mathbf{y}\rangle^2} \text{ subject to } \|\mathbf{y}\|_2 = 1$$

are $\{\boldsymbol{\varphi_k}\}_{i=1}^{n}$ up to a change in sign. □

2.5 Experiments

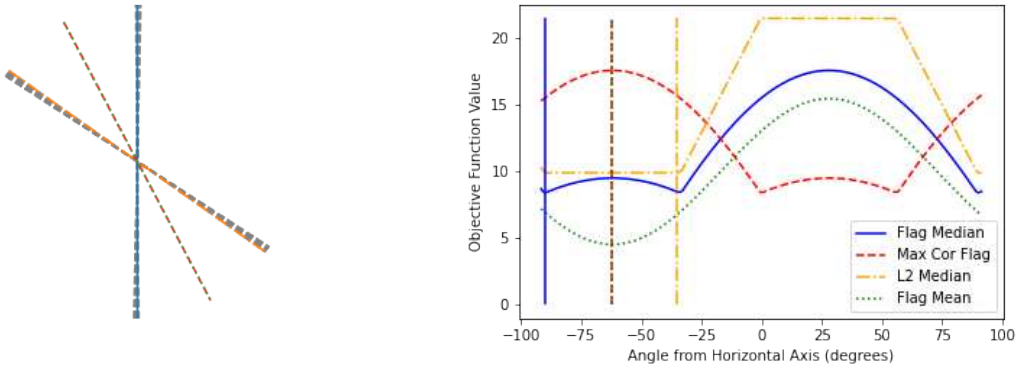
We use examples on $\mathrm{Gr}(1,2)$, $\mathrm{Gr}(1,3)$ and $\mathrm{Gr}(2,4)$ to investigate the behavior of the flag mean, ℓ_2-median, flag median and maximally correlated flag for certain subspace configurations. Section 2.5.1 uses an example on $\mathrm{Gr}(1,2)$ to highlight the difference between the subspace prototypes. Then Section 2.5.2 looks at central subspace prototypes of frames. Finally, Section 2.5.3

elaborates on Section 2.5.1 by looking at central subspace prototype behavior on a non-trivial Grassmannian, $\mathrm{Gr}(2,4)$.

2.5.1 $\mathrm{Gr}(1,2)$

What follows is an example using points on $\mathrm{Gr}(1,2)$ that are generated from 2 processes in Figure 2.1. Process (i) samples 10 vectors from a normal distribution with mean $[0,1]^T$ and standard deviation 0.01. Process (ii) samples 10 vectors normal distribution with mean $[0.66,-0.33]^T$ and standard deviation 0.01. Every vector from each process is normalized to get a data set of 20 points on $\mathrm{Gr}(1,2)$. We compute the flag mean of these data using the SVD. The flag median, maximally correlated flag, and ℓ_2-median are computed by sampling points on $\mathrm{Gr}(1,2)$. Specifically, we sampled 640 points $\{[\mathbf{z}_i]\}_{i=-320}^{319} \in \mathrm{Gr}(1,2)$ by taking $\mathbf{z}_i = [\cos(i),\sin(i)]^T$. Each central prototype is $\mathbf{z}_i$, which optimizes the objective function value for each prototype.

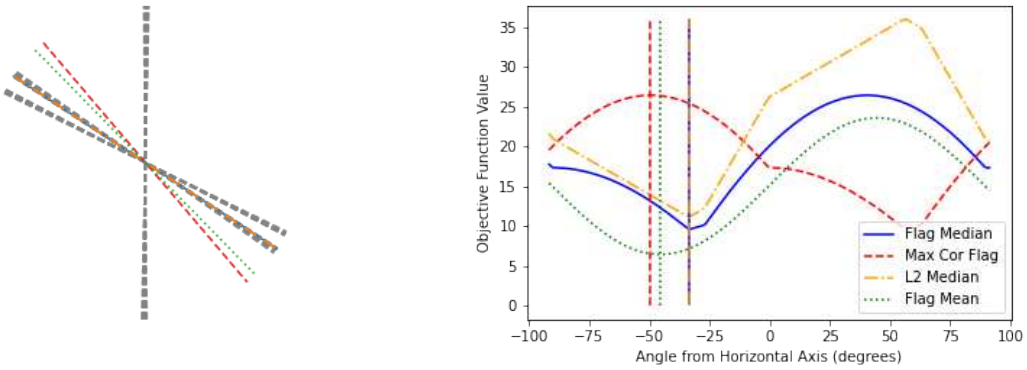
Figure 2.1 provides fuel for understanding when to use each method. Notice points in process (i) and process (ii) are in two distinct clusters with low inter-cluster variance. The maximally correlated flag and flag median behave similarly to a simple mean in $\mathbb{R}^2$ because their optimizers lie halfway between the two point clusters. In contrast, the flag median behaves closer to a Euclidean median because its global minimizers live in the two point clusters. The flag median objective function has a local maximum where the maximally correlated flag and flag median have optimizers. The ℓ_2-median splits the difference between these two objective function behaviors. It has global minimizers for all points near and between points from process (i) and (ii).



(a) **(b)**

Figure 2.1: The grey lines in (a) represent points on Gr(1,2) from process (i) and (ii). Vertical lines in (b) correspond with the colored lines in (a) and represent the calculated solutions for these optimization problems.

We proceed to a second example, where we add a third set of 10 points from process (iii). These points are generated by sampling vectors from a normal distribution with mean $[0.66, -0.33]^T$ and standard deviation 0.01, then normalizing them to get unit vector subspace representatives. Figure 2.2 shows the results from this experiment.



(a) **(b)**

Figure 2.2: The grey lines in (a) represent points on Gr(1,2) from process (i), (ii) and (iii). Vertical lines in (b) correspond with the colored lines in (a) and represent the calculated solutions for these optimization problems.

Adding the points from process (iii) distinguishes between the maximally correlated flag and the flag mean. This is because the flag mean is pulled towards the points in process (iii), whereas the maximally correlated flag is less affected by the new points. The flag median and ℓ_2-median share a global minimizer at the process (ii) side of the cluster of points from process (iii). So the flag median and ℓ_2-median are continuing to behave similarly to a traditional median by remaining closer to the actual points in a cluster.

In both figures, the objective function value curves for the flag median and maximally correlated flag are horizontal translations of one another by 90 degrees because $\sin(\theta) = \cos(\theta + \pi/2)$.

2.5.2 Gr(1,3)

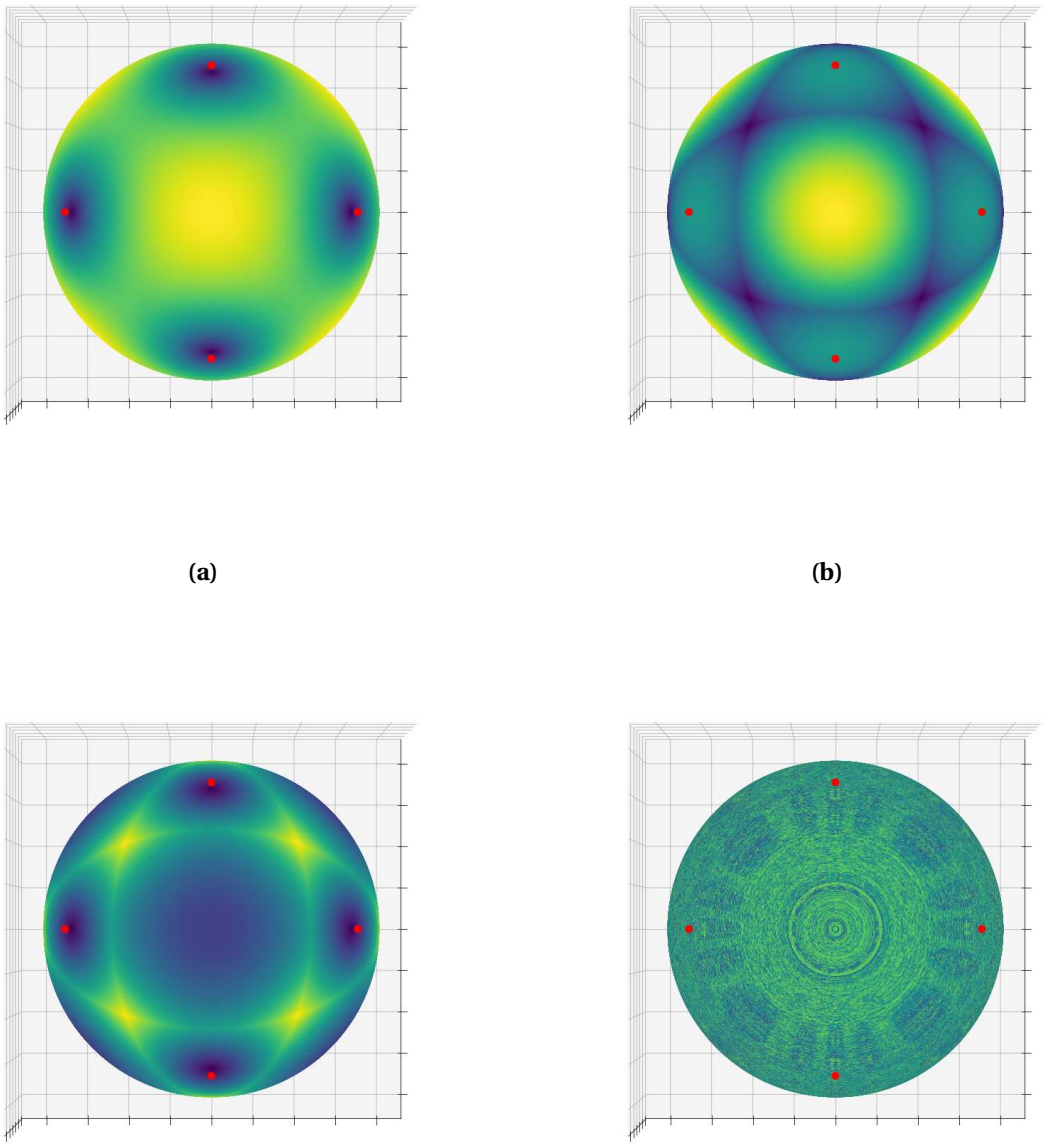
Consider an ETF of $n+1$ points in $\mathbb{R}^3$ that represents the set $A = \{[\mathbf{x}_1], [\mathbf{x}_2], [\mathbf{x}_3], [\mathbf{x}_4]\} \in \mathrm{Gr}(1,3)$. Specifically, these representatives are

$$A = \left\{ \begin{bmatrix} \sqrt{2/3} \\ 0 \\ \sqrt{1/3} \end{bmatrix}, \begin{bmatrix} 0 \\ \sqrt{2/3} \\ \sqrt{1/3} \end{bmatrix} \begin{bmatrix} -\sqrt{2/3} \\ 0 \\ \sqrt{1/3} \end{bmatrix}, \begin{bmatrix} 0 \\ -\sqrt{2/3} \\ \sqrt{1/3} \end{bmatrix} \right\}. \tag{2.28}$$

The flag mean is computed by SVD. The flag median, maximally correlated flag and ℓ_2-median are approximated by sampling objective function values at a set of random points $\{[\mathbf{z}_{i,j}]\} \subset \mathrm{Gr}(1,3)$. The central prototypes are the $[\mathbf{z}_{i,j}]$ which produce the optimal objective function values and are sampled using spherical coordinates as follows.

$$\mathbf{z}_{i,j} = [\sin(j)\cos(i), \sin(j)\sin(i), \cos(j)]^T. \tag{2.29}$$

We sample $i = -320, -319, \ldots 319$ and $j = -320, -319, \ldots 319$.



(a) **(b)**

(c) **(d)**

Figure 2.3: The red dots are in A. The colors in each plot correspond to following objective function values: (a) is the flag median, (b) is the maximally correlated flag, (c) is the ℓ_2-median and (d) is the flag mean. Yellow and blue indicate large and small objective values respectively.

Notice that the flag median, maximally correlated flag and the ℓ_2-medians are exactly the points in A whereas the flag mean is every point in $\mathrm{Gr}(1,3)$. This suggests that Corollary 2.4.5 applies to $\mathrm{Gr}(1,n)$ where n is odd. The fact that the flag mean is every point in $\mathrm{Gr}(1,3)$ the data set follows from Theorem 2.4.1.

2.5.3 $\mathrm{Gr}(2,4)$

In this experiment we sample two clusters of points on $\mathrm{Gr}(2,4)$ with centers $\mathbf{C}_1$ and $\mathbf{C}_2$. Cluster 1 has 30 points and cluster 2 has 70 points. We generate the ith point in cluster j using the following steps:

1. Compute the cluster center, $\mathbf{C}_j$
   (a) Sample a 4×2 random matrix with real entries from a uniform distribution over $[0,1)$
   (b) C_j is the first 2 columns of $\mathbf{Q}$ from the QR-decomposition of the random matrix.
2. Generate the ith point
   (a) Sample $\mathbf{Z}_i \in \mathbb{R}^{4\times 2}$ with real entries from a uniform distribution over $[-.5,.5)$.
   (b) Compute $\mathbf{P}_{i,j} = \mathbf{C}_j + 0.05\mathbf{Z}_i$
   (c) Take the 2 columns of $\mathbf{Q}$ from the QR-decomposition of $\mathbf{P}_{i,j}$

Then we sample the objective function values of the flag median, maximally correlated flag, ℓ_2-median and flag mean for 5000 random points on $\mathrm{Gr}(2,4)$ that are near these clusters. Below is an outline for how we sample these points.

1. Calculate the center, $\mathbf{C}$.
   (a) $\widetilde{\mathbf{C}} = (\mathbf{C}_1 + \mathbf{C}_2)/2$
   (b) $\mathbf{C} = \widetilde{\mathbf{C}}/\|\widetilde{\mathbf{C}}\|_F$
2. Calculate the point

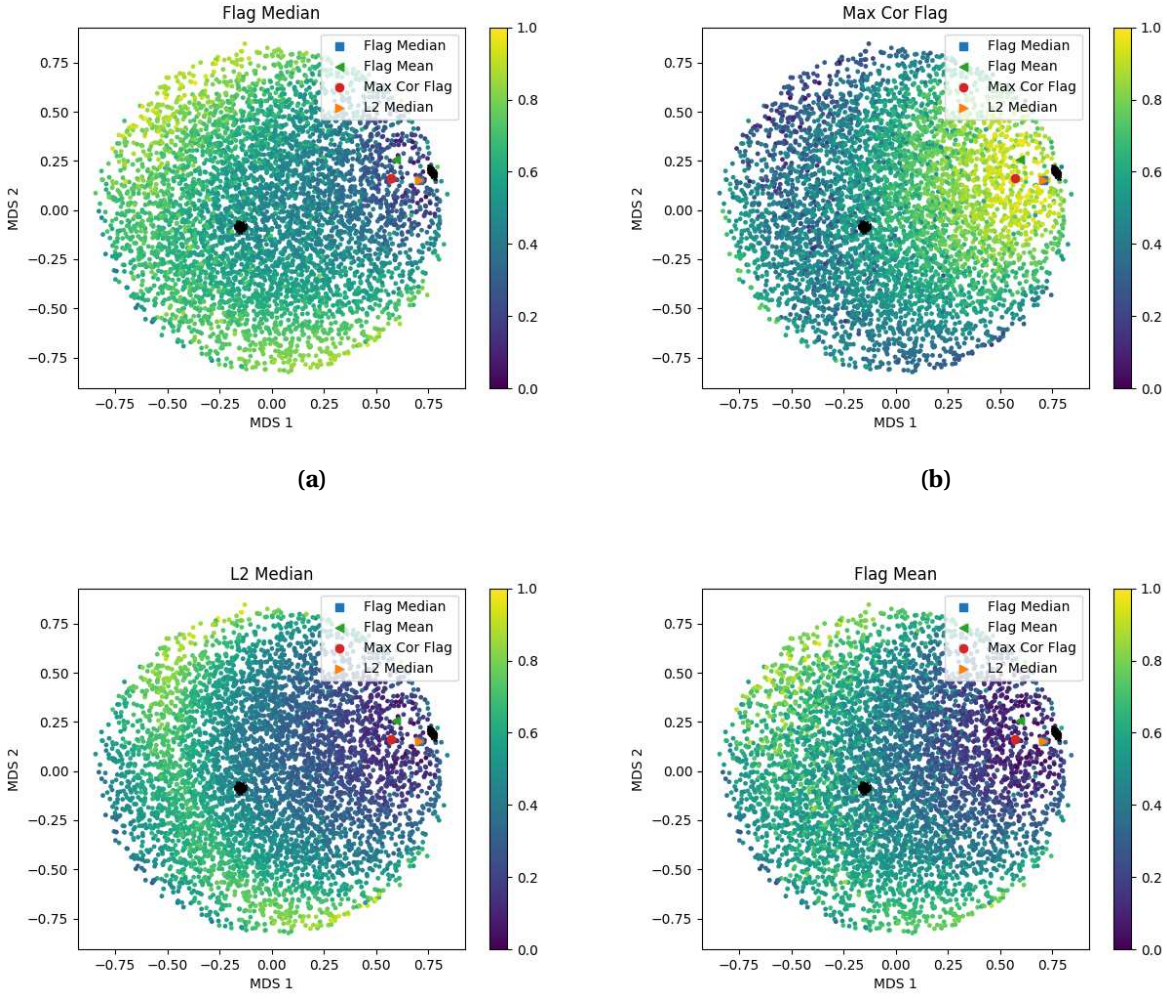
(a) Sample $\mathbf{Z} \in \mathbb{R}^{4\times 2}$ with real entries from a uniform distribution over $[-.5, .5)$.

(b) The random point on $\mathrm{Gr}(2,4)$ is the first 2 columns of $\mathbf{Q}$ from the QR-decomposition of $\mathbf{C} + 0.5\mathbf{Z}$.

Then we do a 2-dimensional Multidimensional Scaling (MDS) embedding of the chordal distance matrix of all the randomly sampled points and the two clusters. We plot this embedding in Figure 2.4.

Notice that the points with the smallest flag median objective function (purple points) have the smallest variance in the MDS plots. The ℓ_2-median and flag mean have similar variance in their low objective function value points (purple points). The maximally correlated flag has a similar variance in its high objective function value points (yellow points) as the ℓ_2-median and flag mean.

Now, we turn our attention to the estimated subspace prototypes from this example. We can use the distance in the MDS embedding to cluster 2 to determine which subspace prototypes are most pulled towards the outlier cluster. The "inlier" (non-outlier) cluster is on the right (cluster 1) and the outlier cluster (cluster 2) is on the left. In order from the closest to cluster 2 to the furthest from cluster 2 in the MDS embedding, we have the 1) flag median and ℓ_2-median, 2) flag mean, 3) maximally correlated flag. This aligns with the notion that medians are the least affected prototype by outliers. Although the sampled ℓ_2-median objective function values have a similar distribution to the flag mean, our estimate of the ℓ_2-median is closer to cluster 2 than the flag mean.



(a) **(b)**

(c) **(d)**

Figure 2.4: The black blobs are the points in each cluster. The blob on the left is cluster 1 (30 points) and the blob on the right is cluster 2 (70 points). The colors in each plot correspond to following objective function values: (a) flag median, (b) maximally correlated flag, (c) ℓ_2-median and (d) flag mean. The central prototypes are chosen as the randomly sampled point with the optimal objective function for each objective function.

2.6 Conclusion

We have presented two new prototypes for clusters of points on the Grassmannian: the flag median and the maximally correlated flag. These prototypes are unified with other subspace prototypes by realizing them as solutions to optimization problems involving matrix norms of

the principal angle matrix. Then we showed the flag median, flag mean and ℓ_2-median lose their uniqueness when presented with a set of evenly spaced subspaces (eg. subspaces spanned by certain frames). Finally, we ran experiments comparing the flag median, maximally correlated flag, flag mean, and the ℓ_2-median. In our experiments, we found the flag median is more robust to outliers than the other central subspace prototypes.

Future work with these prototypes could add details to the mathematical theory. Using the principal angle matrix, we could formulate numerous other subspace prototype optimization problems. Also, we would like to find a domain on which each of the flag median, flag mean and maximally correlated flag problems are convex. Each of these problems boil down to finding the domain where their respective distance or similarity function is convex. Finally, these subspace prototypes could be generalized as prototypes of points on other manifolds where we measure distances using principal angles like the Stiefel manifold.

Chapter 3

FlagIRLS: An Algorithm for Calculating Subspace Prototypes

3.1 Introduction

Euclidean central prototypes vary in their solvability. For example, the Euclidean centroid is a simple average of vectors whereas the geometric median is not as straightforward to compute since it is not a least squares problem. An iterative algorithm for approximating a geometric median is the Weiszfeld algorithm [41]; each iteration of this algorithm is a weighted centroid problem. Thus, the Weiszfeld algorithm falls into a class known as Iteratively Re-weighted Least Squares (IRLS) algorithms.

IRLS algorithms are not only useful to find the geometric median in Euclidean space, they have been applied to solve a number of optimization problems in other spaces. IRLS-type algorithms have been used to find medians on Riemannian manifolds generally using the logarithm and exponential maps [33, 42–44]. One application of IRLS is leveraged to calculate ℓ_2-medians of elements of $SO(3)$ for applications to geometric computer vision problems [42]. More examples of IRLS to solve ℓ_1 optimization problems can be found in numerous popular works including [20, 45–49].

A final example of IRLS in computer vision is the IRLS algorithm to solve the Dual Principal Component Pursuit (DPCP) problem called DPCP-IRLS. The DPCP problem seeks a subspace representative for a data set with outliers. In contrast to the subspace prototype optimization problems from Chapter 2, the optimization problem for DPCP seeks to find orthogonal compliment to the subspace which contains the "inliers" (non-outlier points). This algorithm is used to solve a subspace optimization problem that minimizes the maximally correlated flag objective function and is applied to geometric problems in computer vision [21]. To the best of our

knowledge analysis of the convergence of DPCP-IRLS is an open problem which we will address in this chapter.

Motivated by intuitive optimization problems that minimize distance and maximize similarity, we introduced two new subspace prototypes in Chapter 2. In this chapter we will derive an efficient algorithm called, similar to DPCP-IRLS, called FlagIRLS which is used to calculate these prototypes. FlagIRLS is an IRLS algorithm on the Grassmannian that solves a weighted flag mean problem at each iteration similar to the way an iteration of the Weiszfeld algorithm solves a weighted centroid problem.

We will begin this chapter with showing that FlagIRLS can be used to find the 1-dimensional ℓ_2-median, r-dimensional flag median and r-dimensional maximally correlated flag. Then we will provide a light convergence analysis of FlagIRLS and DPCP-IRLS. The final sections in this chapter will identify the properties of FlagIRLS, the flag median, and maximally correlated flag. Additionally, we determine their utility in synthetic experiments and a computer vision application. A few experiments are used to determine the convergence rate of FlagIRLS. Others are used to provide evidence that the flag median is robust to outliers and can be used effectively in algorithms like Linde-Buzo-Grey (LBG) to produce improved clusterings on Grassmannians. Specifically, we will conduct experiments with these central subspace prototypes on synthetic data sets, the MNIST handwritten digits data set [50], the DARPA (Defense Advanced Research Projects Agency) Mind's Eye data set used in [28], and the UCF YouTube action data set [51]. We find that using FlagIRLS to compute the flag median converges in 4 iterations on a synthetic data set. We also see that Grassmannian LBG clustering with a codebook size of 20 and using the flag median produces at least a 10% improvement in cluster purity over Grassmannian LBG using the flag mean or ℓ_2-median on the Mind's Eye data set.

Contributions:

- FlagIRLS is a new algorithm for calculating novel subspace prototypes.

- Convergence analysis of FlagIRLS with theoretical guarantees for convergence and experiments showing how FlagIRLS converges in few iterations. We also prove a small convergence result about DPCP-IRLS.

- Experiments showing increased subspace clustering performance with the flag median prototype when compared to other subspace prototypes.

- Experiments suggesting that the flag median is more robust to outliers than other subspace prototypes.

3.2 Background

We begin this section with an introduction to the Weiszfeld algorithm and realize it as an IRLS algorithm. Then we mention how a Weiszfeld-type algorithm can be applied to find ℓ_2-medians on Riemannian manifolds. Finally, we provide an introduction to the DPCP problem and the DPCP-IRLS algorithm, an IRLS algorithm to find an optimal subspace representation of inlier data.

For context, the Weiszfeld algorithm for vectors in $\mathbb{R}^n$ is stated in Algorithm 1.

Algorithm 1: Weiszfeld Algorithm in $\mathbb{R}^n$

Data: $\{\mathbf{x}_i\}_{i=1}^p \subset \mathbb{R}^n$
Result: The geometric median $\mathbf{y} \in \mathbb{R}^n$
while *not converged* **do**
  $w_i = \frac{p}{\|\mathbf{x}_i - \mathbf{y}\|_2} \left(\sum_{k=1}^p \frac{1}{\|\mathbf{x}_k - \mathbf{y}\|_2} \right)^{-1}$;
  $\mathbf{y} \leftarrow \sum_{i=1}^p \frac{w_i \mathbf{x}_i}{p}$;
end

Note that each iteration of the Weiszfeld algorithm (see Algorithm 1) is the solution to the least squares problem (2.1) (with $A = \mathbb{R}^n$ and $q = 2$) for the weighted vectors $w_i \mathbf{x}_i$. The weights, w_i,

come from the fact that the geometric median $\mathbf{y}$ satisfies

$$\mathbf{y} = \left(\sum_{i=1}^{p} \frac{\mathbf{x}_i}{\|\mathbf{x}_i - \mathbf{y}\|_2} \right) \Big/ \left(\sum_{k=1}^{p} \frac{1}{\|\mathbf{x}_k - \mathbf{y}\|_2} \right).$$

3.2.1 Weiszfeld-type Algorithm

Fletcher et al. solve (2.3) for $q = 1$ by generalizing the Weiszfeld approach to Riemannian manifolds. In Section 3.5, we use the unweighted Weiszfeld-type algorithm from Fletcher et al. with geodesic distance to calculate the ℓ_2-median on the Grassmannian [33]. Let d be the maximum distance between points in the data set and let δ be the convergence parameter. We define $N_{d,\delta}$ as the number of iterations of one run of our implementation. The complexity of our implementation of this algorithm in Section 3.5 is $O\left(npk^2 N_{d,\delta}\right)$.

3.2.2 DPCP-IRLS Algorithm

DPCP-IRLS is a generalization of the Weiszfeld algorithm for finding a subspace orthogonal to the subspaces that contain the inliers of a dataset in $\mathbb{R}^n$ [20]. We will now provide a brief explanation of the DPCP problem along with the DPCP-IRLS algorithm.

Suppose we have a data set of inliers and outliers $\{\mathbf{x}_i\}_{i=1}^{p} \in \mathbb{R}^n$. Let $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_p]$. Dual PCA seeks a vector $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{y} \neq 0$, that is orthogonal to a hyperplane that contains the inliers of $\{\mathbf{x}_i\}_{i=1}^{p}$ by solving

$$\min_{\mathbf{y}} \left\| \mathbf{X}^T \mathbf{y} \right\|_0. \tag{3.1}$$

This problem is relaxed to the DPCP problem that finds the unit vector $\mathbf{y} \in \mathbb{R}^n$ that solves

$$\min_{\mathbf{y}} \left\| \mathbf{X}^T \mathbf{y} \right\|_1. \tag{3.2}$$

DPCP-IRLS solves a modification of (3.2) by finding a set of k orthogonal unit vectors $\{\mathbf{y}_i\}_{i=1}^{k}$. We do this by stacking $\{\mathbf{y}_i\}_{i=1}^{k}$ into the matrix $\mathbf{Y} \in \mathbb{R}^{n \times k}$. The optimization problem for DPCP-

IRLS is

$$\min_{\mathbf{Y}} \sum_{i=1}^{p} \|\mathbf{Y}^T \mathbf{x}_i\|_2. \tag{3.3}$$

We extend this to be an optimization over subspaces $\{[\mathbf{X}_i]\}_{i=1}^{p} \in \mathrm{Gr}(k_i, n)$. So this, potentially novel, generalized version of DPCP is formulated as

$$\min_{\substack{\mathbf{Y}\in\mathbb{R}^{n\times r}\\ \mathbf{Y}^T\mathbf{Y}=\mathbf{I}}} \sum_{i=1}^{p} \sqrt{\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y})} = \min_{\substack{\mathbf{Y}\in\mathbb{R}^{n\times r}\\ \mathbf{Y}^T\mathbf{Y}=\mathbf{I}}} \sum_{i=1}^{p} \|\cos(\theta(\mathbf{X}_i,\mathbf{Y}))\|_2. \tag{3.4}$$

Define the weights for DPCP-IRLS as

$$w_i(\mathbf{Y}) = \frac{1}{\max\{\|\cos\theta(\mathbf{Y},\mathbf{X}_i)\|_2, \epsilon\}}. \tag{3.5}$$

A generalized version of the DPCP-IRLS algorithm that approximates the solution to 3.4 is in 2. A derivation of this algorithm is outlined by Tsakiris et al. [20] and is similar to the derivation of FlagIRLS in Section 3.3. The complexity of DPCP-IRLS is $O\left(nN_\delta\left(\sum_{i=1}^{p} k_i\right)^2\right)$ where N_δ is the number of iterations and δ is the convergence parameter.

Algorithm 2: The generalized DPCP-IRLS algorithm [20]. The columns of $\mathbf{U}$ are sorted by their associated singular value from largest to smallest.

Input: A set of orthonormal subspace representatives $\{\mathbf{X}_i\}_{i=1}^{p}$ for $\{[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)\}_{i=1}^{p}$.

Output: An orthonormal subspace representative $\mathbf{Y}$ that solves the DPCP-IRLS optimization algorithm $[\mathbf{Y}] \in \mathrm{Gr}(r, n)$

while *not converged* **do**

assign each $w_i(\mathbf{Y})$;

$\mathbf{X} \leftarrow \left[\sqrt{w_1(\mathbf{Y})}\mathbf{X}_1 | \sqrt{w_2(\mathbf{Y})}\mathbf{X}_2 | \cdots | \sqrt{w_p(\mathbf{Y})}\mathbf{X}_p\right]$;

$\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{X}$ % calculate the SVD ;

$\mathbf{Y} \leftarrow \mathbf{U}_{:,n-r:n}$% last r non-zero columns of $\mathbf{U}$;

end

This algorithm is exactly N_δ times slower than the algorithm for finding the flag mean. The complexity of the flag mean algorithm is $O\left(n\left(\sum_{i=1}^{p} k_i\right)^2\right)$.

3.3 Derivations of FlagIRLS

In this section we derive algorithms to approximate solutions to the weighted ℓ_2-median ($r = 1$), the flag median and the maximally correlated flag problems. Each of these problems are $1,2$ matrix norm problems. Recall that the flag mean problem is a squared $2,2$ matrix norm problem. That is, we are deriving algorithms analogous to Weiszfeld and IRLS, that approximate solutions to $1,2$ matrix norm problems by solving iterated weighted $2,2$ matrix norm problems. So rather than find some closed form solution for each optimization problem, we derive an iteration of the FlagIRLS Algorithm.

We will use the following notation for this section:

- $m_i = \min\{r, k_i\}$
- $r = 1$, and $n > k_1, k_2, \ldots, k_p \geq 1$ for the ℓ_2-median problem
- $n > k_1, k_2, \ldots, k_p, r \geq 1$ for both the flag median and the maximally correlated flag problems
- The delta function:

$$\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases}$$

Now we will state a proposition about equivalence of principal angles.

Proposition 3.3.1. *Let* $[\mathbf{X}] \in \mathrm{Gr}(k, n)$ *and* $[\mathbf{Y}] \in \mathrm{Gr}(r, n)$. *Then*

$$\| \cos(\theta([\mathbf{X}], [\mathbf{Y}])) \|_2^2 = tr\big(\mathbf{Y}^T \mathbf{X} \mathbf{X}^T \mathbf{Y}\big).$$

Proof. Let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{Y}^T\mathbf{X}$ be the SVD. Let $m = \min\{r, k\}$. The non-zero singular values of $\mathbf{X}^T\mathbf{Y}$ in descending order are

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m > 0.$$

From [2]

$$\| \cos(\theta([\mathbf{X}],[\mathbf{Y}])) \|_2^2 = \sum_{i=1}^{m} \sigma_i^2.$$

Using the definition of trace, we have

$$\sum_{i=1}^{m} \sigma_i^2 = \operatorname{tr}(\mathbf{\Sigma}\mathbf{\Sigma}).$$

Now we use the facts that $\mathbf{\Sigma}$ is symmetric, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and properties of trace to finish our proof.

$$\begin{aligned}
\operatorname{tr}(\mathbf{\Sigma}\mathbf{\Sigma}) &= \operatorname{tr}\left(\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\right) \\
&= \operatorname{tr}\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\right) \\
&= \operatorname{tr}(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T) \\
&= \operatorname{tr}\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\left(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\right)^T\right) \\
&= \operatorname{tr}\left(\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}\right)
\end{aligned}$$

□

3.3.1 ℓ_2-Median

Proposition 3.3.2. *Let* $\{[\mathbf{X}_i]\}_{i=1}^{p}$ *be a collection of subspaces with each* $[\mathbf{X}_i] \in \operatorname{Gr}(k_i, n)$. *The* ℓ_2*-median optimization problem*

$$\operatorname*{arg\,min}_{[\mathbf{y}]\in\operatorname{Gr}(1,n)} \sum_{i=1}^{p} \omega_i \|\theta([\mathbf{X}_i],[\mathbf{y}])\|_2 \tag{3.6}$$

can be written as

$$\operatorname*{arg\,max}_{[\mathbf{y}]\in\operatorname{Gr}(1,n)} \sum_{i=1}^{p} w_i \|\cos\theta([\mathbf{X}_i],[\mathbf{y}])\|_2^2 \tag{3.7}$$

with weights

$$w_i = \frac{\omega_i}{|\cos\theta(\mathbf{y},\mathbf{X}_i)||\sin\theta(\mathbf{y},\mathbf{X}_i)|} \tag{3.8}$$

as long as $\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y} \in (0,1)$ *for all* i.

Proof. By Proposition 3.3.1 $\cos^2(\theta([\mathbf{X}_i],[\mathbf{y}])) = \mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}$. So $\|\theta([\mathbf{X}_i],[\mathbf{y}])\|_2 = \arccos\left(\sqrt{\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}}\right)$. Hence (3.6) can be rewritten as

$$\min_{\substack{\mathbf{y}\in\mathbb{R}^n \\ \mathbf{y}^T\mathbf{y}=1}} \sum_{i=1}^{p} \omega_i \arccos\left(\sqrt{\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}}\right). \tag{3.9}$$

We move on to solve this ℓ_2-median problem using the Lagrangian

$$\mathscr{L}(\mathbf{y},\lambda) = \sum_{i=1}^{p} \omega_i \arccos\left(\sqrt{\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}}\right) - \lambda(1-\mathbf{y}^T\mathbf{y}). \tag{3.10}$$

We calculate the gradient of $\mathscr{L}(\mathbf{y},\lambda)$ and set it equal to zero,

$$\begin{aligned} &\sum_{i=1}^{p} \frac{-\omega_i}{\sqrt{\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}(1-\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y})}}\mathbf{X}_i\mathbf{X}_i^T\mathbf{y} + 2\lambda\mathbf{y} = 0, \\ &\mathbf{y}^T\mathbf{y} = 1. \end{aligned} \tag{3.11}$$

This allows us to find stationary points of our objective function.

Now we use the definition of w_i from (3.8) and (3.11) to solve to 2λ:

$$2\lambda\mathbf{y} = \sum_{i=1}^{p} \frac{\omega_i}{\sqrt{\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}(1-\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y})}}\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}, \tag{3.12}$$

$$2\lambda\mathbf{y} = \sum_{i=1}^{p} w_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}, \tag{3.13}$$

$$2\lambda = \sum_{i=1}^{p} w_i\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}. \tag{3.14}$$

Notice, we wish to choose $\mathbf{y}$ to maximize 2λ so that we minimize (3.6). Therefore our optimization problem becomes

$$\operatorname*{argmax}_{[\mathbf{y}]\in \mathrm{Gr}(1,n)} \sum_{i=1}^{p} w_i\mathbf{y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}. \tag{3.15}$$

By Proposition 3.3.1, (3.15) is equivalent to (3.8).

□

The next step would be to consider this problem for $r > 1$. The idea would be to proceed in an analogous derivation. However, we cannot formulate this optimization problem using trace since arccos does not distribute across addition, meaning

$$\|\theta([\mathbf{X}],[\mathbf{Y}])\|_2 \neq \arccos\left(\|\cos\left(\theta([\mathbf{X}],[\mathbf{Y}])\right)\|_2^2\right). \tag{3.16}$$

Instead, we must formulate geodesic distance as

$$\sqrt{\sum_{i=1}^{r} \arccos\left(\sqrt{\mathbf{y}_i^T \mathbf{X}\mathbf{X}^T \mathbf{y}_i}\right)^2}.$$

If we move forward, formulate a Lagrangian and take the gradient, we are unable to derive the same equivalence of optimization problems as Proposition 3.3.2. This is largely due to the existence of the double square root.

3.3.2 Flag Median

We now state a similar proposition about the flag median. Except, this time, we are are not restricted to $r = 1$.

Proposition 3.3.3. *Let* $\{[\mathbf{X}_i]\}_{i=1}^p$ *be a collection of subspaces with each* $[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)$. *The flag median optimization problem*

$$\underset{[\mathbf{Y}]\in \mathrm{Gr}(r,n)}{\arg\min} \sum_{i=1}^{p} \omega_i \|\sin\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2 \tag{3.17}$$

can be written as a weighted flag mean problem

$$\underset{[\mathbf{Y}]\in \mathrm{Gr}(r,n)}{\arg\max} \sum_{i=1}^{p} w_i \|\cos\theta([\mathbf{X}_i],[\mathbf{Y}])\|_2^2 \tag{3.18}$$

with weights

$$w_i = \frac{\omega_i}{\| \sin\theta\,(\mathbf{Y}, \mathbf{X}_i)\,\|_2} \tag{3.19}$$

as long as $\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}) \in [0, m_i)$ for all i.

Proof. Using Proposition 3.3.1 and $\sin^2(\theta) = 1 - \cos^2(\theta)$, we re-write (3.17) as

$$\min_{\substack{\mathbf{Y}\in\mathbb{R}^{n\times r}\\ \mathbf{Y}^T\mathbf{Y}=I}} \sum_{i=1}^{p} \omega_i \left(m_i - \mathrm{tr}(\mathbf{Y}_j^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}_j) \right)^{1/2}. \tag{3.20}$$

Then we formulate a Lagrangian from this problem as (3.21) using $\mathbf{\Lambda}$ as a symmetric matrix of Lagrange multipliers with entries λ_{ij}.

$$\mathscr{L}(\mathbf{Y}, \mathbf{\Lambda}) = \sum_{i=1}^{p} \omega_i (m_i - \mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}))^{1/2} - \langle \mathbf{\Lambda}, \mathbf{I} - \mathbf{Y}^T\mathbf{Y} \rangle \tag{3.21}$$

Then the gradient of this Lagrangian is

$$\begin{aligned} \nabla_{\mathbf{y}_j}\mathscr{L} &= \sum_{i=1}^{p} \frac{-\omega_i}{(m_i - \mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}))^{1/2}} \mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j - 2\sum_{i,j=1}^{r} \lambda_{ij}\mathbf{y}_i, \\ \nabla_{\lambda_{ij}}\mathscr{L} &= \mathbf{y}_i^T\mathbf{y}_j - \delta_{ij}. \end{aligned} \tag{3.22}$$

Now, we solve $\nabla\mathscr{L} = 0$. Setting $\nabla_{\lambda_{ij}}\mathscr{L} = 0$ gives us $\mathbf{y}_i^T\mathbf{y}_j = \delta_{ij}$. Then, we make use of $\mathbf{y}_i^T\mathbf{y}_j = \delta_{ij}$ and (3.19).

$$\begin{aligned}
0 &= \nabla_{\mathbf{y}_j}\mathscr{L}, \\
0 &= \sum_{i=1}^{p} \frac{-\omega_i}{(m_i - \mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}))^{1/2}}\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j + 2\lambda_{jj}\mathbf{y}_j - 2\sum_{\substack{q=1 \\ q\neq j}}^{r}\lambda_{jq}\mathbf{y}_q, \\
0 &= \mathbf{y}_j^T\sum_{i=1}^{p} -w_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j + 2\lambda_{jj}\mathbf{y}_j^T\mathbf{y}_j - 2\sum_{\substack{q=1 \\ q\neq j}}^{r}\lambda_{jq}\mathbf{y}_j^T\mathbf{y}_q, \\
0 &= \mathbf{y}_j^T\sum_{i=1}^{p} -w_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j + 2\lambda_{jj}, \\
2\lambda_{jj} &= \sum_{i=1}^{p}\mathbf{y}_j^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j.
\end{aligned} \tag{3.23}$$

We combine these equations for all $j = 1, 2, \ldots r$ and arrive at

$$\sum_{j=1}^{r} 2\lambda_{jj} = \sum_{i=1}^{p} w_i\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}). \tag{3.24}$$

Minimizing (3.17) is equivalent to solving

$$\underset{[\mathbf{Y}]\in\mathrm{Gr}(r,n)}{\arg\max}\sum_{i=1}^{p} w_i\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}), \tag{3.25}$$

Which is equivalent to (3.18) by Proposition 3.3.1.

□

3.3.3 Maximally Correlated Flag

We go onto provide a similar proposition as before, but this time for the maximally correlated flag.

Proposition 3.3.4. *Let* $\{[\mathbf{X}_i]\}_{i=1}^p$ *be a collection of subspaces with each* $[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)$. *The maximally correlated flag optimization problem*

$$\underset{[\mathbf{Y}]\in \mathrm{Gr}(r,n)}{\arg\max} \sum_{i=1}^{p} \omega_i \| \cos\theta([\mathbf{X}_i],[\mathbf{Y}]) \|_2 \tag{3.26}$$

can be written as a weighted flag mean problem

$$\underset{[\mathbf{Y}]\in \mathrm{Gr}(r,n)}{\arg\max} \sum_{i=1}^{p} w_i \| \cos\theta([\mathbf{X}_i],[\mathbf{Y}]) \|_2^2 \tag{3.27}$$

with weights

$$w_i = \frac{\omega_i}{\| \cos\theta(\mathbf{Y},\mathbf{X}_i) \|_2} \tag{3.28}$$

as long as $\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}) \in (0, m_i]$ for all i.

Proof. Using Proposition 3.3.1, we re-write (3.26) as

$$\min_{\substack{\mathbf{Y}\in\mathbb{R}^{n\times r}\\ \mathbf{Y}^T\mathbf{Y}=I}} \sum_{i=1}^{p} \omega_i \left(\mathrm{tr}(\mathbf{Y}_j^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}_j). \right)^{1/2} \tag{3.29}$$

Then we formulate a Lagrangian from this problem as

$$\mathscr{L}(\mathbf{Y},\mathbf{\Lambda}) = \sum_{i=1}^{p} \omega_i (\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}))^{1/2} - \langle \mathbf{\Lambda}, \mathbf{Y}^T\mathbf{Y} - \mathbf{I} \rangle, \tag{3.30}$$

using $\mathbf{\Lambda}$ as a symmetric matrix of Lagrange multipliers with entries λ_{ij}.

Then the gradient of this Lagrangian is

$$\begin{aligned} \nabla_{\mathbf{y}_j}\mathscr{L} &= \sum_{i=1}^{p} \frac{\omega_i}{(\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}))^{1/2}} \mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j - 2\sum_{i,j=1}^{r} \lambda_{ij}\mathbf{y}_i, \\ \nabla_{\lambda_{ij}}\mathscr{L} &= \mathbf{y}_i^T\mathbf{y}_j - \delta_{ij}. \end{aligned} \tag{3.31}$$

Now we solve $\nabla\mathscr{L} = 0$. Setting $\nabla_{\lambda_{ij}}\mathscr{L} = 0$ gives us $\mathbf{y}_i^T\mathbf{y}_j = \delta_{ij}$. Then we make use of $\mathbf{y}_i^T\mathbf{y}_j = \delta_{ij}$ and (3.28) to find

$$\begin{aligned}
0 &= \nabla_{\mathbf{y}_j}\mathscr{L}, \\
0 &= \sum_{i=1}^{p} \frac{\omega_i}{(\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}))^{1/2}}\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j + 2\lambda_{jj}\mathbf{y}_j - 2\sum_{\substack{q=1 \\ q\neq j}}^{r}\lambda_{jq}\mathbf{y}_q, \\
0 &= \mathbf{y}_j^T\sum_{i=1}^{p} w_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j + 2\lambda_{jj}\mathbf{y}_j^T\mathbf{y}_j - 2\sum_{\substack{q=1 \\ q\neq j}}^{r}\lambda_{jq}\mathbf{y}_j^T\mathbf{y}_q, \\
0 &= \mathbf{y}_j^T\sum_{i=1}^{p} w_i\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j - 2\lambda_{jj}, \\
2\lambda_{jj} &= \sum_{i=1}^{p} w_i\mathbf{y}_j^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{y}_j.
\end{aligned} \tag{3.32}$$

We combine these equations for all $j = 1, 2, \ldots r$ and arrive at

$$\sum_{j=1}^{r} 2\lambda_{jj} = \sum_{i=1}^{p} w_i\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}). \tag{3.33}$$

Note that maximizing (3.26) is equivalent to solving

$$\underset{[\mathbf{Y}]\in\mathrm{Gr}(r,n)}{\arg\max} \sum_{i=1}^{p} w_i\mathrm{tr}(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}). \tag{3.34}$$

Which is equivalent to (3.27) by Proposition 3.3.1. □

3.3.4 The FlagIRLS Algorithm

We use the optimization problem equivalences from the previous subsections to derive an iteration of the FlagIRLS algorithm.

Fix $[\mathbf{Z}] \in \mathrm{Gr}(r,n)$ and define $w_i(\mathbf{Z}) \in \mathbb{R}$ for each of the ℓ_2-median, flag median and maximally correlated flag problems as shown in Table 3.1. We can think of w_i as the mapping

$$w_i : \mathbb{R}^{n\times r} \to \mathbb{R}.$$

To avoid singularities, we use a small positive quantity $\epsilon \in \mathbb{R}$ in the denominator of $w_i(\mathbf{Z})$.

Table 3.1: The weights for FlagIRLS labeled by the prototypes for which they are used.

Central Prototype	$w_i(\mathbf{Z})$
ℓ_2-median ($r = 1$)	$\omega_i / \max\{\lvert\cos\theta(\mathbf{Z},\mathbf{X}_i)\rvert\lvert\sin\theta(\mathbf{Z},\mathbf{X}_i)\rvert,\epsilon\}$
flag median	$\omega_i / \max\{\lVert\sin\theta(\mathbf{Z},\mathbf{X}_i)\rVert_2,\epsilon\}$
maximally correlated flag	$\omega_i / \max\{\lVert\cos\theta(\mathbf{Z},\mathbf{X}_i)\rVert_2,\epsilon\}$

Proposition 3.3.5. *Fix* $[\mathbf{Z}] \in \mathrm{Gr}(r,n)$ *to define* $w_i(\mathbf{Z})$ *as in Table 3.1. The solution to*

$$\underset{[\mathbf{Y}]\in\mathrm{Gr}(r,n)}{\arg\max} \sum_{i=1}^{p} w_i(\mathbf{Z}) \lVert \cos\theta([\mathbf{X}_i],[\mathbf{Y}]) \rVert_2^2 \tag{3.35}$$

is the flag mean of $\{[\mathbf{X}_i]\}_{i=1}^p\}$ *with weights* $w_i(\mathbf{Z})$.

Proof. Define the matrix

$$\mathbf{X} = \left[\sqrt{w_1(\mathbf{Z})}\mathbf{X}_1, \sqrt{w_2(\mathbf{Z})}\mathbf{X}_2, \cdots, \sqrt{w_p(\mathbf{Z})}\mathbf{X}_p \right].$$

Using principal angles and properties of trace, we can write the objective function as

$$\begin{aligned}
\sum_{i=1}^{p} w_i(\mathbf{Z}) \lVert \cos\theta([\mathbf{X}_i],[\mathbf{Y}]) \rVert_2^2 &= \sum_{i=1}^{p} w_i(\mathbf{Z}) \mathrm{tr}\left(\mathbf{Y}^T\mathbf{X}_i\mathbf{X}_i^T\mathbf{Y}\right), \\
&= \sum_{i=1}^{p} \mathrm{tr}\left(\mathbf{Y}^T\sqrt{w_i(\mathbf{Z})}\mathbf{X}_i\sqrt{w_i(\mathbf{Z})}\mathbf{X}_i^T\mathbf{Y}\right), \\
&= \mathrm{tr}\left(\mathbf{Y}^T\left(\sum_{i=1}^{p}\sqrt{w_i(\mathbf{Z})}\mathbf{X}_i\sqrt{w_i(\mathbf{Z})}\mathbf{X}_i^T\right)\mathbf{Y}\right), \\
&= tr\left(\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}\right).
\end{aligned} \tag{3.36}$$

So our optimization problem is equivalent to

$$\underset{[\mathbf{Y}]\in\mathrm{Gr}(r,n)}{\arg\max}\, \mathrm{tr}\left(\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}\right). \tag{3.37}$$

The solution, $\mathbf{Y}^*$, is the r left singular vectors of $\mathbf{X}$ associated to the r largest singular values. This is the same as $\mathbf{Y}^*$ being the weighted flag mean of $\{[\mathbf{X}_i]\}_{i=1}^p$ with weights $w_i(\mathbf{Z})$. See [3] for a proof of this equivalence. □

In the previous subsections, we have shown that the ℓ_2-median (with $r = 1$), flag median, and maximally correlated flag can be written as problems of the form (3.35). This form is a weighted flag mean problem by Proposition 3.3.5. We use this to suggest the FlagIRLS algorithm (see Algorithm 3) which approximates the ℓ_2-median (for $r = 1$), flag median, and maximally correlated flag.

Algorithm 3: The FlagIRLS algorithm. See Table 3.1 for the algorithm weights. We assume the columns of $\mathbf{U}$ are sorted from the smallest to the largest singular values of $\mathbf{X}$.

Input: A set of orthonormal subspace representatives $\{\mathbf{X}_i\}_{i=1}^p$ for $\{[\mathbf{X}_i] \in \mathrm{Gr}(k_i, n)\}_{i=1}^p$ with corresponding positive weights $\{\omega_i\}_{i=1}^p \subset \mathbb{R}$

Output: An orthonormal subspace representative $\mathbf{Y}$ for the flag median $[\mathbf{Y}] \in \mathrm{Gr}(r, n)$

while *not converged* **do**

assign each $w_i(\mathbf{Y})$;

$\mathbf{X} \leftarrow \left[\sqrt{w_1(\mathbf{Y})}\mathbf{X}_1 | \sqrt{w_2(\mathbf{Y})}\mathbf{X}_2 | \cdots | \sqrt{w_p(\mathbf{Y})}\mathbf{X}_p\right]$;

$\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{X}$ % calculate the SVD ;

$\mathbf{Y} \leftarrow \mathbf{U}_{:,1:r}$ % first r columns of $\mathbf{U}$;

end

What follows are a few notes on FlagIRLS.

- ***Complexity***

  The complexity of the FlagIRLS algorithm is the complexity of the flag mean times the number of iterations of the algorithm, i.e., $O\left(nN_\delta\left(\sum_{i=1}^p k_i\right)^2\right)$ *where* N_δ *is the number of iterations and* δ *is the convergence parameter.*

- ***Stochastic FlagIRLS***

  We randomly select a subset $B \subseteq \{[\mathbf{X}_i]\}_{i=1}^{p}$, $|B| = m$, to construct $\mathbf{X}$ at each iteration. Stochastic implementations of FlagIRLS are investigated in Figure 3.3.

- ***FlagIRLS outputs a flag of subspaces***

  FlagIRLS is an iterative weighted flag mean algorithm, so the outputs of this algorithm come from the left singular vectors of $\mathbf{X}$. We take r singular vectors associated with the r largest singular values of $\mathbf{X}$, i.e., the first r columns of $\mathbf{U}$. However, there are n columns of $\mathbf{U}$, so FlagIRLS actually outputs a flag of subspaces

  $$[\mathbf{U}_{:,1}] \subset [\mathbf{U}_{:,:2}] \subset \cdots \subset [\mathbf{U}_{:,:r}].$$

  This flag is used to distinguish between different prototypes in Section 3.5.2 with MNIST digits.

- ***Choosing the optimal*** r

  We can find the optimal r for the flag median by choosing r at every iteration in accordance to the criteria outlined by Santamaria et al. [36].

3.4 Convergence Results

In this section we show that iterations of FlagIRLS decrease the unweighted flag median objective function value and an iteration of DPCP-IRLS decreases its respective objective function value.

First we need to define the function $g : \mathbb{R}^{n\times r} \times \mathbb{R}^{n\times k} \to \mathbb{R}$. For FlagIRLS to find the flag median, this function is defined as $g(\mathbf{Y},\mathbf{X}) = \sqrt{\min\{k,r\} - \operatorname{tr}(\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y})}$. For DPCP-IRLS, we define $g(\mathbf{Y},\mathbf{X}) = \sqrt{\operatorname{tr}(\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y})}$. The objective functions for the flag mean and the DPCP-IRLS objective functions can then be written as

$$f(\mathbf{Y}) = \sum_{i=1}^{p} g(\mathbf{Y},\mathbf{X}_i). \tag{3.38}$$

An iteration of FlagIRLS for the flag median and DPCP-IRLS is the mapping $T : \mathbb{R}^{n\times r} \to \mathbb{R}$.

Now we discuss cases where $g(\mathbf{Y}, \mathbf{X}_i) = 0$. For the flag median, $g(\mathbf{Y}, \mathbf{X}_i) = 0$ only when $[\mathbf{Y}] = [\mathbf{X}_i]$. We investigated some cases where the flag median is at lease one of the data points (e.g., $g(\mathbf{Y}, \mathbf{X}_i) = 0$ in Chapter 2. In subsequent experiments, we find that the flag median is not always some $[\mathbf{X}_i]$. For DPCP-IRLS $g(\mathbf{Y}, \mathbf{X}_i) = 0$ only when $[\mathbf{Y}]$ and $[\mathbf{X}_i]$ are orthogonal subspaces. By construction of the DPCP-IRLS problem, we seek a $[\mathbf{Y}]$ that is orthogonal to the inlier subspaces. So, we expect $g(\mathbf{Y}, \mathbf{X}_i) < \epsilon$ for at least one i. In this case, the result in Theorem 3.4.1 becomes

$$f(T(\mathbf{Y})) \leq f(\mathbf{Y}) + \frac{p\epsilon}{2}.$$

Theorem 3.4.1. *Let* $[\mathbf{Y}] \in \mathrm{Gr}(r, n)$. *Suppose* $g(\mathbf{Y}, \mathbf{X}_i) > \epsilon$ *for* $i = 1, 2, \ldots, p$. *Then*

$$f(T(\mathbf{Y})) \leq f(\mathbf{Y}).$$

Proof. Define the function $h : \mathbb{R}^{n\times r} \times \mathbb{R}^{n\times r} \to \mathbb{R}$ as

$$\begin{aligned} h(\mathbf{Z}, \mathbf{Y}) &= \sum_{i=1}^{p} w_i(\mathbf{Y}) g(\mathbf{Z}, \mathbf{X}_i)^2, \\ w_i(\mathbf{Y}) &= \frac{1}{\max\{g(\mathbf{Y}, \mathbf{X}_i), \epsilon\}} = \frac{1}{g(\mathbf{Y}, \mathbf{X}_i)}. \end{aligned} \tag{3.39}$$

We claim that the minimizer of $\min_{\mathbf{Z}} h(\mathbf{Z}, \mathbf{Y})$ is exactly $\mathbf{Z} = T(\mathbf{Y})$. For FlagIRLS, $h(\mathbf{Z}, \mathbf{Y})$ is the weighted flag mean objective function. For DPCP-IRLS, $h(\mathbf{Z}, \mathbf{Y})$ is maximizing the weighted flag mean objective function. So an iteration of these respective algorithms solve the optimization problem

$$T(\mathbf{Y}) = \mathrm{argmin}_{\mathbf{Z}\in\mathbb{R}^{n\times r}} h(\mathbf{Z}, \mathbf{Y}). \tag{3.40}$$

Some algebra reduces $h(\mathbf{Z},\mathbf{Y})$ to

$$h(\mathbf{Z},\mathbf{Y}) = \sum_{i=1}^{p} w_i(\mathbf{Y}) g(\mathbf{Z},\mathbf{X}_i)^2, \tag{3.41}$$

$$= \sum_{i=1}^{p} \frac{g(\mathbf{Z},\mathbf{X}_i)^2}{g(\mathbf{Y},\mathbf{X}_i)}. \tag{3.42}$$

$$\tag{3.43}$$

Now we use the identity from algebra: $\frac{a^2}{b} \geq 2a - b$ for any $a, b \in \mathbb{R}$ and $b > 0$. Let

$$a = g(\mathbf{Z},\mathbf{X}_i) \text{ and } b = g(\mathbf{Y},\mathbf{X}_i). \tag{3.44}$$

Then

$$h(\mathbf{Z},\mathbf{Y}) \geq \sum_{i=1}^{p} \left(2g(\mathbf{Z},\mathbf{X}_i) - g(\mathbf{Y},\mathbf{X}_i)\right), \tag{3.45}$$

$$= 2\sum_{i=1}^{p} g(\mathbf{Z},\mathbf{X}_i) - \sum_{i=1}^{p} g(\mathbf{Y},\mathbf{X}_i), \tag{3.46}$$

$$= 2f(\mathbf{Z}) - f(\mathbf{Y}). \tag{3.47}$$

Now, take $\mathbf{Z} = T(\mathbf{Y})$. This gives us

$$h(T(\mathbf{Y}),\mathbf{Y}) \geq 2f(T(\mathbf{Y})) - f(\mathbf{Y}). \tag{3.48}$$

Using (3.40), we have

$$h(T(\mathbf{Y}),\mathbf{Y}) \leq h(\mathbf{Y},\mathbf{Y}). \tag{3.49}$$

Notice that $h(\mathbf{Y},\mathbf{Y}) \leq f(\mathbf{Y})$ by definition of h. Then

$$h(\mathbf{Y},\mathbf{Y}) = \sum_{i=1}^{p} \frac{g(\mathbf{Y},\mathbf{X}_i)^2}{g(\mathbf{Y},\mathbf{X}_i)}, \tag{3.50}$$

$$\leq \sum_{i=1}^{p} g(\mathbf{Y},\mathbf{X}_i), \tag{3.51}$$

$$= f(\mathbf{Y}). \tag{3.52}$$

This means, we have

$$h(T(\mathbf{Y}),\mathbf{Y}) \leq h(\mathbf{Y},\mathbf{Y}) \leq f(\mathbf{Y}). \tag{3.53}$$

Then we can combine (3.48) with (3.53) and we have

$$2f(T(\mathbf{Y})) - f(\mathbf{Y}) \leq f(\mathbf{Y}), \tag{3.54}$$

$$f(T(\mathbf{Y})) \leq f(\mathbf{Y}). \tag{3.55}$$

□

Corollary 3.4.2. *The objective function values of the FlagIRLS for the flag median and DPCP-IRLS converge as long as $g(\mathbf{Y},\mathbf{X}_i) > \epsilon$ for $i = 1,2,\ldots,p$.*

Proof. Let $[\mathbf{Y}_k] \in \mathrm{Gr}(r,n)$ be an iterate of FlagIRLS or DPCP-IRLS. Notice that the real sequence $f(\mathbf{Y}_k) \in \mathrm{Gr}(r,n)$ is bounded below by 0 and is decreasing by Theorem 3.4.1. Therefore it converges. □

3.5 Experiments

In this section we carry out experiments with synthetic data, the MNIST handwritten digits data set [50], the Mind's Eye data set [28], and the UCF YouTube action data set [51]. One goal is to compare the flag median and maximally correlated flag to the flag mean and ℓ_2-median. Another goal is to establish the efficiency of FlagIRLS as an algorithm for computing some of

these prototypes. For all of this section we use FlagIRLS to compute the flag median, maximally correlated flag, and ℓ_2 median (for $r = 1$), and the Weiszfeld-type algorithm from [33] is used for the ℓ_2-median (for $r > 1$).

Now we discuss the convergence criteria for the iterative algorithms used in this section. We terminate the FlagIRLS algorithm when the objective function values of consecutive iterates are less than $\delta = 10^{-11}$, or if the i^{th} iteration resulted in an increasing objective function value. In the former case, we output the $(i-1)^{\text{st}}$ iterate. We run the FlagIRLS algorithm with FlagIRLS weights with $\epsilon = 10^{-11}$. We also use the Weiszfeld-type algorithm from [33] to calculate the ℓ_2-median. This algorithm is terminated when objective function values of consecutive iterates are less than $\delta = 10^{-11}$. Both FlagIRLS and the Weiszfeld-type algorithm are terminated when we have exceeded 1000 iterations. *Note: FlagIRLS never exceeds* 1000 *iterations in our examples.*

Let d_k be the cluster distortion at iteration k for LBG clustering. We define the clustering error at iteration k as

$$\eta_k = \frac{|d_k - d_{k-1}|}{d_{k-1}}. \tag{3.56}$$

We terminate LBG clustering when $\eta_k < 10^{-5}$.

3.5.1 Synthetic Data

We begin with an experiment on a data set consisting of 10 points from $\text{Gr}(3,20)$ and 10 points from $\text{Gr}(5,20)$. A representative for a point on $\text{Gr}(k,n)$ is sampled in two steps. The first step is to sample an $n \times k$ matrix from a uniform distribution on $[-.5,.5)$, $\mathscr{U}[-.5,.5)$. We then do the QR decomposition of this matrix to get a point on $\text{Gr}(k,n)$. We use this data set to verify the convergence of FlagIRLS.

We run 100 trials of FlagIRLS on these data with different random initializations to calculate 1-dimensional ℓ_2-medians, 3-dimensional flag medians, and 3-dimensional maximally correlated flags. For each of these trials, we verify that convergence by checking 100 points near the FlagIRLS algorithm output. Given one algorithm output, $[\mathbf{X}] \in \text{Gr}(3,20)$, we sample the entries of $\mathbf{Y} \in \mathbb{R}^{20\times 3}$ from $\mathscr{U}[-0.5,0.5)$ and check the objective function value at the first 3 columns of

$\mathbf{Q}$ where $\mathbf{Q}$ comes from the QR decomposition of the matrix $\mathbf{X} + 0.00001\mathbf{Y}$. We call these points "test points" for the algorithm output. We say the FlagIRLS algorithm converges when all the objective function values of the test points are less than or equal to the objective function value for the algorithm output. In this experiment, we find that 100% of the FlagIRLS trials converge for the flag median and ℓ_2-median. However, only 68% of the FlagIRLS trials converge for the maximally correlated flag. Since FlagIRLS does not converge to the maximally correlated flag for all of these trials, we choose not to pursue further experiments with using FlagIRLS to find this prototype.

We now show an example with a similar data set to compare the convergence rate of FlagIRLS to Grassmannian gradient descent. This time we take 20 points from $\mathrm{Gr}(3,20)$. We sample these points in the same way as the first example. Then we run FlagIRLS and Grassmannian gradient descent with 100 random initializations to compute the flag median. The results of this experiment are in Figure 3.1. For this example, Grassmannian gradient descent is implemented with a step size of 0.01. We find that FlagIRLS converges in fewer iterations than Grassmannian gradient descent for the flag median. In addition, we see that the convergence of FlagIRLS is less sensitive to the algorithm initialization than gradient descent due to the standard deviation of the objective function values at each iteration.

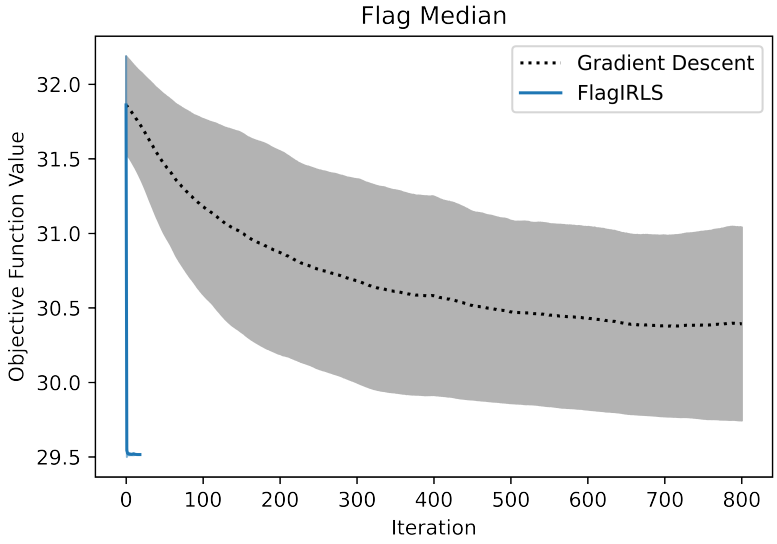


Figure 3.1: The mean objective function values over 100 trials with different random initializations. FlagIRLS converges in fewer iterations than gradient descent for the flag median problem on the synthetic data set. The shaded region is one standard deviation away from the mean objective function value.

Now we compare FlagIRLS to the Weiszfeld-type algorithm to calculate the 1-dimensional ℓ_2-median. We run the same experiment as above with only 1-dimensional subspaces and find that FlagIRLS did not converge to the ℓ_2-median. So, we run a different experiment where we sample 10 random subspaces which are "close" to each other. We do this by first sampling a vector in $\mathbf{c} \in \mathbb{R}^{20}$ from $\mathscr{U}[-0.5, 0.5)$. Then we sample $\mathbf{r_i} \in \mathbb{R}^{20}$ from $\mathscr{U}[-0.05, 0.05)$. Our ith data point is

$$\mathbf{x}_i = \frac{\mathbf{c} + \mathbf{r_i}}{\|\mathbf{c} + \mathbf{r_i}\|_2}.$$

This results in a cluster of 1-dimensional subspaces, namely $[\mathbf{x}_i]$, that are close to one-another. In Figure 3.2 we see that FlagIRLS converges much faster than the Weiszfeld-type algorithm.

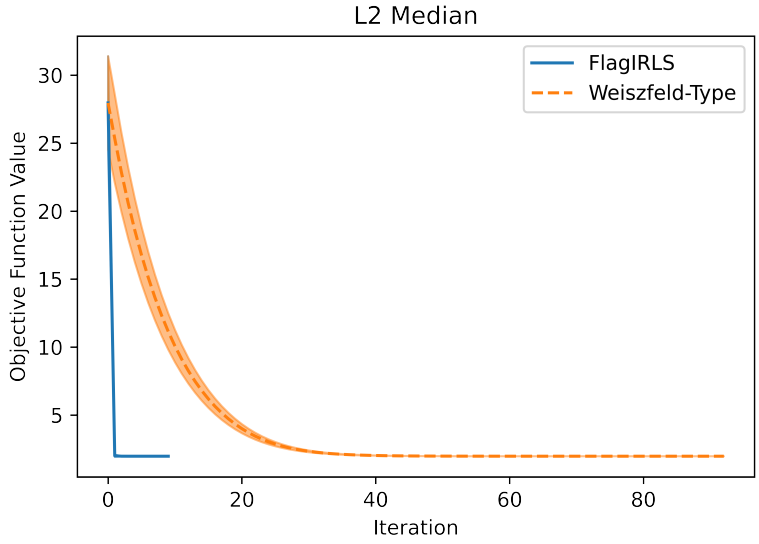


Figure 3.2: The mean objective function values over 100 trials with different random initializations. FlagIRLS converges in fewer iterations than the Weiszfeld-type algorithm for the 1-dimensional ℓ_2-median.

For our next example, we use a data set of 200 points on $\mathrm{Gr}(6,100)$. The points are sampled by first fixing a "center" point for the data set, $[\mathbf{X}_*]$. We do this by taking a random 100×6 matrix with entries from $\mathscr{U}[-.5,.5)$. We then take $\mathbf{X}_*$ as the first 6 columns of $\mathbf{Q}$ from the QR decomposition of this random matrix. The 200 points in the data set are now calculated by the following steps. For each point, we generate $\mathbf{Z}$ by sampling a random 100×6 matrix with entries sampled from $\mathscr{U}[-.5,.5)$ and scaling it by 0.01. We then take the point determined by the first 6 columns of $\mathbf{Q}$ from the QR decomposition of $\mathbf{X}_* + \mathbf{Z}$.

FlagIRLS is run to calculate the flag median and the maximally correlated flag. The Weiszfeld-type algorithm is run to compute the ℓ_2-median. Each of these algorithms are run with 20 different initializations. For the random initializations, we initialize FlagIRLS and the Weiszfeld-type algorithm at the same point. The results of this experiment are in Table 3.2. FlagIRLS converges in far fewer iterations than the Weiszfeld-type algorithm. We terminate the Weiszfeld-type algorithm after 1000 iterations regardless of convergence. So perhaps, many of the high iteration runs of the Weiszfeld-type algorithm still did not converge even after 1000 iterations.

Table 3.2: The mean number of iterations until convergence of 20 random initializations of FlagIRLS and the Weiszfeld-type algorithm on a data set of 200 points on Gr(6, 100). FlagIRLS is converges in far fewer iterations than the Weiszfeld-type algorithm and also sports a much lower standard deviation in the number of iterations.

Algorithm (Prototype)/ Initialization	Mean Iterations
FlagIRLS (flag median)/ random	4.40 ± 0.50
Weiszfeld-Type (ℓ_2-median)/ data point	813.50 ± 261.03
Weiszfeld-Type (ℓ_2-median)/ random	960.65 ± 80.94

In our next experiment we use a data set that consists of 200 points on Gr(3, 20). These points consist of a cluster of 180 points centered around the subspace $[\mathbf{X}_*]$ and 20 outlier points. The points from the 180-point cluster are sampled by the following process. We calculate a fixed "center" point for the data set, $[\mathbf{X}_*]$, by taking a 20×3 matrix with entries from $\mathscr{U}[-.5, .5)$. We then take $\mathbf{X}_*$ as the first 3 columns of $\mathbf{Q}$ from the QR decomposition of this random matrix. We then calculate the points in the cluster by the following steps. The first step is to generate $\mathbf{Z}$ by sampling a random 20×3 matrix with entries sampled from $\mathscr{U}[-.5, .5)$ and scaling it by 0.01. The second step generates one point in the 180 point cluster as the first 3 columns of $\mathbf{Q}$ from the QR decomposition of $\mathbf{X}_* + \mathbf{Z}$. A point from the set of outlier 20 points is the first 3 columns of the QR decomposition of a random 20×3 matrix with entries sampled from $\mathscr{U}[-.5, .5)$.

Table 3.3 shows the results of calculating the flag median, ℓ_2-median, and flag mean of this data set and then computing the chordal distance between $[\mathbf{X}_*]$ and the three different prototypes. Notice the flag median is the least affected by the outliers, the ℓ_2-median is twice as affected, and the flag mean is ten times more affected by the outliers.

Table 3.3: The chordal distance between the algorithm result and $[\mathbf{X}^*]$.

Algorithm	Chordal Distance
flag median	0.0017
ℓ_2-median	0.0022
flag mean	0.0128

Note: for Table 3.3, FlagIRLS converges to the flag median in one iteration.

Our final experiment on synthetic data investigates whether a stochastic implementation of FlagIRLS would result in a faster algorithm with acceptable accuracy for finding the flag median. To do this, we run stochastic FlagIRLS with 100 points on Gr(20, 1000) using anywhere from 1 to 99 points per iteration. The results of running stochastic FlagIRLS with 20 random initializations are in Figure 3.3. Stochastic FlagIRLS, with 80 or more points, produces approximately a 40% reduction in run time with flag median objective function values within 0.1 of the value found using non-stochastic FlagIRLS.

These data are centered around the subspace $[\mathbf{X}_*]$ and generated by the following steps. First, we calculate a fixed "center" point for the data set, $[\mathbf{X}_*]$, by taking a 1000×20 matrix with entries from $\mathcal{U}[-.5, .5)$. We then take $\mathbf{X}_*$ as the first 20 columns of $\mathbf{Q}$ from the QR decomposition of this random matrix. To calculate the points we generate $\mathbf{Z}$ by sampling a random 1000×20 matrix with entries sampled from $\mathcal{U}[-.5, .5)$ and scaling it by 0.01. Then a data point is represented by first 20 columns of $\mathbf{Q}$ from the QR decomposition $\mathbf{X}_* + \mathbf{Z}$.

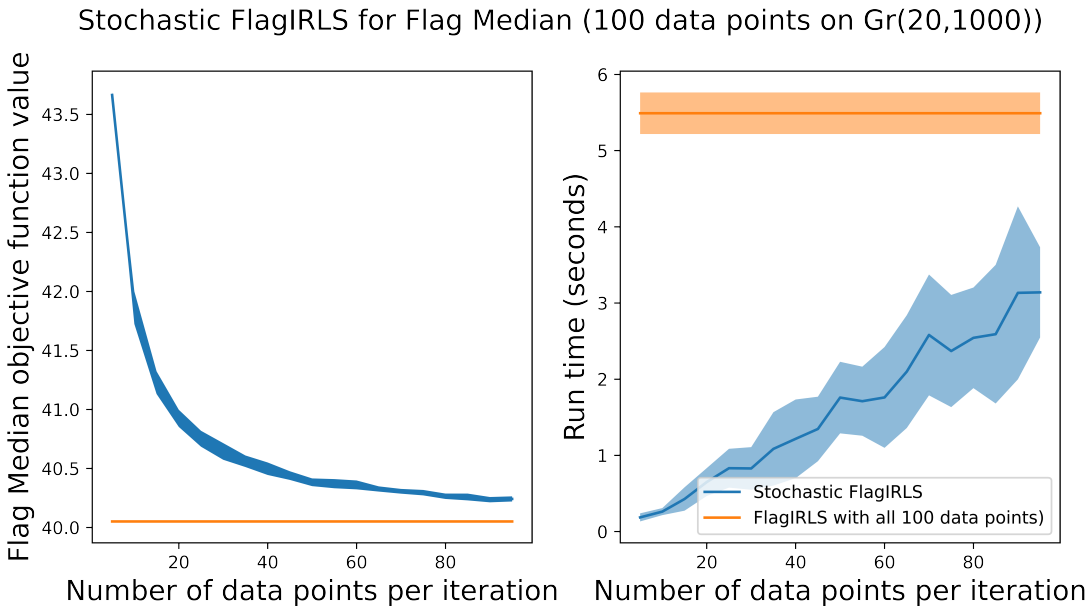


Figure 3.3: The objective function values for the flag median for stochastic implementations of FlagIRLS. The shaded region is 1 standard deviation away from the mean for 20 random initializations.

3.5.2 MNIST Handwritten Digits

The MNIST digits data set is a set of 28×28 single band images of handwritten digits [50]. We represent an MNIST handwritten digit using an element of $\mathrm{Gr}(1,784)$ by taking one image, vectorizing it, then dividing the resulting vector by its norm.

For our first example, we see how the flag median, ℓ_2-median, and flag mean prototypes are classified by a MNIST-trained 3-layer neural network. This trained neural network classifier has a 97% test accuracy on the MNIST test data set. We generate our data sets for this experiment by taking 20 examples of the digit 1 and i examples of the digit 9 from the MNIST training data set. We let $i = 0, 1, 2, 3, \ldots, 19$ and this results in 20 data sets. For each of these data sets, we calculate the flag median, ℓ_2-median and flag mean, then predict the class of each of these prototypes by passing each through the neural network classifier.

In Figure 3.4 we plot the predicted class of each prototype for each data set by the trained neural network. For this figure, we choose to use the random initialization that resulted in the best predictions of the ℓ_2-median.

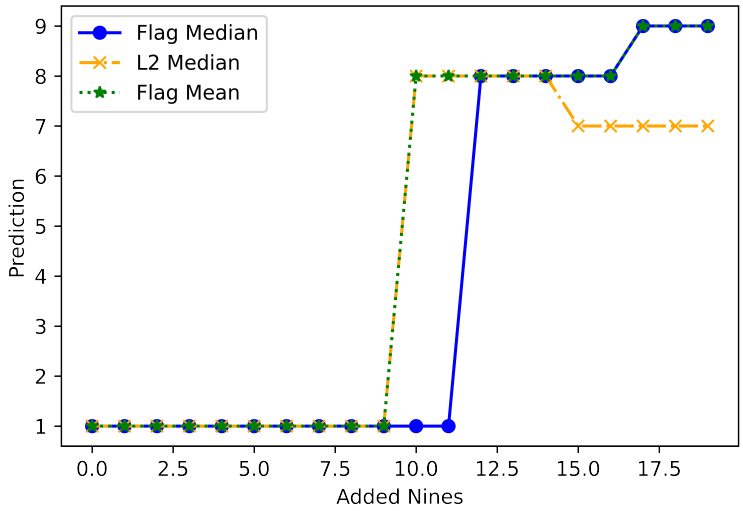


Figure 3.4: The neural network predicted class of the prototype for the data set with 20 examples of 1's and i examples of 9's with $i = 0, 1, 2, \ldots, 19$.

The ℓ_2-median and flag mean are the first prototypes to be misclassified with $i = 10$ added examples of 9's. Finally, the flag median is still classified correctly for $i = 10$ and $i = 11$ added examples 9's. Therefore the flag median is the most robust prototype to outliers in this experiment. The common misclassification as 8 is likely due to the fact that the 1's tend to be at an angle, so when averaged, they look like fuzzy 8's, especially when some 9's have been introduced to the data set. Also, the ℓ_2-median of a data set of 20 1's with 15 to 19 9 digits is misclassified as a 7. This is likely a result of the different angled 1's and the introduction of the examples of 9's adding the top of the digit 7.

Now we use Multi Dimensional Scaling (MDS) [52] to visualize the movement of the prototypes of an MNIST data set that is poisoned with outliers. For this experiment, we use 20 examples of 7's and $i = 0, 2, 4, 6, 8$ examples of 6's. This results in 5 different subspace data sets formed from examples from the MNIST training data set. We then calculate the flag median, ℓ_2-median and the flag mean. We generate a distance matrix for all the examples of 6's and 7's along with the exemplars from each data set using the geodesic distance and pass the distance matrix through a MDS algorithm to visualize relationships between these subspaces in two dimensions. This example is in Figure 3.5.

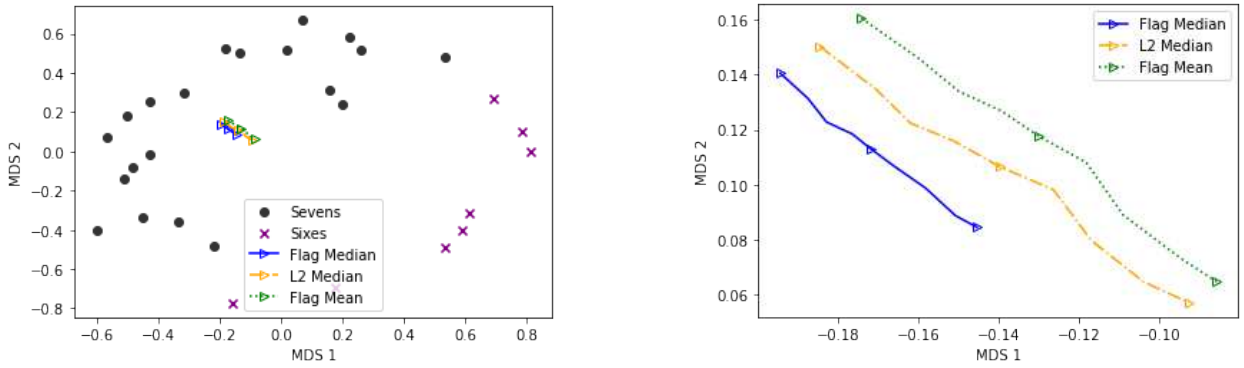


Figure 3.5: MDS embedding of flag median, ℓ_2-median and flag mean with points as one dimensional subspaces. We have 20 examples of 7's and i examples of 6's. Each triangle represents a prototype for $i = 0, 4, 8$. The furthest left triangle is the prototype for the data set with $i = 0$ examples of 6's and the furthest right triangle is the prototype for the data set with $i = 8$ examples of 6's. The lower image is a zoomed in version of the interior of the red box in the upper image to clarify the difference between the central prototypes.

Notice that the ℓ_2-median and flag mean are moving the most as we add examples of 6's. The flag median moves substantially less than the other prototypes and therefore is the least effected by the added examples of 6's.

We now compute the $r = 5$-dimensional flag median and flag mean of a data set with 20 examples of 7's with $i = 8$ 6's. We plot each of the reshaped columns of the matrix representative of these prototypes in Figure 3.6. Notice that the flag mean is more affected by examples of 6's than the flag median. This is particularly noticeable in the final column (dimension 5) where there is a clear 6 in the flag mean whereas the 6 is not clear in the flag median.

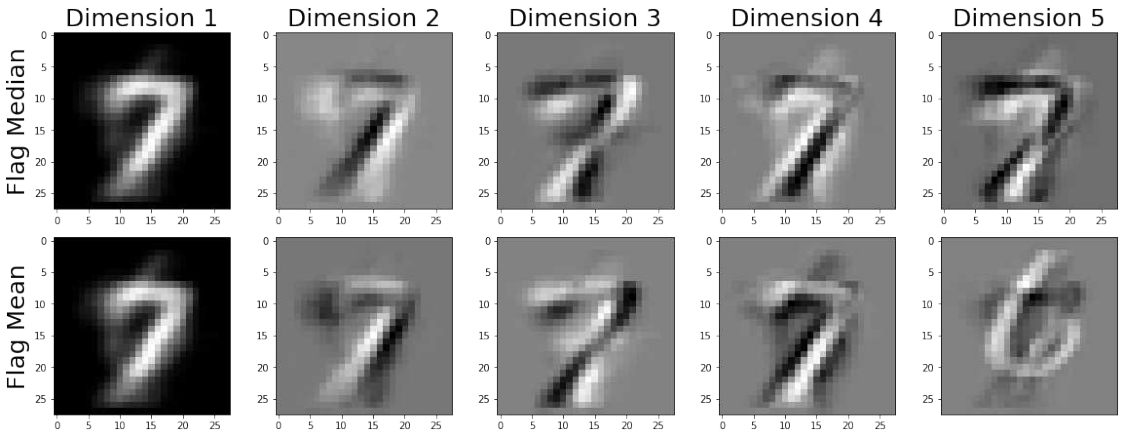


Figure 3.6: Each column of the matrix representative for flag median and flag mean on the data set with 20 examples of 7's and $i = 8$ examples of 6's.

3.5.3 Mind's Eye

The Mind's Eye data set is a set of grey-scale outdoor video clips that are centered on moving objects (mainly humans) and have a subtracted background. Each video clip consists of 48 frames, each rescaled to a size of 32×32 pixels. We use the preprocessed data from the k-means experiment from Marrinan et al. [28]. These data and the scripts for the preprocessing can be accessed at https://www.cs.colostate.edu/~vision/summet. There are 77 labels of the video clips for the action of the centered object in the video. A video clip is represented on $\mathrm{Gr}(48, 1024)$ by the span of the 1024×48 matrix formed by vectorizing and horizontally stacking each frame.

For this example, we use subspaces (points in Gr(48,1024)) that represent clips with action labels bend, follow, pickup, ride-bike and run. There are 27 examples of bend, 32 of follow, 27 of pickup, 17 of ride-bike, and 24 of run. We run the Linde-Buzo-Grey (LBG) algorithm [8, 53] to cluster these data with different sized codebooks (numbers of centers) and prototype calculation using the flag median, ℓ_2-median and flag mean. For each number of centers, we run 20 trials with different LBG initializations. In the LBG algorithm, we calculate distance using chordal distance for all prototypes. The results are in Figure 3.7.

We note that the flag median produces the highest cluster purities for 8, 12, 16 and 20 clusters. In all of the previous experiments we found that the flag median is more robust to outliers which may be the key factor in the success of the flag median prototype LBG implementation. We also note that the ℓ_2-median and the flag mean LBG implementations have similar cluster purities for each of the codebook sizes. Again, this is consistent with the similar behavior of the ℓ_2-median and the flag mean MNIST experiments (see Figure 3.4 and Figure 3.5).

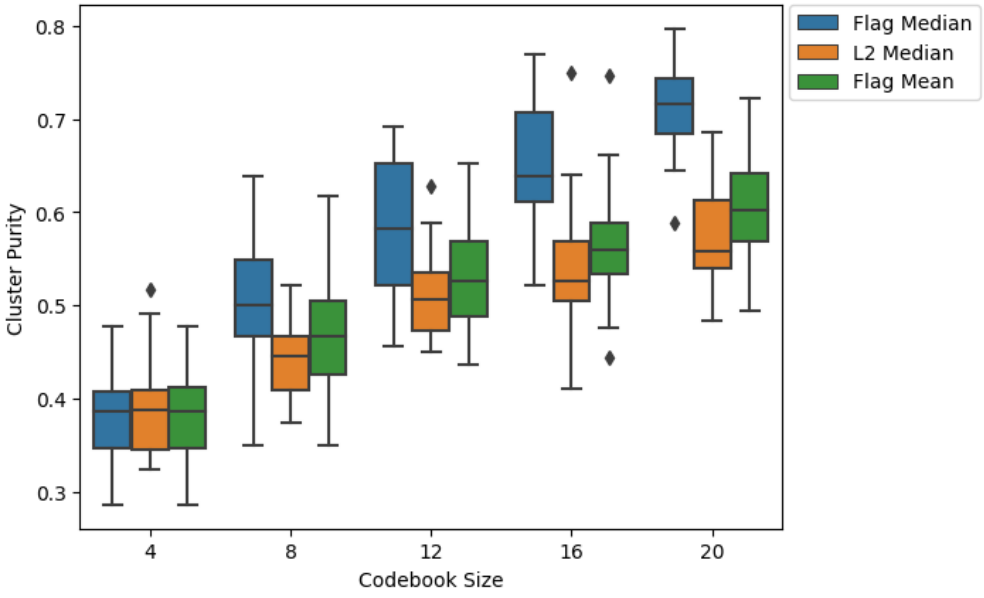


Figure 3.7: An LBG implementation on the Mind's Eye data set. The results of 3 different implementations of LBG for codebook sizes 4,8,12,16 and 20. The flag median is competitive with the ℓ_2-median and flag mean for a size 4 codebook and has higher cluster purity than the ℓ_2-median and flag mean for codebook sizes 8,12,16,20.

3.5.4 UCF YouTube

Our final data set is a subset UCF YouTube Action data set [51]. This data set contains 11 categories of actions. For each category, the videos are grouped into groups with common features. For this experiment, we take approximately one example from each group within an action category. Specifically our data set consists of 23 examples of basketball shooting, 22 of biking/cycling, 25 of diving, 24 of golf swinging, 24 of horse back riding, 24 of soccer juggling, 23 of swinging, 24 of tennis swinging, 24 of trampoline jumping, 22 of volleyball spiking, and 24 of walking with a dog. Since these color videos are quite large, we convert them to greyscale. Then we generate a matrix for each video whose columns are vectorizations of each frame. Finally, we perform the QR decomposition of each video and take the first 10 columns of $\mathbf{Q}$ to be its representative on the Grassmannian.

We then run subspace LBG with 48 dimensional flag mean and flag median. The results are in Figure 3.8. We choose to omit the ℓ_2-median LBG implementation since the Weiszfeld-type algorithm can only compute a 10 dimensional prototype. We run our LBG implementations with 10 trials for each of the following codebook sizes: $4, 8, 12, 16$ and 20. We see the flag median LBG implementation produces higher cluster purities than the flag mean LBG implementation in all trials.

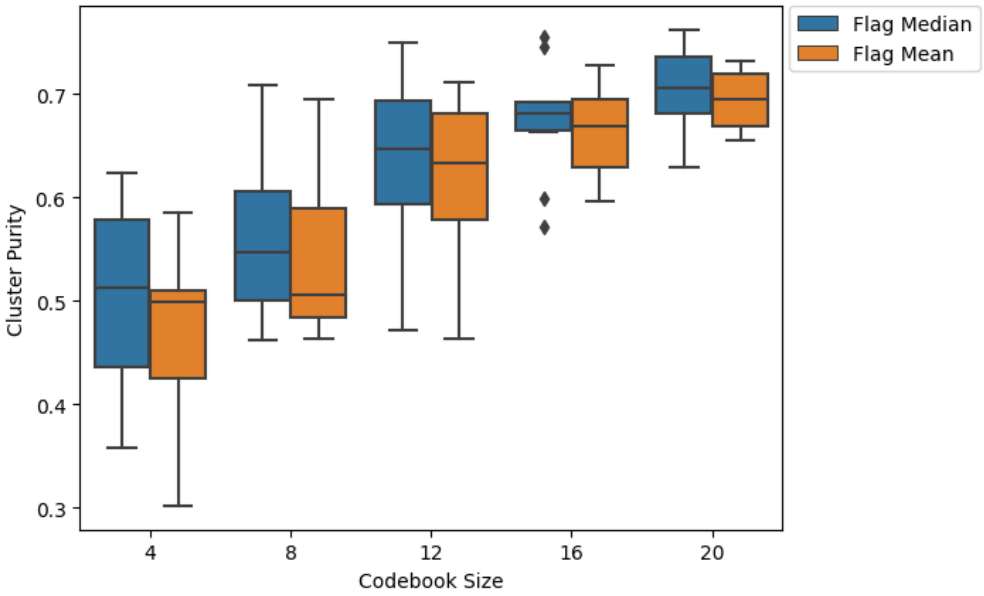


Figure 3.8: An LBG implementation on the YouTube data set. The results of 2 different implementations of LBG for codebook sizes $4, 8, 12, 16$ and 20. The flag median has higher cluster purities than flag mean for all codebook sizes.

3.6 Conclusion

In this chapter we proposed the FlagIRLS algorithm to approximate solutions to the flag median optimization problem. A convergence result was proven regarding FlagIRLS and DPCP-IRLS. Then we ran experiments comparing the flag median, flag mean, and ℓ_2-median. In our experiments, we found the FlagIRLS generally converges faster than gradient descent and the Weiszfeld-type algorithm. In addition, we discovered that the flag median is more robust to outliers and produces higher cluster purities than the flag mean and ℓ_2-median.

Even with our convergence result, there is a lack of proven mathematical guarantees for the flag median and the FlagIRLS algorithm. Although, for all our examples, FlagIRLS converges to a local minimum of the flag median problem, we have not worked out the mathematical theory to find the conditions where the iterates of FlagIRLS converges. We also still need to determine the conditions where an iteration of FlagIRLS is a contraction mapping. On a larger scale, given a data set of subspaces of $\mathbb{R}^n$, we have yet to determine where the flag median problem

is convex. Section 3.3 showed that FlagIRLS is a logical algorithm for finding the flag median, maximally correlated flag, and ℓ_2-median. However, further development of the mathematical theory would give us more intuition about which data sets are good for FlagIRLS, how to initialize FlagIRLS and overall provide us with a better understanding of rates of convergence.

More future work with FlagIRLS could involve machine learning or add details to the mathematical theory. For machine learning, the flag median can be used as a step in a subspace k-means algorithm, Grassmannian n-shot learning or any other machine learning algorithm in which calculating an "average" is a step. Most likely these types of algorithms will be useful for classifying images and videos. In terms of mathematics, we would like to find proofs for the convergence and convergence rates of iterates of FlagIRLS and DPCP-IRLS.

Chapter 4

Pathway Expression Analysis

4.1 Introduction

Influenza and other viruses linked to respiratory illnesses in humans have gained relevance due to the recent COVID-19 pandemic caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Accurate detection of these types of viruses is necessary to isolate infected individuals and consequently slow the spread throughout the population. Moreover, some infected individuals can be shedding the virus without showing symptoms. It is advantageous to have methods of detecting shedding that are robust to symptoms to avoid asymptomatic super spreader scenarios [54].

Analyses of respiratory illnesses aid in understanding the mechanisms of shedding and in developing methodologies that succeed across multiple infectious diseases. Understanding the imprint of viral shedding on human gene expression may uncover latent effects which are beyond disease symptoms. In this paper we will run our experiments on a human microarray data set with multiple respiratory viruses and studies, the GSE73072 data set [55]. Previous work on these respiratory virus data ran various machine learning (ML) models, e.g., neural networks, support vector machines (SVM), centroid encoders (CE), and spectral gene network analysis, to identify discriminatory biomarkers within early shedders challenged with influenza and subsequently classify those subjects [56–59].

Rather than just select significant genes, a biological pathway analysis uses biological relationships between genes, usually by grouping related genes, to build a biologically informed model [60]. Given the numerous definitions of pathway membership and number of ways to relate genes within a pathway, many of these pathway analyses require an a priori set of "important" genes (like those found using ML on gene expression data [56]) to determine significant pathways [61, 62]. Standard tools that rely on an a priori gene set include: over representation

analysis (ORA) [63], gene set enrichment analysis (GSEA) [64–67], Centrality-based pathway enrichment (CePa) [68] and more [69]. ORA determines the statistical significance of the overlap between an a priori set of genes and a given pathway. GSEA improves upon ORA by accounting for the expression levels of the genes. CePa is a further improvement on the ORA method which uses the biological connections between genes in a pathway. Specifically, it uses the centrality of genes in the network generated by a pathway as part of its ranking. Beyond simply ranking pathways by statistical significance, Maglietta et al. offer a method that ranks functional groups of genes, specifically genes associated with the same GO term [70], to predict a phenotype [69]. All of these pathway methods rely on a pathway database which is used to determine pathway membership. There are a plethora of pathway databases including KEGG [71], MetaCyc [72], Reactome [63], Wikipathways [73], BioCarta [74], InnateDB [75] and many more. Most of these databases contain essentially the same information regarding pathway membership. We use the Reactome database, along with their built in tool for ORA in order to produce pathway rankings to compare against our novel pathway expression analysis rankings.

In this paper, we formalize a method called pathway expression analysis which transforms gene expression data into **pathway** expression data. This allows for the determination of pathway significance and subject classification using pathway expression levels rather than gene expression levels. This is not the first time such a translation has been considered. Some of the following works use GO terms or other biologically known collections of genes rather than pathways. For the sake of simplicity, we will use the term pathway loosely in this paragraph to refer to a biologically known collection of genes. Between the early 2000's and 2012, various authors have explored classification and pathway ranking using statistical techniques. Pathway expressions, often referred to as pathway activities, have been calculated using means and medians [76], metagenes (a.k.a. eigengenes) [77], t-scores [78], log-likelihood ratios (LLRs) [79], and FAIME (Functional Analysis of Individual Microarray Expression) [80]. Some of these methods don not use all the genes in the pathway to calculate pathway expression. For example Guo et al. only calculate the average expression level of the differentially expressed genes in a path-

way. Pathway selection in these papers is not always done using pathway expression levels. Su et al. [79] use gene expression levels to select pathways. On the other hand, some works do indeed use their pathway expression data to determine significant pathways and to classify between phenotype. Lee et al. determine significant pathways for classification using the AUC (area under the ROC curve) on a validation data set [78]. Classification and regression of changes in phenotype with pathway expression data has been done using simple models like decision trees [76], linear discriminant analysis [79], logistic regression [78, 79], clustering [77, 80] and other statistical models [80].

The last known work in this field is FAIME in 2012 [80]. Within the last 10 years, the field of machine learning in biology has boomed in popularity and there is a need for **formalization** of these previous pathway expression methods. Our work fills this gap by defining pathway expression data as the result of a simple linear transformation of gene expression data. We offer two forms of pathway expression data: Linear Pathway Expression (LPE) and Centrality Pathway Expression (CPE). Methods like those presented by Guo et al. fall easily into the pathway expression framework as a special case of LPE [76]. CPE, on the other hand, is a novel method for pathway expression that ranks genes based on networks of known and inferred gene interactions rather than statistically inferring a gene ranking using differences between phenotype like the FAIME method [80]. Unlike other previous pathway expression methods (e.g., FAIME), CPE and LPE can be thought of as "unsupervised" pathway expression methods since they do not require phenotype labels for their computation. Due to their simplicity, CPE and LPE use less computational resources than most previous pathway expression methods since our methodology involves a linear transformation rather than more complicated non-linear mappings. Instead of using one of the classifiers from the previous works, we choose to use Sparse Support Vector Machines (SSVMs) for our classifier. This sparse classification scheme allows us to preform our pathway selection and classification using pathway expression data in one step, without any parameter tuning.

We use our pathway expression methods (LPE and CPE with SSVM) to select pathways which discriminate between uninfected subjects (controls) and eventual shedders infected with various respiratory viruses: H1N1, H3N2, HRV (Human Rhinoviruses), RSV (Respiratory Syncytial Virus), in the early stages of infection using the GSE73072 data set. O'Hara et al. found that gene expression data from some pathways like B Cell Maturation and Activation + Cell Adhesion Molecules were used to produce 100% classification accuracy for certain experiments with this data set but analyses with pathway expression has never been done [59]. This work compliments the early detection analysis done by Aminian et al. [56] by utilizing "pathway expression" with a simple feature selector and classifier in place of gene expression with optimal feature selection and classification techniques. By using pathway expression as an alternative to gene expression, we extract pathways as features using SSVMs and subsequently rank pathways by their weight in the SSVM model. We do this by extracting the top pathways for each pathway expression method on training data, restricting the test pathway expression data to those top pathways, and finally classifying controls vs. shedders (from 4 evenly spaced time bins within 32 hours after infection) using the test pathway expression data with an SVM. We also benchmark these selected pathways against two known pathway ranking methodologies ORA and CePa. As an aside, our implementation of CePa in Python, to the best of our knowledge, is one of the first Python implementations of such a workflow.

In our experiments on the GSE73072 dataset we find that pathway expression methods generally produce higher classification rates than gene expression methods with the same type of SSVM feature selector and classifier. We also find that using CPE, which adds gene network information, increases classification rates over simple LPE. Classification rates with CPE and LPE are found to be robust LIMMA normalization of the gene expression data and the pathways selected by these methods prove to be appropriately robust across training dataset partitions. The pathways which are selected using CPE and LPE methods tend to be distinct from the pathways selected using CePa and ORA. Hence, pathway expression is a useful method for discovering biological pathways which are not traditionally associated with a disease of interest. The

code for LPE and CPE is implemented in Python and available at https://github.com/nmank/PathwayAnalysis. This work is meant to revive work in pathway expression analysis by adding a simple framework for generating pathway expression data, introducing a new method for pathway expression, CPE, which leverages gene networks to produce pathway expression data and finally implementing a simple sparse classifier to select pathways and discriminates between phenotype at the same time.

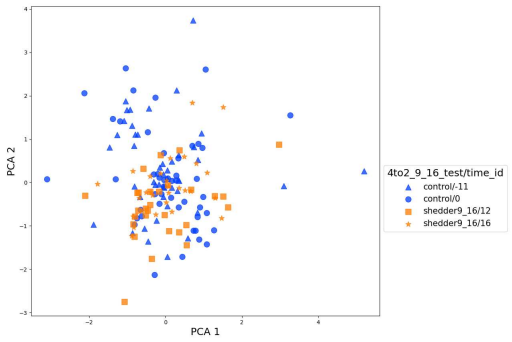
4.2 Results

In this section we provide a number of results using pathway expression on GSE73072. Specifically, we investigate two types of pathway expression: Centrality Pathway Expression (CPE) and Linear Pathway Expression (LPE) and compare these methods to Gene Expression (GE) methods. We begin this section with some visualizations of pathway expression data Section 4.2. In Section 4.2.1 we provide classification statistics including Balanced Success Rates (BSRs) for experiments using the features selected on pathway expression on test studies. We also pose an argument for the best CPE method in Section 4.2.1. In Section 4.2.2, we compare the pathways selected using pathway expression to those selected using CePa and ORA. Lastly, in Section 4.2.3, we list the top pathways found using the best CPE methods.

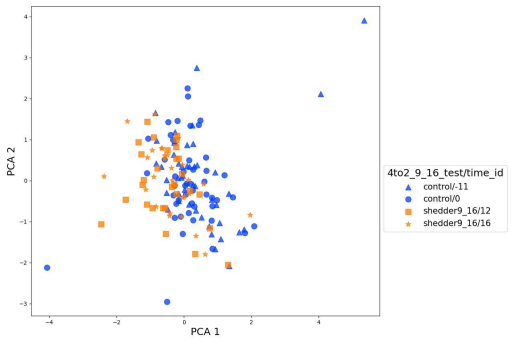
Visualization

For our PCA (Principle Component Analysis) visualization, we use gene expression data from 9 to 16 hours after infection that have been batch corrected for subject identifier using LIMMA. We select features using 4 studies containing H1N1 and H3N2 strains of influenza. Then we do two PCA plots of the first two principle components of data from 2 HRV test studies. One plot uses all the features for the PCA. The second plot uses only the features found using the 4 H1N1 and H3N2 studies. The feature selection is done with gene expression data and with pathway expression data using the same SSVM feature selector methodology. The objective of this experiment is a head to head comparison between gene expression to pathway expression

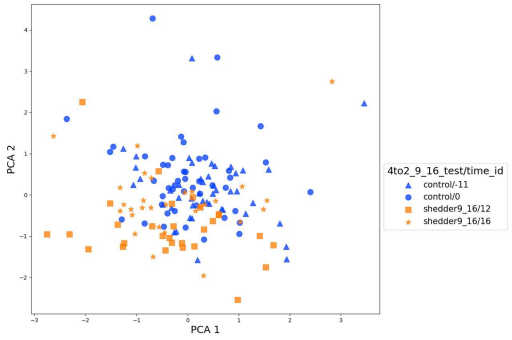
using the linear separation between the controls (subjects before infection) and shedders (subjects from 9 to 16 hours after infection). These PCA plots are in Figure 4.1. For this experiment and time bin, we notice better linear separation between the pathway expression data than the gene expression data.



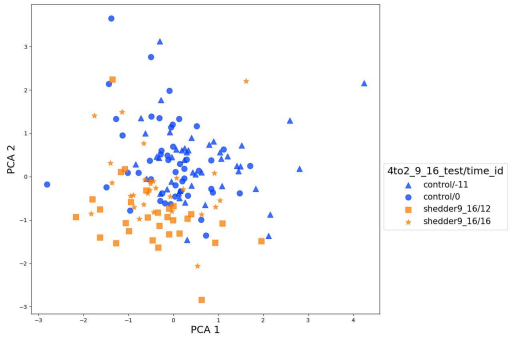
(a) GE test data with all the features.



(b) GE test data with the selected features.



(c) CPE test data with all the features.



(d) CPE test data with the selected features.

Figure 4.1: A PCA embedding of the 2 HRV test studies using GE and CPE in the time bin 9 to 16 hours after infection. The first column is an embedding with all the features and the second is with the selected features. These CPE data are generated using pre- computed, undirected edges with out-degree centrality. The data have been batch corrected for subject identifier using LIMMA.

When we produce the same types of PCA visualizations using each train/test split, along with each of the different time bins and pathway expression types, we notice that the gene expression data appears to linearly separate better than pathway expression data. But, upon

further investigation, we find that the explained variance ratio of the first two principle components for pathway expression data in these plots is generally less than 0.5. So most of the variance of pathway expression data is captured in the 3rd to n^{th} principle components. This provides some explanation for the poor linear separation of pathway expression data using the first two principle components of PCA.

4.2.1 Classification Results

In our classification experiments we draw samples from the GSE73072 data set and distinguish between controls and shedders within 32 hours after infection. For our experiments, we separate these data into 4 evenly sized time bins. Controls are all subjects at times before infection. We perform 3 different data set splits. For two of these splits, we find features using data from 4 studies and test on the remaining 2 or 3 studies. For the third split, we find features using data from 6 studies then test on 1 study. The details of the data sets used for these experiments are in Table 4.1.

Table 4.1: The train/test splits by study ID for the 4 to 2 and 6 to 1 experiments. The parenthetical after each study ID is the associated virus.

Partition Name	Train	Test
4 to 2	DEE2 (H3N2), DEE3 (H1N1), DEE4 (H1N1), DEE5 (H3N2)	UVA (HRV), Duke (HRV)
4 to 3	DEE2 (H3N2), DEE3 (H1N1), DEE4 (H1N1), DEE5 (H3N2)	UVA (HRV), Duke (HRV), DEE1 (RSV)
6 to 1	UVA (HRV), Duke (HRV), DEE1 (RSV) DEE3 (H1N1), DEE4 (H1N1), DEE5 (H3N2)	DEE2 (H3N2)

We use these classification experiments to compare LPE and CPE methods to each other and benchmark these pathway expression methods against GE methods. Classification results on the test studies from the 4 to 2, 4 to 3 and 6 to 1 experiments with LIMMA batch correction by subject ID are reported Table 4.2. A comparison of test classification BSRs between uncorrected and LIMMA corrected data is in Figure 4.2. It indicates that, using our pipelines, pathway expression classification BSR is more robust to LIMMA correction than gene expression. Using pathway expression (either LPE or CPE) we produce the same or higher classification statistics on the test data than GE for more than 81% of all the test statistics, experiments and time bins.

Table 4.2: Classification statistics by method, experiment and time bin on the test data sets. Pathway expression methods generally produce a higher BSR, precision, recall and accuracy over gene expression methods with the SVM feature selection technique. CPE is computed using pre- computed edges with PageRank centrality. The highest statistic for each time bin is bold face. All experiments apply LIMMA using subject identifier to the data. Standard deviation is not available for these statistics due to the design of our LOSO experiments. See Section 4.4 for details.

Time Bin	Experiment	Method	BSR	Precision	Recall	Accuracy
1 to 8	4 to 2	GE	0.6	0.71	0.71	0.64
1 to 8	4 to 2	LPE	0.72	0.79	0.81	0.73
1 to 8	4 to 2	CPE	**0.8**	**0.85**	**0.86**	**0.8**
9 to 16	4 to 2	GE	0.74	0.81	0.76	0.72
9 to 16	4 to 2	LPE	**0.82**	**0.87**	**0.85**	**0.8**
9 to 16	4 to 2	CPE	0.74	0.8	0.8	0.73
17 to 24	4 to 2	GE	0.67	0.74	0.76	0.69
17 to 24	4 to 2	LPE	0.7	0.77	0.77	0.7
17 to 24	4 to 2	CPE	**0.77**	**0.83**	**0.82**	**0.77**
25 to 32	4 to 2	GE	**0.96**	**0.98**	**0.99**	**0.98**
25 to 32	4 to 2	LPE	0.92	0.97	0.96	0.94
25 to 32	4 to 2	CPE	0.9	0.96	0.94	0.92
1 to 8	4 to 3	GE	0.61	0.74	0.71	0.65
1 to 8	4 to 3	LPE	**0.73**	**0.83**	0.8	0.76
1 to 8	4 to 3	CPE	**0.73**	0.82	**0.84**	**0.77**
9 to 16	4 to 3	GE	0.74	0.83	0.78	0.74
9 to 16	4 to 3	LPE	0.79	0.86	**0.85**	0.78
9 to 16	4 to 3	CPE	**0.81**	**0.88**	0.84	**0.81**
17 to 24	4 to 3	GE	0.59	0.72	0.65	0.6
17 to 24	4 to 3	LPE	0.69	0.79	0.75	0.69
17 to 24	4 to 3	CPE	**0.77**	**0.85**	**0.83**	**0.78**
25 to 32	4 to 3	GE	0.8	0.91	0.87	0.84
25 to 32	4 to 3	LPE	0.8	0.91	0.87	0.84
25 to 32	4 to 3	CPE	**0.83**	**0.92**	**0.92**	**0.88**
1 to 8	6 to 1	GE	0.62	0.8	**0.97**	0.81
1 to 8	6 to 1	LPE	0.8	0.89	**0.97**	0.89
1 to 8	6 to 1	CPE	**0.89**	**0.94**	**0.97**	**0.93**
9 to 16	6 to 1	GE	**0.91**	**0.94**	**1.0**	**0.96**
9 to 16	6 to 1	LPE	0.89	**0.94**	0.97	0.93
9 to 16	6 to 1	CPE	0.82	0.91	0.91	0.86
17 to 24	6 to 1	GE	0.77	0.87	**1.0**	**0.9**
17 to 24	6 to 1	LPE	0.74	0.86	0.94	0.85
17 to 24	6 to 1	CPE	**0.8**	**0.89**	0.97	**0.9**
25 to 32	6 to 1	GE	0.76	0.87	0.97	0.87
25 to 32	6 to 1	LPE	0.68	0.84	0.91	0.81
25 to 32	6 to 1	CPE	**0.91**	**0.94**	**1.0**	**0.96**

We now compare the effects of normalization via LIMMA on subject identifier on the classification results in Figure 4.2. The y-axis, Difference, is the difference in BSRs on the test studies with LIMMA normalized data for subject identifier and un-normalized data for subject identifier. LIMMA normalization using subject identifier increases the classification BSR for each method. When comparing LPE and CPE to GE, we notice that the inter-quartile range and median difference for pathway expression methods is smaller than those from the gene expression methods. This implies that the BSRs for gene expression methods change less uniformly and more on average from LIMMA batch correction on subject identifier than pathway expression methods. Consequently, we claim that pathway expression methods are more robust to subject differences within a class. In addition, linear pathway expression methods appear to be the most robust to subject differences within a class because the median difference for LPE in Figure 4.2 is nearer to 0 than the median difference for CPE and GE.

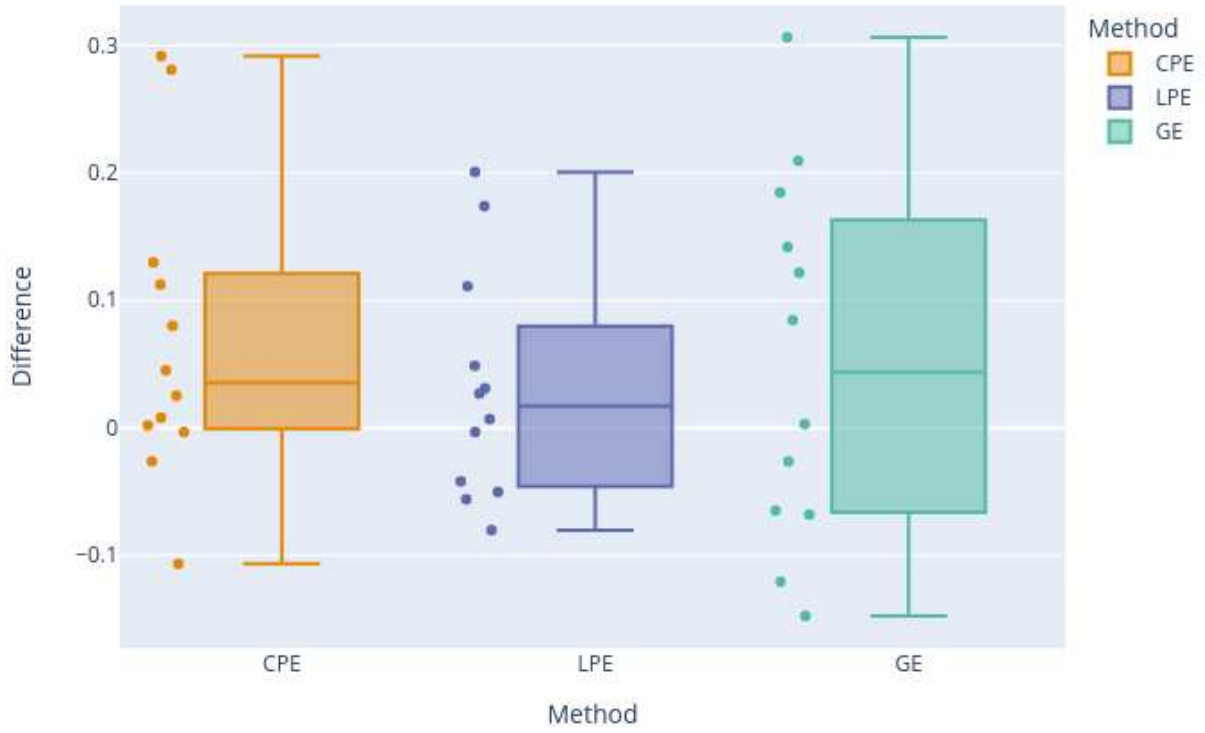


Figure 4.2: The difference between test BSR with LIMMA normalization using subject identifier and test BSR without LIMMA. Each box represents the distribution of these differences across experiment, time bin, and data partition for each method. A positive difference in BSR indicates that LIMMA normalization using subject identifier increased the classification accuracy. For CPE, we use pre- computed, directed edges with PageRank centrality.

CPE requires two parameters to be set: a pathway gene network edge type and a centrality measure. We test CPE by using either correlation edges or pre-computed (from Reactome) edges, which may be directed or undirected. For centrality measures, we use either PageRank or out-degree centrality. Table 4.3 details the CPE configuration which produces the highest test BSR. Using this table, we observe that pre-computed edges with PageRank centrality is the most common method across all experiments and time bins to produce the highest BSR.

Table 4.3: The CPE centrality and similarity configurations which resulted in the highest test BSR for CPE given each data partition and time bin. The data have been batch corrected for subject identifier using LIMMA.

Experiment	Time Bin	Centrality	Similarity	Directed
4 to 2	1 to 8	PageRank	pre-computed	True
4 to 2	9 to 16	out-degree	pre-computed	False
4 to 2	17 to 24	out-degree	correlation	False
4 to 2	25 to 32	PageRank	pre-computed	False
4 to 3	1 to 8	out-degree	correlation	False
4 to 3	9 to 16	PageRank	pre-computed	False
4 to 3	17 to 24	out-degree	pre-computed	True
4 to 3	25 to 32	PageRank	pre-computed	False
6 to 1	1 to 8	PageRank	pre-computed	True
6 to 1	9 to 16	PageRank	correlation	False
6 to 1	17 to 24	PageRank	pre-computed	True
6 to 1	25 to 32	PageRank	pre-computed	False

Finding the “best” CPE configurations by using those which produce the maximum test BSR is not the best way to select the “best” CPE configuration since it doesn’t take into account the performance of each CPE configuration across all experiments. So, in order to choose the best CPE edge type and centrality ranking, we look at the distributions of test BSRs for each edge type and centrality ranking across experiments. The box plot from this investigation is in Figure 4.3. We believe that the best CPE configuration should have the lowest variance across experiments and time bins while maintaining one of the highest median test BSRs. By this criteria, we see pre-computed, directed edges with PageRank centrality is the “best" CPE configuration.

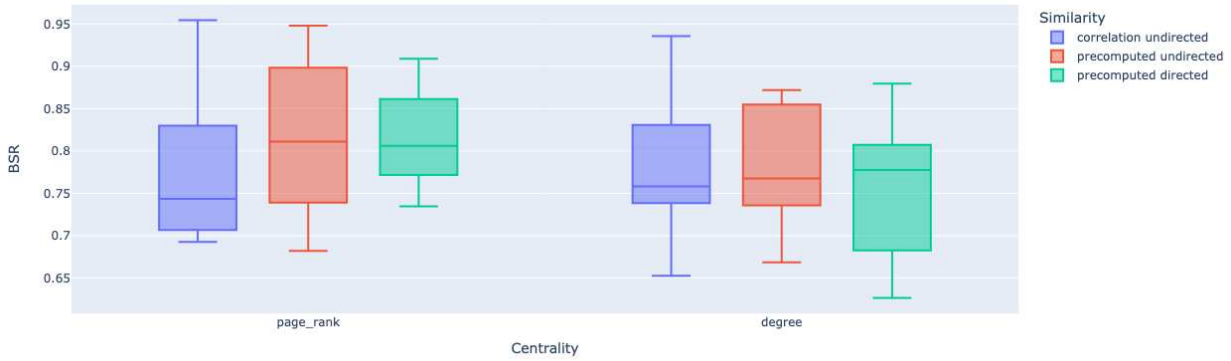


Figure 4.3: The distribution of BSR across experiments for each CPE configuration. Notice pre- computed, directed edges with PageRank centrality has one of the highest BSRs while maintaining the lowest variance. All experiments apply LIMMA using subject identifier to the data.

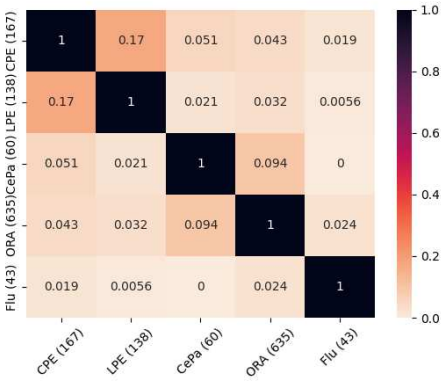
4.2.2 Comparing Pathway Selection Methodologies

We perform two pathway selection experiments using two data sets: 1) 4 studies (the training data for the 4 to 2 and the 4 to 3 experiments) and 2) 6 studies (the training data for the 6 to 1 experiments).

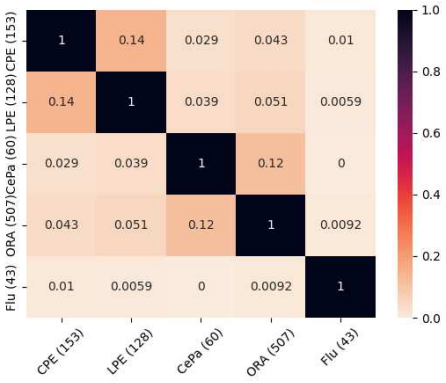
In these experiments, we compare the pathways that are selected by the pathway expression methods (CPE and LPE) to the pathways selected by standard pathway ranking algorithms (CePa and ORA) as well as a list of influenza related pathways (labeled Flu) from Reactome. We find this list of influenza related pathways by simply searching for influenza on the Reactome website. We do not expect high overlap with these influenza pathways since they are likely only activated at later time points during infection. For methodological consistency we use the same edge and centrality methods for CePa and CPE, namely pre-computed, directed edges with PageRank centrality.

For comparison, we use the Jaccard/Tanimoto similarity coefficient as a measure of overlap between the sets of pathways from different methodologies. These comparisons for the 6 study features are in Figure 4.4 and the 4 training features are in Figure 4.5.

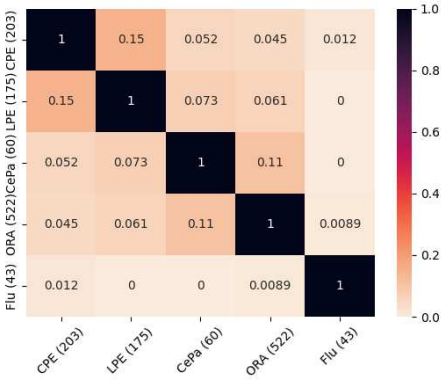
Over all methods and experiments, we notice LPE and CPE have a high Jaccard similarity, and CePa and ORA have a relatively high Jaccard similarity. CePa and ORA both have a small Jaccard similarity with LPE and CPE. We generally see low overlap between all methods and the influenza pathways. Generally, ORA is the method with the highest overlap with the influenza pathways since it detects at least twice as many pathways as each of the other pathway selection methods. In fact, CPE has absolutely no overlap with influenza pathways for the 4 study experiments. In the 6 study experiments, we see that CPE has a higher overlap with the flu pathways than all other methods at the 9 to 16 and 17 to 24 time bins even though it detects less than half as many pathways as ORA. LPE has the highest overlap with the flu pathways at the 1 to 8 hour time bin for the 4 study experiment even though there are only 60 LPE pathways and 254 ORA pathways.



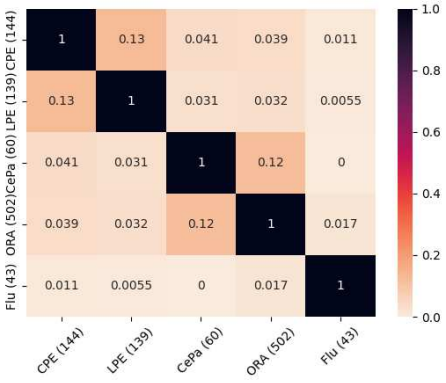
(a) 1 to 8 time bin, 6 studies



(b) 9 to 16 time bin, 6 studies

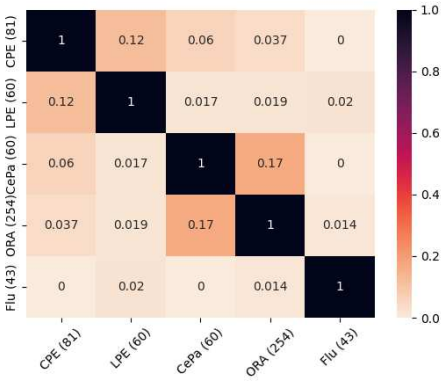


(c) 17 to 24 time bin, 6 studies

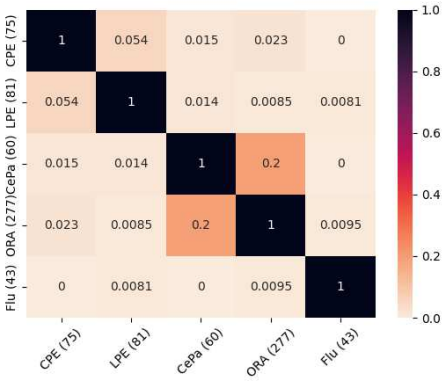


(d) 25 to 32 time bin, 6 studies

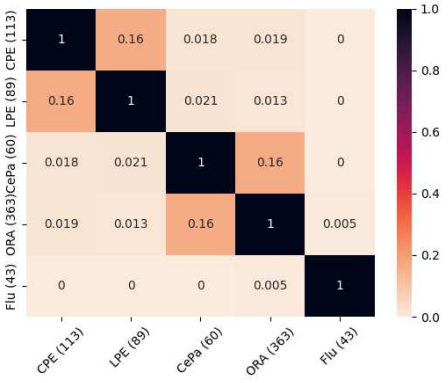
Figure 4.4: Jaccard overlap between the selected pathways for different methodologies. Pathways are selected using the 6 training studies. Each plot is for a different train/test experiment with LIMMA using subject identifier. The CPE configuration is pre- computed, directed edges with PageRank centrality.



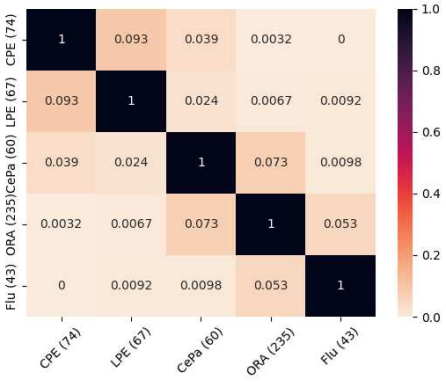
(a) 1 to 8 time bin, 4 studies



(b) 9 to 16 time bin, 4 studies



(c) 17 to 24 time bin, 4 studies



(d) 25 to 32 time bin, 4 studies

Figure 4.5: Jaccard overlap between the selected pathways for different methodologies. Pathways are selected using the 4 training studies. Each plot is for a different train/test experiment with LIMMA using subject identifier. The CPE configuration is pre- computed, directed edges with PageRank centrality.

Next, we investigate the robustness of these selected pathways across different studies in Table 4.4. To do this, we look at the Jaccard overlap between the pathways from 4 training studies and those from 6 training studies for each method and time bin. The pathways found by ORA are the most robust to this change in training data. Generally, the overlap between the pathway expression pathways is half of that from ORA. The CePA overlap is far smaller than the other methods and generally 10 times smaller than the overlap from ORA.

Table 4.4: Jaccard overlap between the selected pathways across the 4 study and the 6 study pathways. LIMMA was used to normalize the data for subject identifier. The final two columns are the number of pathways selected by the method using the stated training dataset (4 or 6 studies). The CPE configuration is pre- computed, directed edges with PageRank centrality.

Method	Time Bin	Jaccard Overlap	4 Studies	6 Studies
CPE	1 to 8	0.0877	81	167
LPE	1 to 8	0.0588	60	138
CEPA	1 to 8	0.0169	60	60
ORA	1 to 8	0.2095	254	635
CPE	9 to 16	0.1014	75	153
LPE	9 to 16	0.0885	81	128
CEPA	9 to 16	0.0169	60	60
ORA	9 to 16	0.2288	227	507
CPE	17 to 24	0.1408	113	203
LPE	17 to 24	0.1186	89	175
CEPA	17 to 24	0.0435	60	60
ORA	17 to 24	0.3574	363	552
CPE	25 to 32	0.0846	74	144
LPE	25 to 32	0.1196	67	139
CEPA	25 to 32	0.0256	60	60
ORA	25 to 32	0.2222	235	502

4.2.3 Top CPE Pathways

So far, we have seen that CPE generally produces the highest classification rates in our experiments over all methods examined in Table 4.2. For this entire section we will be using the CPE configuration with pre-computed, directed edges with PageRank centrality. We use the SSVM weights on the training data to determine the top pathways for each experiment. Figure 4.6 is a

heat map of these SSVM weights for the union of the selected pathways from each experiment and time bin. The most discriminatory pathways on the training data are the pathways with the highest SSVM weights. We notice that some pathways remain activated across all times and studies. This is indicated by a dark streak in one column which persists across all rows.

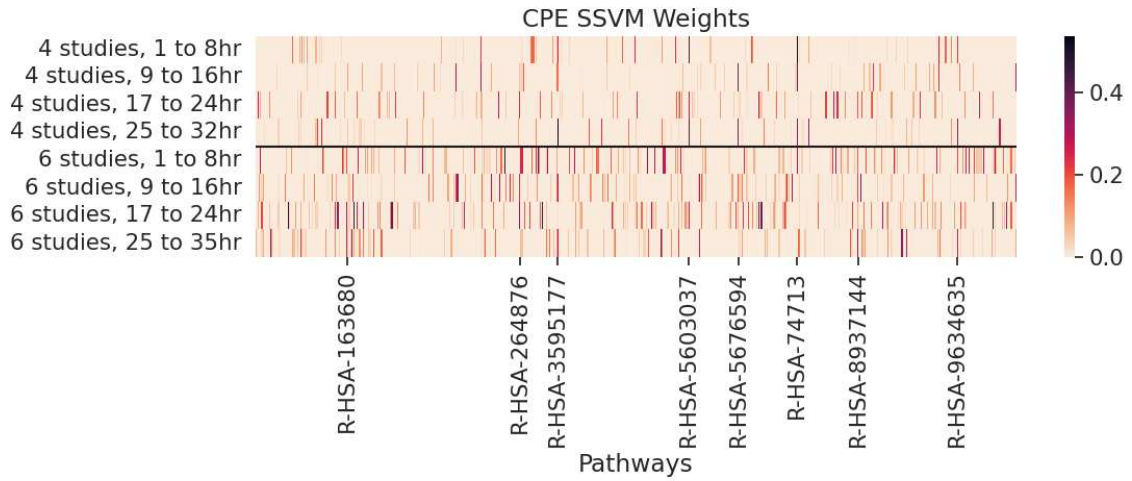


Figure 4.6: SSVM weights for the 4 and 6 study training datasets by pathway. The CPE configuration is with pre- computed, directed edges with PageRank centrality.

The top pathways from CPE across all experiments are found by adding the SSVM weights across all experiments and time bins are in Table 4.5. This amounts to adding the values of each column in Figure 4.6 to compute a pathway score for each selected pathway. Figure 4.7 is plot of these sorted scores that is used to determine a total SSVM weight threshold of .7 to identify the pathways listed in Table 4.5. A brief discussion of the relationships between the pathways in Table 4.5 and influenza appears in Section 4.3.

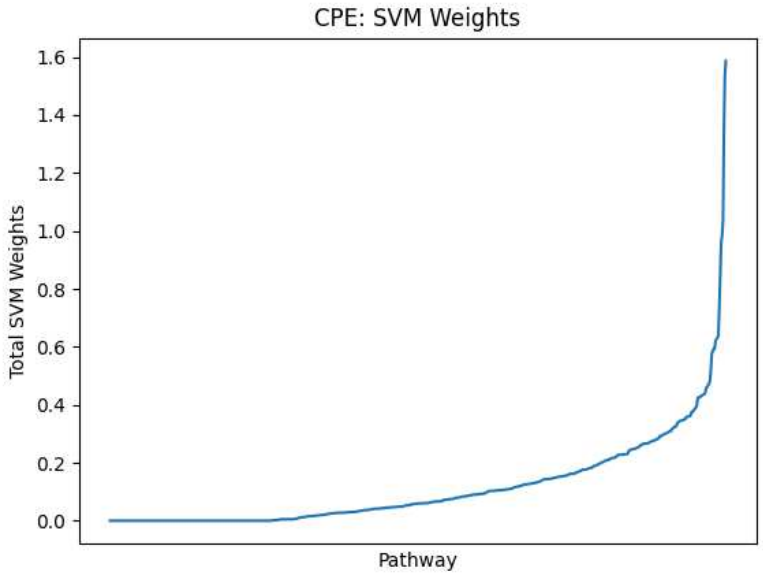


Figure 4.7: The sorted weights of the SSVM classifiers over all experiments and time bins for CPE. The CPE configurations in all experiments are pre- computed, directed edges with PageRank centrality.

Table 4.5: The pathways with the highest magnitude SVM weights summed over all experiments and times. CPE configurations are pre- computed, directed edges with PageRank centrality. All experiments apply LIMMA using subject identifier to the data.

Reactome ID	Pathway Name
R-HSA-163680	AMPK inhibits chREBP transcriptional activation activity
R-HSA-264876	Insulin processing
R-HSA-8937144	Aryl hydrocarbon receptor signalling
R-HSA-5676594	TNF receptor superfamily (TNFSF) members mediating non-canonical NF-kB pathway
R-HSA-9634635	Estrogen-stimulated signaling through PRKCZ
R-HSA-5603037	IRAK4 deficiency (TLR5)
R-HSA-74713	IRS activation
R-HSA-3595177	Defective CHSY1 causes TPBS

Now we present the top pathways from CPE for each experiment in Table 4.6. We define these top pathways as those pathways with the highest SSVM weight in the feature selection

process for their experiment and time bin. We notice that R-HSA-74713 appears in two time bins for the 4 study experiment. From our searches on Reactome, none of these pathways are directly labeled as influenza pathways. However, the Jaccard overlap heat map in Figure 4.4 suggests that the pathways selected using this CPE configuration contains some of the influenza virus signal because they have the highest overlap with the influenza pathways out of all pathway selection techniques at the 9 to 16 and 17 to 24 hour time bins. Therefore, further investigation into R-HSA-8939242 and R-HSA-9694631 pathways and their relationship to respiratory viruses is needed. Additionally, R-HSA-3595177 and R-HSA-74713 appear in the both Table 4.5 and Table 4.6 and therefore there may be a link between these pathways and respiratory viruses.

Table 4.6: The pathways with the highest magnitude SVM weights from CPE for each experiment and time. CPE configurations are pre- computed, directed edges with PageRank centrality. All experiments apply LIMMA using subject identifier to the data.

Experiment	Time	Reactome ID	Pathway Name	Genes	Probes
4 studies	1-8	R-HSA-74713	IRS activation	5	10
4 studies	9-16	R-HSA-74713	IRS activation	5	10
4 studies	17-24	R-HSA-2179392	EGFR Transactivation by Gastrin	9	22
4 studies	25-32	R-HSA-3595177	Defective CHSY1 causes TPBS	8	21
6 studies	1-8	R-HSA-2485179	Activation of the phototransduction cascade	11	20
6 studies	9-16	R-HSA-8939242	RUNX1 regulates transcription of genes involved in differentiation of keratinocytes	8	25
6 studies	17-24	R-HSA-9694631	Maturation of nucleoprotein	16	27
6 studies	25-32	R-HSA-5218921	VEGFR2 mediated cell proliferation	21	53

4.3 Discussion

In the results we provided a comparison between pathway expression and gene expression methods with the same linear feature selection and classification methodology on train/test partitions of the GSE73072 data set. Both methods selected features (pathways or genes) using

influenza training data, then use these selected features in a LOSO (Leave One Subject Out) cross validation experiment with an SVM classifier on testing data. In these experiments, we found that pathway expression (CPE and LPE) methods generally produce higher test BSRs than gene expression methods. Specifically, we found that pathway expression produces higher test BSR than gene expression on 10 out of 12 classification experiments.

We used the distributions of test BSRs across experiments and time bins to conclude that pre-computed, directed edges with PageRank centrality is the "best" centrality configuration. We found that CPE with pre-computed edges and PageRank centrality produced the highest test BSR out of all CPE configurations for most experiments and time bins. Consequently, we reported the pre-computed, directed edges with PageRank centrality CPE configuration in all our classification rates and pathway ranking comparisons.

In addition to the feature selection and classification experiments, we compared these selected pathways from pathway expression methods to two standard gene expression pathway analysis methods: CePa and ORA. We found that the pathways selected from pathway expression methods generally have little similarity to pathways from these gene expression methods. This suggests that the pathway expression methods along with SSVM feature selection provides a unique pipeline that selects discriminatory pathways which are not detected by standard pathway analyses on gene expression data.

Pathway expression methods are also pulling out some respiratory virus signal since LPE produced a non-zero overlap with influenza pathways on 6 out of 8 experiments and time bins and CPE (with pre-computed, directed edges and PageRank centrality) had a non-zero overlap with the influenza pathways for each time bin in the 6 study experiment while sporting the highest overlap in 2 out of the 4 time bins. Since CPE has a non-zero overlap with the influenza pathways for the each of the time bins in the 6 study experiments, we suggest investigation into the links between respiratory viruses and the top pathways from this method, namely R-HSA-2485179, R-HSA-8939242, R-HSA-9694631 and R-HSA-5218921. R-HSA-8939242 and R-HSA-

9694631 have the most promise since they were the top pathways at the time bins where CPE had the highest overlap with the influenza pathways out each one of the tested methods.

When we look at both the highest magnitude SVM weight pathways from CPE by experiment. Overall, we see that R-HSA-74713 and R-HSA-3595177 appear in both lists. Therefore these are the most discriminatory of the pathways listed in Section 4.2. In fact, the R-HSA-74713 pathway appeared in more than one experiment and time bin. Our methods detected two insulin-related pathways: R-HSA-74713 and R-HSA-264876. R-HSA-74713 is a mediator of insulin signaling events and R-HSA-264876 is an insulin processing pathway. Insulin signaling is related to the influenza virus because the influenza virus impairs insulin signaling and down-regulates the expression of genes in the insulin pathway [81, 82]. Additionally, R-HSA-3595177 is involved in the synthesis of chondroitin sulfate which has been shown to be associated with pulmonary immune response to influenza infection by Brune et al. [83].

Many of the other pathways that are found to have the highest total SSVM weights across all experiments and time bins have links to respiratory viruses. The R-HSA-163680 pathway involves AMPK signaling which is known to be significant in the modulation of viral infections [84]. Aryl hydrocarbon receptor (ARH) signaling (the R-HSA-8937144 pathway) is an important part of the immune system, and ARH specifically regulates the immune response which is directly related to infection [85,86]. We find that the R-HSA-5676594 pathway is related to NF-κB which is known to activate during RSV infection, especially early during the infection [87,88]. The R-HSA-9634635 pathway involves signaling through PRKCZ and it was found in one study that suppressing the expression of PRKCZ reduces RSV infection [89]. The R-HSA-5603037 pathway is associated with IRAK4 deficiency. Kim et al. found that IRAK4 kinase activity is involved with TLR-dependent immune responses and the influenza virus is dependent on IRAK4 kinase activity [90].

Not only does our implementation of pathway expression produce interesting pathways, it also is robust to perturbation in theory and in practice. Linear pathway expression for a given pathway is robust to perturbation of the gene expression levels in a pathway by mean 0 noise

because it is a mean of gene expression levels in a pathway. Centrality pathway expression is only approximately robust to such an addition of noise since it is a weighted mean of gene expression levels. In our experiments we found pathway expression methods proved to be more robust to subject batch effect than gene expression methods. Linear pathway expression was the pathway expression method that was the least affected by subject batch affect.

We also examine the robustness of pathway expression with regards to the pathways selected using SSVM feature selection on pathway expression data. We do this by comparing these selected pathways to those selected using gene expression features along with standard pathway ranking methods. Specifically, we look at the overlap between the pathways found with 4 studies and those found with 6 studies. We find that the pathways selected by ORA have the highest overlap, the pathway expression method have the second highest and CePa has the lowest overlap. It is expected that this overlap is somewhat proportional to the number of pathways that were selected by each method since we are sampling from a background set of pathways. In an extreme example, if the each of the two pathway sets have more than half of the total pathways, then they must overlap. We see this proportionality of the number of pathways selected and the overlap for each of the methods. Additionally, the viruses in the 4 study training dataset are H3N2 and H1N1, and the viruses in the 6 training dataset are H3N2, H1N1, RSV and HRV. We note that the relatively high overlap in the pathways detected by ORA indicates that it pulls out the general respiratory virus signal but does not detect the differences between the different viruses in the training datasets. On the other hand, the low overlap for CePa suggests that it detects the difference between the datasets but does not capture the overall respiratory virus signal. Pathway expression methods take the middle ground since their overlap magnitude is between the standard pathway analysis methods (ORA and CePa). That is to say, the pathways found using pathway expression encapsulate the difference between the viruses in the training datasets while still maintaining the overall respiratory virus signal.

Now we return to analyzing our methodologies. CPE and LPE are simple linear models for translating gene expression data to pathway expression data which result in improved classifi-

cation rates of our tested machine learning models. The linear nature of our implementation of pathway expression, especially LPE, makes it possibly one of the simplest methods available for transforming vectors from the gene space to the pathway space. A different non-linear pathway expression formulation may improve upon the results with pathway expression presented in this paper. One non-linear modification of LPE has been done by using the absolute values of the entries in the gene expression matrix instead. This modification collapses the gene expression data into the positive hyper-octant and results in a loss of information. However, there are many possible variants of non-linear pathway expression that require further investigation.

The pathway expression methods in this paper are meant to re-inspire a wide variety of investigations into the pathway expression pipeline including batch correction, pathway expression generation and downstream pathway expression analyses. Investigation into the robustness of pathway expression methods to batch effects could be done by running experiments with corrections for study or strain effects. In this paper, we applied batch normalization to gene expression vectors before calculating pathway expression. Another interesting experiment in future work could be to apply batch normalization to the pathway expression vectors themselves. Within the framework of CPE, different methods for network generation, and centrality measures still needs to be tested. LPE and CPE are arguably the simplest pathway expression methods since they are effectively an average of gene expression levels. From a bird's eye view, pathway expression methods are any method that transforms a gene expression matrix to a pathway expression matrix. Therefore any type of pathway transition matrix, as well as a non-linear transform, can be applied to gene expression data to produce pathway expression data. This leaves the door wide open to the use of other central prototypes for generating pathway expression matrices. The pathway expression transformation can be seen as just a pre-processing step. Hence, any machine learning algorithm that has been used on gene expression data can be used on pathway expression data to select pathways, classify pathways, cluster pathways, etc. Hence, the opportunity for novel work with pathway expression can include an investigation into pathway expression pre-processing, batch correction techniques, methods for pathway

expression generation, and downstream pathway expression analyses using statistics, machine learning algorithms and more. The bottom line is, creativity and/or applying sound biological principles while designing a pathway expression workflow will be key in selecting meaningful pathways and perhaps improve classification rates.

Our 4 to 2, 4 to 3 and 6 to 1 classification experiments in Chapter 4.2 are designed to mimic the experiments that were done by Aminian et al. [56]. For a direct comparison of best CPE results between this paper to the gene expression results from Aminian et al. see Table 4.7. Our results with LPE, CPE **and GE** generally produce lower SVM BSRs than those from the same experiments in Aminian et al. This is not surprising since the feature selection technique in this paper is far more simple and less robust than what was done by Aminian et al. However, at the 25 to 32 hour time bin, the pathway expression methods produce higher BSR than Aminian et al. by 1 to 4 percent in two out of three experiments. The higher classification BSRs at the latest time bin are consistent with the concept that genes generally don not work in concert in pathways during the early hours after infection. A PCA of all the probes related to genes in immune response pathway, α/β Interferon, by Aminian et al. highlights the inactivity of the pathway during early hours of infection and activation of the pathway during later hours after infection. Perhaps we would see pathway expression methods produce even higher BSRs by running experiments at later time bins and/or using the same feature selection technique that was used by Aminian et al. Additionally, any biologically informed modification of the pathway expression pipeline presented in this paper could increase test classification rates while detecting even more biologically informative pathways. We choose to include this table and it's analysis for completeness and to inspire future work on improving pathway expression transformations and optimization of downstream machine learning feature selection and classification architectures on pathway expression data.

Table 4.7: Classifications rates of SVM in a LOSO experiment on test data across different experiments within 32 hours after infection. All experiments in this table use LIMMA normalization on subject identifier. The majority of the best results from this paper are using CPE.

Time Bin	Paper	4 to 2	4 to 3	6 to 1
1 to 8	This dissertation	80.04	73.45	**89.44**
1 to 8	Aminian et al.	**84.74**	**82.06**	87.97
9 to 16	This dissertation	82.13	80.84	89.44
9 to 16	Aminian et al.	**93.21**	**90.37**	**100.00**
17 to 24	This dissertation	77.07	77.27	80.35
17 to 24	Aminian et al.	**81.58**	**78.82**	**86.36**
25 to 32	This dissertation	**92.49**	82.83	**90.91**
25 to 32	Aminian et al.	88.21	**85.46**	89.44

Overall, the pathway expression analysis framework provides a concise approach for characterizing the biological processes associated with the host response to infection. Although we produce lower classification rates than those in Aminian et al., in a head to head comparison with gene expression, we see that CPE and LPE improve classification rates. Moreover, we envision that pathway selection using this approach may provide additional insights into biological mechanisms associated with the host response to infection. The pathways detected using CPE are connected to the immune response and to some specific respiratory viruses. Our preliminary experiments addressing time-evolving human clinical data provide some insight into the pathway activity for humans infected with respiratory viruses. Finally, we offer a Python package for computing CPE, LPE and CePa: https://github.com/nmank/PathwayAnalysis.

4.4 Methods

In this section we detail the models, metrics, experiments, and data sets used in this paper. We run two mirrored classification and pathway selection pipelines on pathway expression data and gene expression data.

Our pipeline used in the pathway expression (CPE and LPE) classification experiments is the following:

1. Batch normalize each train and test partition separately by subject ID using LIMMA.
2. Select pathways, viewed as features, in pathway expression data using the training data (LPE-SSVM or CPE-SSVM).
3. Run a LOSO classification experiment using SVM on test pathway expression data restricted to the selected pathways (generated by LPE-SSVM or CPE-SSVM) and record mean classification statistics (eg. BSR).

For pathway selection using pathway expression we simply use the pathways selected by SSVM on the training data and rank these selected pathways by their SSVM weights.

The pipeline for the GE classification experiments is:

1. Batch normalize each train and test partition separately by subject ID using LIMMA.
2. Select genes, viewed as features, using the an SSVM on the training data.
3. Run a LOSO classification experiment using SVM on test gene expression data restricted to the selected genes and record the mean classification statistics (eg. BSR).

For pathway selection using gene expression data we use the genes selected by SSVM on the training data as input ORA or CePa pathway selection. ORA is implemented using the p-value from the analysis.identifiers function reactome2-
py, (https://github.com/reactome/reactome2py). CePa is implemented in the GLPE.simple_ transform function in the PathwayAnalysis package:

(https://github.com/nmank/PathwayAnalysis). Details on the ORA and CePa methodologies are provided in Section 4.4.4.

4.4.1 Data Set (GSE73072)

We perform experiments on the GSE73072 data set [55] from the NCBI Gene Expression Omnibus (GEO). This data set is a microarray gene expression data set for human subjects challenged with the influenza virus. These data were collected from 7 studies by Duke, UVA and hVIVO and was funded by the Defense Advanced Research Projects Agency (DARPA). The entire data set consists of 22277 probe identifiers and 148 human subjects infected with four different types of respiratory viruses: HRV, RSV, H1N1 and H3N2. The data samples are collected at irregular time intervals from 38 hours before infection to 680 hours after infection. The data can be found here https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE73072.

We run two binary classification experiments with GSE73072 data set that are copies of experiments by Aminian et al. [56]. In these experiments we classify between control subjects and shedding subjects. Our control group consists of all subjects between, and including, 38 to 0 hours before infection. We define (early) shedders as pre-symptomatic subjects within 32 hours after infection who will eventually be characterized as shedders over the course of the immune response. We break our data sets down further into 4 sets of shedders from evenly spaced time bins within 32 hours after infection. Our train/test splits are described in Table 4.1 using the format: study identifier (virus).

For data pre-processing, we perform two steps which follow the experimental design of Aminian et al. so that we can make a faithful comparison of classification rates [56]. First, we normalize the entire data set using the robust multi-array average (RMA) [91] method. We then correct for subject differences by applying normalization across subject identifier using the LIMMA package [92] to each train/test partition separately.

We use the Reactome Pathway Database [62] to determine pathway membership. The graphite package [93], using the information from Reactome, gives protein-based edges which are

then translated using Entrez gene identifiers to generate pathway networks with edges. We also directly use the Reactome database information, available on the Reactome website, to ensure genes are in appropriate pathways and edges are within Reactome pathways only (any edges outside of a pathway are removed before analysis).

The GSE73072 data set is a microarray data set with features given as microarray probe identifiers (probe IDs). We choose to do all our analyses on the probe IDs rather than determine a mapping to Entrez identifiers (Entrez IDs) in order to retain as much feature information as possible. However, this means that the pathway membership and network edge information needs to be converted to probe IDs. We use the following mapping from Entrez ID pathway networks to probe ID pathway networks. Although this method results in a loss of some information, *we retain all the probe IDs that correspond to Entrez IDs from in affymetrix platform file.* The edges between probe IDs derived from this method are used for the edges and edge weights for the pre-computed directed and undirected edges for CPE and CePa workflows.

1. For every probe ID in the affymetrix platform file, map it to the first Entrez ID in its associated list of Entrez IDs (in the event of multiple maps). Since some probe IDs are not mapped to Entrez IDs via the platform file, we loose 37.1% of the probe IDs in the original data set.

2. Use the pathway network information to draw an Entrez ID network

3. Map all the Entrez ID nodes from the pathway networks to probe IDs using this mapping. *Note: this means some nodes are mapped to multiple probe IDs.* Assign edges in the pathway networks with probe ID nodes according to the edges between Entrez IDs. Multiple probe IDs that map to the same Entrez ID will have no edges between them. Probe IDs with no Entrez ID to map to are dropped. Entrez IDs with no probe IDs mapping to them are dropped.

For a visual of this mapping see Figure 4.8.

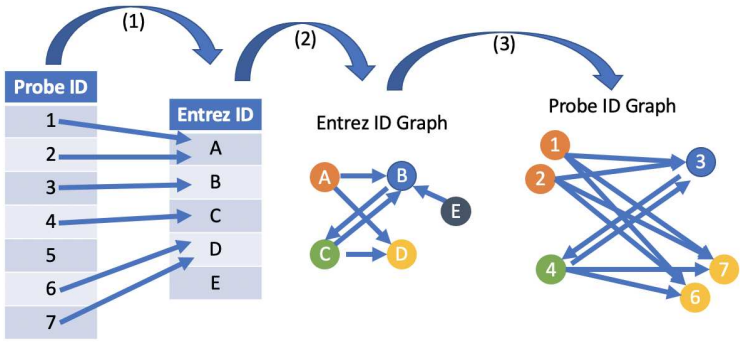


Figure 4.8: How we generate Probe ID pathway networks using the platform file and Entrez ID pathway networks.

4.4.2 Pathway Expression (LPE and CPE)

Pathway expression is a method to represent biological data as a pathway expression matrix with pathway expression levels as features. In this section we develop the mathematical underpinnings of the pathway expression methods used in this paper.

Let $\mathbf{X} \in \mathbb{R}^{p \times n}$ be a GE matrix with n gene expression levels for p subjects. In a gene expression matrix, each gene is assigned a gene index and each subject is assigned a subject index. We can map $\mathbf{X}$ to a "pathway expression matrix" $\mathbf{Y} \in \mathbb{R}^{p \times m}$ with m pathway expression levels for p subjects. In a pathway expression matrix each pathway is assigned a pathway index and each subject is assigned a subject index. We define such a linear mapping in (5.8) using the pathway transition matrix $\mathbf{P} \in \mathbb{R}^{n \times m}$.

$$\mathbf{Y} = \mathbf{XP} \tag{4.1}$$

Let $P^{(i,j)}$ be the entry in the ith row and jth column of $\mathbf{P}$. The most simple definition of $\mathbf{P}$ is pathway membership in (4.2) and is used to compute a linear pathway expression (LPE) matrix.

$$P^{(i,j)} = \begin{cases} 1 & \text{if gene } i \text{ is in pathway } j \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

A slightly more involved definition of pathway expression is centrality pathway expression (CPE) where we weight gene expression levels by their centrality within a pathway network. This path-

way transition matrix is denoted $\mathbf{P}_c$. We construct $\mathbf{P}_c$ using the centrality of gene i in the jth pathway network $c_j(i)$ in (4.3) where $\|\mathbf{c}_j\|_1 = \sum_{i=1}^{n} |c_j(i)|$.

$$P_c^{(i,j)} = \begin{cases} \frac{c_j(i)}{\|\mathbf{c_j}\|_1} & \text{if gene } i \text{ is in pathway } j \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

The full CPE algorithm is in Figure 4.9. For our implementations of CPE data sets, we use the same 6 combinations of pathway network edges and centrality measures that are used in CePa: pre-computed directed, pre-computed undirected or correlation edges with either out-degree (normalized by maximum out-degree) or PageRank centrality methods.

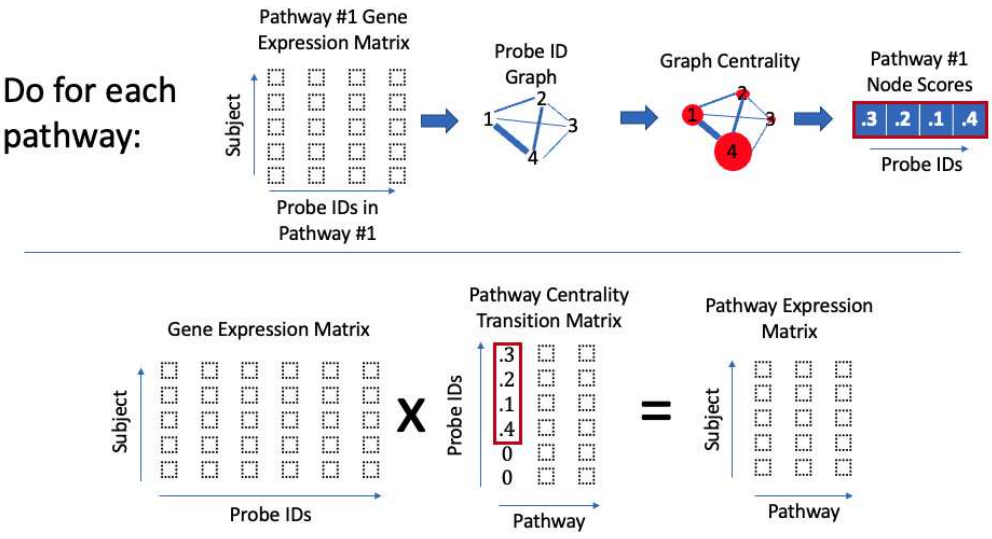


Figure 4.9: The workflow for CPE.

Now we will investigate CPE further by formulating the calculation of pathway expression as an optimization problem. Let $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m]$ where $\mathbf{y}_i \in \mathbb{R}^p$. We call $\mathbf{y_j}$ the pathway expression vector for pathway j. Analogously, let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ where $\mathbf{x}_i \in \mathbb{R}^p$ where we call $\mathbf{x}_i$ the gene expression vector for gene i. We claim

$$\mathbf{y}_j = \arg\min n \sum_{i=1}^{n} P^{(i,j)} \|\mathbf{z} - \mathbf{x}_i\|_2^2. \tag{4.4}$$

The obvious solution to this problem is the scaled average of the set $\{P^{(i,j)}\mathbf{x}_i\}_{i=1}^{n}$. But notice that this is exactly

$$\mathbf{y}_j = \sum_{i=1}^{p} P^{(i,j)}\mathbf{x_i}. \tag{4.5}$$

Doing this for every j we recover (5.8).

This outlines a methodology for translating a data set from the gene space to the pathway space using types of pathway expression. The LPE and CPE methods used in this paper can be found at on GitHub in the PathwayAnalysis repository https://github.com/nmank/PathwayAnalysis. The code in this repository can be used as an out of the box method for pathway expression analysis on other data sets. We can now leverage our data sets in the pathway expression matrix format to determine discriminatory pathways for a classification problem.

4.4.3 SSVM Feature Selection

We use SSVM feature selection with gene and pathway expression matrices to find the best genes and pathways. This is done by running SSVM on the training data, then selecting features based on the SSVM weights. For simplicity, the SSVM feature selection in our experiments is limited to a rendition of the first iteration of iterated feature removal (IFR) from O'Hara et al. [59]. Throughout this section, features can be genes or pathways depending on whether the experiment uses gene expression or pathway expression.

Our feature selection methodology starts with all the m features, $P_1, P_2, \ldots, P_m$, ordered with respect to the corresponding magnitude of their SSVM weights, $w_1 \geq w_2 \geq \cdots \geq w_m \geq 0$, calculated on whatever classification experiment we are analyzing. Our goal is to take only the features that have significant weights from the SSVM model. To determine which weights are significant, we calculate the weight ratios $r_i = w_{i-1}/w_i$ and look for a "jump", that is, we find where the weights rapidly decrease for the first time over a certain threshold. This will be reflected in the ratios as a large "jump" in value. For our experiments we set our "jump" ratio at 5. After we have isolated the top features by weight, we add in the features that are at least .9 correlated to these features using the training data (either GE, LPE or CPE depending on the ex-

periment). The number of added correlated features changes depending on the initial feature set.

4.4.4 Pathway Ranking Using Gene Feature Sets

Let G be the set of all genes in the data set and $F \subseteq G$ be a feature set of genes. In this paper, the gene feature sets are calculated using SSVM feature selection in Section 4.4.3. In this subsection we will introduce two pathway ranking methods using a feature set of genes (or probe IDs): 1) ORA and 2) CePa. Let $P \subseteq G$ be the set of genes in a given pathway. These methods, ORA and CePa, assign a score to each pathway by leveraging the genes in F. The higher the score, the more important the pathway. In summary, each of these methods are a map $\phi : P \to \mathbb{R}$ where P is the set of all pathways.

ORA is one of the most simple and widely used pathway scoring methods, so it is an ideal ground-truth pathway ranking method. Generally, ORA is a methodology for investigating the statistical significance of the overlap of genes in the feature set with known pathways. Using the hypergeometric distribution, ORA determines the p-value of the significance of the overlap, which is the score of the pathway, by

$$p = 1 - \sum_{k=0}^{f-1} \frac{\binom{\hat{F}}{k}\binom{N-\hat{F}}{n-k}}{\binom{N}{n}}. \tag{4.6}$$

In (4.6), f is the number of genes in the overlap of the feature set and pathway, $\hat{F}$ is the number of genes in the feature set, N is the total number of genes possible, and n is the number of genes in the pathway. We implement this method using the Python package reactome-2py to use the ORA analysis tools on the Reactome website. The GitHub page for reactome2py is https://github.com/reactome/reactome2py.

CePa is a method which uses network centrality for pathway ranking on a given feature set, developed by Gu et al. [68]. This method combines the statistical notions that are used in ORA with biological pathway network information, specifically network centrality. For a given pathway P, we generate a pathway network where genes are the nodes. These edges are generated

the same as those generated for CPE. Specifically, edges are either correlation between gene expression levels across subjects or pre-computed edges, specifically those edges generated from known biological connections using the graphite package [93].

Let $c_P : G \to \mathbb{R}$ be a pathway centrality map from the gene space to the real numbers. Given a gene $g \in G$, $c_P(g)$ is the centrality of g within its pathway network. Then the rank for pathway P is just the sum of the centralities of the genes in both P and the feature set F as in (4.7).

$$\text{CEPA}(P) = \sum_{n \in P \cap F} c_P(n) \tag{4.7}$$

We perform six CePa experiments where we use pre-computed directed, pre-computed undirected or correlation edges with either out-degree (normalized by maximum out-degree) or PageRank centrality methods. To the best of our knowledge, this is the first time CePa has been used with PageRank centrality.

The final step in CePa is determining the pathway significance score relative to a collection of null feature sets. To do this, a large number of $m \in \mathbb{N}$ null trials are run with $|F|$ genes selected from a uniform distribution over all the genes. Then CePa is run for each of these null trials resulting in a set of null pathway rankings. Let the vector of the set of null pathway rankings for pathway P be denoted $\mathbf{n_P}$ and ordered by null feature set. Define the null value indicator map, denoted I, in (4.8).

$$I(\mathbf{n_P}) = \begin{cases} 1 & \text{if } \mathbf{n_P} > \text{CEPA}(P) \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

We then use the indicator map to find the “null value" for pathway P as the mean of the entries in $I(\mathbf{n_P})$ in (4.9).

$$\text{null value}(P) = \frac{\sum_{i=1}^{m} I(\mathbf{n_P})_i}{m} \tag{4.9}$$

Finally, the significant pathways from CePa are the pathways with the highest CePa scores and lowest null values. So given a pathway score threshold $\alpha \in \mathbb{R}$ and null value threshold $\epsilon \in \mathbb{R}$, we say a pathway P is significant if CEPA(P) $> \alpha$ and null value(P) $< \epsilon$. For this paper, we take the top 60 pathways with null value less than .05.

Overall, CePa consists of three steps. First determine pathway scores, then find null values and select pathways using pathway score and null value thresholds. An overview of the CePa algorithm workflow is provided in Figure 4.10.

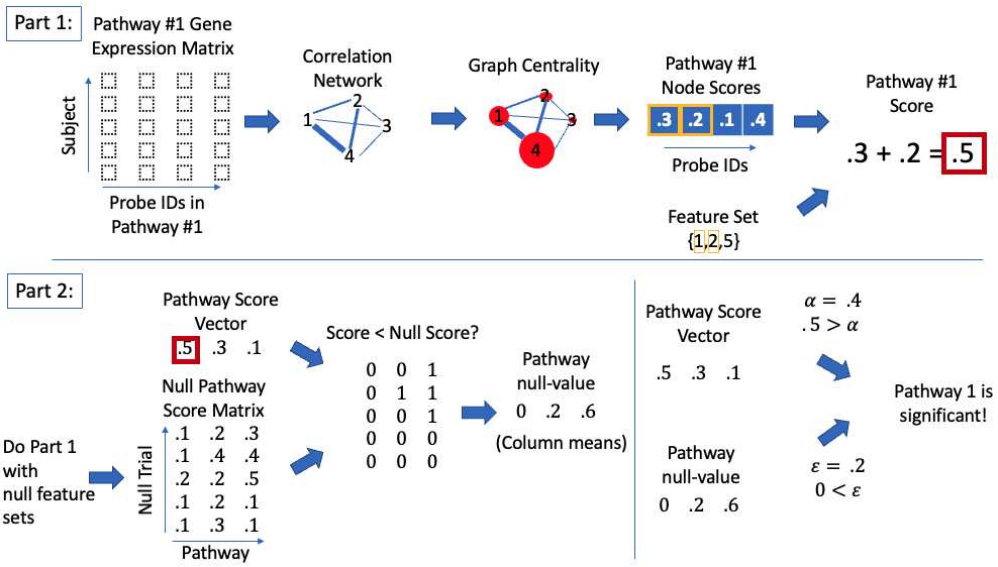


Figure 4.10: The workflow for CePa.

4.4.5 Evaluation

For comparison between sets of pathways, we use the Jaccard/Tanimoto similarity coefficient. Given two sets of pathways, P and P', the Jaccard/Tanimoto similarity coefficient between these two sets is defined in (4.10), as

$$J(P,P') = \frac{|P \cap P'|}{|P \cup P'|} \tag{4.10}$$

We evaluate the novel linear pathway expression data by using SVM to classify between controls and shedders in a LOSO experiment on the test data set using the only the pathways that were selected on the training data. We use the mean test BSR, precision, recall and accuracy of these SVMs to determine the "best" method.

We compute mean test BSR for these LOSO experiments on test data in three steps. 1) Compute a confusion matrix for each subject's SVM experiment on the test data. 2) Sum all the subject confusion matrices. 3) Compute the average of the true positive rate and the true negative rate. This metric provides a better model assessment than accuracy on data sets with imbalanced class sizes. Precision, recall and accuracy are computed in a similar manner leveraging the sum of all the confusion matrices across experiments. *Note: that standard deviation for these statistics is not available since we are computing them from a sum of the confusion matrices across the LOSO experiments.*

These pathway expression results are compared to results using the same workflow on gene expression data. We run an SSVM feature selection on the training gene expression matrices to select discriminatory genes. Then we restrict the gene expression matrices of the test data to the discriminatory genes and run a LOSO SVM experiment on the test data set and use the BSR to compare against the BSRs from the LPE methods.

Chapter 5

Module Representatives for Refining Gene Co-Expression Modules

5.1 Introduction

Modules of genes are useful structures to group genes into their downstream biological functions. Among their many applications, gene modules have been used as tools for finding prognostic markers by classifying subtypes of cancer and identifying groups of genes which are related to heart failure [94, 95]. A gene module is generally defined as a set of "co-expressed genes." Gene expression levels are used to determine gene co-expression and are measured using transcriptomics data collected using either microarrays or the newer technology, RNA sequencing (RNA-seq).

Early work in the detection of co-expression modules used clustering of biological co-expression networks [96–99]. Edge weights in these networks are found either using known biological connections between genes or statistical methods to measure similarity between gene expression levels, most commonly calculating correlation. In 2005, Zhang et al. proposed WGCNA (weighted gene co-expression network analysis). It uses a branch cut of an agglomerative hierarchical clustering tree of a correlation gene co-expression network to detect modules [100]. In 2008, Langfelder et al. released a comprehensive R package to perform WGCNA [101]. In the same year, these methodologies were interpreted geometrically by Horvath et al. [102]. The R package provided a stable platform for the WGCNA workflow. In 2016, Bailey et al. used WGCNA to detect modules and provide a detailed analysis of pancreatic cancer [103]. Other approaches for module detection include techniques like FLAME (Fuzzy clustering by Local Approximation of Memberships) and ICA (Independent Component Analysis) [104, 105]. A 2016 survey of mod-

ule detection methods provides a comparison of more than 40 methods and highlights some of the shortcomings of WGCNA [106]. A more recent survey paper of gene co-expression analysis by Van Dam et al. referenced 9 methods for co-expression module detection and underlined the significance of the WGCNA workflow for co-expression module detection [107]. Even the simple correlation edge weight techniques from WGCNA are used in the popular protein-protein interaction network database STRING [108]. In summary, module detection using co-expression networks has at least a 20 year history in bioinformatics and the WGCNA algorithm has played a central role in applications.

WGCNA also addresses the question of module representation by using the "eigengene." Zhang et al. use the "eigengene" as a single vector representative of the genes in a module [100]. Correlation between eigengenes is used by Zhang et al. to compare modules to one another. These eigengenes are also used to define "fuzzy" module membership by calculating similarities between gene expression vectors and each eigengene. Langfelder et al. use eigengenes to compare different biological groups (eg. tissue types, species and other biological delineations) using 3 steps: 1) "consensus module" detection, 2) hierarchically clustering the eigengenes of each module, and 3) comparing these clusterings and eigengenes to one another [109]. An overview of additional methods for module level analysis is provided by Wang et al. This includes references to module comparison methods that incorporate protein-protein interaction and transcriptomics data to generate module networks [110].

More recently, researchers have been building upon hierarchical clustering for module detection using module refinement. These methods use WGCNA modules as an initialization for another clustering algorithm to generate refined modules from WGCNA. For example, k-means is one of these module refinement algorithms. The application of k-means clustering is sensitive to center initialization; so using the WGCNA hierarchical clustering technique as a method for initialization provides a significant head start for the clustering. k-means clustering with eigengenes to refine WGCNA the module membership was introduced in 2017 and was shown to increase biological significance of the modules through functional enrichment analysis [111].

In 2021, Hou et al. modified the module refinement algorithm by clustering using module connectivity which is measured by "correlation distance," while eliminating the eigengene [112].

The focus of this paper is on developing new methods for the representation of modules, analyzing their properties, and implementing them in a k-means-type (Linde-Buzo-Gray) clustering algorithm for gene co-expression module refinement. We present a methodology which seeks to improve upon the WGCNA method for module representation by introducing eigengene subspaces, flag mean [3], flag median [113], and module expression vectors [114]. We seek to develop methods which improve biological significance of the resulting modules, at the possible expense of Machine Learning classification rates. The main contributions of this paper are listed below.

- A generalization of the eigengene for module representation.
- The application of the Linde-Buzo-Gray (LBG) subspace clustering schemes with the flag mean, flag median, eigengene subspace and module expression vector to refine WGCNA modules.
- A demonstration that the proposed refinements to WGCNA may lead to enhanced biological significance of the resulting modules.

5.2 Methods

Our workflow is detailed in Figure 5.1 and summarized in the following three bullets:

- Calculate initial modules by running WGCNA.
- Find refined modules by running LBG clustering with different central prototypes and distances with initial clusters from WGCNA.
- Evaluate biological significance of the refined modules.

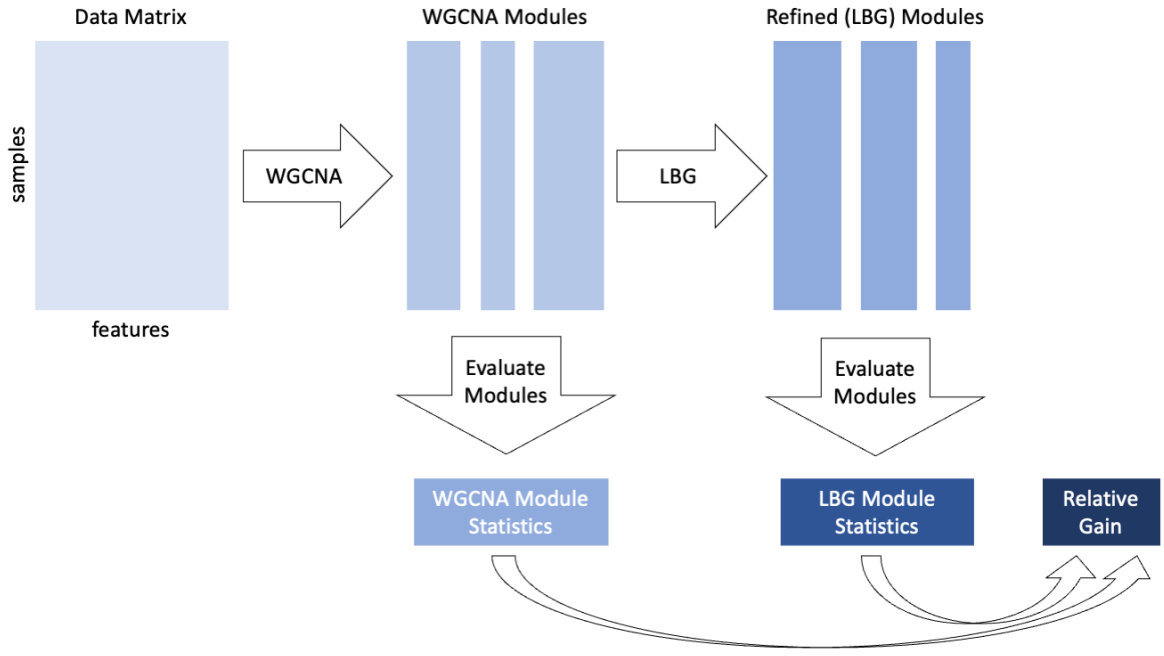


Figure 5.1: The general workflow for module refinement and evaluation.

In this section, we will describe how to use the new module representatives along with LBG clustering to compute refined modules. Then we explain how we use a classification problem and gene ontological significance to evaluate these modules. Section 5.2.2 describes the flag mean, flag median and the eigengene subspace module representatives. The module expression module representative is introduced in Section 5.2.3. Our LBG clustering implementation is explained in Section 5.2.4. The final section, Section 5.2.5 outlines our methods for determining the viability of the modules we find.

5.2.1 Introduction to the Grassmann Manifold

The Grassmann manifold, denoted $\mathrm{Gr}(k,n)$, is the manifold whose points parameterize k dimensional subspaces of n dimensional space. Let $[\mathbf{X}] \in \mathrm{Gr}(k,n)$ denote a k-dimensional subspace of n dimensional space that is spanned by the columns of the matrix $\mathbf{X}$. We restrict our matrix representatives for points in $\mathrm{Gr}(k,n)$ to be $\mathbf{X} \in \mathbb{R}^{n\times k}$ with k orthonormal columns.

Suppose we have a gene expression matrix with n samples of m genes in a module $\mathbf{G} \in \mathbb{R}^{n\times m}$. $\mathbf{G}$ holds m gene expression vectors, $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_m]$. Then we can transform these m gene expression vectors to m points on $\mathrm{Gr}(k,n)$ using k-nearest neighbors with cosine similarity. Given two gene expression vectors $\mathbf{g}_i, \mathbf{g}_j \in \mathbb{R}^n$, we measure the cosine distance between $\mathbf{g}_i$ and $\mathbf{g}_j$ as

$$d(\mathbf{g}_i, \mathbf{g}_j) = 1 - \frac{\mathbf{g}_i^T \mathbf{g}_j}{\|\mathbf{g}_i\|_2^2 \|\mathbf{g}_j\|_2^2}. \tag{5.1}$$

Then take the subspace representative for the gene i as the span of it's k-nearest neighbors. Suppose $\mathbf{h}_1, \mathbf{h}_2, \ldots \mathbf{h}_k \in \mathbb{R}^n$ are the k nearest neighbors to gene i. Note: this means that $\mathbf{g}_i = \mathbf{h}_j$ for some $j = 1, 2, \ldots, k$. Now we construct the matrix $\mathbf{H}_i = [\mathbf{h}_1, \mathbf{h}_2, \ldots \mathbf{h}_k] \in \mathbb{R}^{n\times k}$. We generate a representative for a point on $\mathrm{Gr}(k,n)$ by taking the first k columns of the QR-decomposition of $\mathbf{H}_i$. We denote this representative $\mathbf{X}_i$. So, we have $[\mathbf{X}_i] \in \mathrm{Gr}(k,n)$. Note: in our experiments we will take $k = 1, 2$ dimensional subspaces for each of our samples. On the other hand, we will be calculating $r = 1, 2, 4, 8$ dimensional module representatives. We will refer to the subspace dimensions k as the *data dimension* and r as the *center dimension*.

Principal angles between subspaces are a common dissimilarity measure that is invariant to orthogonal transformations [2, 3]. Take $[\mathbf{X}] \in \mathrm{Gr}(k,n)$ and $[\mathbf{Y}] \in \mathrm{Gr}(r,n)$. We denote the vector of principal angles between $[\mathbf{X}]$ and $[\mathbf{Y}]$ as $\boldsymbol{\theta}([\mathbf{X}], [\mathbf{Y}]) \in \mathbb{R}^{\min\{k,r\}}$. The entries in this vector are between 0 and $\pi/2$. For details on the calculation of principal angles, see [2].

The vector of cosines of the principal angles between $[\mathbf{X}]$ and $[\mathbf{Y}]$ is used to measure similarity between subspaces and can be computed by $\|\cos(\boldsymbol{\theta}([\mathbf{X}], [\mathbf{Y}]))\|_2^2 = \mathrm{tr}(\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X})$. When $r = k = 1$, namely when $\mathbf{X}$ and $\mathbf{Y}$ are just a unit vectors, this similarity amounts to the correlation between $\mathbf{X}$ and $\mathbf{Y}$.

The chordal distance on $\mathrm{Gr}(k,n)$ is $\|\sin(\boldsymbol{\theta}([\mathbf{X}], [\mathbf{Y}]))\|_2$ and the geodesic distance is $\|\boldsymbol{\theta}([\mathbf{X}], [\mathbf{Y}])\|_2$ [4]. We can calculate the chordal distance by using $\sin^2(\theta([\mathbf{X}], [\mathbf{Y}])) = \min(r,k) - \mathrm{tr}(\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X})$. This is preferred to the canonical Riemannian geodesic distance since it can be calculated without needing to find the actual principal angles. In addition, the algorithms for chordal distance prototypes (e.g., SVD and FlagIRLS [3, 113]) are generally faster to compute

than algorithms for geodesic distance prototypes (e.g., Karcher and Riemannian-Weiszfeld [31, 33]).

5.2.2 Subspace Prototypes

In order to allow more flexibility in our generalization of optimization problems to subspaces, we work with points that are not all necessarily on the same Grassmannian but are in the same ambient space. Suppose we have a set of subspaces of k-dimensional space,

$$\{[\mathbf{X}_1], [\mathbf{X}_2], \ldots, [\mathbf{X}_m]\} \subset \mathrm{Gr}(k, n).$$

We want to find an r-dimensional subspace of $\mathbb{R}^n$, $[\mathbf{Y}^*] \in \mathrm{Gr}(r, n)$, that is the center of these points, i.e., that $[\mathbf{Y}^*]$ is a solution to

$$\underset{[\mathbf{Y}]\in\mathrm{Gr}(r,n)}{\arg\min} \sum_{i=1}^{m} d([\mathbf{X}_i], [\mathbf{Y}]) \tag{5.2}$$

where $d : \mathrm{Gr}(k, n) \times \mathrm{Gr}(r, n) \to \mathbb{R}$ measures dissimilarity between its arguments.

The r-dimensional **flag mean** is the subspace central prototype that solves

$$\min_{[\mathbf{Y}]\in\mathrm{Gr}(r,n)} \sum_{i=1}^{m} \|\sin(\theta([\mathbf{X}_i], [\mathbf{Y}]))\|_2^2. \tag{5.3}$$

The flag mean is just the r left singular vectors of the block matrix $[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m]$ associated with the r largest singular values [3]. The complexity of this algorithm is $O\left(n\left(\sum_{i=1}^{m} k_i\right)^2\right)$.

The r-dimensional **flag median** is the subspace central prototype that solves

$$\min_{[\mathbf{Y}]\in\mathrm{Gr}(r,n)} \sum_{i=1}^{m} \|\sin(\theta([\mathbf{X}_i], [\mathbf{Y}]))\|_2. \tag{5.4}$$

The flag median is found by iteratively re-weighted flag means in the algorithm FlagIRLS [113]. The complexity of this algorithm depends on N_δ, the number of iterations and δ, the convergence parameter. The complexity of FlagIRLS is $O\left(nN_\delta\left(\sum_{i=1}^{m} k_i\right)^2\right)$.

The **eigengene** comes from the Principal Component Analysis (PCA) of the gene expression matrix. Instead of representing the columns of $\mathbf{G}$ as subspaces, we only mean-center the rows of $\mathbf{G}$. Let $\widehat{\mathbf{G}}$ denote $\mathbf{G}$ with mean centered rows. The eigengene $\mathbf{y}^*$ is the solution to the optimization problem

$$\mathbf{y}^* = \underset{\mathbf{y}^T\mathbf{y}=1}{\arg\max} \mathbf{y}^T \widehat{\mathbf{G}}\widehat{\mathbf{G}}^T\mathbf{y}. \tag{5.5}$$

We define the *eigengene subspace as the generalization of the eigengene to higher dimensional subspsaces*. Specifically, the eigengene subspace is the weights for the first r principal components of the PCA of the gene expression matrix. The eigengene subspace is the solution to

$$\max_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \mathrm{tr}(\mathbf{Y}^T \widehat{\mathbf{G}}\widehat{\mathbf{G}}^T\mathbf{Y}). \tag{5.6}$$

The solution to this problem is exactly the r right singular vectors of the matrix $\widehat{\mathbf{G}}^T$ that are associated with the r largest singular values. This is the same as the r left singular vectors of the matrix $\widehat{\mathbf{G}}$ that are associated with the r largest singular values. The complexity of computing the eigengene subspace is the same as the flag mean: $O\left(nm^2\right)$.

There is a direct link between the flag mean and the eigengene subspace which can be seen by rewriting (5.6) as

$$\min_{[\mathbf{Y}]\in \mathrm{Gr}(r,n)} \sum_{i=1}^{p} 1 - \widehat{\mathbf{g}}_i^T \mathbf{Y}\mathbf{Y}^T \widehat{\mathbf{g}}_i. \tag{5.7}$$

Notice, this is the r-dimensional flag mean optimization problem of a set of 1-dimensional subspaces if we require each $\widehat{\mathbf{g}}_i$ to be a unit vector.

5.2.3 Module Expression

Pathway expression is a methodology for dimensionality reduction of gene expression data which translates the gene expression feature space into a pathway expression feature space [114]. Linear pathway expression (LPE) methods generate a pathway expression value by averaging gene expression levels for the genes in the pathway. Centrality pathway expression (CPE)

also calculates an average, but weights the genes in the pathway by their centralities in a gene network for the pathway.

In this paper we translate the notion of pathway expressions to module expression by using collections of genes in modules rather than pathways. Suppose we have a gene expression matrix for a module $\mathbf{G} \in \mathbb{R}^{n \times m}$ with n subjects and m genes. In a gene expression matrix, each gene is assigned a gene index and each subject is assigned a subject index. We can map $\mathbf{G}$ to a "module expression vector" $\mathbf{y} \in \mathbb{R}^n$. We define such a linear mapping

$$\mathbf{y} = \mathbf{G}\mathbf{m} \tag{5.8}$$

using the module transition vector $\mathbf{m} \in \mathbb{R}^n$. For linear module expression, we take $\mathbf{m}_i = 1$ for all i. For centrality module expression, we take $\mathbf{m}_i = c_i$ where $c_i \in \mathbb{R}$ is the centrality of gene i in a network of genes in the module.

In this paper, we will use correlation between genes expression levels to generate edges for our module gene network and PageRank centrality [115] for our centrality measure. We choose this because PageRank centrality with a correlation network produced a higher inter-quartile range of balanced success rate than degree centrality in the initial pathway expression analysis paper [114]. With these parameters, the computational complexity of module expression for one module is the same as the computational complexity of PageRank. PageRank is essentially the complexity of computing the largest eigenvalue of a $m \times m$ matrix which we can do using the power iteration algorithm. Let N_δ be the number of iterations and δ be the convergence parameter. Then, the computational complexity is $O\left(m^2 N_\delta\right)$.

5.2.4 Module Refinement with LBG Clustering

We mathematically formalize module refinement as a mapping between sets of modules. Let F be the set of all features for a data set and $\{W_1, W_2, \ldots, W_s\}$ be a set of WGCNA modules. Note that $W_i \subseteq F$ for each i and $\bigcup_{i=1}^{s} W_i \subseteq F$. Also say $|W_i| = m_i$ and require $W_i \cap W_j = \emptyset$ for all $i \neq j$. Let $\mathbb{P}(F)$ denote the power set of F. Consider the mapping $\psi : \mathbb{P}(F) \to \mathbb{P}(F)$. Let

$\psi(\{W_1, W_2, \ldots, W_s\}) = \{R_1, R_2, \ldots, R_{s'}\}$ where $R_i \subset F$ and $R_i \cap R_j = \emptyset$ for all $i \neq j$. We say ψ is a *module refinement method* when $\bigcup_{i=1}^{s} W_i = \bigcup_{i=1}^{s'} R_i$. So module refinement is a mapping between two set partitions. We call $\{R_1, R_2, \ldots, R_{s'}\}$ the refined modules and R_i the ith refined module. Note: the number of modules can change since s' and s are not necessarily equal.

Given the training data for one fold, we use PyWGCNA [116] to compute WGCNA modules. Then we run LBG clustering with each of the four types of centers (eigengene subspace, flag mean, flag median and module expression) initialized with these WGCNA modules. See Figure 5.2 for our data partitioning scheme and Figure 5.1 for a visual of our general workflow. We define a set of refined modules is the modules output from a run of LBG clustering that was initialized with WGCNA. Specifically, the set of features found using this methodology is a refined module.

The centers (e.g., module representatives) in our LBG implementations are the eigengene subspace, flag mean, flag median and module expression. We run LBG clustering with $r = 1, 2, 4, 8$ dimensional eigengene subspaces, flag means, and flag medians. Eigengene subspaces and module expression are only run with $k = 1$ dimensional subspace representations for the features (e.g., genes or probe identifiers). We use $k = 1$ and $k = 2$ dimensional representations for the features for the flag mean and flag median LBG clustering implementations. With all of these various module representatives and their corresponding subspace dimensions, we have a total of 21 methods for module refinement.

For our LBG implementation, we calculate similarity between subspaces by using the square of the cosine of the smallest principal angle between the subspaces. A detailed explanation of principal angles can be found in [2].

Let d_k be the cluster distortion at iteration k for LBG clustering. We terminate LBG clustering when $|d_k - d_{k-1}|/d_{k-1} < 10^{-11}$ or we have reached 20 iterations. If we have reached 20 iterations and $|d_k - d_{k-1}|/d_{k-1} \geq 10^{-11}$ then we consider this a failed clustering and do not report the results of this module refinement.

5.2.5 Module Evaluation

We compare the refined modules with LBG clustering to the modules that were detected using just WGCNA. We will evaluate these modules by 1) comparing their ability to discriminate between phenotype and 2) comparing the gene ontological significance of the modules.

We begin by partitioning our data into 5 stratified folds. Then we detect modules on the training data for each fold. We choose to run this cross validation scheme in order to examine module robustness across folds. We only calculate modules on the training data so we can transfer these features over to the test data to see how well they discriminate between phenotype.

First, we will describe our methods for determining the ability of a set of modules to discriminate between phenotypes. Since we do 5-fold cross validation for our module detection, we have a set of modules found on the training data and a sequestered test data set for each of the 5-folds. So, for each fold and using only features in 1 module, we train and test an SVM (Support Vector Machine) classifier to classify between phenotype on the sequestered test set and record the Balanced Success Rate (BSR.) This give us a BSR for each module found by each module refinement method and WGCNA for each fold.

We now explain how to compare the Gene Ontological (GO) significance between a refined module detection scheme and WGCNA. This methodology mirrors what was done by Botia et al. [111]. For this method, we use the Python package gprofiler to determine the p-value of the overlap between the genes in the module and each gene ontological terms. Let A be the set of gene ontological terms. We denote the p-value for the ith module (set of features) with the gene ontological term $a \in A$ as $p(i,a)$. Now define the piecewise function $\sigma : \mathbb{R} \to \mathbb{R}$ as

$$\sigma(x) = \begin{cases} x & x < 0.05 \\ 0 & x \geq 0.05 \end{cases} \tag{5.9}$$

Then we compute the ontological significance of the ith module as

$$\sum_{a \in A} -\log_{10} \sigma(p(i,a)) \tag{5.10}$$

Suppose we have m_j modules in fold j. Given a module detection method, we call the score (BSR or ontological significance) for the ith module of fold j, $s(i,j) \in \mathbb{R}$. Then the score for this method at fold j is

$$s(j) = \sum_{i=1}^{m_j} s(i,j). \tag{5.11}$$

The we can compare the relative gain $RG(j)$ of the score of a refined module $s_r(j)$ to the score of the original WGNCA module $s_w(j)$ by computing

$$RG(j) = \frac{s_r(j)}{s_w(j) + \epsilon} - 1 \tag{5.12}$$

where $\epsilon = 0.0001$. We do this for each fold to get a 5 different values of $RG(j)$ and estimate the relative gain in score of a given module refinement technique over WGCNA. That is to say, the score of a module refinement technique is better than WGCNA if $RG(j) > 0$ for each $j = 1,2,\ldots,5$.

Now we can calculate the mean relative gain for a given module refinement technique and data set as the mean of the relative gains across all 5 folds for both BSR and GO significance. We use this mean to rank the module refinement techniques by BSR and GO significance.

5.3 Experiments

In this section we introduce the GSE73072 data set [55], and the Salmonella collaborative cross mice data set from Texas A&M [117]. Then we will go on to evaluate our module refinement techniques with respect to their ability to classify between phenotype and their significance to gene ontological terms. For the sake of simplicity, we will refer to both probe identifiers and genes identifiers as "features" for the rest of this section.

For Figure 5.8, Figure 5.12, Figure 5.13, Figure 5.14, and Table 5.2 we denote a module refinement method by it's central prototype along with the dimensions. The method for this identification is {central prototype}_{center dimension}_{data dimension}. For example, flag median with a center dimension of 8 with data dimension of 2 is flag_median_8_2.

5.3.1 Data Sets

The GSE73072 data set [55] is a microarray gene expression data set with 22277 probe identifiers and 148 human subjects. The humans subjects in GSE73072 are infected with various respiratory viruses and are sampled at irregular time intervals from 38 hours before infection to 680 hours after infection. The data are available on the NCBI Gene Expression Omnibus (GEO). We normalize the entire GSE73072 data set with the robust multi-array average method [91]. Our experiments are preformed on 2 studies from the data set that only have subjects with HRV (human rhinovirus). Since this is a microarray data set, we do all our analysis on the probe identifiers. For these data, we classify between *controls* and *shedders*.. Controls are subjects before infection (eg. 38 hours to 0 hours before infection) and shedders are subjects shedding HRV between 48 to 64 hours after infection. This gives us an HRV data set with 138 samples from human subjects with 93 controls and 45 shedders.

The second data set is an RNA-sequencing (RNA-seq) data set of collaborative cross mice. A collaborative cross population of mice are mice that are bred to simulate the genetic diversity of human populations. An initial study with these data was done by Scoggin et al. [117]. All the mice in this data set are infected with Salmonella enterica serotype Typhimurium (STm) for 21 days and RNA-seq samples are taken from the liver and the spleen. The phenotypes in this data set are "delayed susceptible," "tolerant" and "resistant." A tolerant mouse is one that has few signs of infection even with high pathogen loads. The "delayed susceptible" phenotype are mice that lived 7 days in the initial experiments, but died quickly in the second set of experiments (21 days.) For this paper we refer to the "delayed susceptible" phenotype as just "susceptible." There are 26332 gene identifiers in this data set. For our experiments, we restrict ourselves to

working with only the data which are sampled from the liver and only those samples which are labeled *tolerant* and *susceptible*. With this restriction to liver samples and only tolerant and susceptible mice, we are left with a total of 29 samples with 19 labeled tolerant and 10 labeled susceptible.

We use the data partitioning methodology in Figure 5.2 below.

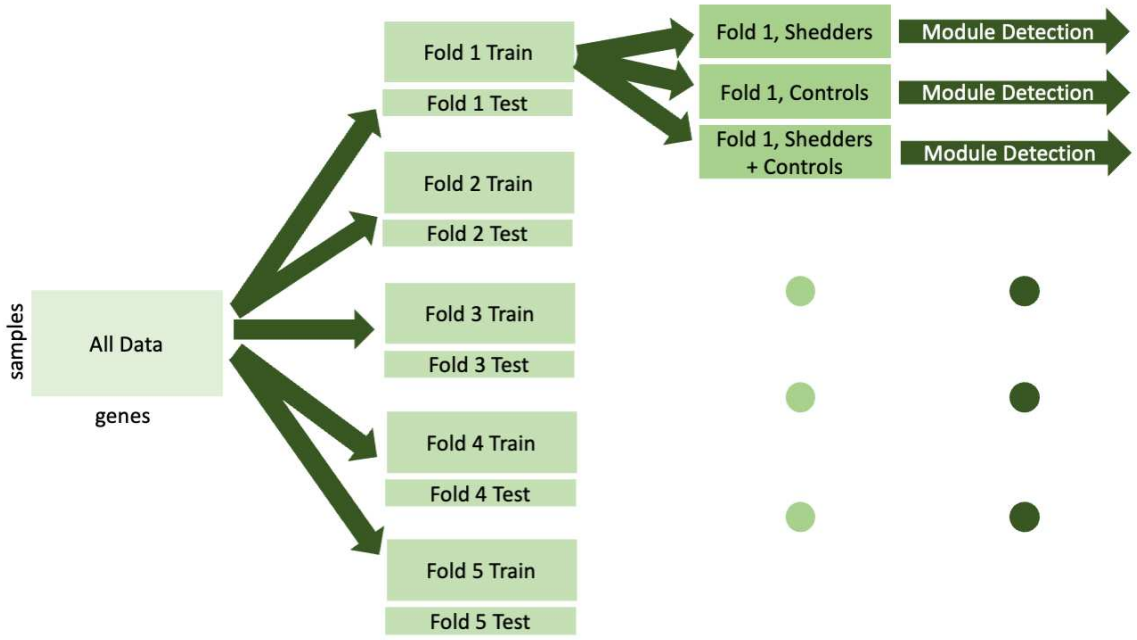


Figure 5.2: The method for partitioning data for module detection. Here we use the GSE73072 labels, but the same scheme is used for the Salmonella data with susceptible and tolerant as labels rather than control and shedder. We use 5-fold cross validation. Then, for each fold, we divide the training data into three sets depending on class. Then we use these data for module generation. The $\cdots$ indicate that we preform the same process for each fold.

The identifiers for these data partitions are below.

Table 5.1: The naming scheme ("Identifier" column) for the data for module generation. "Subject Labels" are the subjects that are used for module generation.

Identifier	Dataset	Subject Labels
gse73072_hrv_48_64	GSE73072	shedding and control
gse73072_hrv_48_64_shedder48_64	GSE73072	shedding
gse73072_hrv_48_64_control	GSE73072	control
salmonella_Liver	Salmonella	susceptible and tolerant
salmonella_Liver_susceptible	Salmonella	susceptible
salmonella_Liver_tolerant	Salmonella	tolerant

A summary of the cross validation methodology in Figure 5.2 is as follows.

1. Separate the data into 5 stratified folds.

2. Break the training data in each fold into three sets. The first set of modules is generated using all the data. The other two data sets are generated using only the data from the subjects in a class (e.g., for GSE73072 data we have shedders or controls).

3. Then we detect modules on the training data from each of the 3-sets for each of the 5-folds.

Using 5 fold cross validation in step (i) allows us to analyze module robustness across folds. Since we are detecting modules for each fold, we are able to test each module set on a sequestered test set. Additionally, a stratified split corrects for the class imbalance. In step (ii), we detect modules using each of the 3 different training data sets for each fold (e.g., control, shedder, control and shedder) to examine the affect of phenotype on module generation.

5.3.2 Module Sizes

Since LBG clustering can change the number of modules, there are some experiments where our module refinement techniques detected only 1 enormous module. Specifically, there are 7

modules across all module refinement methods and folds with more than 20000 features. Since we tested 21 module refinement techniques on 30 partitions of our 2 data sets, we only detect one large module containing all the features in less than 1% of data partitions and module refinement methods. These modules are in Table 5.2 and are not included in Figure 5.3.

Table 5.2: The modules detected by LBG module refinement which resulted in more than 20000 features. The values in the Data and Center columns are the data and center dimensions.

Data Set	Prototype	Data	Center	Folds	Size
salmonella_Liver_susceptible	flag_median	1.0	8.0	All	26332
salmonella_Liver_susceptible	eigengene	1.0	8.0	3	26331
salmonella_Liver_susceptible	flag_mean	1.0	8.0	1	24744

Note: the 8-dimensional subspace prototypes can not be employed with the salmonella_Liver_susceptible given there are only 10 subjects.

In Figure 5.3, we see our module refinement methods increase the variance in module size while reducing the size of extremely large modules. The only data set where module refinement significantly decreases module size is salmonella_Liver_susceptible. Module size is maintained for 2-dimensional subspace data representations for the flag mean and the flag median across all data sets other than salmonella_Liver_susceptible.

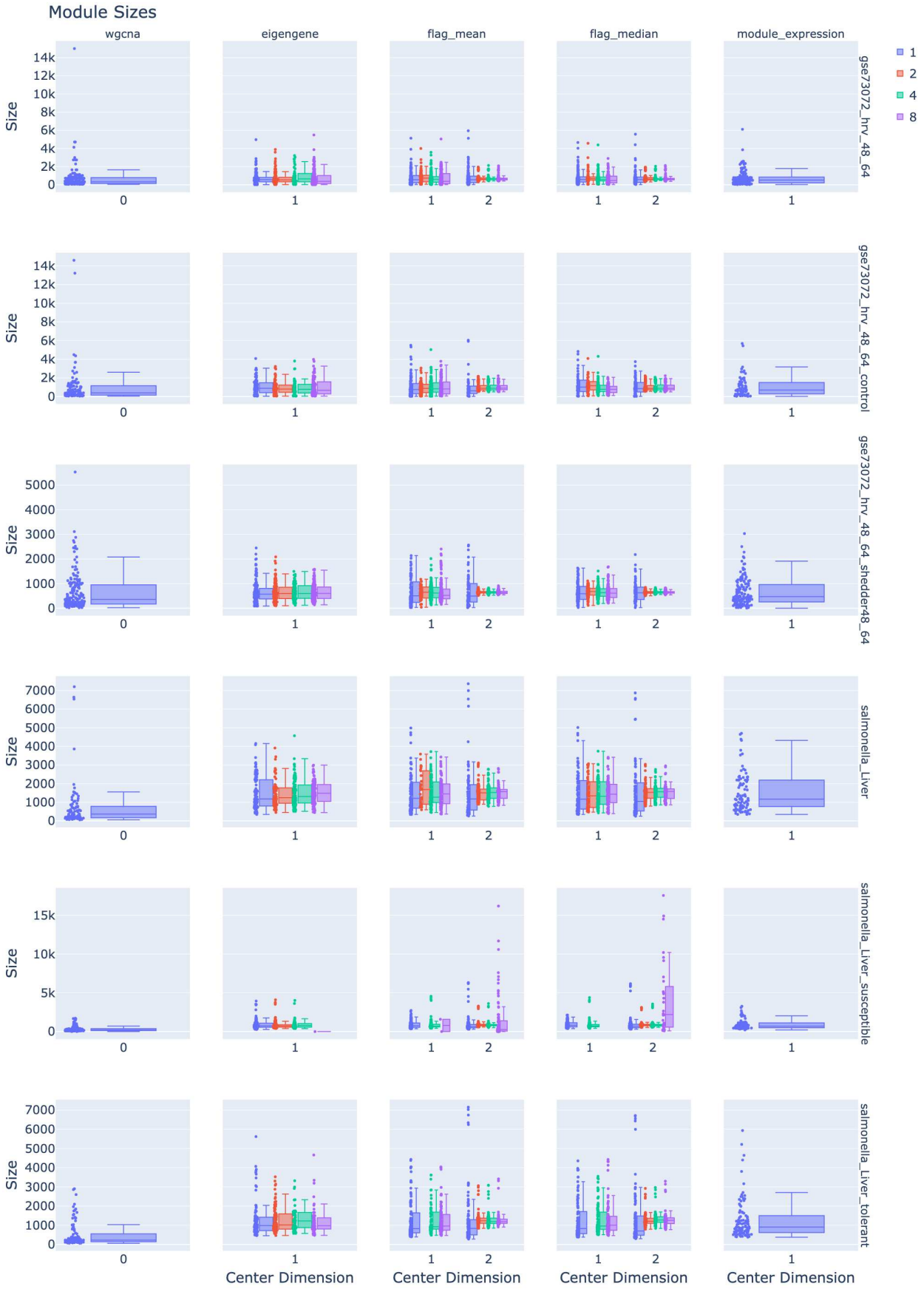


Figure 5.3: Module size as a function of different method. The data subspace dimension for the flag median and flag mean is on the x-axis. Colors correspond to the central prototype dimension (e.g., $1, 2, 4, 8$).

5.3.3 Eigengene Subspace Captures More Variance

We plot the explained variance ratio (EVR) for the modules on the training data in Figure 5.4. As we increase the dimension of the eigengene subspace, we see a logarithmic increase in EVR for modules across all data sets and folds.

For the salmonella_Liver and the salmonella_Liver_tolerant data sets, the interquartile range of the EVR decreases as the subspace dimension increases. The EVR of the modules on the GSE73072 data sets is generally lower than that of the modules for the Salmonella data sets.

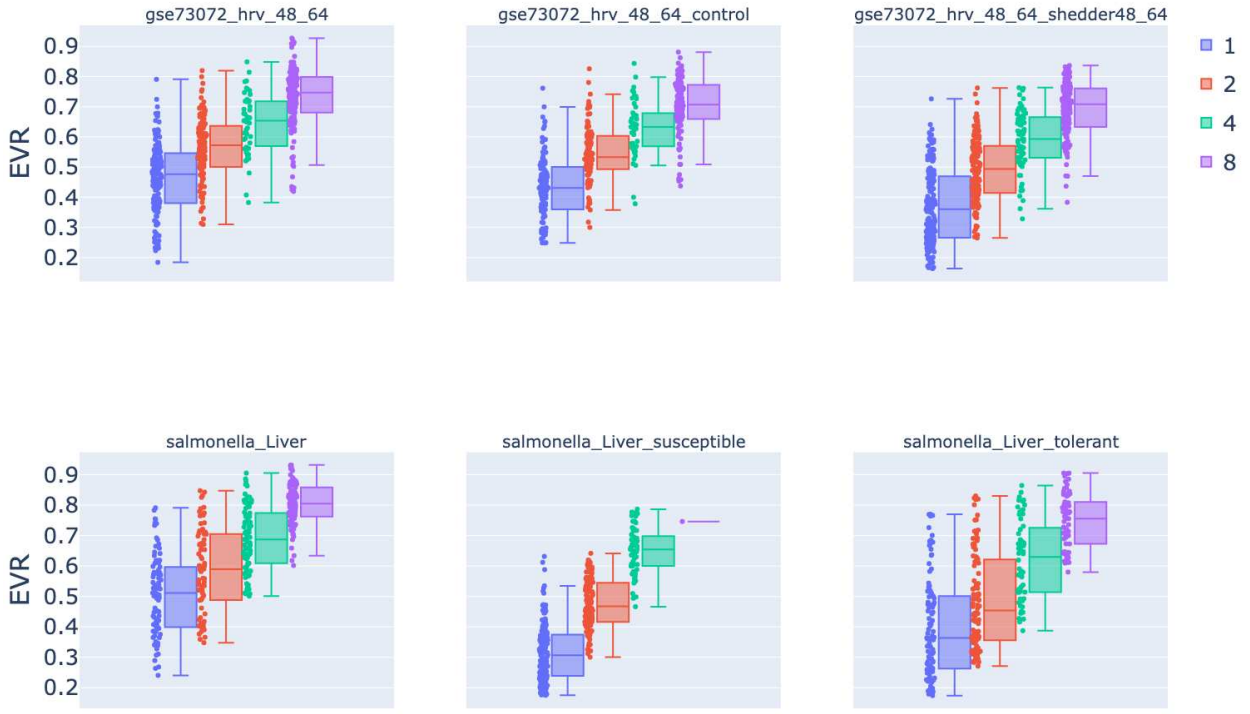


Figure 5.4: EVR colored by eigengene subspace dimension (e.g., 1, 2, 4, 8). The explained variance increases as the dimension of the eigengene subspace increases.

5.3.4 Classification with Modules

We evaluate the modules using the BSR for a Support Vector Machine (SVM) classifier on the test data for each fold restricted to the features in a module. In terms of data normalization,

we subtract by the mean and divide by the variance. We do an additional $\log_2$ normalization for only the GSE73072 data sets.

First we plot the test BSR from each module across data sets for WGCNA in Figure 5.5. The BSR of the modules detected on the different GSE73072 data sets is essentially the same regardless of the data used to generate the modules. The Salmonella data set with all the data has the highest median BSR followed by the modules from the tolerant and the susceptible Salmonella data sets.

Figure 5.5: BSR as a function of dataset over all folds for WGCNA modules.

We plot the relative gain in classification BSR for each method in Figure 5.6. A robust module detection technique will have low variance in relative gain across folds.

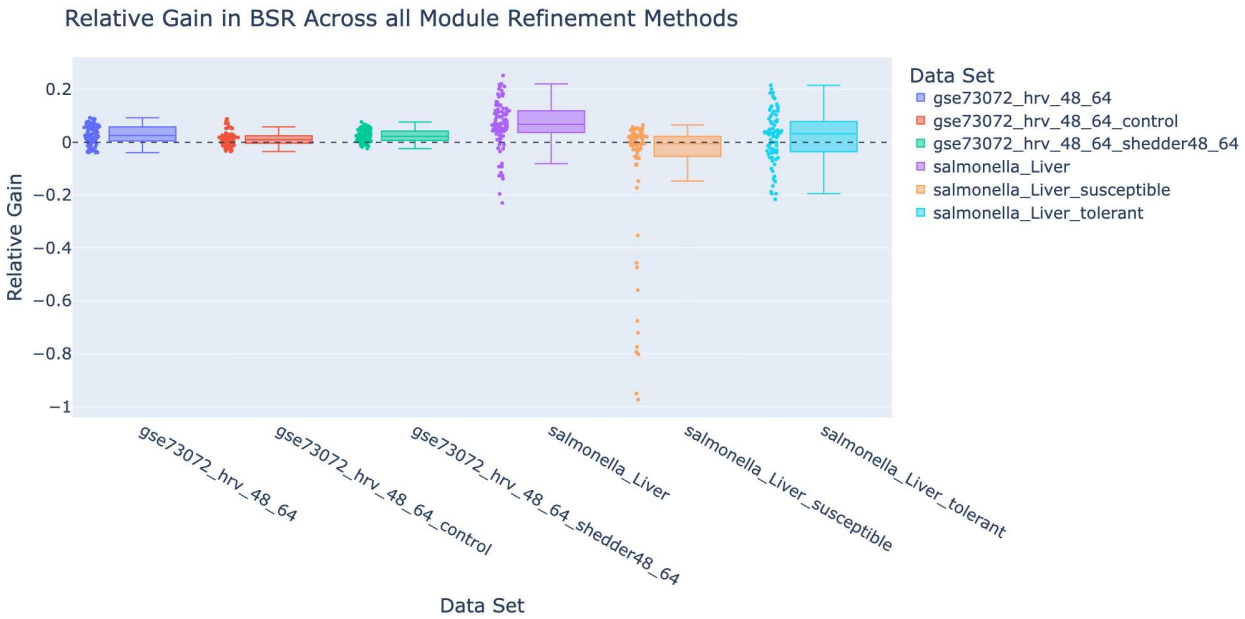


Figure 5.6: Relative gain in BSR as a function of dataset over all module refinement methods and folds.

Figure 5.6 is used to determine which dataset to use for a closer analysis of the different module refinement methods. Most methods have a small variance in their relative gain for each dataset. However, the salmonella_Liver_susceptible dataset has a high variability in relative gain across module refinement methods and folds. All the GSE73072 data sets and the salmonella_Liver_susceptible data set have a small negative median relative gain in BSR.

We investigate the relative gain of the module refinement methods on the data sets which produce the highest median relative gain in Figure 5.7. For the GSE73072 data set, the only methods which result in a positive relative gain in BSR are the flag mean and median with 2-dimensional subspace data representations and center subspaces with dimension higher than 1. Only the 4-dimensional flag median with a data dimension of 2 results in a positive relative gain for each fold.

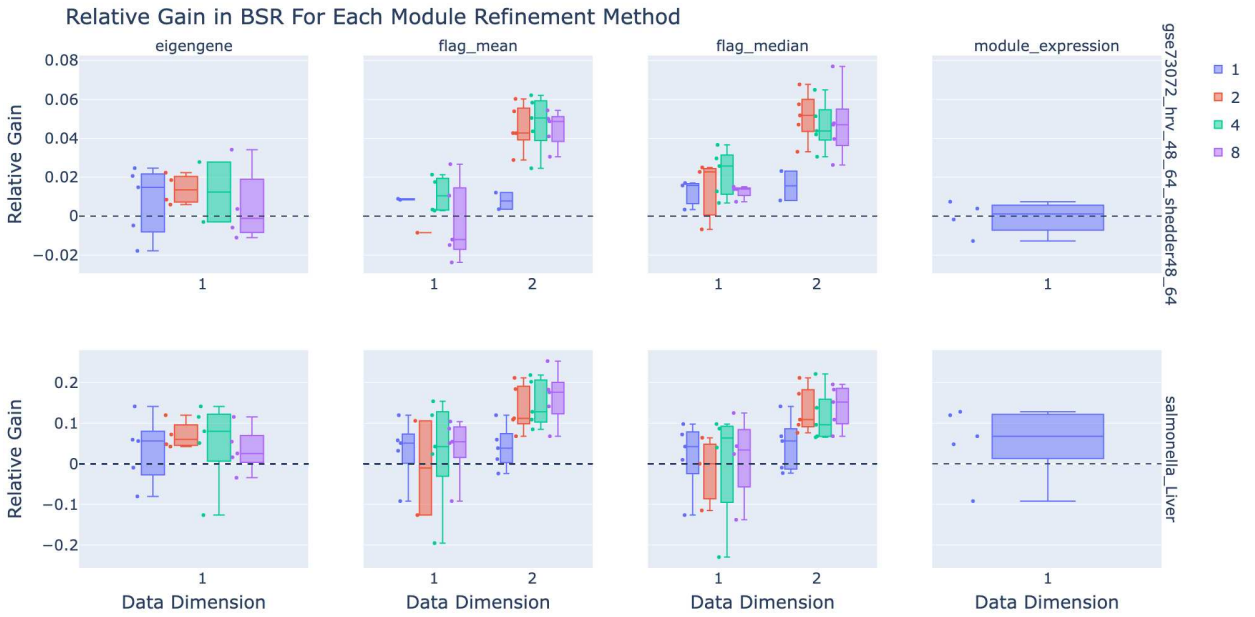


Figure 5.7: Relative gain in BSR as a function of module refinement parameters. The colors correspond to the central prototype subspace dimension.

The mean relative gain in BSR for each method on each data set is in Figure 5.8. In general, 2 dimensional subspace data representations result in module refinement with the highest mean relative gain in BSR across data sets.

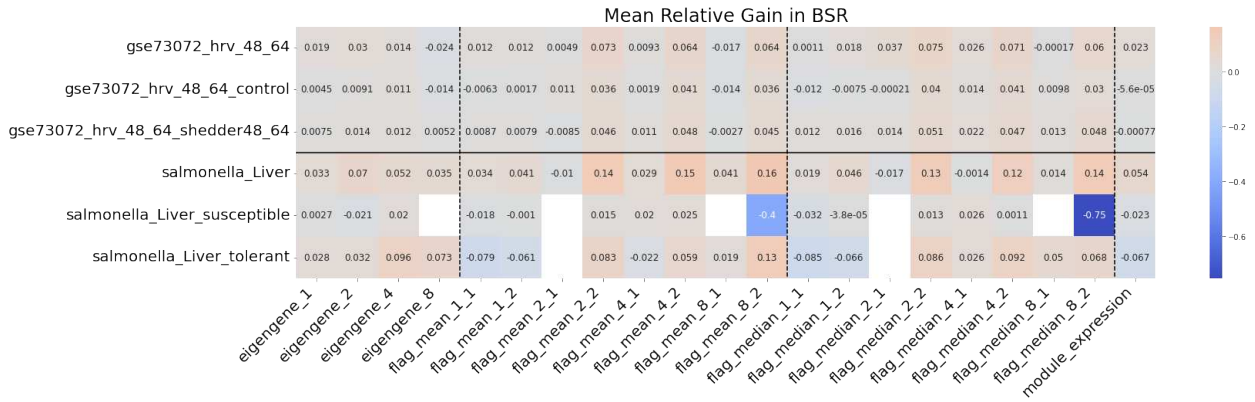


Figure 5.8: Mean relative gain in BSR across all folds for each module refinement method. The missing entries are those where the WGCNA BSR is 0 or are methods that appear in Table 5.2.

5.3.5 Gene Ontological Significance

Before we consider the relative gains in Gene Ontological (GO) significance we will consider the GO Significance of the WGCNA modules in Figure 5.9.

Figure 5.9: GO significance as a function of dataset for WGCNA modules over all folds. We remove the one module that has a GO significance higher than 10000 that was detected on the salmonella_Liver_susceptible dataset.

Now we plot the relative gain in GO significance for each method in Figure 5.10. There are at most 5 points for each method which correspond to the relative gain over each fold. A module refinement method improves classification GO significance over WGCNA when it has a positive relative gain and is robust when it has a low variance in relative gain across folds.

In Figure 5.10 we see that there is little difference between relative gain in GO significance across the GSE73072 data sets; but the highest median gain is with the modules generated using data from both shedders and controls. On the other hand, there is some variance in the relative gain in GO significance across different Salmonella data sets. There are some extremely high outlier relative gains with the modules detected with only the susceptible subjects in the Salmonella data set. These likely due to the large modules listed in Table 5.2.

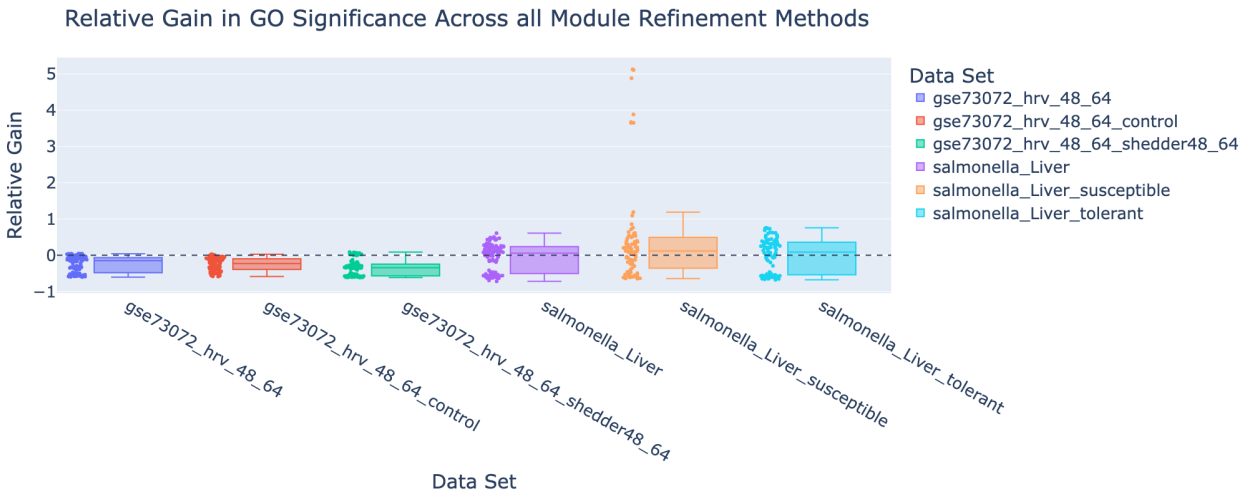


Figure 5.10: Relative gain in GO significance as a function of dataset over all module refinement methods and folds.

We use Figure 5.10 to find the data sets which are used to find modules with the highest median relative gains. Then we plot the relative gain of the refined modules from these data sets in Figure 5.11.

For the eigengene, we see an increase in relative gain by using higher dimensional eigengene subspaces with the GSE73072 data set and a decrease in relative gain for higher dimensional eigengene subspaces for the Salmonella data set. For 1 dimensional data representations with the flag mean and flag median we see a decrease in GO significance. Only module expression, 1-dimensional flag mean and flag median with 1-dimensional data representations result in a positive relative gain in GO significance on the GSE73072 data set. All of the 1-dimensional subspace prototypes and the module expression vector result in much higher relative gains in GO significance on the Salmonella data set.

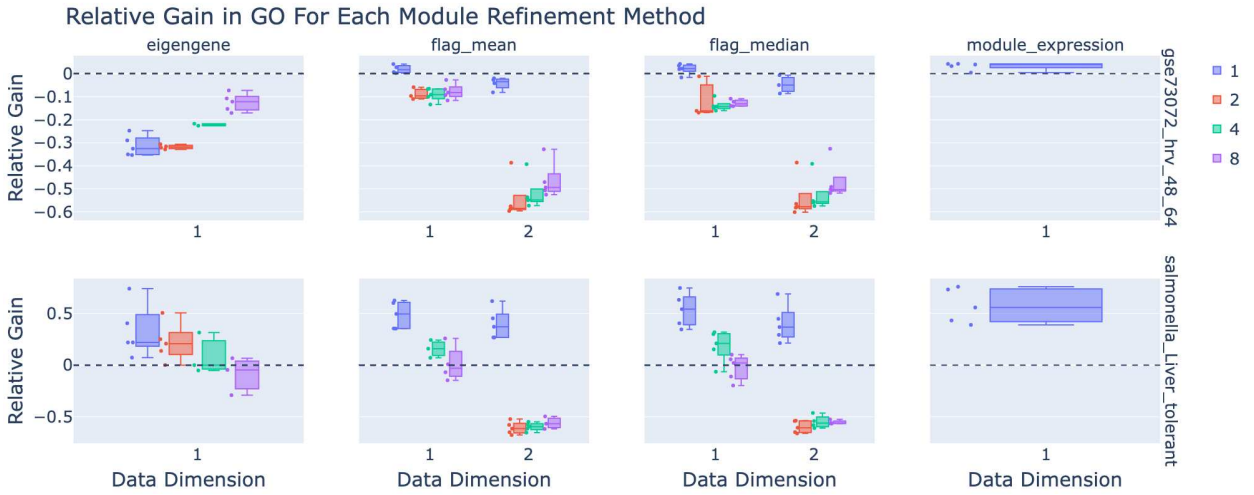


Figure 5.11: The relative gain of each module refinement method across all 5 folds. A positive value indicates higher GO significance than WGCNA. The colors correspond to the central prototype subspace dimension.

We summarize the relative gains in GO significance for each data set and central prototype in Figure 5.12. The values of the pixels in this heat map correspond to the mean relative gain in GO significance across folds. Overall, the module refinement techniques that represent the data with 2 dimensional subspaces result in negative mean relative gains in GO significance. In contrast, 1-dimensional module representatives generally result in positive mean relative gain in GO significance.

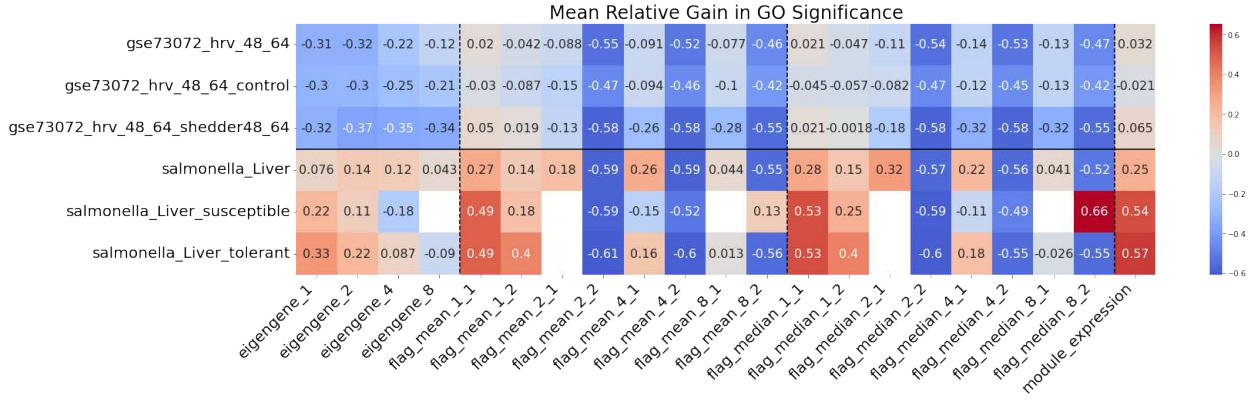


Figure 5.12: Mean relative gain in GO Significance across all folds for each module refinement method. The missing entries are those where the WGCNA GO significance is 0 or are methods that appear in Table 5.2.

5.3.6 Results Summary

We rank these module refinement techniques across all data sets using the mean relative gain. The top 3 methods with the highest relative gains for GSE73072 and Salmonella are in Figure 5.13 and Figure 5.14. Notice that flag_median_1_1 and module_expression appear in the top 3 for both GO mean relative gain for both data sets. This suggests that this is one of the better module refinement techniques for GO significance. For BSR, we see that 2-dimensional subspace data representations result in the best relative gains.

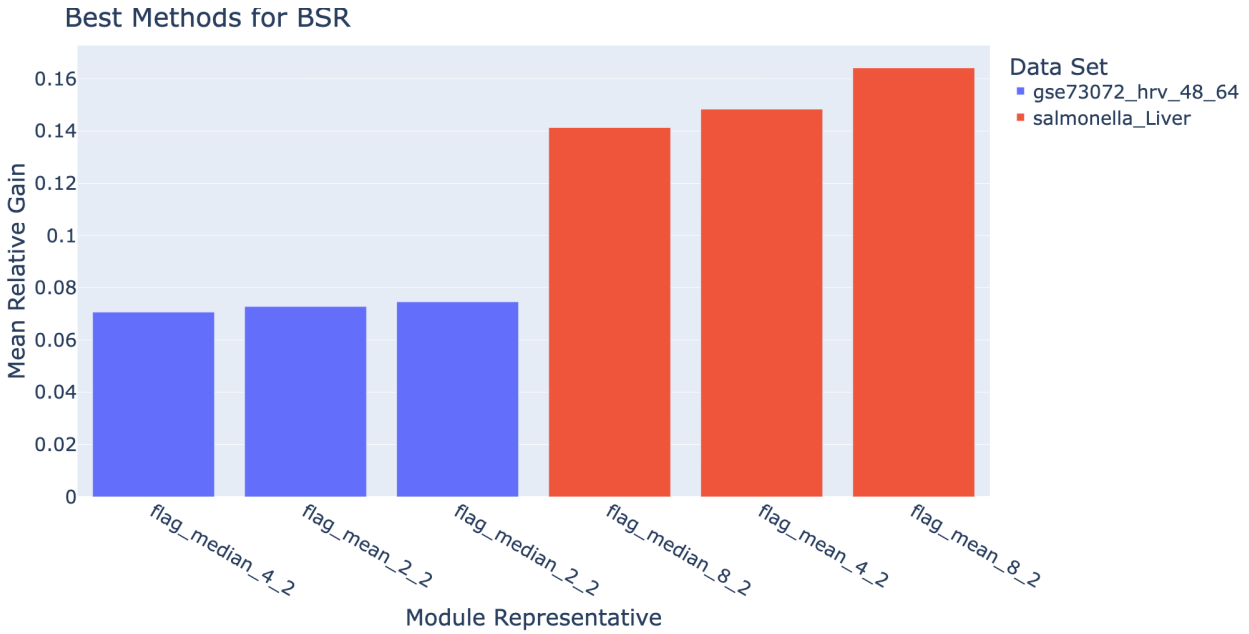


Figure 5.13: The module representatives which produced refined modules with the top mean relative gains in BSR over WGCNA modules. The modules in Table 5.2 are omitted.

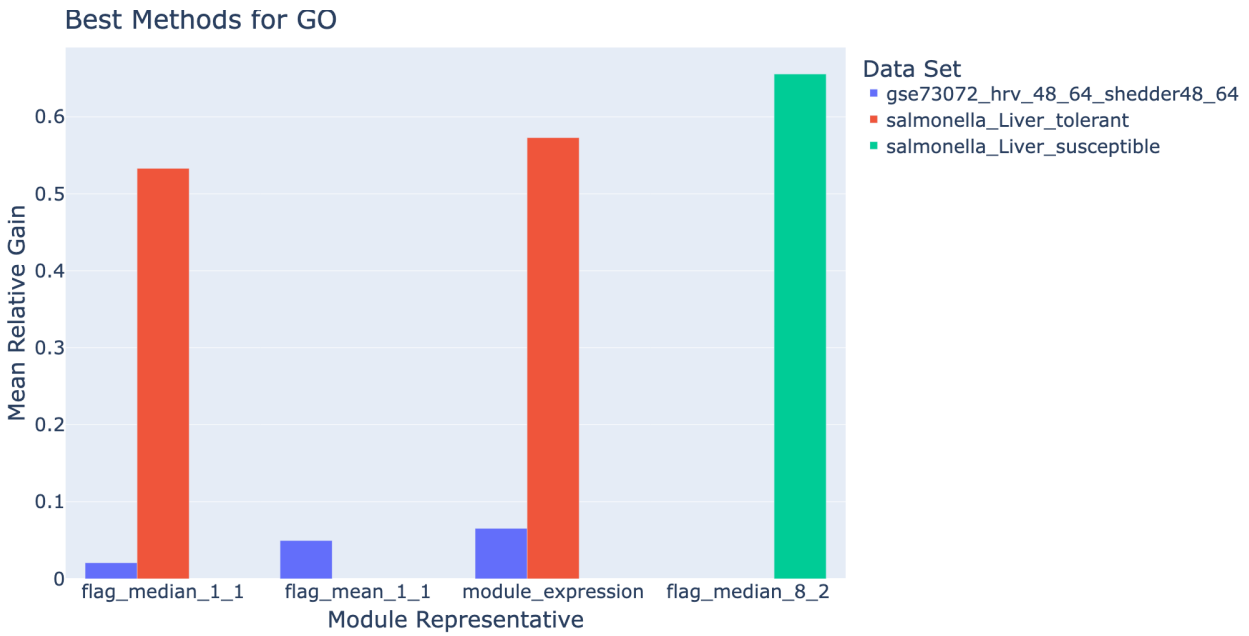


Figure 5.14: The module representatives which produced refined modules with the top mean relative gains in GO significance over WGCNA modules. The modules in Table 5.2 are omitted.

Certain module representatives result in modules with increased average GO significance and classification BSR between for both the GSE73072 and Salmonella data sets. For example, module expression module representation with LBG clustering on the Salmonella data set with all subjects results in modules with an average 5% relative gain in BSR and an average 25% relative gain in GO significance over WGNCA. Figure 5.15 highlights all such methods which produce positive mean relative gains in classification BSR and GO significance.

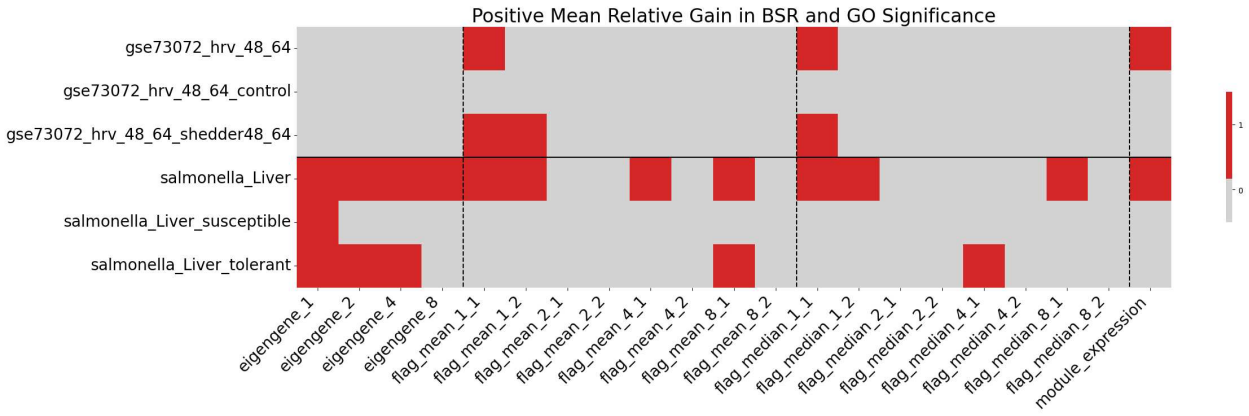


Figure 5.15: The module representatives which produced refined modules with positive mean relative gains in classification BSR and GO significance over WGCNA modules are colored red (with a value of 1).

5.4 Discussion

In this work we have introduced multiple new methods for gene co-expression module representation and used these module representatives to refine WGCNA gene co-expression modules. We evaluated these refined gene co-expression modules by their relative gain over WGCNA modules in BSR of a SVM classifier on a sequestered test set and GO significance according to a statistical test. Our examples with the GSE73072 and Salmonella mice data set provided insight into the optimal data set for calculating modules along with the ideal module representation technique. Given the proper data set and module representative, we were able to increase classification rates *and* gene ontological significance over WGCNA modules, even when averaged across 5 folds of modules.

We do recognize the limitations of 5-fold cross validation with these data. For example, we have different samples of the same subjects in training and test. This is acceptable since we are not seeking state of the art accuracy. Instead, we are using this metric as a statistic for module viability. We leave such a search for state of the art classification rates using features from refined modules to future work.

Our choice of data set for module detection had a small effect on the relative gains of the refined modules in terms of classification BSR and GO significance. We found that the refined

using both subjects in either the GSE73072 or Salmonella data sets produced the highest classification BSRs. On the other hand, using only the tolerant subjects for the Salmonella liver data set produced the highest mean relative gains in GO significance. So perhaps, we found more improvement in GO signal by detecting refined modules using only data from tolerant mice.

Now we turn to discussing the best module refinement techniques in our experiments. The best module refinement method had an increase relative gain in BSR and GO significance across both data sets. This means, the modules from this method needed to capture enough variance in signal to discriminate between phenotype while still maintaining enough of a uniform biological signal to result in high GO significance. These modules were colored red in Figure 5.15. The modules computed on data sets with both phenotypes of subjects had positive relative gains in both classification BSR and GO significance (e.g., gse73072_hrv_48_64). The module representatives for these ideal module refinement methods were the 1 dimensional flag mean and 1 dimensional flag median with 1 dimensional subspace data representations and the module expression vector.

For both the Salmonella and GSE73072 data sets, the methods which had the highest relative gains in BSR did not also have the highest gains in GO significance. Specifically, in Figure 5.13 and Figure 5.14, there was no central prototype in the top 3 relative gains for both BSR and GO significance. Specifically, 1-dimensional subspace flag mean and medians with 1-dimensional subspace data representations and module expression were the best LBG cluster refinement methods for increasing GO significance. On the other hand, 2, 4 and 8 dimensional subspace prototypes all with 2-dimensional data representations produced the highest relative gains in BSR. In general we understand that lower dimensional subspace prototypes and module expression result in modules with uniform biological signal; whereas higher dimensional subspace prototypes result in modules with higher variance signal which improves classification rates.

We also notice that the 2-dimensional flag mean and median along with the 8-dimensional flag median resulted in LBG which did not converge in 20 iterations. Due to the scope of our

study, we chose to not include these experiments. Future work could increase the number of iterations of LBG clustering for these methods to hopefully result in eventual convergence. Even better, one could work on proving convergence results for LBG clustering with subspace prototypes (e.g., flag mean and flag median.) Then, using this criteria, one could determine the optimal central prototype and subspace dimension for module refinement using LBG clustering.

There are numerous directions for potential future work with these new module representatives. For example, one could compute an initial clustering with other algorithms for co-expression modules like FLAME [104]. This could lead to a survey summarizing prominent methods for detecting and refining gene co-expression modules.

Chapter 6

Conclusion

In this dissertation we studied central prototypes of clusters of subspace data and network data. Some of our novel prototypes were used to improve clustering algorithms and others were leveraged for feature extraction and dimensionality reduction. Our applications included both computer vision data sets like MNIST and UFC YouTube Action and biological data sets like GSE73072. Chapter 1 had some background for working with the Grassmannian and frame theory. Then, in Chapter 2, we provided a survey of central prototypes of subspaces (e.g., Grassmannian) which resulted in phrasing two new subspace prototypes: the flag median and the maximally correlated flag. Next, we derived and analyzed the FlagIRLS algorithm in Chapter 3. We found that it quickly converges to the flag median in our experiments. This section also included a few computer vision applications of the flag median and FlagIRLS. After the subspace prototypes and FlagIRLS discussion, we pivoted to looking at central prototypes for features (e.g., genes) in biological data sets. Chapter 4 provided a framework: pathway expression, which produced a low-dimensional, biologically informed, feature representation of gene expression data that increased phenotype classification rates. Finally, in Chapter 5, we combined the subspace prototypes and pathway expression theory in a module refinement task with biological data sets. Using these refined prototypes, we were able to find modules (clusters of genes) which increased phenotype classification rates and biological significance over the genes from standard (WGCNA) modules. Each of these chapters filled in gaps in previous work with subspace averaging, biological pathway representation and module detection.

Although there have been works summarizing the utility of different subspace prototypes for clustering in computer vision, there has not been a work which unifies their theoretical underpinnings while simultaneously analyzing their properties. In Chapter 2 we used optimization problems involving the principal angle matrix to formulate central subspace prototype optimization problems as simply matrix norms. Then we used this formulation to realize two new

subspace prototypes: the flag median and the maximally correlated flag. Frame theory is used to investigate properties of the flag median, flag mean and ℓ_2-median for even configurations of subspaces. Finally, small synthetic examples allowed us to study the properties of the prototypes with even subspace configurations and find that the flag median is the more robust to outliers than other central subspace prototypes.

Weiszfeld algorithms have been adapted to Riemannian manifolds (e.g., the Grassmannian), but they are implemented using computationally expensive Riemannian manifold log and exp maps. In Chapter 3 we introduced the novel FlagIRLS algorithm which shortcuts log and exp maps by using iteratively re-weighted singular value decompositions. We proved that the objective function values of iterations of FlagIRLS (for the flag median) and DPCP-IRLS both converge given some assumptions on the subspace configurations. Through experimentation, we found that FlagIRLS converges in fewer iterations than other subspace prototype algorithms. Finally, FlagIRLS is used to compute the flag median as a part of LBG clustering to improve cluster purities over other standard subspace prototypes in two action recognition data sets.

Now we pivot to pathway expression. We introduced this method in Chapter 4. Our implementations of pathway expression averaged expression levels of genes in a biological pathway network (network of genes). Numerous methods already existed which represent biological pathways using averages and other statistical methods. Pathway expression offered a simple transformation-based approach which translated gene expression levels into pathway expression levels. Our implementations of pathway expression utilized averages and weighted averages of gene expression levels using known biological networks of genes within a pathway. This method, centrality pathway expression (CPE), resulted in a reduction of gene expression data from the gene expression space to the pathway expression space while increasing phenotype classification rates on human respiratory virus data.

In Chapter 5 we also utilized averages to analyze biological datasets, specifically gene expression (a.k.a. transcriptomics) data. We used the averages developed throughout this paper: the flag median and module expression (a form of pathway expression) along with the previ-

ously defined flag mean and a generalization of the eigengene as module representatives to improve WGNCNA modules (clusters of genes). We took WGCNA modules and used subspace LBG clustering with the four module representatives to change module membership. In our experiments we found that LBG subspace clustering with our 4 module representatives improved the biological significance over WGNCA.

Future work on these projects includes finding all possible subspace averages that can be realized as solutions to matrix norms of the principal angle matrix and deriving algorithms for calculating each these subspace averages. Then one could find where these optimization problems are convex and analyze the convergence of these algorithms. Additionally, one can run more pathway expression experiments with novel notions of the pathway expression mapping (e.g., non-linear maps) and/or run experiments on more datasets with improved feature selection algorithms. Finally, running our module refinement pipeline on larger datasets may result in greater improvements in biological significance of the refined modules.

Bibliography

[1] 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.

[2] Ake Bjorck and Gene H Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973.

[3] 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.

[4] John H Conway, Ronald H Hardin, and Neil JA Sloane. Packing lines, planes, etc.: Packings in Grassmannian spaces. *Experimental Mathematics*, 5(2):139–159, 1996.

[5] Emily King. Lecture notes in frame theory, December 2020.

[6] Peter G Casazza, Matthew Fickus, Dustin G Mixon, Yang Wang, and Zhengfang Zhou. Constructing tight fusion frames. *Applied and Computational Harmonic Analysis*, 30(2):175–187, 2011.

[7] Matthew Fickus, John Jasper, Emily J King, and Dustin G Mixon. Equiangular tight frames that contain regular simplices. *Linear Algebra and its Applications*, 555:98–138, 2018.

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

[9] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[10] 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.

[11] Paul Horst. Relations among sets of measures. *Psychometrika*, 26(2):129–149, 1961.

[12] Moody T Chu and J Loren Watterson. On a multivariate eigenvalue problem, part i: Algebraic theory and a power method. *SIAM Journal on Scientific Computing*, 14(5):1089–1106, 1993.

[13] Lei-Hong Zhang and Moody T Chu. On a multivariate eigenvalue problem: II. global solutions and the Gauss-Seidel method. *Submitted, available at http://www4. ncsu. edu/mtchu/Research/Papers/Readme. html*, 2009.

[14] Lei-Hong Zhang. Riemannian Newton method for the multivariate eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2972–2996, 2010.

[15] Lei-Hong Zhang and Li-Zhi Liao. An alternating variable method for the maximal correlation problem. *Journal of Global Optimization*, 54(1):199–218, 2012.

[16] Lei-Hong Zhang and Moody T Chu. Computing absolute maximum correlation. *IMA Journal of Numerical Analysis*, 32(1):163–184, 2012.

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

[18] 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.

[19] 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.

[20] Manolis C Tsakiris and René Vidal. Dual principal component pursuit. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–18, 2015.

[21] Manolis C Tsakiris and René Vidal. Hyperplane clustering via dual principal component pursuit. In *International conference on machine learning*, pages 3472–3481. PMLR, 2017.

[22] Yuchen Xie, Baba C Vemuri, and Jeffrey Ho. Dictionary learning on Riemannian manifolds. In *MICCAI workshop on STMI*, volume 1800, page 1800, 2012.

[23] Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel methods on Riemannian manifolds with gaussian RBF kernels. *IEEE PAMI*, 37(12):2464–2477, 2015.

[24] Anoop Cherian, Suvrit Sra, Stephen Gould, and Richard Hartley. Non-linear temporal subspace representations for activity recognition. In *Proceedings of the IEEE CVPR*, pages 2197–2206, 2018.

[25] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds. *NIPS*, 29:4592–4600, 2016.

[26] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[27] Khurrum Aftab, Richard Hartley, and Jochen Trumpf. Generalized Weiszfeld algorithms for Lq optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(4):728–745, 2014.

[28] Tim Marrinan, J Ross Beveridge, Bruce Draper, 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, 2014.

[29] Sofya Chepushtanova and Michael Kirby. Classification of hyperspectral imagery on embedded Grassmannians. *arXiv preprint arXiv:1502.00946*, 2015.

[30] Jiayao Zhang, Guangxu Zhu, Robert W Heath Jr, and Kaibin Huang. Grassmannian learning: Embedding geometry awareness in shallow and deep learning. *arXiv preprint arXiv:1808.02229*, 2018.

[31] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977.

[32] Ziv Yavo, Joseph M Francos, Ignacio Santamaria, and Louis L Scharf. Estimating the mean manifold of a deformable object from noisy observations. In *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5. IEEE, 2016.

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

[34] Thomas R Knapp. Canonical correlation analysis: A general parametric significance-testing system. *Psychological Bulletin*, 85(2):410, 1978.

[35] David Monk. The geometry of flag manifolds. *Proceedings of the London Mathematical Society*, 3(2):253–286, 1959.

[36] Ignacio Santamaría, Louis L Scharf, Chris Peterson, Michael Kirby, and J Francos. An order fitting rule for optimal subspace averaging. In *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pages 1–4. IEEE, 2016.

[37] Ignacio Santamaria, Javier Vía, Michael Kirby, Tim Marrinan, Chris Peterson, and Louis Scharf. Constrained subspace estimation via convex optimization. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1200–1204. IEEE, 2017.

[38] 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.

[39] Yehuda Vardi and Cun-Hui Zhang. The multivariate L1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4):1423–1426, 2000.

[40] Jon R Kettenring. Canonical analysis of several sets of variables. *Biometrika*, 58(3):433–451, 1971.

[41] Endre Weiszfeld. Sur le point pour lequel la somme des sistances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

[42] Richard Hartley, Khurrum Aftab, and Jochen Trumpf. L1 rotation averaging using the Weiszfeld algorithm. In *CVPR 2011*, pages 3041–3048. IEEE, 2011.

[43] Bijan Afsari. Riemannian $\mathscr{L}^p$ center of mass: existence, uniqueness, and convexity. *Proceedings of the American Mathematical Society*, 139(2):655–673, 2011.

[44] Le Yang. Riemannian median and its estimation. *LMS Journal of Computation and Mathematics*, 13:461–479, 2010.

[45] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted $\ell - 1$ minimization. volume 14, pages 877–905. Springer, 2008.

[46] Rick Chartrand and Wotao Yin. Iteratively reweighted algorithms for compressive sensing. In *2008 IEEE international conference on acoustics, speech and signal processing*, pages 3869–3872. IEEE, 2008.

[47] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 63(1):1–38, 2010.

[48] Gilad Lerman, Michael B McCoy, Joel A Tropp, and Teng Zhang. Robust computation of linear models by convex relaxation. *Foundations of Computational Mathematics*, 15(2):363–410, 2015.

[49] Gilad Lerman and Tyler Maunu. Fast, robust and non-convex subspace recovery. *Information and Inference: A Journal of the IMA*, 7(2):277–336, 2018.

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

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

[52] Joseph B Kruskal. *Multidimensional scaling*. Number 11. Sage, 1978.

[53] Shannon Stiverson, Michael Kirby, and Chris Peterson. Subspace quantization on the Grassmannian. In *International Workshop on Self-Organizing Maps*, pages 251–260. Springer, 2019.

[54] Julien Riou and Christian L Althaus. Pattern of Early Human-to-Human Transmission of Wuhan 2019 Novel Coronavirus (2019-nCoV), December 2019 to January 2020. *Eurosurveillance*, 25(4):2000058, 2020.

[55] TY Liu, T Burke, L Park, C W, A Zaas, G Ginsburg, and A Hero. An Individualized Predictor of Health and Disease Using Paired Reference and Target Samples. *BMC bioinformatics*, 17(1):1–15, 2016.

[56] M Aminian, T Ghosh, A Peterson, AL Rasmussen, S Stiverson, K Sharma, and M Kirby. Early Prognosis of Respiratory Virus Shedding in Humans. *Scientific Reports*, 11(1):1–15, 2021.

[57] Mmanu Chaturvedi, Tomojit Ghosh, Michael Kirby, Xiaoyu Liu, Xiaofeng Ma, and Shannon Stiverson. Explorations in Very Early Prognosis of the Human Immune Response to Influenza, 2016.

[58] Nathan Mankovich. Methods for Network Generation and Spectral Feature Selection: Especially on Gene Expression Data. Master's thesis, Colorado State University, 2019.

[59] Stephen O'Hara, Kun Wang, Richard A Slayden, Alan R Schenkel, Greg Huber, Corey S O'Hern, Mark D Shattuck, and Michael Kirby. Iterative Feature Removal Yields Highly Discriminative Pathways. *BMC Genomics*, 14(1):1–15, 2013.

[60] Purvesh Khatri, Marina Sirota, and Atul J Butte. Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges. *PLoS computational biology*, 8(2):e1002375, 2012.

[61] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. The STRING Database in 2021: Customizable Protein–Protein Networks, and Functional Characterization of User-Uploaded Gene/Measurement Sets. *Nucleic acids research*, 49(D1):D605–D612, 2021.

[62] Marc Gillespie, Bijay Jassal, Ralf Stephan, Marija Milacic, Karen Rothfels, Andrea Senff-Ribeiro, Johannes Griss, Cristoffer Sevilla, Lisa Matthews, Chuqiao Gong, Chuan Deng, Thawfeek Varusai, Eliot Ragueneau, Yusra Haider, Bruce May, Veronica Shamovsky, Joel Weiser, Timothy Brunson, Nasim Sanati, Liam Beckman, Xiang Shao, Antonio Fabregat, Konstantinos Sidiropoulos, Julieth Murillo, Guilherme Viteri, Justin Cook, Solomon Shorser, Gary Bader, Emek Demir, Chris Sander, Robin Haw, Guanming Wu, Lincoln Stein, Henning Hermjakob, and Peter D'Eustachio. The Reactome Pathway Knowledgebase 2022. *Nucleic Acids Research*, 50(D1):D687–D692, 11 2021.

[63] Guangchuang Yu and Qing-Yu He. ReactomePA: an R/Bioconductor Package for Reactome Pathway Analysis and Visualization. *Mol. BioSyst.*, 12:477–479, 2016.

[64] Aravind Subramanian, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, Scott L Pomeroy, Todd R Golub, Eric S Lan-

der, et al. Gene Set Enrichment Analysis: A Knowledge-Based Approach for Interpreting Genome-Wide Expression Profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

[65] Maxim V Kuleshov, Matthew R Jones, Andrew D Rouillard, Nicolas F Fernandez, Qiaonan Duan, Zichen Wang, Simon Koplev, Sherry L Jenkins, Kathleen M Jagodnik, Alexander Lachmann, et al. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic acids research*, 44(W1):W90–W97, 2016.

[66] Jing Wang, Suhas Vasaikar, Zhiao Shi, Michael Greer, and Bing Zhang. WebGestalt 2017: A more comprehensive, powerful, flexible and interactive gene set enrichment analysis toolkit. *Nucleic acids research*, 45(W1):W130–W137, 2017.

[67] Enrico Glaab, Anaïs Baudot, Natalio Krasnogor, Reinhard Schneider, and Alfonso Valencia. EnrichNet: Network-based gene set enrichment analysis. *Bioinformatics*, 28(18):i451–i457, 2012.

[68] Z Gu, J Liu, K Cao, J Zhang, and J Wang. Centrality-Based Pathway Enrichment: a Systematic Approach for Finding Significant Pathways Dominated by Key Genes. *BMC Systems Biology*, 6(1):1–13, 2012.

[69] Rosalia Maglietta, Ada Piepoli, Domenico Catalano, F Licciulli, Massimo Carella, Sabino Liuni, Graziano Pesole, Francesco Perri, and Nicola Ancona. Statistical assessment of functional categories of genes deregulated in pathological conditions by using microarray data. *Bioinformatics*, 23(16):2063–2072, 2007.

[70] Gene Ontology Consortium. The gene ontology (GO) database and informatics resource. *Nucleic acids research*, 32(suppl_1):D258–D261, 2004.

[71] Minoru Kanehisa and Susumu Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[72] Ron Caspi, Tomer Altman, Kate Dreher, Carol A Fulcher, Pallavi Subhraveti, Ingrid M Keseler, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Lukas A Mueller, et al. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic acids research*, 40(D1):D742–D753, 2012.

[73] Alexander R Pico, Thomas Kelder, Martijn P Van Iersel, Kristina Hanspers, Bruce R Conklin, and Chris Evelo. WikiPathways: pathway editing for the people. *PLoS biology*, 6(7):e184, 2008.

[74] Darryl Nishimura. BioCarta. *Biotech Software & Internet Report: The Computer Software Journal for Scient*, 2(3):117–120, 2001.

[75] David J Lynn, Geoffrey L Winsor, Calvin Chan, Nicolas Richard, Matthew R Laird, Aaron Barsky, Jennifer L Gardy, Fiona M Roche, Timothy HW Chan, Naisha Shah, et al. InnateDB: facilitating systems-level analyses of the mammalian innate immune response. *Molecular systems biology*, 4(1):218, 2008.

[76] Zheng Guo, Tianwen Zhang, Xia Li, Qi Wang, Jianzhen Xu, Hui Yu, Jing Zhu, Haiyun Wang, Chenguang Wang, Eric J Topol, et al. Towards precise classification of cancers based on robust gene functional expression profiles. *BMC bioinformatics*, 6(1):1–12, 2005.

[77] Andrea H Bild, Guang Yao, Jeffrey T Chang, Quanli Wang, Anil Potti, Dawn Chasse, Mary-Beth Joshi, David Harpole, Johnathan M Lancaster, Andrew Berchuck, et al. Oncogenic pathway signatures in human cancers as a guide to targeted therapies. *Nature*, 439(7074):353–357, 2006.

[78] Eunjung Lee, Han-Yu Chuang, Jong-Won Kim, Trey Ideker, and Doheon Lee. Inferring pathway activity toward precise disease classification. *PLoS computational biology*, 4(11):e1000217, 2008.

[79] Junjie Su, Byung-Jun Yoon, and Edward R Dougherty. Accurate and reliable cancer classification based on probabilistic inference of pathway activity. *PloS one*, 4(12):e8161, 2009.

[80] Xinan Yang, Kelly Regan, Yong Huang, Qingbei Zhang, Jianrong Li, Tanguy Y Seiwert, Ezra EW Cohen, H Rosie Xing, and Yves A Lussier. Single sample expression-anchored mechanisms predict survival in head and neck cancer. *PLoS computational biology*, 8(1):e1002350, 2012.

[81] Marumi Ohno, Toshiki Sekiya, Naoki Nomura, Masashi Shingai, Hiroshi Kida, et al. Influenza virus infection affects insulin signaling, fatty acid-metabolizing enzyme expressions, and the tricarboxylic acid cycle in mice. *Scientific reports*, 10(1):1–12, 2020.

[82] Gary K Geiss, Mahru C An, Roger E Bumgarner, Erick Hammersmark, Dawn Cunningham, and Michael G Katze. Global impact of influenza virus on cellular pathways is mediated by both replication-dependent and-independent events. *Journal of Virology*, 75(9):4321–4331, 2001.

[83] Jourdan E Brune, Mary Y Chang, William A Altemeier, and Charles W Frevert. Type i interferon signaling increases versican expression and synthesis in lung stromal cells during influenza infection. *Journal of Histochemistry & Cytochemistry*, 69(11):691–709, 2021.

[84] Maimoona Shahid Bhutta, Elisa S Gallo, and Ronen Borenstein. Multifaceted role of AMPK in viral infections. *Cells*, 10(5):1118, 2021.

[85] Emily A Stevens, Joshua D Mezrich, and Christopher A Bradfield. The aryl hydrocarbon receptor: a perspective on potential roles in the immune system. *Immunology*, 127(3):299–311, 2009.

[86] Cristina Gutiérrez-Vázquez and Francisco J Quintana. Regulation of the immune response by the aryl hydrocarbon receptor. *Immunity*, 48(1):19–33, 2018.

[87] Vira Bitko and Sailen Barik. Persistent activation of RelA by respiratory syncytial virus involves protein kinase c, underphosphorylated iκbβ, and sequestration of protein phosphatase 2a by the viral phosphoprotein. *Journal of Virology*, 72(7):5610–5618, 1998.

[88] Karl W Thomas, Martha M Monick, Janice M Staber, Timor Yarovinsky, A Brent Carter, and Gary W Hunninghake. Respiratory syncytial virus inhibits apoptosis and induces NF-κB activity through a phosphatidylinositol 3-kinase-dependent pathway. *Journal of Biological Chemistry*, 277(1):492–501, 2002.

[89] Cameron D Griffiths, Leanne M Bilawchuk, John E McDonough, Kyla C Jamieson, Farah Elawar, Yuchen Cen, Wenming Duan, Cindy Lin, Haeun Song, Jean-Laurent Casanova, et al. IGF1R is an entry receptor for respiratory syncytial virus. *Nature*, 583(7817):615–619, 2020.

[90] Tae Whan Kim, Kirk Staschke, Katarzyna Bulek, Jianhong Yao, Kristi Peters, Keun-Hee Oh, Yvonne Vandenburg, Hui Xiao, Wen Qian, Tom Hamilton, et al. A critical role for IRAK4 kinase activity in Toll-like receptor–mediated innate immunity. *The Journal of Experimental Medicine*, 204(5):1025–1036, 2007.

[91] R Irizarry, B Hobbs, F Collin, Y Beazer-Barclay, K Antonellis, U Scherf, and T Speed. Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data. *Biostatistics*, 4(2):249–264, 2003.

[92] M Ritchie, B Phipson, D Wu, Y Hu, C Law, W Shi, and G Smyth. *limma* Powers Differential Expression Analyses for RNA-Sequencing and Microarray Studies. *Nucleic Acids Research*, 43(7):e47–e47, 2015.

[93] G Sales, E Calura, D Cavalieri, and C Romualdi. graphite-A Bioconductor Package to Convert Pathway Topology to Gene Network. *BMC Bioinformatics*, 13(1):1–12, 2012.

[94] Cheng Fan, Aleix Prat, Joel S Parker, Yufeng Liu, Lisa A Carey, Melissa A Troester, and Charles M Perou. Building prognostic models for breast cancer patients using clinical variables and hundreds of gene expression signatures. *BMC medical genomics*, 4(1):1–15, 2011.

[95] Xiaowei Niu, Jingjing Zhang, Lanlan Zhang, Yangfan Hou, Shuangshuang Pu, Aiai Chu, Ming Bai, and Zheng Zhang. Weighted gene co-expression network analysis identifies critical genes in the development of heart failure after acute myocardial infarction. *Frontiers in genetics*, page 1214, 2019.

[96] Patrik D'haeseleer, Shoudan Liang, and Roland Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[97] Eran Segal, Michael Shapira, Aviv Regev, Dana Pe'er, David Botstein, Daphne Koller, and Nir Friedman. Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature genetics*, 34(2):166–176, 2003.

[98] Ziv Bar-Joseph, Georg K Gerber, Tong Ihn Lee, Nicola J Rinaldi, Jane Y Yoo, François Robert, D Benjamin Gordon, Ernest Fraenkel, Tommi S Jaakkola, Richard A Young, et al. Computational discovery of gene modules and regulatory networks. *Nature biotechnology*, 21(11):1337–1342, 2003.

[99] Joshua M Stuart, Eran Segal, Daphne Koller, and Stuart K Kim. A gene-coexpression network for global discovery of conserved genetic modules. *science*, 302(5643):249–255, 2003.

[100] Bin Zhang and Steve Horvath. A general framework for weighted gene co-expression network analysis. *Statistical applications in genetics and molecular biology*, 4(1), 2005.

[101] Peter Langfelder and Steve Horvath. WGCNA: An R Package for Weighted Correlation Network Analysis. *BMC bioinformatics*, 9(1):1–13, 2008.

[102] Steve Horvath and Jun Dong. Geometric interpretation of gene coexpression network analysis. *PLoS computational biology*, 4(8):e1000117, 2008.

[103] Peter Bailey, David K Chang, Katia Nones, Amber L Johns, Ann-Marie Patch, Marie-Claude Gingras, David K Miller, Angelika N Christ, Tim JC Bruxner, Michael C Quinn,

et al. Genomic analyses identify molecular subtypes of pancreatic cancer. *Nature*, 531(7592):47–52, 2016.

[104] Limin Fu and Enzo Medico. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC bioinformatics*, 8(1):1–15, 2007.

[105] Andrew E Teschendorff, Michel Journée, Pierre A Absil, Rodolphe Sepulchre, and Carlos Caldas. Elucidating the altered transcriptional programs in breast cancer using independent component analysis. *PLoS computational biology*, 3(8):e161, 2007.

[106] Wouter Saelens, Robrecht Cannoodt, and Yvan Saeys. A comprehensive evaluation of module detection methods for gene expression data. *Nature communications*, 9(1):1–12, 2018.

[107] Sipko Van Dam, Urmo Vosa, Adriaan van der Graaf, Lude Franke, and Joao Pedro de Magalhaes. Gene co-expression analysis for functional classification and gene–disease predictions. *Briefings in bioinformatics*, 19(4):575–592, 2018.

[108] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613, 2019.

[109] Peter Langfelder and Steve Horvath. Eigengene networks for studying the relationships between co-expression modules. *BMC systems biology*, 1(1):1–17, 2007.

[110] Xuewei Wang, Ertugrul Dalkic, Ming Wu, and Christina Chan. Gene module level analysis: Identification to networks and dynamics. *Current opinion in biotechnology*, 19(5):482–491, 2008.

[111] Juan A Botía, Jana Vandrovcova, Paola Forabosco, Sebastian Guelfi, Karishma D'Sa, John Hardy, Cathryn M Lewis, Mina Ryten, and Michael E Weale. An additional k-means clustering step improves the biological features of WGCNA gene co-expression networks. *BMC systems biology*, 11(1):1–16, 2017.

[112] Jie Hou, Xiufen Ye, Chuanlong Li, and Yixing Wang. K-module algorithm: an additional step to improve the clustering results of WGCNA Co-expression networks. *Genes*, 12(1):87, 2021.

[113] 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.

[114] Nathan Mankovich, Eric Kehoe, Amy Peterson, and Michael Kirby. Pathway expression analysis. *Scientific Reports*, 12(1):1–13, 2022.

[115] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.

[116] Narges Rezaie, Fairlie Reese, and Ali Mortazavi. PyWGCNA: A Python package for weighted gene co-expression network analysis. *bioRxiv*, 2022.

[117] Kristin Scoggin, Jyotsana Gupta, Rachel Lynch, Aravindh Nagarajan, Manuchehr Aminian, Amy Peterson, L Garry Adams, Michael Kirby, David W Threadgill, and Helene L Andrews-Polymenis. Elucidating mechanisms of tolerance to salmonella typhimurium across long-term infections using the collaborative cross. *Mbio*, 13(4):e01120–22, 2022.

Appendix A

Frame Theory Proof

Lemma A.0.1. Suppose we have two ETFs of size p with frame bound A in $\mathbb{R}^n$ with respective synthesis matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ then $\mathbf{U\Phi V} = \mathbf{\Psi}$ for some orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{p\times p}$.

Proof. Consider the SVDs for each synthesis matrix. $\mathbf{\Phi} = \mathbf{U_1\Sigma_1{V_1}}^T$ and $\mathbf{\Psi} = \mathbf{U_2\Sigma_2{V_2}}^T$. We know $\mathbf{\Sigma_1} = \mathbf{\Sigma_2}$ because these frames are both tight. As a side note, this means both of these frames have the same frame bound. Using the orthogonality of the left and right singular matrices we have

$$\mathbf{{U_1}}^T\mathbf{\Phi V_1} = \mathbf{{U_2}}^T\mathbf{\Psi V_2} \implies \mathbf{U_2{U_1}}^T\mathbf{\Phi V_1{V_2}}^T = \mathbf{\Psi}.$$

Notice $\mathbf{U} = \mathbf{U_2{U_1}}^T \in O(p)$ because$(\mathbf{U_2{U_1}}^T)^T(\mathbf{U_2{U_1}}^T) = \mathbf{U_1{U_2}}^T\mathbf{U_2{U_1}}^T = \mathbf{U_1{U_1}}^T = \mathbf{I}$ and $\mathbf{V} = \mathbf{V_1{V_2}}^T \in O(n)$ for similar reasons. □