Winter 1-15-2021

Machine Learning Morphisms: A Framework for Designing and Analyzing Machine Learning Work ows, Applied to Separability, Error Bounds, and 30-Day Hospital Readmissions

Eric Zenon Cawi
Washington University in St. Louis

Washington University in St. Louis

McKelvey School of Engineering

Department of Electrical and Systems Engineering

Dissertation Examination Committee:
Arye Nehorai, Chair
Shantanu Chakrabartty
ShiNung Ching
Ulugbek Kamilov
Patricio S. La Rosa
Neal Patwari

Machine Learning Morphisms: A Framework for Designing and Analyzing Machine Learning Workflows, Applied to Separability, Error Bounds, and 30-Day Hospital Readmissions

by
Eric Cawi

A dissertation presented to
the Graduate School
of Washington University
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

January 2021
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

I would like to first thank my advisor, Prof. Arye Nehorai, for his mentorship and insights over the course of this dissertation, as well as financial support in the final months. Next, thank you to Dr. Patricio S. La Rosa for our close collaboration, and for always being gentle when dismantling my wilder ideas. Thank you to the committee, Professors ShiNung Ching, Ulegbek Kamilov, Neal Patwari, and Shantanu Chakrabartty, and Dr. Patricio S. La Rosa for providing feedback. Thank you to Dr. Ann-Marcia Tukpah and Dr. Lenise Cummings-Vaughn, for close collaboration on the hospital readmissions projects, which became the genesis for the MLM. Thank you to my friends in the department and St. Louis area, who have shared in the trials and tribulations of the Ph.D. process. Finally, huge thank you to my parents, who have always supported my work and provided commentary on my research and papers, and listened to my practice talks.

I wish to acknowledge the financial support of the Preston M. Green Department for the academic year 2015-2016, the NSF Graduate Research Fellowship Program from 2016-2019, the Los Alamos National Laboratory Applied Machine Learning Summer school for summer 2018, and Dr. Naveen Singla and Bayer Decision Sciences Emerging Talent for Summer 2020.

I also wish to thank the McDonnell International Scholars Academy for housing support and five years of opportunites from 2015-2020.

Eric Cawi

Washington University in Saint Louis

January 2021

Dedicated to my parents.

ABSTRACT OF THE DISSERTATION

Machine Learning Morphisms: A Framework for Designing and Analyzing Machine Learning Workflows, Applied to Separability, Error Bounds, and 30-Day Hospital Readmissions

by

Eric Cawi

Doctor of Philosophy in Systems Science and Mathematics

Washington University in St. Louis, January 2021

Research Advisor: Professor Arye Nehorai

A machine learning workflow is the sequence of tasks necessary to implement a machine learning application, including data collection, preprocessing, feature engineering, exploratory analysis, and model training/selection. In this dissertation we propose the Machine Learning Morphism (MLM) as a mathematical framework to describe the tasks in a workflow. The MLM is a tuple consisting of: Input Space, Output Space, Learning Morphism, Parameter Prior, Empirical Risk Function. This contains the information necessary to learn the parameters of the learning morphism, which represents a workflow task.

In chapter 1, we give a short review of typical tasks present in a workflow, as well as motivation for and innovations in the MLM framework.

In chapter 2, we first define data as realizations of an unknown probability space. Then, after a brief introduction to statistical learning, the MLM is formally defined. Examples

of MLM's are presented, including linear regression, standardization, and the Naive Bayes Classifier. Asymptotic equality is defined between MLM's by analyzing the parameters in the limit of infinite training data. Two definitions of composition are proposed, output and structural. Output composition is a sequential optimization of MLM's, for example standardization followed by regression. Structural composition is a joint optimization inspired by backpropagation from neural nets. While structural compositions yield better overall performance, output compositions are easier to compute and interpret.

In Chapter 3, we define the property of separability, where an MLM can be optimized by solving lower dimensional sub problems. A separable MLM represents a divide and conquer strategy for learning without sacrificing optimality. We show three cases of separable MLM's for mean-squared error with increasing complexity. First, if the input space consists of centered, independent random variables, OLS Linear Regression is separable. This is extended to linear combinations of uncorrelated ensembles, and ensembles of non-linear, uncorrelated learning morphisms. The example of principal component regression is explored thoroughly as a separable workflow, and the choice between equivalent linear regressions is discussed. These separability results apply to a wide variety of problems via asymptotic equality. Functions which can be represented as power series can be learned via polynomial regression. Further, independent and centered power series can be generated using an orthogonal extension of principal component analysis (PCA).

In Chapter 4, we explore the connection between generalization error and lower bounds used in estimation. We start by defining the "Bayes MLM", the best possible MLM for a given problem. When the loss function is mean-squared error, Cramer-Rao lower bounds exist for an MLM which depend on the bias of the MLM and the underlying probability distribution. This can be used as a design tool when selecting candidate MLM's, or as a tool for sensitivity

analysis to examine the error of an MLM across a variety of parameterizations. A lower bound on the composition of MLM's is constructed by applying a nonlinear filtering framework to the composition. Examples are presented for centering, PCA, ordinary least-squares linear regression, and the composition of these MLM's.

In Chapter 5 we apply the MLM framework to design a workflow that predicts 30-day hospital readmissions. Hospital readmissions occur when a patient is admitted less than 30 days after a previous hospital stay. We examine readmissions for a group of medicare/medicaid patients with the four most common diagnoses at Barnes Jewish Hospital. Using MLM's, we incorporate the Mapper algorithm from topological data analysis into the predictive workflow in a novel ensemble. This ensemble first performs fuzzy clustering on the training set, and then trains models independently on each cluster. We compare an assortment of workflows predicting readmissions, and workflows featuring mapper outperform other standard models and current tools used for risk prediction at Barnes Jewish. Finally, we examine the separability of this workflow. Mapper workflows incorporating AdaBoost and logistic regression create node models with low correlation. When PCA is applied to each node, Random Forest node models also become decorrelated. Support Vector Machine node models are highly correlated, and do not converge when PCA is applied. This is consistent with their worse performance.

In Chapter 6 we provide final comments and future work.

Chapter 1

Machine Learning Workflows: Background and Motivation

This dissertation introduces a fundamental mathematical framework to describe and analyze machine learning workflows called the Machine Learning Morphism (MLM). In this chapter we will provide background and motivation for the MLM. In Chapter 2 we formally define the MLM and introduce basic properties of composition and asymptotic equality for workflows. In Chapter 3 we investigate separability properties of MLM's, and show that a wide variety of workflows can be approximated in a separable manner. In Chapter 4, we develop Cramer-Rao style lower bounds for the generalization error of MLM's, and investigate the contribution of compositions to that error, In Chapter 5 we develop a workflow to predict Hospital Readmissions using the MLM framework, improving over current methods.

This prompts a key question: What is a machine learning workflow? Fundamentally, machine learning establishes a *model* between a *Feature Space* and an outcome of interest. The *Feature Space*, or *Input Space*, is a space whose elements represent aspects of the process, system, or data under study. The outcome of interest could be a number (regression), a discrete object (classification, clustering), a function, or any other mathematical object. To learn, we assume

that there is a true but unknown relationship between the feature and the outcomes, and learn the proper model between input and outcome based on previous observations.

Ideally, we would collect data, choose a set of candidate model types based on the problem at hand, and train the models accordingly. However, the structure of data is generally not immediately conducive to model training. For example, in Chapter 5 the data is a mixture of real valued and categorical variables, and in order to train models we chose to embed the categorical variables into real-valued space. This suggests that the mapping between inputs and outputs is a sequence of learning tasks, which we will call a *Machine Learning Workflow*. Expressing a model with a workflow allows us to improve performance at a variety of stages.

Workflows begin with tasks associated to data collection. One task in data collection is data discovery, the process of acquiring a dataset [88]. Sources include open source benchmark sets such as the UCI Repository [27], data from experiments, and/or data from processes such as website clicks. Another task is augmentation, which is the addition of external data to previously collected data. Augmentation is used widely in neural nets to increase the size of the training set, mainly featuring warping the data (rotation, shifting, flipping for images) or using sampling. As a final example of a data collection task, consider data generation. This is used when sufficient data does not exist, and collection must be crowdsourced, for example, Amazon Mechanical Turk [75]; or generated synthetically via packages such as sythpop [72] or SMOTE Sampling [23]. Errors in data collection propagate through the entire workflow. For example, if collected data does not represent the entire space of features and outcomes, then a model will never learn parts of the relationship, and never correctly predict certain input [11].

After data has been collected, preprocessing tasks such as standardization "clean" the data and transform it into a format compatible with later tasks. Different problems necessitate

different preprocessing, tasks. For example, missing data is endemic in real applications, and must be handled. Common tasks which handle missing data include deletion and imputation [40]. Sampling is also used in preprocessing, as we wish to remove outliers and select the most informative subsets of the collected data [36]. In categorical variables, embeddings are used to project discrete data into real valued vector spaces [81]. Conversely, sometimes real features are discretized or binned into intervals [31]. In Chapters 3 and 4, we will see that centering, or subtracting the mean from data, is a fundamentally important preprocessing task.

Another potential task in a workflow, exploratory analysis deals with the visualization, statistical properties, and qualitative analysis of data [50]. One goal is to elucidate patterns or properties which can be exploited later in the workflow. Another goal of exploratory analysis is to present insights to a wider audience using tools such as Tableau. Visualization tasks include plotting high dimensional data [17], representation and interpretation of learning models [58], or exploring matrix structures such as covariance [77]. Principal Component Analysis (PCA) [85] is used for dimension reduction, decorrelating data, and visualization, and is a commonly used tool in exploratory analysis and feature engineering. In Chapter 5, we also examine the TDA Mapper algorithm [93], a relatively recent exploratory tool which creates a graph representation that captures the "shape" of high dimensional data.

Feature extraction builds off of exploratory analysis by either selecting the most relevant features, engineering new features from the data, or both. By building new features, the goal is to extract more relevant information and elucidate better training [42]. In addition to PCA, other examples of feature extraction used for dimension reduction include Linear Discriminant Analysis [6] and Multidimensional Scaling [25]. Autoencoders use a neural net to generate a representation of the data [101]. Manifold learning assumes that data lies

on a manifold, and learns the representation of data on that manifold. Once features are extracted, they must be selected, using an importance metric such as the Akaike Information Criterion [102] or the Random Forest variable importance score [103]. Feature extraction and selection are critical to overall performance.

Model training is what comes to mind with the phrase "machine learning," and handles the actual prediction of the newly created features. In this work, we assume that model parameters are learned by optimizing a loss function over a training set. Common examples of models include neural networks, random forests, and support vector machines. Models often come with hyperparameters, which specify important aspects of the model but are not optimized directly by the loss function. Instead, hyperparameter selection and model comparison are done via processes such as cross validation. Cross validation works by setting aside some of the training data as a validation set, training a model, and computing a metric. This process is repeated many times and the metric is averaged to provide a more robust estimate. In addition to hyperparameter and model selection, cross validation is used to provide estimates of the overall model performance as well [110].

Workflow development is often seen as a combination of art and science, as there is no "ultimate" workflow that is best suited for every situation [48]. Often, new ideas for feature extraction or model training necessitate changes to the preprocessing steps or model selection. Mathematically, the individual components of workflows are well studied, but there is no rigorous theory for design of workflows as a whole.

The field of Auto-ML seeks automated design of workflows from an algorithmic perspective. Packages such as TPOT [73] and Auto-sklearn [32] create wrappers around the popular scikit-learn package in python, while Auto-WEKA [54] performs hyperparameter optimization over the WEKA platform [109]. TPOT constructs a graphical model of a workflow, and then

uses a genetic algorithm to search the space of possible workflows [73], Auto-WEKA uses a bayesian framework to iteratively optimize model hyperparameters. Auto-sklearn builds off of Auto-WEKA, but also incorporates model performance on past datasets and creates ensemble models out 15 classifiers available in scikit-learn [32]. Google's Cloud AutoML is the tech giant's entry into the AutoML arena, which performs adaptive architecture search [57].

Auto-ML packages are based off of different systems and languages, but they are required to establish some sort of *ontology* on the space of machine learning. Mathematically, an ontology is a set of objects (for example, functions in a library) and a set of operations, properties, and relations between them (continuing the example: input/output compatibility, equality, summation of functions). The packages I have mentioned above are examples of algorithmic ontology, which feature a set of functions and rules about which functions can interface as inputs/outputs. Then they search the space of workflows created by these rules. The field of "Ontology Learning" uses machine learning to create ontologies across different sets of data, and features prominently in text and language processing [63]. This, however, is not an ontology on learning itself. In [104] and [105], Wang and others define a "Concept Algebra" as an ontology for knowledge and machine learning modeling. A *concept* is an abstract structure that could represent data, functions, or algorithms, and a large set of formal algebraic operations are defined. Some challenges with this approach include mathematical density, and the need to describe a huge array of concepts to cover the various tasks present in a workflow over a huge array of data types. Therefore, there is a need for a simplified mathematical representation of learning.

To address this need, we proposed the MLM. Every workflow task features an input/output relationship, a mapping with parameters, and optimization over an objective function (or

equivalent selection method) which learns those parameters. An MLM is a tuple of five elements which encodes the information necessary to perform a machine learning task [19], and is endowed with several properties of equality, composition, and separability [20].. In the coming chapters we will show that MLM's can be used to design workflows, analyze separability conditions, and investigate error bounds.

The MLM framework is inspired first and foremost by statistical learning, which we briefly review in Chapter 2. In statistical learning theory, models are called *learning machines*, and models are selected and parameters learned via the process of empirical risk minimization [98]. Statistical learning makes several assumptions on the regularity of the learning machines, and specifically studies the model training portion of a workflow [99]. The MLM framework loosens assumptions on the structure of the learning machines with the goal of incorporating a larger variety of workflow tasks. For example, we use morphisms instead of functions in order to account for operations transforming from one category to another. The MLM couples the choice of learning morphism to a risk function because the parameters of the workflow are intrinsically coupled to the choice of risk function. Other fields which define mathematical formalizations of machine learning include but are not limited to Vapnik-Chervonenkis (VC) Dimension theory [45], probably approximately correct learning [44], and algorithmic learning theory [5]. We will rely heavily on the study of empirical risk minimization in VC Theory, which sets conditions for convergence to minimizing the expected value of the risk [97].

The innovations of the MLM framework lie in its construction as a novel mathematical object. We are able to define operations and comparisons across workflows, and propose structural composition as a novel type of workflow optimization. Asymptotic equality lays

the foundation for algebraic operations and linear vector spaces or other ontological formulations of machine learning workflows. The separability results propose a useful divide and conquer strategy based on functional approximation and asymptotic equality. To the best of our knowledge, Cramer Rao bounds are not fully explored for general workflows, and the propogation of error across steps is novel. To the best of our knowledge, the incorporation of Mapper into a workflow of this type is novel as well.

In the next chapter, we begin with a brief review of statistical learning, and then formally define the MLM and the basic properties of asymptotic equality, output composition, and structural composition.

Chapter 2

The Machine Learning Morphism

This chapter formally defines the Machine Learning Morphism (MLM). This starts with a treatment of the definition of data by Mieske and Liese [67]. We continue with a brief overview of Statistical Learning and Empirical Risk Minimization. Then, we present the MLM as a mathematical object which attempts to generalize the learning machine from statistical learning. We explore examples of MLMs, and define the concept of asymptotic equality between MLM's, as well as two types of composition.

2.1 Statistical Learning Overview

Mieske and Liese [59] define a *statistical model* as a probability space with an unknown distribution paired with a set of candidate distributions that attempt to capture the unknown

Table 2.1: Notation for sets, spaces, functions, etc used throughout this dissertation.

Notation	Meaning	Example
Script Capital	Space	$\mathbb{S}$
Bold, Non-Italic	Set	$\mathbf{X}$
Bold, Italic, Lower Case, Index	Array	$\mathbf{x}, \quad \mathbf{x}_1, \quad \mathbf{y}$
Capital Italic	Function	$P(\cdot)$
Italic, lower case	Scalar	n

distribution. We will use this formulation to explore the idea of machine learning from a statistical perspective.

First define a triple $(\Omega, \mathcal{S}_\Omega, P_\Omega)$, where Ω represents the universe, $\mathcal{S}_\Omega$ is a sigma algebra on Ω, and $P_\Omega : \mathcal{S}_\Omega \to \mathbb{R}$ is the corresponding probability measure. Next, Mieske and Liese define a mathematical representation of the universe called the *Sample Space*, $(\mathbb{S}, \mathcal{S}_\mathbb{S})$. Elements $\sim \in \mathbb{S}$ are realizations of a random variable $S : \mathcal{S}_\Omega \to \mathbb{S}$. $\mathbb{S}$ is also equipped with a sigma algebra $\mathcal{S}_\mathbb{S}$ and unknown probability measure $P_S : \mathcal{S} \to \mathbb{R}$. Finally, a model is a triple

$$(\mathbf{S}, \mathcal{S}_S, \mathbb{P}_\Theta) \tag{2.1}$$

where $\mathbb{P}_\mathbf{A}$ is a collection of probability measures indexed over a set $\mathbf{A}$. The key assumption is that the true probability measure is contained in the model, i.e. $P_S \in \mathbb{P}_\mathbf{A}$, or $\mathbb{P}_\mathbf{A}$ contains a very close approximation to the unknown distribution. By "close", we may mean a small Kullbach-Leibler (KL) Divergence, or directly compare the moments of the true and approximating distributions.

In machine learning, we impose additional structure on the sample space, i.e. $\mathbb{S} = (\mathbb{X}, \mathbb{Y})$. Here $\mathbb{Y}$ represents a variable or outcome of interest, and $\mathbb{X}$ represents data collected which relates to the outcome of interest via an unknown process. The elements of $(\mathbb{X}, \mathbb{Y})$ are realizations of the joint random variable (X, Y). We again have a sigma algebra $\mathcal{S}_{\mathbb{X},\mathbb{Y}}$ with elements $(\mathbf{x}, \mathbf{y})$ and an unknown joint distribution $P_{(X,Y)}$.

Our fundamental objective, however, is different than that presented in statistical decision theory. Instead of learning the unknown probability distribution, we generally wish to predict the output Y given an observation of X. Essentially, we wish to learn the conditional mean $\mathrm{E}_{Y|X}(Y|X)$. Therefore instead of a set of probability distributions, in statistical learning

we have a set of mappings called *learning machines*, $F : \mathbb{X} \to \mathbb{Y} \in \mathbb{F}$, which attempt to approximate $\mathrm{E}_{Y|X}(Y|X)$. For simplicity we will present an overview of *supervised learning*, which is the case when we have observations of both the input and the output. However, the fundamental idea remains the same in the unsupervised and semi-supervised cases.

In order to pick a learning machine from $\mathbb{F}$, we assume that our observed data is a set of realization from the sample space $(\mathbb{X}, \mathbb{Y})$. We will denote the realizations of the outcome of interest as $\mathbf{Y} \in \mathbb{Y} \times \mathbb{Y} \times \cdots \times \mathbb{Y} = \mathbb{Y}^n$, and we will call the realizations of the features as $\mathbf{X} \in \mathbb{X} \times \mathbb{X} \times \cdots \times \mathbb{X} = \mathbb{X}^n$. In [98], Vapnik et al. give three criterion for a supervised learning problem. The first is the underlying distribution of the features, $P(X = \mathbf{x})$. Next is the unknown conditional distribution $P(Y = \mathbf{y}|X = \mathbf{x})$. Finally, given a parameter space Θ with elements $\boldsymbol{\theta}$, denote the function

$$F : \mathbb{X} \to \mathbb{Y}, \quad \hat{\mathbf{y}} = F(\mathbf{x}; \boldsymbol{\theta}) \tag{2.2}$$

used to approximate $P(Y = \mathbf{y}|X = \mathbf{x})$ as a *learning machine*. The parameters $\boldsymbol{\theta}$ are learned using a loss function

$$L : \mathbb{Y} \times F(\mathbb{X}) \to \mathbb{R}, \quad l = L(\mathbf{y}, F(\mathbf{x}; \boldsymbol{\theta})) \tag{2.3}$$

as a measure of discrepancy between the predictions $F(\mathbf{x}; \boldsymbol{\theta})$ and the observed values $\mathbf{y}$.

Then define the *expected risk function* as:

$$R : \Theta \to \mathbb{R}, \quad r = R(\boldsymbol{\theta}; X, Y, F) = \int_{\mathbb{S}} L(Y, F(X; \boldsymbol{\theta}))dP(X, Y) = \mathrm{E}(L(\boldsymbol{\theta}; X, Y, F)) \tag{2.4}$$

where $P(X = \mathbf{x}, Y = \mathbf{y}) = P(Y = \mathbf{y}|X = \mathbf{x})P(X = \mathbf{x})$ is the unknown probability measure on $\mathbb{X}$ and $\mathbb{Y}$. This represents the expected value of the lost function across all realizations,

and the optimal parameters are defined as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} R(\boldsymbol{\theta}; X, Y, F) \tag{2.5}$$

Statistical learning theory has a rich history, and the properties of learning machines are widely studied and researched [78]. For example, Linear Regression [99], Support Vector Machines (SVMs) [112], and Neural Networks [68] all use this optimization framework with different choices of learning machine, loss, and risk functions to train their parameters. In practice the parameters are learned by approximating Eq. 2.4 with an *empirical risk function [97]* defined on the realizations:

$$\bar{R} : \Theta \to \mathbb{R}, \quad \bar{r} = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{y}_i, F(\mathbf{x}_i; \boldsymbol{\theta})) \tag{2.6}$$

and the optimal parameters are given by:

$$\bar{p}^* = \arg\min_{p\in\boldsymbol{\Theta}} \bar{R}(p; X, Y, F) \tag{2.7}$$

Empirical risk is a valid approximation for the expected risk when Eq. 2.6 converges to Eq. 2.4 in probability as n goes to ∞. The conditions for this convergence are discussed thoroughly in [97], [99], and [98].

Learning workflows consist of a sequence of operations acting on the realizations in the statistical space. We distinguish two types of operations in these workflows. The first set consists of processes such as standardization and sampling, which help guarantee the convergence of the empirical risk function to the continuous risk function defined in Eq. 2.4. The second set of operations learn the parameters of the learning machine F, which can itself be defined as a composition of operations acting on the sample space. These

operations are often separated into stages such as preprocessing, feature extraction/selection, model training, etc., and it can be difficult to understand what is actually happening to the original data or if one has built the best workflow for the task at hand.

In our approach, we define a fundamental building block based off the idea of learning machines and risk minimization in statistical learning theory called the *Machine Learning Morphism (MLM)*, that can be used to systematically build and analyze each step in a machine learning application, and keep track of the data at each step in the process. Since some data operations do not necessarily fit the mathematical definition of functions, for example splitting the data into multiple training and hold-out sets for cross-validation, we use morphisms as the building block rather than functions. This approach allows us to define a workflow as a composition of morphisms acting on the sample space, whose parameters are learned using a risk function acting on the statistical space.

2.2 Defining the Machine Learning Morphism

The MLM was defined in [19] as a mathematical object to describe transformations acting on data in machine learning workflows. Formally the MLM is an object with five components:

Definition 1. Let:

- $\mathbb{X}$ be an input space, where $\mathbb{X}$ is part of a sample space $\{\mathbb{S}, \mathbb{U}\}$ as defined above,
- $\mathbb{Y}$ be an output space,
- $F : \mathbb{X} \to \mathbb{Y}$ a morphism from input to output spaces, with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, and
- $P_\Theta(\boldsymbol{\theta})$ a probability distribution on $\boldsymbol{\theta}$ representing prior knowledge of the parameters

- $\bar{R} : \boldsymbol{\Theta} \to \mathbb{R}$ an empirical risk function of the form $\bar{R} = \frac{1}{n}\sum_{i=1}^{n} L(\mathbf{y}_i, F(\mathbf{x}_i; \boldsymbol{\theta}))$, where $L : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$ is a loss function. We assume that the empirical risk function converges to the expected value of the loss function, i.e. $\lim_{n\to\infty} \bar{R} = \mathrm{E}_{(X,Y)}(L)$. This is a standard assumption in statistical learning [97].

Then the *Machine Learning Morphism* $\mathcal{ML} : \mathbb{X} \to \mathbb{Y}$ is defined as the tuple:

$$\mathcal{ML} : (\mathbb{X}, \mathbb{Y}, F(\mathbf{x}; \boldsymbol{\theta}), P_{\Theta}(\boldsymbol{\theta}), L(\mathbf{y}, F)) \tag{2.8}$$

and the *output* of the MLM is the learning morphism with parameters learned by optimizing the loss function:

$$F(\mathbf{x}; \boldsymbol{\theta}^*) \tag{2.9}$$

where

$$\boldsymbol{\theta}^* = \arg\min_{p \in \mathbb{P}} \bar{R}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Y}, F(\cdot; \boldsymbol{\theta}), P(\boldsymbol{\theta})) \tag{2.10}$$

and $\mathbf{X} \subset \mathbb{X}^n$ and $\mathbf{Y} \subset \mathbb{Y}^n$ are n realizations of the input and output spaces used to learn the parameters with prior distribution $P_{\Theta}(\boldsymbol{\theta})$.

□

The Machine Learning Morphism consists of the morphism F, whose parameters have been optimized over an empricial risk function $\bar{R}$ on the set of realizations from the statistical space. The empirical risk function, or equivalently the choice of loss function, controls the

learning objective. For example in regression tasks, the most common loss function is mean-squared error (MSE), $L = ||\mathbf{y} - F(\mathbf{x}; \boldsymbol{\theta})||_2^2$, which rewards learning machines for staying close to the mean of Y and heavily penalizes error due to outliers. Mean absolute error, on the other hand, seeks the L_1 norm of the error, which applies less weight on the error due to outliers in the data and rewards reducing the overall average error on the dataset.

The risk function also performs any necessary operations on the data in the statistical space to ensure convergence in probability to the expected value risk function in Eq. 2.4. One example of this is applying sampling techniques such as oversampling, undersampling, SMOTE [23], or ROSE [62] to the training data in a classification task. This helps "learn" a better representation of the training data conditioned on each class.

The morphism F represents an operation acting on data, such as data standardization, feature extraction, or regression, In supervised tasks, we assume that the realizations of $\mathbb{Y}$ are matched to corresponding realizations of $\mathbb{X}$. In an unsupervised task or if training outputs are unknown (for example, k-means clustering), the realizations $\mathbf{Y} = \emptyset$ are empty.

In Figure 2.1 we represent a diagram of the interplay between the various spaces, distributions, and functions at work in an MLM. We start with the unknown probability space $((\mathbb{X}, \mathbb{Y}), \mathcal{S}_{(\mathbb{X},\mathbb{Y})}, P_{(X,Y)}(x, y))$. In learning our main goal is to elucidate the relationship between $\mathbb{X}$ and $\mathbb{Y}$ which is probabilistically measured with the unknown conditional expectation $\mathrm{E}(Y|X)$. To do this we introduce a learning morphism F, which has parameters $\boldsymbol{\theta}$ belonging to a probability space $(\Theta_{\mathrm{F}}, \mathcal{S}_{\mathrm{F}}, P(\boldsymbol{\theta}))$. The probability space over the parameters allows us to encode information on the parameters and represent Bayesian operations. To control the learning, we introduce a loss funtion L, for example MSE, which defines how well F approximates values in $\mathbb{Y}$. Ideally, the optimal parameters are learned by minimizing the expected value of the loss function, but in practice this is done using Empirical Risk Minimization.

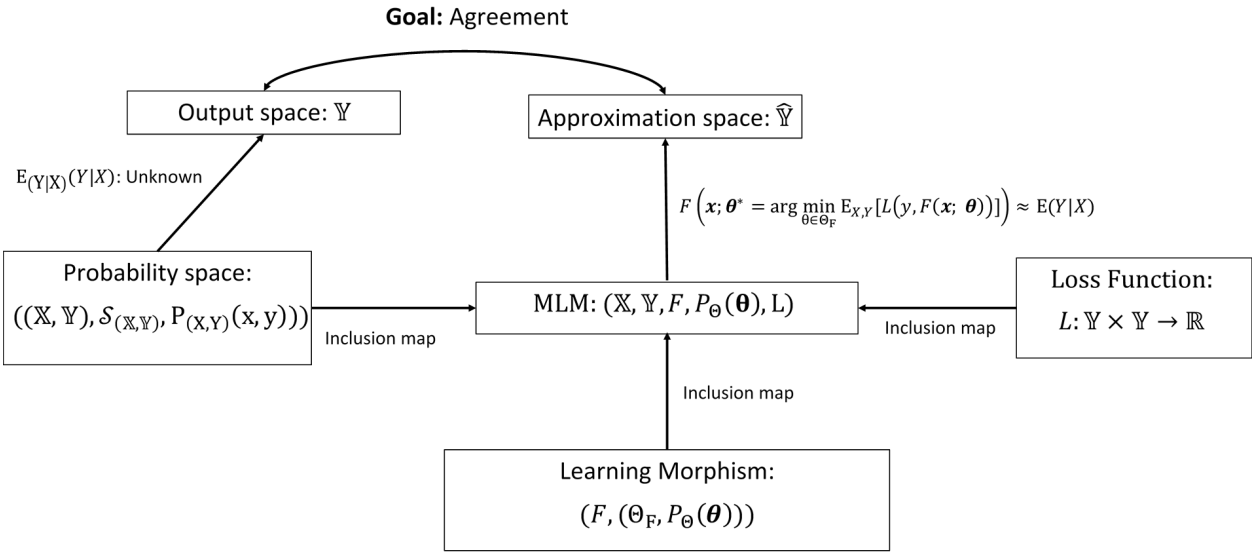


Figure 2.1: Probabilistic representation of an MLM, which can be characterized by the five elements in Def. 1. The learning and choice of optimal parameters are controlled by the loss function. The morphism outputs into an approximation space which may not be exactly the same as the true output space, but for properly designed workflows the approximations $\hat{\mathbf{y}}$ will be close to the true value $\mathbf{y}$.

Further, the range of F is not necessarily the output space $\mathbb{Y}$. Instead, we introduce an "Approximation Space", $\hat{\mathbb{Y}}$. In desirable models, the approximation $\hat{\mathbf{y}} = F(\mathbf{x};\boldsymbol{\theta}^*)$ will be close to the true observation $\mathbf{y}$ for all $\mathbf{x} \in \mathbb{X}$.

2.2.1 Notation with Regards to Priors

Throughout this dissertation we will use two conventions with regard to the parameter prior. In a MLM, there are many cases where we have no prior information on the parameters. In this case we will use a standard non-informative prior:

$$P(\boldsymbol{\theta}) = 1 \quad \forall \boldsymbol{\theta} \tag{2.11}$$

The other important case is when we desire to fix the parameter at a certain value, $\boldsymbol{\theta}_0$. We will use Dirac delta notation for this case:

$$P(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \tag{2.12}$$

Intuitively, a prior of this form says that our knowledge of these parameters is absolute.

2.2.2 Examples of MLM's: Linear Regression, Standardization, Naive Bayes

Consider the example of least squares linear regression. The input space $\mathbb{X}$ is $\mathbb{R}^m$. Denote $\mathbf{X} \in \mathbb{R}^{n \times m}$ as the matrix made from n realizations of $\mathbb{X}$. The output space $\mathbb{Y}$ is $\mathbb{R}$ and denote the n-dimensional vector of all output realizations as $\mathbf{Y}$. We assume no prior knowledge on the parameters of F, so we use the improper prior $P(\boldsymbol{\theta}) = 1 \, \forall \boldsymbol{\theta} \in \mathbb{R}^m$. The morphism F is defined as:

$$F(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{x} \cdot \boldsymbol{\theta} \tag{2.13}$$

is the dot product of $\mathbf{x}$ and $\boldsymbol{\theta} \in \mathbb{R}^M$, and the empirical risk function R is the sum of squared error across all of the datapoints:

$$\bar{R}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Y}, F(\cdot; \boldsymbol{\theta}), P(p)) = ||\mathbf{Y} - \mathbf{X}\boldsymbol{\theta}||_2^2 \tag{2.14}$$

where $||\cdot||_2$ is the standard 2-norm for real-valued vectors. The Machine Learning Morphism is

$$\mathcal{ML}(\mathbf{x}) = \mathbf{x} \cdot \boldsymbol{\theta}^* \tag{2.15}$$

where $\boldsymbol{\theta}^* = \arg\min \ ||\mathbf{Y} - \mathbf{X}\boldsymbol{\theta}||_2^2 = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ is the optimal least squares solution, and $\mathbf{X}^T \in \mathbb{R}^{m \times n}$ is the transpose of $\mathbf{X}$.

Continuing the example of linear regression, many times the data matrix $\mathbf{X}$ consists of shifted and scaled columns. The process of shifting and scaling is an MLM. The input space is $\mathbb{R}^m$, and the output space is $\mathbb{R}^m$. The parameter $\boldsymbol{\theta}$ is a vector containing shifting constants $\mathbf{c} \in \mathbb{R}^m$ and scaling constants $\mathbf{s} \in \mathbb{R}^m$, and again P is the non-informative prior. The morphism

$$F(\mathbf{x}; \boldsymbol{\theta}) = (\mathbf{x} - \mathbf{c})\text{diag}(\mathbf{s})^{-1} \tag{2.16}$$

where $\text{diag}(\mathbf{s})$ is the diagonal matrix whose main diagonal is $\mathbf{s}$. This morphism shifts and scales each element of x. The risk function depends on the type of scaling implemented. For example, the goal of standardization is to transform data into standard Gaussian random variables to ensure that each element of a data point lies on the same scale for comparison, statistical tests, or model training later on in the workflow. For standardization, one choice of empirical risk function is

$$\bar{R}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Y}, F(\cdot; \boldsymbol{\theta}), P(\boldsymbol{\theta})) = KL(P((\mathbf{X} - \mathbf{1}^m \otimes \mathbf{c}^T)\text{diag}(\mathbf{s})^{-1}), \mathcal{N}(0, \mathbf{I}_n)) \tag{2.17}$$

where $KL(\cdot)$ is the KL divergence [84] between the distribution of their shifted and scaled data and the standard multivariate normal distribution, $\mathbf{I}_n$ is an $n \times n$ identity matrix, and $\otimes$ is the Kronecker product. Let $\bar{\mathbf{x}}$ be the vector of column means of $\mathbf{X}$, and $\tilde{\mathbf{x}}$ be the vector of standard deviations of each columns of $\mathbf{X}$. If the data follow a multivariate normal distribution, then the optimal parameters in this case are $\mathbf{c} = \bar{\mathbf{x}}$ and $\mathbf{s} = \tilde{\mathbf{x}}$.

Because we specify a prior distribution on parameters in Definition 1, we can express Bayesian operations as MLMs. Consider the Naive Bayes classifier [86] with input space is a set $\mathbb{X}$.

Similarly, the output space is the set $\mathbb{Y} = \{y_1, ..., y_k\}$ for $k < \infty$. Assuming conditional independence between the input data, the morphism chooses the class which maximizes the posterior distribution, $\arg\max_{i \in \{1,...,k\}} P(Y = y_i | X = \mathbf{x}; \mathbf{p}^*)$, and the empirical risk function is 0-1 loss [86]. The parameter prior $P(\boldsymbol{\theta})$ is a distribution on the parameters of the likelihood $P(X = \mathbf{x}|Y = y; \boldsymbol{\theta})$ and prior $P(y; \boldsymbol{\theta})$.

Further examples of MLM's can be found in Table 2.2. It should be noted that operations commonly performed on the training data, such as sampling, are not MLM's. As discussed above, these operations are handled in the definition of the empirical risk function. For example in the case of sampling the training data is augmented before the risk minimization.

2.2.3 Composition of MLM's

Now that we have defined MLM's and provided several examples, we will define two ideas of composition found in machine learning workflows. These are based on the notion of functional composition, but differ in how to optimize parameters across both MLM's.

The first type of composition is *Output Composition*

Definition 2. Let $\mathcal{ML}_1 : (\mathbb{X}_1, \mathbb{X}_2, F_1, P_{\Theta_1}(\boldsymbol{\theta}_1), L_1)$, $\mathcal{ML}_2 : (\mathbb{X}_2, \mathbb{X}_3, F_2, P_{\Theta_2}(\boldsymbol{\theta}_2), L_2)$, Denote the elements of $\mathbb{X}_i$ with the vector $\mathbf{x}_i$ for $i \in \{1, 2\}$.

The *output composition*, denoted $\mathcal{ML}_2 \circ_{\mathrm{O}} \mathcal{ML}_1$, is the morphism:

$$F_2(F_1(\mathbf{x}_1; \boldsymbol{\theta}_1^*); \boldsymbol{\theta}_2^*) \tag{2.18}$$

Table 2.2: Common operations expressed as MLM's

Operation	Input Space $\mathbb{X}$	Output Space $\mathbb{Y}$	Parameters	Morphism	Empirical Risk Function
Data Encoding	Abstract Space $\mathbb{X}$	$\mathbb{R}^m$	embedding parameters	Injective map: $F : \mathbb{X} \hookrightarrow \mathbb{R}^m$	trivial (one - hot encoding) or cost function
PCA	$\mathbb{R}^m$	$\mathbb{R}^c$	$c \in \mathbb{N}$ $\mathbf{A} \in \mathbb{R}^{m \times c}$	$\mathbf{xA}$	$\|\|\mathbf{X} - \mathbf{XA}^T\|\|_{\mathbb{F}}^2$ Such that: $\mathbf{AA}^T = \mathbf{I}$
Linear Regression	$\mathbb{R}^m$	$\mathbb{R}$	$\mathbf{p} \in \mathbb{R}^m$	$\mathbf{x} \cdot \mathbf{p}$	$\|\|\mathbf{Y} - \mathbf{Xp}\|\|_2{}^2$
Logistic Regression	$\mathbb{R}^m$	$[0, 1]$	$p \in \mathbb{R}^m$	$F(\mathbf{x}.\mathbf{p}) = \frac{1}{1+\exp(-\mathbf{x}\cdot\mathbf{p})}$	Maximum Likelihood $\prod_{i=1}^{N} F(\mathbf{x}_i;\mathbf{p})^{y_i}(1 - F(\mathbf{x}_i;\mathbf{p}))^{1-y_i}$
SVM	$\mathbb{R}^m$	$\{-1, 1\}$	$(\mathbf{w}, \mathbf{b}) \in (\mathbb{R}^m, \mathbb{R}^m)$ slack variables $s \in \mathbb{R}^m,\ c \in \mathbb{R}$	$\mathbf{w} \cdot \mathbf{x} - \mathbf{b}$	$\|\|\mathbf{w}\|\|_2 + c\|\|\mathbf{s}\|\|_2$
Decision Tree	Set $\mathbf{X}$	$\{y_1, y_2, ..., y_k\}$ for finite k	splitting criterion	Tree	Gini Impurity $\sum_{j=1}^{N} 1 - \sum_{i=0}^{1} P(Y_j = i \mid X_j = \mathbf{x}_j)^2$ Information Gain
Standardization	$\mathbb{R}^m$	$\mathbb{R}^m$	$(\mathbf{c}, \mathbf{s}) \in (\mathbb{R}^m, \mathbb{R}^m)$	$(\mathbf{x} - \mathbf{c})\text{diag}(\mathbf{s})^{-}1$	KL Divergence, Eq.
Adaboost	$\mathbb{R}^m$	$\mathbb{R}$	parameters associated with weak learners	$\sum_{i=1}^{k} F_i(\mathbf{x})$	Exponential Loss: $\sum_{i=1}^{N} \exp(-y_i \sum_{j=1}^{k} F_j(\mathbf{x}_i))$
Neural Networks	$\mathbb{R}^m$	$\mathbb{R}$	Weights in $\mathbb{R}^w$	$F = F_k(F_{k-1}(F_{k-2}(\ldots F_1(\mathbf{x}))))$	Loss functions, examples include mean-squared error: $\|\|\mathbf{Y} - F(\mathbf{X})\|\|^2$, Cross Entropy: $\frac{-1}{N}\sum_{i=1}^{N} y_i \log(F(\mathbf{x}_i)) -$ $(1 - y_i)\log(1 - F(\mathbf{x}_i))$
Model Evaluation	Collection $V_{\mathbb{Y}}^k$	$\mathbb{R}$ or $\mathbb{R}^k$	Evaluation parameters Test/validation set	Performance Metric e.g. Accuracy, Sensitivity, etc.	Complexity Criterion or other objective e.g. Aikeke Information Criterion

where

$$\boldsymbol{\theta}_1^* = \underset{\boldsymbol{\theta}_1 \in \Theta_1}{\arg\min} \quad \frac{1}{n} \sum_{i=1}^{n} L_1(\mathbf{y}_i, F_1(\mathbf{x}_{i,1}; \boldsymbol{\theta}_1)) \tag{2.19}$$

and

$$\boldsymbol{\theta}_2^* = \underset{\boldsymbol{\theta}_2 \in \Theta_2}{\arg\min} \quad \frac{1}{n} \sum_{i=1}^{n} L_2(\mathbf{y}_i, F_2(F_1(\mathbf{x}_{i,1}; \boldsymbol{\theta}_1^*); \boldsymbol{\theta}_2)) \tag{2.20}$$

This represents a sequence of two optimizations. The first parameters are learned with the loss function L_1. The second parameters are trained using the loss function L_2 but the argument $F_2(F_1(\mathbf{x}_1; \boldsymbol{\theta}_1^*); \boldsymbol{\theta}_2)$ incorporates the optimal parameters from the first MLM. Figure 2.2 shows a graphical representation of an output composition, and highlights the sequential nature of the learning operations. It is important to note that this operation is an MLM with structure:

$$\mathcal{ML}_3 = \mathcal{ML}_2 \circ_{\mathrm{O}} \mathcal{ML}_1 = (\mathbb{X}_1, \mathbb{Y}, F_2(F_1(\mathbf{x}; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2), P(\boldsymbol{\theta}_2)\delta(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_1^*), L_2(\mathbf{y}, F_2(F_1(\mathbf{x}; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2))) \tag{2.21}$$

where we have used the dirac delta function, $\delta(\cdot)$, to fix $\boldsymbol{\theta}_1$ at the optimal value with respect to L_1, as mentioned in Section 2.2.1.

Note that this MLM is not necessarily "optimal" with respect to L_2. In [19], the authors defined the operation of composition between two MLM's as a joint optimization. To reflect that we have introduced a definition of output composition, we modify the name of this definition:

Definition 3. Let $\mathcal{ML}_1 : (\mathbb{X}_1, \mathbb{X}_2, F_1, P(\boldsymbol{\theta}_1), L_1)$ and $\mathcal{ML}_2 : (\mathbb{X}_2, \mathbb{Y}, F_2, P(\boldsymbol{\theta_2}), L_2)$

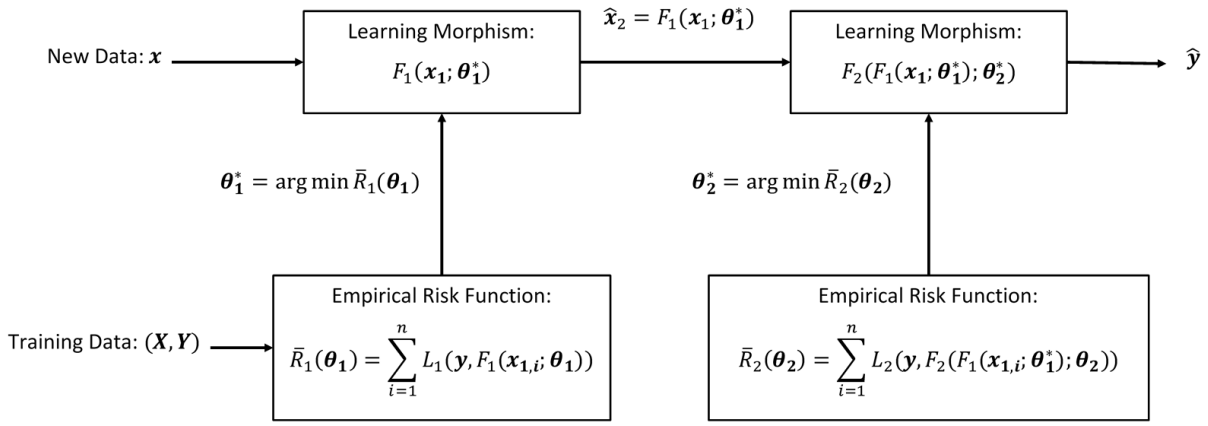


Figure 2.2: The output composition $\mathcal{ML}_2 \circ_{\mathrm{O}} \mathcal{ML}_1$ first estimates the optimal parameters of $\mathcal{ML}_1$ using empirical risk function $\bar{R}_1$. Then the optimal parameters of $\mathcal{ML}_2$ are estimated using the risk function $\bar{R}_2$, using the output of $\mathcal{ML}_1$ on the training set.

The operation of *structural composition*, $\mathcal{ML}_2 \circ_{\mathrm{S}} \mathcal{ML}_1$, is an MLM with structure:

$$\mathcal{ML}_{comp} : (\mathbb{X}_1, \mathbb{Y}, F = F_2 \circ F_1, P(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2), L_2(\mathbf{y}, F_2(F_1(\mathbf{x}; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2)) \tag{2.22}$$

and the corresponding output is:

$$F_2(F_1(\mathbf{x}; \boldsymbol{\theta}_1^*); \boldsymbol{\theta}_2^*) \tag{2.23}$$

with:

$$(\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^*) = \underset{\boldsymbol{\theta}_1 \in \Theta_1, \boldsymbol{\theta}_2 \in \Theta_2}{\arg\min} \quad \sum_{i=1}^{k} L(\mathbf{y}_i, F_2(F_1(\mathbf{x}_{1,i}; \boldsymbol{\theta}_1); \boldsymbol{\theta}_2)) \tag{2.24}$$

where $\mathbf{x}_{1,i} \in \mathbb{X}_1$ are realizations from $\mathbb{X}_1$.

This defines a joint optimization over both parameter sets, which reflects a focus on the final task of interest. A visualization of this idea is presented in Figure 2.3, where it can be seen that instead of a sequential relationship between MLM's, $\mathcal{ML}_1$ and $\mathcal{ML}_2$ both contribute

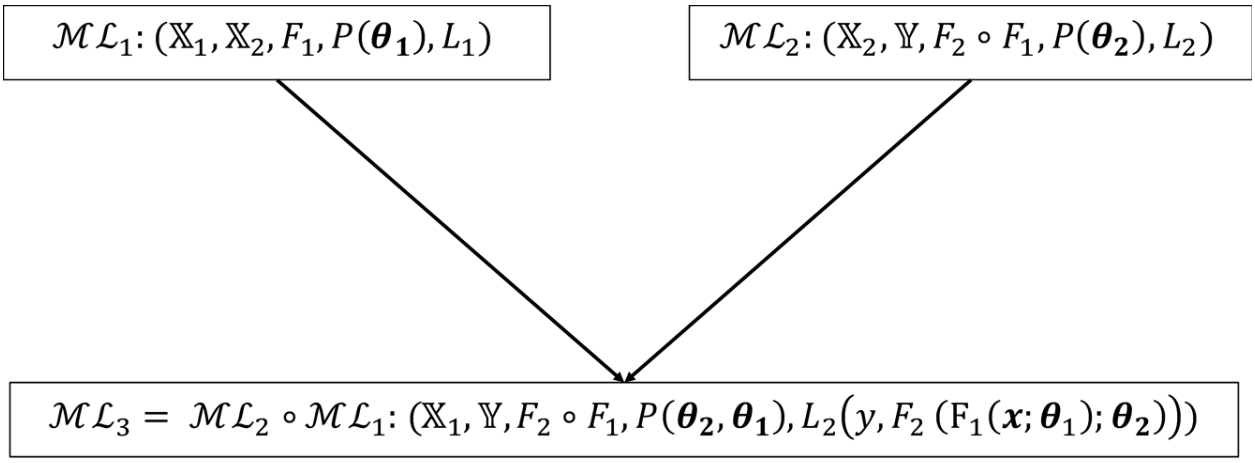


Figure 2.3: The structural composition $\mathcal{ML}_2 \circ_{\mathrm{S}} \mathcal{ML}_1$ estimates the parameters jointly. As an operation on two MLM's it combines elements of both $\mathcal{ML}_1$ and $\mathcal{ML}_2$. This composition will always be more optimal with respect to L_2 but may be more difficult to compute.

elements to $\mathcal{ML}_2 \circ_{\mathrm{S}} \mathcal{ML}_1$. Compared to output composition, structural composition will always be at least as optimal optimal with respect to L_2. However, introducing more parameters means the optimization problem is much harder than either of the optimizations solved in an output composition. Further, learning the parameters of L_1 may be desirable in their own right. For example, PCA is also used as an exploration/visualization tool as well as in predictive models.

These two types of composition allow us to express a workflow:

Definition 4. Let $\mathcal{ML}_i : (\mathbb{X}_i, \mathbb{Y}_i, F_i, P_{\Theta_i}(\boldsymbol{\theta}_i), L_i)$ for $i = 1, 2, ..., p$ be a set of MLM's such that the input spaces and output spaces have the structure $\mathbb{X}_i = \mathbb{Y}_{i-1}$ for $i \geq 2$.

Then a *Machine Learning workflow* is a sequence of $p - 1$ compositions, either output or structural:

$$\mathcal{W} = \mathcal{ML}_k \circ_{p-1} \mathcal{ML}_{p-1} \circ_{p-2} \cdots \circ_2 \mathcal{ML}_2 \circ_1 \mathcal{ML}_1 \tag{2.25}$$

where $\circ_j \in \{\circ_{\mathrm{O}}, \circ_{\mathrm{S}}\}$ for $j = 1, ..., p - 1$

This expresses a workflow as a sequence of learning tasks, some of which are learned independently, and some of which are learned jointly. If all of the compositions are structural compositions, then all the parameters are learned jointly through a single loss function. Conversely, if all the compositions are output compositions, each set of parameters is learned independently.

As an example, take a binary classification task using logistic regression. Let the input space be $\mathbb{X}_1 = \mathbb{R}^k$ and final output space$\mathbb{Y} = \{0, 1\}$. First, perform the operation of centering, which is an MLM that can be expressed as:

$$\mathcal{ML}_{\mathrm{C}} : (\mathbb{R}^k, \mathbb{R}^k, \mathbf{x} - \boldsymbol{\theta}_{\mathrm{C}}, P(\boldsymbol{\theta}_{\mathrm{C}}) = 1, L_{\mathrm{C}} = ||\mathbf{x} - \boldsymbol{\theta}_0)||_2^2 \tag{2.26}$$

To minimize $\mathrm{E}_X(L_{\mathrm{C}})$, simply take $\boldsymbol{\theta}^*_{\mathrm{C}} = E_X(\mathbf{x})$. For finite samples use the sample mean, $\theta^*_{\mathrm{C}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$.

Next, perform a Principal Component Analysis (PCA) on the data to perform dimensionality reduction and decorrelate the input space, with the loss function taken as reconstruction error [43]. This is an MLM with structure:

$$\mathcal{ML}_{\mathrm{P}} : (\mathbb{R}^k, \mathbb{R}^h, F_{\mathrm{P}} = \boldsymbol{\Theta}_{\mathbf{P}}^T \mathbf{x}, P(\Theta_{\mathbf{P}}) = 1, L_P = ||\mathbf{x} - \Theta_{\mathbf{P}}^T \Theta_{\mathbf{P}} \mathbf{x}||_2^2) \tag{2.27}$$

Here $\Theta_{\mathbf{P}}$ is a $h \times k$ matrix with $h \leq k$. The optimal parameters $\Theta^*_{\mathbf{P}}$ are given by the first h eigenvectors of the sample covariance matrix.

This matrix reduces the dimension of the feature space from k to h by taking linear combinations of the original features. Further, this matrix is orthonormal.

Next, perform a logistic regression trained on log-likelihood:

$$\mathcal{ML}_{\mathrm{R}} : (\mathbb{R}^h, [0,1], F_{\mathrm{L}} = \frac{1}{1+\exp(-\boldsymbol{\theta}_{\mathrm{R}}^T \mathbf{x})}, P(\boldsymbol{\theta}_{\mathrm{R}}) = 1, L_{\mathrm{R}} = -F_{\mathrm{R}}(\mathbf{x}_{\mathrm{P}}; \boldsymbol{\theta}_{\mathrm{R}})^y (1 - F_{\mathrm{R}}(\mathbf{x}_{\mathrm{P}}; \boldsymbol{\theta}_{\mathrm{R}}))^{1-y}) \tag{2.28}$$

Denote the optimal parameters of the logistic regression as $\boldsymbol{\theta}_{\mathrm{R}}^* \in \mathbb{R}^h$, which can be solved via Maximum likelihood estimation. The logistic function transforms the feature space to the probability that the output $y = 1$.

Finally, for classification, a threshold is chosen to transform the class probability into a prediction of 0 or 1. This is an MLM with structure:

$$\mathcal{ML}_{\mathrm{T}} : ([0,1], \{0,1\}, F_{\mathrm{T}} = \begin{cases} 1 & x \geq \theta_{\mathrm{T}} \\ 0 & \text{else} \end{cases}, P(\theta_{\mathrm{T}}) = 1, L_{\mathrm{T}} = |y - F_{\mathrm{T}}|) \tag{2.29}$$

This loss function simply measures the accuracy of the classifier, but there are many possible loss functions for the threshold.

Then the workflow is an MLM

$$\begin{aligned} \mathcal{W} = \mathcal{ML}_{\mathrm{T}} \circ_{\mathrm{O}} \mathcal{ML}_{\mathrm{R}} \circ_{\mathrm{O}} \mathcal{ML}_{\mathrm{P}} \circ_{\mathrm{O}} \mathcal{ML}_{\mathrm{C}} = \\ (\mathbb{R}^k, \{0,1\}, F_{\mathrm{W}} = F_{\mathrm{T}}(\frac{1}{1+\exp(-\boldsymbol{\theta}_{\mathrm{R}}^{*\mathrm{T}}((\Theta_{\mathbf{P}}^{*T})(\mathbf{x} - \theta_{\mathrm{C}}^*)))}; \theta_{\mathrm{T}}), ... \\ P(\boldsymbol{\theta}_{\mathrm{T}}) \delta(\boldsymbol{\theta}_{\mathrm{R}} - \boldsymbol{\theta}_{\mathrm{R}}^*) \delta(\Theta_{\mathbf{P}} - \Theta_{\mathbf{P}}^*) \delta(\boldsymbol{\theta}_{\mathrm{C}} - \boldsymbol{\theta}_{\mathrm{C}}^*), L_{\mathrm{T}}(y, F_{\mathrm{W}})) \end{aligned} \tag{2.30}$$

Note in this example that the loss functions L_{C} and L_{P} do not depend on the final output y. This suggests that $\mathcal{ML}_{\mathrm{C}}$ and $\mathcal{ML}_{\mathrm{P}}$ are not learning the same problem as $\mathcal{ML}_{\mathrm{R}}$ and $\mathcal{ML}_{\mathrm{T}}$, and that workflows are a sequence of different learning objectives coming together to

support one or more tasks of interests. For example, when performing testing we need to keep the mean estimates and principal components of the training data stored in order to evaluate the test set.

2.2.4 Asymptotic Equality of MLM's

Since the MLM is a novel type of object, it is natural to envision a linear algebra on MLM's. The first step towards a linear space of MLM's is the notion of equality. Let $\mathcal{ML}_1$ and $\mathcal{ML}_2$ be two mlm's. Denote $\boldsymbol{\theta}_i^{\infty} = \arg\min \lim_{n\to\infty} \bar{R}_i = E(L_i)$, for $i = 1, 2$ as the *asymptotic parameters*, which are learned when optimizing the expected value of the loss function rather than the empirical risk function.

To define the concept of equality between MLM's, it is intuitive that the input spaces, output spaces, and learning morphisms should be the same. However, the empirical risk function is defined over potentially different sets of training realizations, meaning that the learned parameters are themselves realizations of a random variable, and will never be equal with finite data.

Therefore we first attempt to define equality using the asymptotic parameters:

Definition 5. $\mathcal{ML}_1$ and $\mathcal{ML}_2$ are strictly equivalent when

- $\mathbb{X}_1 = \mathbb{X}_2 = \mathbb{X}$: This equality means that the input spaces are exactly the same.
- $\mathbb{Y}_1 = \mathbb{Y}_2 = \mathbb{Y}$: This equality denotes that the output spaces are exactly the same.
- $P(\boldsymbol{\theta}_1) = P(\boldsymbol{\theta}_2)$: This equality denotes equality in distribution.

- $F_1(\mathbf{x};\boldsymbol{\theta}_1) = F_2(\mathbf{x};\boldsymbol{\theta}_2) = F(\mathbf{x};\boldsymbol{\theta})$: This equality is a structural equality, e.g. F_1 and F_2 have the same functional form.
- $L_1(\mathbf{y}, F_1) = L_2(\mathbf{y}, F_2) = L(\mathbf{y}, F)$: This equality also denotes equality in functional form.

This definition states that the underlying loss function is the same, so the asymptotic parameters are the same, i.e. $\boldsymbol{\theta}_1^\infty = \mathrm{E}(L) = \boldsymbol{\theta}_2^\infty$. It utilizes the several different forms of equality, which restricts some attempts at analysis. To ease some of these restrictions, note that as $n \to \infty$, the prior information becomes less and less impactful. Further, in many workflows two functions which give equal output are acceptable, therefore we formulate a less strict definition of asymptotic equality.

Definition 6. $\mathcal{ML}_1$ and $\mathcal{ML}_2$ are asymptotically equivalent, denoted $\mathcal{ML}_1 =_\infty \mathcal{ML}_2$ when

- $\mathbb{X}_1 = \mathbb{X}_2 = \mathbb{X}$
- $\mathbb{Y}_1 = \mathbb{Y}_2 = \mathbb{Y}$
- For a given $\mathbf{x} \in \mathbb{X}$, the output of each learning morphism is the same: $F_1(\mathbf{x};\boldsymbol{\theta}_1^\infty) = F_2(\mathbf{x};\boldsymbol{\theta}_2^\infty)$

Since we are motivated by equivalence in the final output, we do not need equality in functional form, loss function, or parameter priors. We do need the input space and output space

to be the same. This constrains equality to the space of morphisms that are all working on the same sample space $((\mathbb{X}, \mathbb{Y}), \mathcal{S}_{(\mathbb{X},\mathbb{Y})}, P_{(X,Y)})$, and the morphisms F_1 and F_2 are "learning" the same moment.

This is a broader equivalence class of MLM's,, and the advantage of this is that one MLM may be asymptotically equivalent to the other but computationally simpler or more desirable. Asymptotic equality also has the advantage that two models which are identical except for their training realizations are considered the same object. Further, note that the parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ do not have to have the same dimensions or have the same units of measurements. This means that learning morphisms with different but equivalent functional representations are considered asymptotically equivalent.

For example, take the case of univariate logistic regression trained using maximum likelihood and an uninformative prior. Let $\mathcal{ML}_1$ have the structure:

$$\mathcal{ML}_1 : (\mathbb{R}, \mathbb{R}, F_1(x; \theta_0, \theta_1) = \frac{1}{1 + \exp(-(\theta_0 + \theta_1 x))},$$
$$P_\Theta(\theta_1, \theta_0) = 1, L(y, F) = -F(x; \theta_0, \theta_1)^y (1 - F(x; \theta_0, \theta_1))^{1-y}) \tag{2.31}$$

Now let $z = \theta_0 + \theta_1 x$, and assume z is centered around a neighborhood of 0. Then we can express $F_1(x; \theta_0, \theta_1)$ using a Maclaurin series:

$$F(x; \theta_0, \theta_1) = \frac{1}{2} + \frac{1}{4}z - \frac{1}{1/48}z^3 + \frac{1}{480}z^5 - \cdots \tag{2.32}$$

$$= \frac{1}{2} + \frac{1}{4}(\theta_0 + \theta_1 x) - \frac{1}{48}(\theta_0 + \theta_1 x)^3 + \frac{1}{480}(\theta_0 + \theta_1 x)^5 + \cdots \tag{2.33}$$

$$= (\frac{1}{2} + \frac{\theta_0}{4} - \frac{\theta_0^3}{48} + \cdots) + (\frac{\theta_1}{4} - \frac{3\theta_0^2\theta_1}{48} + \cdots)x + (-\frac{-3\theta_1^2\theta_0}{48} + \ldots)x^2 + (\frac{\theta_1^3}{480} + \cdots)x^3 + \cdots \tag{2.34}$$

Define a new set of parameters $\bar{\theta}_0 = \frac{1}{2} + \frac{\theta_0}{4} - \frac{\theta_0^3}{48} + \ldots$, $\bar{\theta}_1 = \frac{\theta_1}{4} - \frac{3\theta_0^2\theta_1}{48} + \ldots$, and so on up to a finite k, we can create a new MLM:

$$\mathcal{ML}_2 : (\mathbb{R}, \mathbb{R}, F_2(x; \bar{\boldsymbol{\theta}}) = \sum_{i=0}^{k} \bar{\theta}_i x^i, P_\Theta(\bar{\boldsymbol{\theta}}) = 1, L(y, F) = -F(x; \bar{\boldsymbol{\theta}})^y (1 - F(x; \bar{\boldsymbol{\theta}}))^{1-y}) \tag{2.35}$$

Again noting that we are centered around a point of interest and for large k, $\mathcal{ML}_1 =_\infty \mathcal{ML}_2$ are asymptotically equivalent because F_1 and F_2 are equivalent functional representations

2.3 Chapter Review

In this chapter we provide a review of statistical learning, formal definitions for an MLM, as well as the property of asymptotic equality and operations of output/structural composition.

Composition is used to formally define a workflow as a type of MLM itself. The innovations of the MLM framework are:

- We generalize the assumption of learning machines from continuous real valued objects to functions or morphisms acting on other categories. In this way we can establish operations acting on diverse sets of inputs and outputs. For example, when working with categorical variables, feature extraction is concerned with embedding discrete sets into $\mathbb{R}^n$ for use in the next stage in the workflow. This can be accomplished very simply by dummy coding, other embeddings such as Multiple Correspondence Analysis, or Weight of Evidence Encoding [90]. Each of these types of encodings optimizes a different loss function, and we can express the transformation as an MLM.

- The definitions of workflows as compositions is a powerful design tool. For each element of the composition, we have a set of candidate morphisms. Further, we must make the choice of whether to define a structural or sequential composition, based on optimality and computational cost.

- The tools of equality and composition(s) can be applied to any MLM's, either whole workflows or MLM's which are part of a larger workflow. Other operations defined on MLM's can be similarly defined and form a set of operations used to manipulate spaces of workflows.

In the next chapter, we will see that the principal of equality can be used in a strategy for scalable learning. Specifically, the idea is to find equivalent MLM's, where one MLM is computationally more desirable than the other.

Chapter 3

Separable Machine Learning Morphisms are Parallel without Sacrificing Optimality

In this Chapter we build upon the concepts defined in Chapter 2 to show that MLM's can be used to define scalable machine learning workflows using separability properties of the loss function. Then we focus on supervised learning problems trained on mean-squared error (MSE) and propose a strategy to achieve scalable workflows in ordinary least squares regression. This result is extended to the case of non-linear orthogonal decompositions and randomized models. Each of these cases divide a workflow into a set of lower dimensional MLM's, which can be solved in parallel. This provides opportunities for scaling while still learning the optimal parameters.

Scalability is a paramount concern in machine learning workflows [41][4]. Training data might have enough samples that learning parameters takes days of compute time, or overloads the working memory of regular computers. Further, the data might have a huge number of dimensions (for example, gene expression data), making matrix computations especially intense. One solution is to purchase time on high performance cores, but that is expensive and still potentially time consuming. Therefore, much research is done to build algorithms

that can handle large amounts of data and/or high dimensional feature spaces by spreading the load across cores/machines, saving time and money.

One method to scale is to find algorithmic speedups. For example, stochastic gradient descent is one method used to make neural network backpropagation more computationally tractable [12][13]. Parallel architectures have been created for many problems, such as Support Vector Machines (SVM) [39], decision trees [7], and genetic algorithms [3][80]. Cross validation, an important technique for model validation and hyperparameter selection, has been parallelized in the popular WEKA platform [21].

Another important strategy for scalability is the *Divide and Conquer* or *Distributed Learning* approach, which breaks a model down into several sub-problems, each of which are faster to solve and can be solved in parallel [79][82]. These models can be *data parallel* where different data is used to train each submodel, *feature parallel* where different features are used to train each submodel, *model parallel*, where different models are trained on the same data, or some combination thereof [100]. Once the sub-problems are trained they are combined, for example as a weighted linear combination or other form of ensemble.

Many distributed learning architectures exist, for example, tree based architectures like [2] or a non-centralized peer-to-peer network [92]. These architectures cover the set of data, feature, and model parallel. Many existing frameworks exist for distributed data management, such as MapReduce [26], Hadoop [107], or Apache Spark [114] which manage data and algorithms in a distributed framework. Similarly, there are many libraries designed to implement models in a distributed manner, perhaps the most famous of which is Tensorflow [1]. Many models make use of random sampling to train ensembles on different subsets of the data or feature space [74], and divide and conquer methods have been used in feature engineering to generate compressed representations of data [10].

Assume we have identified a workflow which we wish to implement in a scalable manner. There are several challenges which must be addressed. How do we know that a worfklow is scalable to begin with? If scalability is possible, which type of scalability do we use? Further, how do we guarantee that the scaled workflow converges to the optimal parameters learned without scaling? In order to answer these questions, we will expand our MLM framework by developing the property of MLM separability. Separability allows us to learn the parameters via divide and conquer while preserving asymptotic equality with our desired workflow.

Then, we investigate the separability of machine learning workflows incorporating MSE loss functions. We explore three different equivalent workflows for linear regression, and the use cases for each workflow. We also investigate separability in the case of non-trivial prior information, and leverage orthogonal decomposition of learning morphisms as a strategy for separability for Mean Square-Error.

3.1 Defining Scalable Machine Learning via Asymptotic Equivalence

Building on asymptotic equality, we now define a divide and conquer framework for MLM's. In this definition we define "scalability" via separability of a loss function into loss functions defined on lower dimensional parameter space:

Definition 7. Let $\mathcal{ML}_\Sigma : (\mathbb{X}_\Sigma, \mathbb{Y}, F_\Sigma, P_\Theta(\boldsymbol{\theta}_\Sigma), L_\Sigma)$ be an MLM. Partition the parameters into m subsets, $\boldsymbol{\theta}_\Sigma = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_m)$.

For $i = 1, ..., m$ Let $F_i : \mathbb{X}_i \to \mathbb{Y}$, for $\mathbb{X}_i \subset \mathbb{X}$, be a set of morphisms with parameters $\boldsymbol{\theta}_i$. Let $\{L_i\}_{i=1}^m$ be a set of m loss functions corresponding to the morphisms F_i.

Then we define $\mathcal{ML}_\Sigma$ as *separable* if:

$$\boldsymbol{\theta}_\Sigma^\infty = \underset{\boldsymbol{\theta}_\Sigma \in \Theta_\Sigma}{\arg\min} \quad \mathrm{E}_{(X,Y)}(L_\Sigma(\mathbf{y}, F_\Sigma(\mathbf{x}; \boldsymbol{\theta}_\Sigma))) = \tag{3.1}$$

$$(\underset{\boldsymbol{\theta}_1 \in \Theta_1}{\arg\min} \quad \mathrm{E}_{(X_1,Y)}(L_1(\mathbf{y}, F_1(\mathbf{x}_1; \boldsymbol{\theta}_1))), \underset{\boldsymbol{\theta}_2 \in \Theta_2}{\arg\min} \quad \mathrm{E}_{(X_2,Y)}(L_2(\mathbf{y}, F_2(\mathbf{x}_2; \boldsymbol{\theta}_2))), \cdots,$$

$$\underset{\boldsymbol{\theta}_m \in \Theta_m}{\arg\min} \quad \mathrm{E}_{(X_m,Y)}(L_k(\mathbf{y}, F_j(\mathbf{x}_m; \boldsymbol{\theta}_j)))) = \tag{3.2}$$

$$(\boldsymbol{\theta}_1^\infty, \boldsymbol{\theta}_2^\infty, \cdots, \boldsymbol{\theta}_m^\infty) \tag{3.3}$$

This definition states that we can learn the asymptotic parameters of $\mathcal{ML}_\Sigma$ by optimizing over several other loss functions independently. The loss functions L_i and morphisms F_i don't necessarily need to belong to other MLM's, they just need to capture the parameters. However, an intuitive strategy we will explore in section 3 is to take F_i as the learning morphism and L_i as the loss functions from a set of lower dimensional MLM's.

From this definition we can use asymptotic equivalence to find scalable MLM's. Let $\mathcal{ML}_1$ be an MLM in a workflow, and let $\mathcal{ML}_\Sigma$ be a scalable MLM. If $\mathcal{ML}_1 =_\infty \mathcal{ML}_\Sigma$, then we can use $\mathcal{ML}_\Sigma$ in place of $\mathcal{ML}_1$ and learn the parameters in parallel. The scaling here is the natural parallelization of the parameters, to choose this strategy one must ensure that optimizing each sub-problem is less computationally costly than learning the parameters of $\mathcal{ML}_1$.

In the next section, we will present some examples of scalable MLM's and workflows featuring scalable MLM's. The examples are organized according to the type of separability: feature, data, and model.

3.2 Strategies for Separability in Mean-Squared Error Problems

3.2.1 Separable Linear MLM's are Feature Parallel

In feature parallel operations, the input space $\mathbb{X}$ is projected down into lower dimensional spaces $\{\mathbb{X}_i\}$, and learning morphisms are trained on each input space before being aggregated together.

Mean-squared error (MSE) is possibly the most common loss function used across machine learning, statistics, and signal processing. Therefore it is natural to investigate separability properties of MLM's with MSE loss functions. To do this, first define a class of MLM's with an MSE loss function and a linear learning morphism. We will call these *least squares linear MLM's*.

Theorem 1 (Orthogonal Decomposition of Least Squares Linear MLM's)**.** Assume $X = \cup_{i=1}^{k} \mathbb{X}_i$ is a union of 1-Dimensional vector spaces over a field $\mathbb{F}$, $\boldsymbol{\theta} = (\theta_1, ..., \theta_k)$ is a set of scalar parameters. Further assume that $E(x_i) = 0$ for $i = 1, ..., k$ and that the random variables X_i are uncorrelated, which then implies $E(X_i X_l) = 0$ for $i \neq l$. Let the output space $\mathbb{Y}$ be a scalar vector space.

Define an MLM with structure:

$$\mathcal{ML}_{\Sigma} : (\mathbb{X}, \mathbb{Y}, F_{\Sigma} = \theta_0 + \sum_{i=1}^{k} \theta_i x_i, P(\boldsymbol{\theta}) = 1, L = (y - F_{\Sigma}(\mathbf{x}; \boldsymbol{\theta}_{\Sigma}))^2) \tag{3.4}$$

The MLM $\mathcal{ML}_{\Sigma}$ is separable.

Proof. See Appendix

□

This theorem easily extends to the case when subsets of $\mathbb{X}$ are block independent.

The MLM in Theorem 1 is an example of a feature parallel implementation, since the same family of model is trained across different features independently. One immediate implication is that centering the input space, a step commonly used in workflows, is important for separabililty in this case. To see why, note that a set of centered random variables forms a hilbert space, and the MSE loss function is proportional to the standard L-2 inner product. The structure of $\mathcal{ML}_\Sigma$ as a sum of orthogonal elements then naturally separates the loss function.

How to Achieve the Sufficient Conditions to leverage Theorem 1

In order to apply the results of Theorem 1 to an MLM with MSE loss, we need to engineer two conditions: an uncorrelated input space and a linear least squares learning morphism. Formally, this means we are looking to satisfy the equality on a learning morphism F:

$$F(\mathbf{x};\boldsymbol{\theta}) = G(\mathbf{z}) = \begin{bmatrix} \boldsymbol{\theta}_2^T & \theta_{2,0} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix} \tag{3.5}$$

where $\mathbf{z}$ is a vector of centered, uncorrelated random variables.

Suppose $\mathbb{X} = \mathbb{R}^k$. If morphism F is already linear with an intercept term, then one strategy is $\mathbf{z} = \mathbf{A}\mathbf{x}$ where $\mathbf{A} \in \mathbb{R}^{h \times k}$ is a matrix representing an orthogonal linear transformation. An

example of this, which we will examine in the next section, is Principal Component Analysis (PCA). PCA scales and rotates the data to a set of orthogonal coordinates, as well as potentially performing dimension reduction. Then the multivariate regression is equivalent to the sum of m univariate linear regressions [64].

Other transformations include data whitening, which first estimates the covariance structure of $\mathbb{X}$ and then uses a whitening matrix to transform the data into a set of independent standard normal variables [53]. Random orthogonal projections are used as a scalable method for dimensionality reduction, which project data into a random low dimensional orthogonal space [9].

When F is not linear, we can leverage the idea of equivalent functional representations to achieve scalability. For example, let x be a centered scalar gaussian random variable, then we know that the moments x^2, x^3, ..., x^k are independent of each other. Therefore, we can define centered variables:

$$z_i = x^i - \mathrm{E}(x^i) \tag{3.6}$$

for $i = 0, 1, ..., k$ for some finite k and form a power series:

$$G = \theta_0 + \sum_{i=1}^{k} \theta_i z_i \tag{3.7}$$

Power series constructed in this way can be used to approximate analytic learning morphisms in a separable way.

For real valued functions of scalar variables, there are many different orthogonal decompositions. For band-limited functions, frequency domain based decompositions such as discrete fourier transforms, wavelet transforms, and time frequency analysis transform the domain

from $\mathbb{R}$ to the frequency domain. To choose which one to use, one must consider the tradeoff of changing the domain of the problem and potentially increasing the number of parameters versus the cost of solving the original optimization problem.

The main takeaway from these examples is that general families of decomposable MLM's with MSE risk need some sort of orthogonal *Feature Space Decomposition*(FSD). FSD's like the Random Subspace Method are often used specifically to decorrellate the feature space to improve ensemble model performance. For linear MLM's, a very common FSD is Principal Component Analysis (PCA). PCA seeks to learn the eigenvectors of the covariance matrix of the input space. This decorrelates the columns of the training matrix, and regression is then separable, which we will demonstrate in the next section.

Example: Principal Component Regression

In this example, we will showcase three different workflows involving least squares linear MLM's. The first is Principal Component Regression, represented as a workflow involving output compositions. This workflow features first centering, then PCA, and then a linear regression. Because PCA results in centered, decorrelated random variables, PCR is a separable workflow by Theorem 1. The second workflow is linear regression with an intercept term. The final workflow is PCR, but featuring a sequence of structural compositions which jointly optimizes all workflow parameters. At the end of this example, we will then think about scenarios to pick different workflows.

When the linear regression is feasible, these workflows are equivalent. To see this, first define a similar workflow as the earlier example featuring logistic regression. Let $\mathbb{X} = \mathbb{R}^k$, $\mathbb{Y} = \mathbb{R}$, and the output space of PCA be denoted as $\mathbb{X}_2 = \mathbb{R}^h$ for $h \leq k$. Let $\mathcal{ML}_0$ with optimal

parameters $\boldsymbol{\theta}_0^* = \mathrm{E}(\mathbf{x})$ be the centering MLM defined in Eq. 2.26, and $\mathcal{ML}_1$ be PCA as defined in Eq. 2.27, with optimal parameter matrix $\boldsymbol{\Theta}_1^*$.

The final MLM is linear regression with an intercept term, which has structure:

$$\mathcal{ML}_2 : (\mathbb{R}^h, \mathbb{R}, F_2 = \boldsymbol{\theta}_2^T \mathbf{x} + \theta_{\mu_y}, P(\boldsymbol{\theta}_2) = 1, L_2 = (y - F_2(\mathbf{x}; \boldsymbol{\theta}_2)^2) \tag{3.8}$$

which has optimal parameters learned on the centered, PCA-transformed data given as $\boldsymbol{\theta}_2^*$ and $\theta_{\mu_y}^*$. We denote the intercept parameter of $\mathcal{ML}_2$ by θ_{μ_y} to reflect that since $\mathbf{x}$ is centered, the intercept learns the mean of the output, $\mathrm{E}(y)$.

Then the output of the workflow is a MLM:

$$\mathcal{W} = \mathcal{ML}_2 \circ_{\mathrm{O}} \mathcal{ML}_1 \circ_{\mathrm{O}} \mathcal{ML}_0 = \tag{3.9}$$

$$(\mathbb{R}^k, \mathbb{R}, F_{\mathcal{W}} = (\boldsymbol{\theta}_2)^T \boldsymbol{\Theta}_1^{*T} (\mathbf{x} - \boldsymbol{\theta}_0 *) + \theta_{\mu_y}, P(\boldsymbol{\theta}_2) \delta(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_1^*) \delta(\boldsymbol{\theta}_0 - \boldsymbol{\theta}_0^*), L_2(y, F_{\mathcal{W}})) \tag{3.10}$$

Now $\mathbf{z} = \boldsymbol{\Theta}_1^{*T}(\mathbf{x} - \boldsymbol{\theta}_0^*)$ represents uncorrelated random variables. Further $\mathcal{ML}_2$ has a learning morphism that is linear with an intercept term. Therefore the machine learning workflow $\mathcal{W}$ is separable by Theorem 1.

Exploring this example further, we note that the learning morphism of workflow $\mathcal{W}$ is functionally equivalent to a linear regression with an intercept. Let $\bar{\boldsymbol{\theta}}_0 = -\boldsymbol{\theta}_2^T \boldsymbol{\Theta}_1^T \boldsymbol{\theta}_0 + \theta_{\mu_y}$ and $\bar{\boldsymbol{\theta}} = \boldsymbol{\Theta}_1 \boldsymbol{\theta}_2$. Then

$$F_{\mathcal{W}} = F_3 = \bar{\theta_0} + \bar{\boldsymbol{\theta}}^T \mathbf{x} \tag{3.11}$$

Now define an MLM with structure

$$\mathcal{ML}_3 : (\mathbb{R}^k, \mathbb{R}, F_3, P(\bar{\boldsymbol{\theta}}_1) = 1, L_3 = (y - F_3)^2) \tag{3.12}$$

The optimal paramters of $\mathcal{ML}_3$, $\bar{\theta}_0^*$ and $\bar{\boldsymbol{\theta}}^*$, are found via the least squares linear regression formula.

A key question: are the parameters of $\mathcal{W}$ equivalent to learning the parameters of $\mathcal{ML}_3$? In other words, is this sequence of output compositions equivalent to an ordinary least squares linear regression?

To have asymptotic equality between $\mathcal{W}$ and $\mathcal{MLM}_3$, the input and output spaces are the same, but we need $F_3(\mathbf{x}; \bar{\boldsymbol{\theta}}^*) = F_{\mathcal{W}}(\mathbf{x}; \boldsymbol{\theta}_2^*, \boldsymbol{\Theta}_1^*, \boldsymbol{\theta}_0^*, \theta_{\mu_y}^*)$, which implies :

$$= \bar{\theta}_0^* + \bar{\boldsymbol{\theta}}^{*T}\mathbf{x} = (\boldsymbol{\theta}_2^*)^T \boldsymbol{\Theta}_1^{*T}(\mathbf{x} - \boldsymbol{\theta}_0 *) + \theta_{\mu_y} = \tag{3.13}$$

In $\mathcal{W}$, we fix the parameters of the first two morphisms $\boldsymbol{\Theta}_1 = \boldsymbol{\Theta}_1^*$ and $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_0^*$.

This leaves $\boldsymbol{\theta}_2$ and θ_{μ_y} as the "free"parameters, and the dimension of $\boldsymbol{\theta}_2$ is h.

Since these are both affine transformations of $\mathbf{x}$, equality will occur when the coefficients are equal.

Then for equality we have a system of linear equations in $\begin{bmatrix} \boldsymbol{\theta}_2 \\ \theta_{\mu_y} \end{bmatrix}$:

$$\begin{bmatrix} \bar{\theta}_0^* \\ \bar{\theta}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_0^{*T} \boldsymbol{\Theta}_1^* & 1 \\ \boldsymbol{\Theta}_1^* & \mathbf{0}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_2 \\ \theta_{\mu_y} \end{bmatrix} \tag{3.14}$$

When $h < k$ (the case of dimension reduction) this system is overdetermined, since there are $p + 1$ unknowns and $k + 1$ equations. This case represents dimension reduction, and depending on the rank of the augmented matrix will have zero, one, or infinite solutions.

When $h = k$, the coefficient matrix is given as:

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{\theta}_0^{*T} \boldsymbol{\Theta}_1^* & 1 \\ \boldsymbol{\Theta}_1^* & \mathbf{0}^k \end{bmatrix} \tag{3.15}$$

where $\mathbf{0}^k$ is a $k \times 1$ vector of zeros.

$\mathbf{A}$ is a $(k+1) \times (k+1)$ matrix. Because Θ_1 is an orthonormal matrix, its columns are linearly independent. Further, the first row is independent of rows 2 through $(k + 1)$ because of the last column which has one "1" and k zeros. Therefore, the rank of $\mathbf{A}$ is $k + 1$, which means this system is consistent. Hence, in the case when there is no dimension reduction, $\mathcal{W}$ is equivalent to $\mathcal{ML}_3$. For a full proof that the optimal solutions of $\mathcal{W}$ and $\mathcal{ML}_3$ are equal, see the appendix.

The next eploration with this example is to compare the workflow $\mathcal{W}$ to a structural composition, $\mathcal{W}_2$ with structure:

$$\mathcal{W}_2 = \mathcal{ML}_2 \circ_S \mathcal{ML}_1 \circ_S \mathcal{ML}_0 =$$

$$(\mathbb{R}^k, \mathbb{R}, F_{\mathcal{W}_2} = (\boldsymbol{\theta}_2)^T \boldsymbol{\Theta}_1^T (\mathbf{x} - \boldsymbol{\theta}_0) + \theta_{\mu_y}, P(\boldsymbol{\theta}_0, \boldsymbol{\Theta}_1, \theta_{\mu_y}, \boldsymbol{\theta}_2) = 1, L_2(y, F_{\mathcal{W}_2})) \tag{3.16}$$

In this workflow, the parameters are learned jointly rather than sequentially. Further, there are no constraints on the shifting parameters $\boldsymbol{\theta}_0$ or matrix $\boldsymbol{\Theta}_1$. In the appendix, we show that the optimal parameters of $\mathcal{W}$ are a critical point of the gradient of the loss function. So the parameters of $\mathcal{W}$ are a local optima of $\mathcal{W}_2$. Further, when $\boldsymbol{\theta}_0^*$ are fixed as the column means of $\mathbf{X}$, and we assume $\boldsymbol{\Theta}_1$ is full rank, then from Eq. A.79 we have

$$\boldsymbol{\theta}_2 = \boldsymbol{\Theta}_1^{-1} (\mathbf{X}_{\mathrm{C}}^T \mathbf{X}_{\mathrm{C}})^{-1} \mathbf{X}_C^T \mathbf{Y} \tag{3.17}$$

and

$$\boldsymbol{\Theta}_1^T (-\frac{1}{n} \mathbf{X}^T \mathbf{1}_{n\times 1} \mathbf{1}_{n\times 1}^T \mathbf{X} (\mathbf{X}_{\mathrm{C}}^T \mathbf{X}_{\mathrm{C}})^{-1} \mathbf{X}_{\mathrm{C}}^T \mathbf{Y}) = \mathbf{0}^k \tag{3.18}$$

The vector multiplying $\boldsymbol{\theta}_1$ in Eq. 3.18 features combinations of the coefficients of a centered linear regression weighted by the square of the column means. This system has k^2 unknowns and k equations. Each of the equations features one of the rows of $\boldsymbol{\Theta}_1$, which has an independed set of parameters from the other equations. This equation has an infinite number of solutions.

When the mean parameters are not fixed, one must solve the system of nonlinear equations in Eq. A.49-A.52. Preliminary MATLAB simulations using *fminunc* suggest that the optimal parameters multiply out to the same value as in $\mathcal{ML}_3$ and have the same MSE. This is

intuitive because composing linear and affine transformations results in another linear transformation, so any combination of parameters resulting in the optimal regression coefficients will have optimal MSE.

Choosing Between Three Equivalent Workflows

$\mathcal{ML}_3$, $\mathcal{W}$, and $\mathcal{W}_2$ have the same predictive value, but $\mathcal{W}$ and $\mathcal{W}_2$ involve many more parameters, so why use PCR? Practically, the regression $\mathcal{ML}_3$ requires computing the inverse of $\mathbf{X}^T\mathbf{X}$, which is $k \times k$ and requires $0(k^3)$ operations for Gauss-Jordan Elimination or $0(k^{2.373})$ when using the Coppersmith-Winograd algorithm [108]. The workflow $\mathcal{W}$ requires computing a Singular Value Decomposition of $\mathbf{X}$, which takes $O(nk^2)$ operations.

There are three cases at work here. The first is when the training matrix $\mathbf{X}$ has full rank k and k is not very large. In this case, the inverse is computationally feasible, and will take less time than PCR. Therefore, we would choose the ordinary least squares mlm $\mathcal{ML}_3$ over $\mathcal{W}$.

In the second case, $\mathbf{X}$ is full rank but the dimension of the feature space is very large, $k >> n$. In this case the SVD is computationally more tractible than the matrix inverse, so we choose $\mathcal{W}$ over $\mathcal{ML}_3$. In this case, however, neither case may be tractable, so we need to perform dimensionality reduction by choosing less than k principal components. Scalable low rank approximations have been researched, for example via random decompositions [94]. Here, however, there is a tradeoff between optimal MSE and computational feasibility.

The third case occurs when $\mathbf{X}$ is not full rank, i.e. there is collinearity between predictors. In this case $\mathcal{ML}_3$ can't be computed, so we choose $\mathcal{W}$ or $\mathcal{W}_2$. In this case we can perform dimensionality reduction without losing predictibility.

Now, the complexity of the structural composition depends on the algorithm used and the initial conditions used to start the iterations. However, in the case of no dimensionality reduction, this workflow is attempting to optimize a nonlinear equation with $2k + 1 + k^2$ parameters, which can be computationally expensive. Further, we know that the optimal parameters will result in the same predictability as $\mathcal{ML}_3$ and $\mathcal{W}$, but lose much of the interpreability inherent in the other two cases. Therefore we will restrict the choice to $\mathcal{ML}_3$ and $\mathcal{W}$. We might choose the structural composition based workflow in the case of a non-linear learning morphism, where it might have a better optimum than an output composition based workflow.

3.2.2 Incorporating Prior Information Using Maximum A Posteriori Estimation

In the previous examples, we have explored cases with uninformative priors. Now, we will extend the case of least squares linear regression to a maximum posteriori problem.

Let $\mathbb{X} = \mathbb{R}^k$ be a set of random variables, $\mathbb{Y} = \mathbb{R}$ also be a random variable. Let $F_{\mathrm{B}}(\mathbf{x}; \boldsymbol{\theta}_{\mathrm{B}})$ be a learning morphism with parameters $\boldsymbol{\theta}_{\mathrm{B}}$ with prior $P(\boldsymbol{\theta}_{\mathrm{B}})$.

In bayesian operations, the key probability distribution is the posterior:

$$P(\boldsymbol{\theta}_{\mathrm{B}}|y, \mathbf{x}) \propto P(y|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{B}})P(\boldsymbol{\theta}_{\mathrm{B}}) \tag{3.19}$$

For multiple i.i.d. samples $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ with corresponding i.i.d. observations $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, the posterior becomes

$$P(\boldsymbol{\theta}_{\mathrm{B}}|y, \mathbf{x}) \propto \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \boldsymbol{\theta}_{\mathrm{B}}) P(\boldsymbol{\theta}_{\mathrm{B}}) \tag{3.20}$$

The Maximum A Posteriori (MAP) estimator is the set of parameters:

$$\theta^*_{\mathrm{B}} = \arg\max_{\theta_{\mathrm{B}} \in \Theta_B} \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \boldsymbol{\theta}_{\mathrm{B}}) P(\boldsymbol{\theta}_{\mathrm{B}}) = \tag{3.21}$$

$$\arg\max_{\theta_{\mathrm{B}} \in \Theta_{\mathrm{B}}} \log(\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \boldsymbol{\theta}_{\mathrm{B}}) P(\boldsymbol{\theta}_{\mathrm{B}})) = \tag{3.22}$$

$$\arg\max_{\theta_{\mathrm{B}} \in \Theta_{\mathrm{B}}} \sum_{i=1}^{n} (\log(P(y_i|\mathbf{x}_i, \boldsymbol{\theta}_{\mathrm{B}})) + \log(P(\boldsymbol{\theta}_{\mathrm{B}}))) = \tag{3.23}$$

$$\arg\min_{\theta_{\mathrm{B}} \in \Theta_{\mathrm{B}}} \sum_{i=1}^{n} -(\log(P(y_i|\mathbf{x}_i, \boldsymbol{\theta}_{\mathrm{B}})) + \log(P(\boldsymbol{\theta}_{\mathrm{B}}))) \tag{3.24}$$

MAP estimators are used very commonly in bayesian and machine learning applications as a computationally tractable alternative to Bayesian MSE estimation.

We can define a *MAP MLM* as follows:

$$\mathcal{ML}_{\mathrm{B}} : (\mathbb{X}, \mathbb{Y}, F_{\mathrm{B}}, P(\boldsymbol{\theta}_{\mathrm{B}}), L_{\mathrm{B}} = -(\log(P(y|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{B}})) + \log(P(\boldsymbol{\theta}_{\mathrm{B}}))) \tag{3.25}$$

which has optimal parameters given as θ^*_{B}.

As before, we will examine the case where $\mathbb{X} = \mathbb{R}^k$ is a set of centered, independent random variables, $\mathbb{Y} = \mathbb{R}$, and $F_{\mathrm{B}} = \boldsymbol{\theta}^T_{\mathrm{B},x}\mathbf{x} + \theta_{\mathrm{B},0}$. Now, however, let the parameter prior be normally distributed as $P(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathrm{B}}, \mathbf{C}_{\mathrm{B}})$ for known hyperparameters $\boldsymbol{\mu}_{\mathrm{B}}, \mathbf{C}_{\mathrm{B}}$. To formulate the MAP MLM, let $\boldsymbol{\theta}_{\mathrm{B}} = \begin{bmatrix} \boldsymbol{\theta}_{\mathrm{B},x} & \theta_{\mathrm{B},0} \end{bmatrix}$. For a MAP linear regression, we will assume that the conditional distribution of the output is gaussian: $P(y|\mathbf{x}, \boldsymbol{\theta}_{\mathrm{B}}) \sim \mathcal{N}(\boldsymbol{\theta}^T_{\mathrm{B}}\mathbf{x}, \sigma^2_y)$ for a known

variance σ_y^2. This means that the MAP loss function is:

$$(\frac{1}{2\sigma_y^2}(y - \boldsymbol{\theta}_{\mathrm{B}}^T\mathbf{x} - \theta_{\mathrm{B},0})^2 + (\boldsymbol{\theta}_{\mathrm{B}} - \boldsymbol{\mu}_{\mathrm{B}})^T\mathbf{C}_{\mathrm{B}}^{-1}(\boldsymbol{\theta}_{\mathrm{B}} - \boldsymbol{\mu}_{\mathrm{B}})) \tag{3.26}$$

We can define a *MAP MLM* as follows:

$$\mathcal{ML}_{\mathrm{B}} : (\mathbb{R}^k, \mathbb{R}, \boldsymbol{\theta}_{\mathrm{B}}^T\mathbf{x}+\theta_{\mathrm{B},0}, \mathcal{N}(\boldsymbol{\mu}_{\mathrm{B}}, \mathbf{C}_{\mathrm{B}}), L_{\mathrm{B}} = (\frac{1}{2\sigma_y^2}(y-\boldsymbol{\theta}_{\mathrm{B}}^T\mathbf{x}-\theta_{\mathrm{B},0})^2+(\boldsymbol{\theta}_{\mathrm{B}}-\boldsymbol{\mu}_{\mathrm{B}})^T\mathbf{C}_{\mathrm{B}}^{-1}(\boldsymbol{\theta}_{\mathrm{B}}-\boldsymbol{\mu}_{\mathrm{B}})) \tag{3.27}$$

The first term is proportional to the MSE loss function explored in Theorem 1. The second term is a weighted 2-norm. If $\mathbf{C}_{\mathrm{B}}$ is block diagonal, then both terms in this loss function are separable with respect to blocks of $\boldsymbol{\theta}_{\mathrm{B}}$, and we have a separable MLM. Formally:

Corollary 1. If $\mathbf{C}_{\mathrm{B}}$ is block diagonal with blocks $\mathbf{C}_{\mathrm{B},i}$ for $i = 1, ..., p$, and $\mathbf{x}$ are realizations of centered, independent random variables, then $\mathcal{ML}_{\mathrm{B}}$ is separable.

Proof. First define the individual regression coefficients as $\boldsymbol{\theta}_{\mathrm{B}} = \begin{bmatrix}\theta_{\mathrm{B},1} & ... & \theta_{\mathrm{B},k} & \theta_{\mathrm{B},0}\end{bmatrix}$. Next, let $\boldsymbol{\theta}_i \subset [\boldsymbol{\theta}_{\mathrm{B}}\ \theta_{\mathrm{B},0}]$ for $i = 1, 2, ..., p$ be the vector of parameters corresponding to the blocks $\mathbf{C}_{\mathrm{B},i}$, and let $\boldsymbol{\mu}_i \subset \boldsymbol{\mu}_{\mathrm{B}}$ be the corresponding vector of prior means.

Define $\mathbf{x}_i \subset \mathbf{x}$ as the set of variables with regression coefficients $\boldsymbol{\theta}_i$.

Because $\mathbf{x}$ are independent and centered random variables, we can separate

$$\frac{1}{2\sigma_y^2}\mathrm{E}(y - \boldsymbol{\theta}_{\mathrm{B}}^T\mathbf{x} - \theta_{\mathrm{B},0})^2 = \tag{3.28}$$

$$\frac{1}{2\sigma_y^2}(\mathrm{E}((y - \theta_{\mathrm{B},1}x_1)^2) + \mathrm{E}((y - \theta_{\mathrm{B},2}x_2)^2) + \cdots + \mathrm{E}((y - \theta_{\mathrm{B},k}x_k)^2) + \mathrm{E}((y - \theta_{\mathrm{B},0})^2) - (k)\mathrm{E}(y^2)) \tag{3.29}$$

by Eq. A.11.

Because $\mathbf{C}_{\mathrm{B}}$ is block diagonal, we can expand the second term as:

$$(\boldsymbol{\theta}_{\mathrm{B}} - \boldsymbol{\mu}_{\mathrm{B}})^T\mathbf{C}_{\mathrm{B}}^{-1}(\boldsymbol{\theta}_{\mathrm{B}} - \boldsymbol{\mu}_{\mathrm{B}}) = \tag{3.30}$$

$$\sum_{i=1}^{p}(\boldsymbol{\theta}_i - \boldsymbol{\mu}_i)^T\mathbf{C}_{\mathrm{B},i}^{-1}(\boldsymbol{\theta}_i - \boldsymbol{\mu}_i) \tag{3.31}$$

Then we can define loss functions

$$L_i = (y - \boldsymbol{\theta}_i^T\mathbf{x}_i)^2 + (\boldsymbol{\theta}_i - \boldsymbol{\mu}_i)^T\mathbf{C}_{\mathrm{B},i}^{-1}(\boldsymbol{\theta}_i - \boldsymbol{\mu}_i) \tag{3.32}$$

noting that one of the loss functions must also include the intercept term $\theta_{\mathrm{B},0}$.

With these loss functions we satisfy the definition of separability as desired. □

In conclusion, when incorporating prior information to gaussian linear models, we also need parameters to be block independent of each other. This is an intuitive result, as dependent parameters . The term $(\boldsymbol{\theta}_{\mathrm{B}} - \boldsymbol{\mu}_{\mathrm{B}})^T\mathbf{C}_{\mathrm{B}}^{-1}(\boldsymbol{\theta}_{\mathrm{B}} - \boldsymbol{\mu}_{\mathrm{B}})$ is a type of Generalized Tikhonov

Regularization, which uses the prior information as a regularization term for the learning objective.

3.2.3 Uncorrellated Ensembles are Model Parallel

The example of PCR belies an idea of how to express feature space decompositions for MSE. Let $\{\phi_i : \mathbb{X} \to \mathbb{R}^k\}_{i=1}^m$ be a set of morphisms which are uncorellated:

$$\mathrm{E}_X(\phi_i(\mathbf{x})^T \phi_j(\mathbf{x})) = 0,\ i \neq j \tag{3.33}$$

Then the family of morphisms with structure:

$$\begin{aligned} \mathcal{ML}_\phi : (\mathbb{X}, \mathbb{Y}, F_\phi = \sum_{i=1}^m \theta_i \phi_i(\mathbf{x}) + \theta_0, P(\theta_0, \theta_1, \theta_2, ...\theta_m) = 1, \dots \\ L_\phi = (\mathbf{y} - \sum_{i=1}^m \theta_i \phi_i(\mathbf{x}))^T (\mathbf{y} - \sum_{i=1}^m \theta_i \phi_i(\mathbf{x}))) \end{aligned} \tag{3.34}$$

is separable.

To see this, define $\mathbf{z}_i = \phi_i(\mathbf{x})$ for $i = 1, ..., m$, and the vector

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_m \end{bmatrix} \tag{3.35}$$

Then the learning morphism takes the form

$$F_\phi = \sum_{i=1}^{k} \theta_i \mathbf{z}_i + \theta_0 = \begin{bmatrix} \boldsymbol{\theta} & \theta_0 \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix} \tag{3.36}$$

which satisfies the idea of Eq. 3.5 and since $\mathbf{z}$ are uncorrelated by definition, this workflow is separable.

In the example of PCR, each ϕ_i is the vector $(\mathbf{x} - \boldsymbol{\theta}_0^*)\boldsymbol{\theta}_{1,i}^*$, where $\boldsymbol{\theta}_{1,i}^*$ are the columns of the principal component matrix defined in section 3.1.2 and $\boldsymbol{\theta}_0^* = E_X(\mathbf{x})$ is the mean of $\mathbf{x}$.

From this idea, if a learning morphism can be expressed via an orthogonal decomposition. For scalars, decompositions such as Fourier series or wavelets can be used. Because the orthogonality is taken with respect to the expected value, we can also use sequences of orthogonal polynomials as basis functions for different distributions.

For example, let x be a realization of a standard normal random variable. The univariate hermite polynomials:

$$\phi_0 = 1 \tag{3.37}$$

$$\phi_1 = x \tag{3.38}$$

$$\phi_2 = x^2 - 1 \tag{3.39}$$

$$\phi_3 = x^3 - 3x \tag{3.40}$$

$$\vdots \tag{3.41}$$

are orthogonal with respect to the standard normal distribution. Therefore, if our workflow first centers the data, then divides by the standard deviation, any learning morphism which can be expressed as a linear combinations of Hermite Polynomials is separable.

Similarly, the Legendre polynomials are orthogonal with respect to a uniform distribution on the interval $[0, 1]$ and the Laguerre polynomials are orthogonal with respect to the Gamma distribution.

For multivariate data, we can still leverage the idea of function equivalence. For example, fuzzy logic systems have been shown to be functionally equivalent to neural nets, CART, or regressions [51][111]. For MSE problems we require functional equivalence with an orthogonal decomposition. This can be found for real valued data via an eigenvector decomposition such as PCA. PCA creates a set of centered and uncorrelated gaussian variables, which we will denote $z_1, z_2, ..., z_h$. In [56], an orthogonal extension to the principal components is developed showing that

$$\mathrm{E}((z_i^2 - \mathrm{E}(z_i^2))(z_j^2 - \mathrm{E}(z_j^2)) = 0 \tag{3.42}$$

for $i \neq j$. This extension can be extended to form a multivariate power series in terms of the principal components, which will allow us to define a set of basis functions based on $(z_1, z_2, ..., z_h, z_1^2, z_1 z_2, ...)$.

3.2.4 Ensemble Models Built with Randomization are Model, Data, and Feature Parallel

In Equation 3.34 we defined an orthogonal decomposition with basis functions defined on the entire input space. Some ensemble workflows, such as random forests, use randomization

strategies to build ensembles of learning morphisms trained on subsets of the input space. Ensembles are used in machine learning workflows to improve the performance compared to learning a single MLM.

First, define a set of input subspaces: $\{\mathbb{X}_i\}_{i=1}^m \subset \mathbb{X}$. Let $F_i : \mathbb{X}_i \to \mathbb{Y}$, for $i = 1, ..., m$ be a set of learning morphisms with corresponding parameters $\boldsymbol{\theta}_i$. Then if

$$\mathrm{E}_{X_i X_j}(F_i(\mathbf{x}_i; \boldsymbol{\theta}_i) F_j(\mathbf{x}_j; \boldsymbol{\theta}_j)) = 0 \tag{3.43}$$

for $i \neq j$, then an MLM with structure:

$$(\mathbb{X} = \cup_{i=1}^m \mathbb{X}_i, \mathbb{Y}, F = \sum_{i=1}^m F_i(\mathbf{x}_i; \boldsymbol{\theta}_i), P(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_m) = 1, L = (\mathbf{y} - F)^T(\mathbf{y} - F)) \tag{3.44}$$

is separable by the logic in Equations A.2-A.11. This represents an ensemble of uncorrelated learning morphisms, whose parameters need not appear linearly. The challenge is now to determine the subsets $\mathbb{X}_i$.

In section 3.1.2, we used an orthogonal transformation to create an orthogonal input space, which naturally decorrelated the least squares linear MLM. A diffferent way to decorrelate the morphisms in the workflow in Equation 3.44 is to use randomization to define the subsets $\mathbb{X}_i$. For example, bagging, or "bootstrap aggregating" randomly samples the training data with replacement to create m training sets, trains a workflow on each training set, and then averages the outputs with the goal of lowering the variance of the ensemble [14]. The Random Subspace method [47] randomly samples the features with replacement to create an ensemble with the goal of decorrelating the outputs of each member.

One of the most famous ensemble models, random forests [15], employs both bagging and the random subspace method to create an ensemble of decision or regression trees. There are many variants in random forests, for example Extremely Randomized Trees [37] and Rotation Forests [87], but the key idea is that proper randomization to decorrelate the ensemble will lead to better performance than a single decision tree [60].

The trees in a random forest are trained independently, and can be expressed as MLM's. First, denote the projection of a vector $\mathbf{x} \in \mathbb{X}$ to subspace $\mathbb{X}_i$ as $\mathbf{x}_i$. These subsets incorporate projection into a lower dimensional feature space as a result of the random subspace method. Next, let $\{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^m$ represent n_i samples drawn with replacement from the training data $(\mathbf{X}, \mathbf{Y})$ with features corresponding to the subspace $\mathbb{X}_i$. Let $F_i : \mathbb{X}_i \to \mathbb{Y}$ represent a decision tree acting on the subspace $\mathbb{X}_i$ with splitting parameters $\boldsymbol{\theta}_i$. Assuming we have no prior information on the splitting parameters, each tree is an MLM with structure:

$$\begin{aligned} \mathcal{ML}_{\mathrm{RF},i} &= (\mathbb{X}_i, \mathbb{Y}, F_i(\mathbf{x}_i; \boldsymbol{\theta}_i), P(\boldsymbol{\theta}_i) = 1, \dots \\ \bar{R}_i = \frac{1}{n_i} \sum_{(\mathbf{x},\mathbf{y}) \in (\mathbf{X}_i, \mathbf{Y}_i)} L(\mathbf{y}, F_i) &= \frac{1}{n_i} \sum_{(\mathbf{x}_i,\mathbf{y}) \in (\mathbf{X}_i, \mathbf{Y}_i)} ||\mathbf{y} - F_i(\mathbf{x}_i; \boldsymbol{\theta}_i)||_2^2) \end{aligned} \tag{3.45}$$

The random forest is then the average of the outputs of the decision tree MLM's

$$\hat{\mathbf{y}} = F_{\mathrm{RF}}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} F_i(\mathbf{x}_i; \boldsymbol{\theta}_i^*) \tag{3.46}$$

or equivalently

$$m\hat{y} = \sum_{i=1}^{m} F_i(\mathbf{x}_i; \boldsymbol{\theta}_i^*) \tag{3.47}$$

If the outputs of F_i are uncorrelated by the randomization, then they solve the separable sub problems from Equation 3.44. In practice, the correlation between the output of the

trees will depend on the efficacy of the randomization and correlation between the elements of the input space, and will not be truly zero. However, randomization as a strategy is still used to both improve ensemble performance and provide inherent parralelization, and in the ideal situation will be equivalent to a separable workflow.

3.3 Chapter Review

In this chapter we expanded the MLM framework in order to define separability of MLM's. We then examined the separability of MLM's trained on MSE loss, starting with linear models, extending to uncorrelated orthogonal decompositions, and ending with randomized ensembles. For MSE problems, engineering orthogonality in the input space or learning morphism is necessary to separate the workflow's loss function, allowing parallel parameter optimization. Finally, we explored randomization as a strategy originally developed to improve MSE in workflows

Separable linear least squares MLM's as defined in Theorem 1 are an example of feature parallel learning, as independence allows us to train regressions on blocks of features in parallel. We examined the case of Principal Component Regression as a workflow defined as a sequence of output compositions, and proved it was asympotically equivalent to a linear regression with an intercept term. We also examined the learning morphisms of PCR as a sequence of structural compositions, and showed that the optimal points of PCR is a critical point of the joint loss function. Then we discussed the set of use cases where each option might be chosen. When the prior is informative, parameter independence was needed in addition to the conditions of Theorem 1. This result is particularly powerful because we can approximate a wide variety of functions using power series of Gaussian random variables.

We extended this result in Equation 3.34 to separability for linear combinations of orthogonal basis functions. This is a model parallel implementation as the basis functions are defined over the entire input space. Examples of basis functions are Fourier Series or orthogonal polynomials, including orthogonal extensions of the principal component space. The final approach we explored is the use of randomization to build workflows such as random forests on subsets of the input space and training data. This combines aspects of model, data, and feature parallel implementations.

Once a separable MLM is identified, the challenge is then to design a workflow to satisfy any necessary conditions, such as a decorrelated input space for Theorem 1. Separability is one of many tools for scalable learning, and other strategies may scale more effectively. For example, the process of finding an orthogonal decomposition such as PCA requires roughly the same order of computations as computing the matrix inverse in linear regression, and other decompositions of multivariate data may be computationally expensive. Further, there is no guarantee that a workflow has an equivalent separable representation, or that it can be found. Finally, the idea of separability does not address the question: What is the best workflow for the problem at hand? Instead it is one of many tools used to scale workflows that have already been chosen.

The results in this chapter give hints towards linear algebra for MLM's. Ideally, we would construct a basis set of uncorrelated workflows across the same input and output space. Then, if MSE is the main objective, we know that adding the outputs is the same as training an MLM on the addition of the learning morphisms. The key question then becomes, what is the cardinality of the space of workflows defined by linear combinations of uncorrelated learning morphisms? To answer this, in future work we will attempt to leverage concepts such as Rademacher Complexity, which is a generalization of the VC dimension. This,

combined with ideas from functional analysis should yield a class of MLM's which belong to a vector space of naturally parallelizable MLM's.

Chapter 4

Performance Bounds and Generalization Error

In this chapter we will use MLM's to connect the idea of generalization error in Machine Learning to the concept of the Cramer-Rao Lower Bound in estimation. Given a loss function L, the *generalization error* is defined as:

$$\mathrm{GE}_{\mathrm{L}}(F) = \mathrm{E}_{X,Y}(L(Y, F(X; \boldsymbol{\theta}^*))) \tag{4.1}$$

In estimation, we are concerned with parameters $\boldsymbol{\alpha}$, which parameterize the true underlying distribution $P(X, Y; \boldsymbol{\alpha})$. Recall that the observations are given by the set $(\mathbf{X}, \mathbf{Y})$, and assume that they are n i.i.d. samples. The Cramer-Rao Bound on estimators $\hat{\alpha}(\mathbf{X}; \mathbf{Y})$ provides a lower bound on the MSE:

$$\mathrm{E}_{X,Y}((\hat{\alpha} - \boldsymbol{\alpha})^2) \geq \frac{1}{n}\mathbf{I}^{-1}(\boldsymbol{\alpha}) \tag{4.2}$$

where $\mathbf{I}(\boldsymbol{\alpha}) = -\mathrm{E}_{X,Y}[\frac{\partial}{\partial \boldsymbol{\alpha}\boldsymbol{\alpha}^T} \log(P(X, Y; \boldsymbol{\alpha}))]$ is the Fisher Information Matrix.

In this Chapter we will bound the generalization error of an MLM with MSE loss using the underlying parameters $\boldsymbol{\alpha}$. This leverages the Bias-Variance Decompsition for MSE. For MLM's with other loss functions, we can still apply this bound to characterize the variance of

an MLM around the best possible workflow. This approach treats an MLM as an "estimator" of the best possible workflow. Finally, we derive a bound for a composition of MLM's in terms of the lower bounds of MLM's in the composition.

4.1 The "Bayes" Workflow is the MLM with the Lowest Generalization Error

In order to bound the error of an MLM, we must first establish the best possible workflow, which has the lowest possible error. To do this, we will pointwise minimize the loss function. Following the nomenclature of Louppe [60], define:

Definition 8. Let $(\mathbb{X}, \mathbb{Y})$ be an input/output space, and let L be a loss function. Let $\mathbb{F} = \{F : \mathbf{X} \to \mathbb{Y}\}$ be the set of morphisms from $\mathbb{X}$ to $\mathbb{Y}$. Define a set of MLM's with common input space, output space, and loss function:

$$\mathbb{M} = \{\mathcal{ML} : (\mathbb{X}, \mathbb{Y}, F \in \mathbb{F}, P(\boldsymbol{\theta}_F), L)\} \tag{4.3}$$

The *Bayes MLM* is the MLM with the lowest generalization error:

$$GE_{\mathrm{L}}(F_{\mathrm{B}}) \leq GE_{\mathrm{L}}(F) \quad \forall F \in \mathbb{F} \tag{4.4}$$

We can define the learning morphism by pointwise minimization:

$$F_{\mathrm{B}} = arg\min_{\hat{\mathbf{y}}} \mathrm{E}_{X,Y}(L(\mathbf{y}, \hat{y})) = \mathrm{E}_X(\mathrm{E}_{Y|X}(L(\mathbf{y}, \hat{\mathbf{y}}))) \tag{4.5}$$

For most loss functions, the minimization occurs at the point $L(\mathbf{y}, \mathbf{y})$. So

$$\hat{\mathbf{y}} = \mathrm{E}_{Y|X}(\mathbf{y}|\mathbf{x}) = \Gamma_{Y|X}(\mathbf{x}_1; \boldsymbol{\alpha}) \tag{4.6}$$

is the best prediction we can make as a function of the input $\mathbf{x}$. This is parameterized by $\boldsymbol{\alpha}$, which are the underlying parameters of the true distribution $P(X, Y; \boldsymbol{\alpha})$. Then the Bayes Model has structure:

$$\mathcal{ML}_{\mathrm{B}} : (\mathbb{X}, \mathbb{Y}, F_{\mathrm{B}}, P(\boldsymbol{\alpha}), L) \tag{4.7}$$

If we know the full distribution, then the Bayes MLM is the best workflow for a given loss function, and we we call $GE_{\mathrm{L}}(F_{\mathrm{B}}) = \mathrm{E}_{X,Y}(L(\mathbf{y}, \Gamma_{Y|X}))$ the "Bayes Error". For mean-squared error, the Bayes error is:

$$GE_L(F_{\mathrm{B}}) = \mathrm{E}_{X,Y}((\mathbf{y} - \Gamma_{Y|X})(\mathbf{y} - \Gamma_{Y|X})^T) = \mathrm{E}_X[\mathrm{E}_{Y|X}[(\mathbf{y} - \Gamma_{Y|X})(\mathbf{y} - \Gamma_{Y|X})^T]] = \tag{4.8}$$

$$\mathrm{E}_X[\mathrm{var}(Y|X)] \tag{4.9}$$

Which is simply the natural variance of $\mathbf{y}$ around the conditional mean $\Gamma_{Y|X}$. This term is the well known irreducible error in the Bias-Variance Decomposition of MSE [60].

4.2 Bounding MLM MSE

In this section, we apply the main result of Nayak [69] to bound the MSE of an MLM based on the underlying distribution. Let $\mathcal{ML}$ have structure $(\mathbb{X}, \mathbb{Y}, F, P(\boldsymbol{\theta}_{\mathrm{F}}), L = ||\mathbf{y} - F||_2^2)$.

Then define the *bias* of $\mathcal{ML}$ as

$$\mathbf{B}_{X,Y}(F) = \mathrm{E}_{X,Y}[\mathbf{y} - F(\mathbf{x}; \boldsymbol{\theta}^*)] \tag{4.10}$$

The bias is simply the discrepancy between the true value of $\mathbf{y}$ and the prediction $F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}})$.

The MSE can be decomposed as:

$$GE_{\mathrm{MSE}}(F) = \mathrm{trace}\left[\mathrm{E}_{X,Y}((\mathbf{y} - F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}}))(\mathbf{y} - F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}}))^T)\right] = \tag{4.11}$$

$$\mathrm{trace}\left(\mathrm{E}_X[\mathrm{var}(Y|X)] + \mathbf{B}_{X,Y}(F)\mathbf{B}_{X,Y}(F)^T + \mathrm{var}_X[F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}}) - \Gamma_{X|Y}]\right) \tag{4.12}$$

This is the Bias-Variance Decomposition. If $\mathbf{A}$ and $\mathbf{B}$ are matrices, let $\mathbf{A} \succcurlyeq \mathbf{B}$ denote that $\mathbf{A} - \mathbf{B}$ is positive semidefinite. This is called the Loewner order. Then applying the main result from Nayak [69], we can bound the MSE:

$$\mathrm{var}_X[F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}}) - \Gamma_{X|Y}] \succcurlyeq \tag{4.13}$$

$$\left(\frac{\partial \mathbf{B}_{X,Y}(F)}{\partial \boldsymbol{\alpha}} + \mathrm{E}_X[\frac{\partial \Gamma_{Y|X}(\mathbf{x})}{\partial \boldsymbol{\alpha}}]\right)^T \mathbf{I}(\boldsymbol{\alpha})^{-1} \left(\frac{\partial \mathbf{B}_{X,Y}(F)}{\partial \boldsymbol{\alpha}} + \mathrm{E}_X[\frac{\partial \Gamma_{Y|X}(\mathbf{x})}{\partial \boldsymbol{\alpha}}]\right) \tag{4.14}$$

Then the MSE matrix is lower bounded:

$$\mathrm{E}_{X,Y}((\mathbf{y} - F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}}))(\mathbf{y} - F(\mathbf{x}; \boldsymbol{\theta}^*_{\mathrm{F}}))^T) \succcurlyeq \tag{4.15}$$

$$\mathrm{E}_X[\mathrm{var}(Y|X)] + \mathbf{B}_{X,Y}\mathbf{B}^T_{X,Y} +$$

$$\left(\frac{\partial \mathbf{B}_{X,Y}(F)}{\partial \boldsymbol{\alpha}} + \mathrm{E}_X[\frac{\partial \Gamma_{Y|X}(\mathbf{x})}{\partial \boldsymbol{\alpha}}]\right)^T \mathbf{I}(\boldsymbol{\alpha})^{-1} \left(\frac{\partial \mathbf{B}_{X,Y}(F)}{\partial \boldsymbol{\alpha}} + \mathrm{E}_X[\frac{\partial \Gamma_{Y|X}(\mathbf{x})}{\partial \boldsymbol{\alpha}}]\right) = \tag{4.16}$$

$$\mathbf{L}\mathbf{B}_{X,Y} \tag{4.17}$$

and the MSE is bounded by:

$$GE_{\text{MSE}}(F) \geq \text{trace}(\mathbf{L}\mathbf{B}_{X,Y}) \tag{4.18}$$

The learning morphism F and parameters $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ interact in the bias function $\mathbf{B}_{X,Y}$, and the other terms only depend on $\boldsymbol{\alpha}$. Therefore, to reduce the bound we are incentivized to reduce the bias of the workflow. This generally has the unfortunate consequence of increasing $\text{cov}_X[F(\mathbf{x};\boldsymbol{\theta}^*) - \Gamma_{X|Y}]$.

This bound is analogous to the Cramèr-Rao Lower bound for estimators of $\mathbf{y}$. To briefly illustrate how it is derived, let $\alpha_i \in \boldsymbol{\alpha}$, and denote the dimension of $\boldsymbol{\alpha}$ as d_α. Define the *score function* as

$$S_i = \frac{\partial \log(P(X,Y;\boldsymbol{\alpha}))}{\partial \alpha_i} \tag{4.19}$$

for $i = 1, ..., d_\alpha$. The score functions have zero mean and covariance matrix given by the fisher information $\mathbf{I}(\boldsymbol{\alpha})$. Further, the covariance between a score function and the term $F - \Gamma_{Y|X}$ is:

$$\text{cov}[F - \Gamma_{Y|X}, S_i] = \frac{\partial \mathbf{B}_{X,Y}(F)}{\partial \alpha_i} + \frac{\partial \Gamma_{Y|X}}{\partial \alpha_i} \tag{4.20}$$

Let $\mathbf{S} = [S_1, S_2, ..., S_{d_\alpha}]$. Then the bound follows from the positive semidefiniteness of:

$$\text{cov}(F - \Gamma Y|X, \mathbf{S}) = \begin{bmatrix} \text{var}_X[F(\mathbf{x};\boldsymbol{\theta}^*) - \Gamma_{X|Y}] & \text{cov}[F - \Gamma_{Y|X}, \mathbf{S}] \\ \text{cov}[F - \Gamma_{Y|X}, \mathbf{S}]^T & \mathbf{I}(\boldsymbol{\alpha}) \end{bmatrix} \tag{4.21}$$

Other covariance bounds also use this score function formulation. For the Barankin bound, we select a set of test points $\boldsymbol{\alpha}_j$ for $j = 1, ..., k$. Then the score function for the Barankin

Bound is

$$S_{\mathrm{B},j} = \frac{P(X,Y;\boldsymbol{\alpha}_j) - P(X,Y;\boldsymbol{\alpha})}{P(X,Y;\boldsymbol{\alpha})} \tag{4.22}$$

The covariance term for the barankin bound will be different from the S_j of course, but the Barankin bound is also tighter. For the rest of this chapter, we will assume we are using bounds of the same type as Equation 4.14.

For MSE, we have connected the bound on the generalization error of workflows with a lower bound based. Other loss functions, however, do not possess such convenient decompositions. The term $\mathrm{var}[F - \Gamma_{Y|X}]$ does not depend on the loss function, and can act as a metric to compare other workflows to the Bayes workflow. If this covariance is low, then the learning morphism F is consistently making predictions close to the Bayes Model.

4.3 Example Bounds for Individual MLM's

In this section we present three motivating examples of error bounds: Centering, PCA, and Ordinary Least-Squares Linear Regression.

To visualize the bounds, for $j = 1, .., 1000$, we use two loops. The inner loop takes as input a realization of training data, $(\mathbf{X}_j, \mathbf{Y}_i) = (\mathbf{x}_i, y_i)_{i=1}^n$, and varies n from 50 to 500. In this case the dimension of $\mathbf{x}_i$ is $d_x = 10$, and we will treat the means as "unknown" and covariance structure as known. Then we perform centering, pca, and linear regression. For these MLM's we compute the lower bound, and estimate the covariance of the error from a set of test data.

In the outer loop, we generate the realizations of training and testing data from a distribution with mean $\begin{bmatrix} \boldsymbol{\alpha}_X \\ \alpha_y \end{bmatrix}$ and covariance structure $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{XX} & \boldsymbol{\Sigma}_{XY} \\ \boldsymbol{\Sigma}_{XY}^T & \boldsymbol{\Sigma}_{YY} \end{bmatrix}$. Then for n samples the fisher information is $\mathbf{I}(\boldsymbol{\alpha}_x, \alpha_y) = n\boldsymbol{\Sigma}^{-1}$.

Then we plot trace(cov[error]) and trace(cov[Bound]) as a visualization of how each MLM behaves.

This process is repeated by sampling the same covariance structure multiple times and averaging to estimate the true MSE, while the covariance bounds remain unchanged. To plot, we computed the trace of the MSE matrix and corresponding lower bound matrix in Figures 4.1, 4.2, and 4.3.

4.3.1 Centering

Let $\mathbb{X} = \mathbb{R}^m$ and have elements distributed as: $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\alpha}_X, \sigma_{XX})$, where $\boldsymbol{\alpha}_X$ is the unknown mean and Σ_{XX} is a known covariance matrix. Next, assume we wish to center our data, so let $\mathbf{x}_2 = \mathbf{x}_1 - \boldsymbol{\alpha}_X$. Recall that the centering MLM has structure:

$$\mathcal{ML}_{\mathrm{C}} : (\mathbb{X}, \mathbb{Y}, F_{\mathrm{C}} = \mathbf{x} - \boldsymbol{\theta}_{\mathrm{C}}, P(\boldsymbol{\theta}_{\mathrm{C}}) = 1, L_{\mathrm{C}} = ||\mathbf{x} - \boldsymbol{\theta}_{\mathrm{C}})||_2^2 \tag{4.23}$$

Denote the set of training data as $\{\mathbf{x}_{1,i}\}_{i=1}^n$ The optimal parameters are given by the sample mean: $\boldsymbol{\theta}_C^* = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_{1,i}$.

The MLM is unbiased:

$$\mathbf{B}_{X_1,X_2} = \mathrm{E}_{X_1,X_2}(\mathbf{x}_2 - F_{\mathrm{C}}(\mathbf{x}_1)) = \tag{4.24}$$

$$\mathrm{E}(\boldsymbol{\alpha}_X - \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_{1,i}) = 0 \tag{4.25}$$

The irreducible error is also zero:

$$\mathrm{E}_{X_1}(\mathrm{var}(\mathbf{x}_2|\mathbf{x}_1)) = \tag{4.26}$$

$$\mathrm{E}_{X_1}(\mathrm{var}(\mathbf{x}_1 - \boldsymbol{\alpha}_x|\mathbf{x}_1)) = \mathrm{E}_{X_1}(0) = 0 \tag{4.27}$$

The phenomenon of zero irreducible error occurs when there is a direct functional correspondence between the two spaces. Here all we are doing from $\mathbb{X}_1$ to $\mathbb{X}_2$ is subtracting the mean, so if we know the mean, then we have completely perfect information and there is no error.

The Bayes MLM is simply $\Gamma_{X_2|X_1} = \mathbf{x}_1 - \boldsymbol{\alpha}_X$. Then the derivative of the Bayes MLM with respect to $\boldsymbol{\alpha}_x, \alpha_y$ is:

$$\frac{\partial \Gamma^T_{X_2|X_1}}{\partial[\boldsymbol{\alpha}_x, \alpha_y]} = \begin{bmatrix} -\mathbb{I}_{d_x \times d_x} & \mathbf{O}_{d_x \times 1} \end{bmatrix} \tag{4.28}$$

Then from Equation 4.14 the lower bound is given by:

$$\mathrm{E}((\mathbf{x}_2 - F_{\mathrm{C}})(\mathbf{x}_2 - F_{\mathrm{C}})^T) \succcurlyeq 0 + 0 + \begin{bmatrix} -\mathbb{I}_{d_x \times d_x} & \mathbf{O}_{d_x \times 1} \end{bmatrix} \mathbf{I}(\boldsymbol{\alpha}_x, \alpha_y)^{-1} \begin{bmatrix} -\mathbb{I}_{d_x \times d_x} \\ \mathbf{O}_{d_x \times 1} \end{bmatrix} = \tag{4.29}$$

$$\frac{1}{n} Sigma_{XX} \tag{4.30}$$

This is a classical result and the bound is visualized in Figure 4.1.

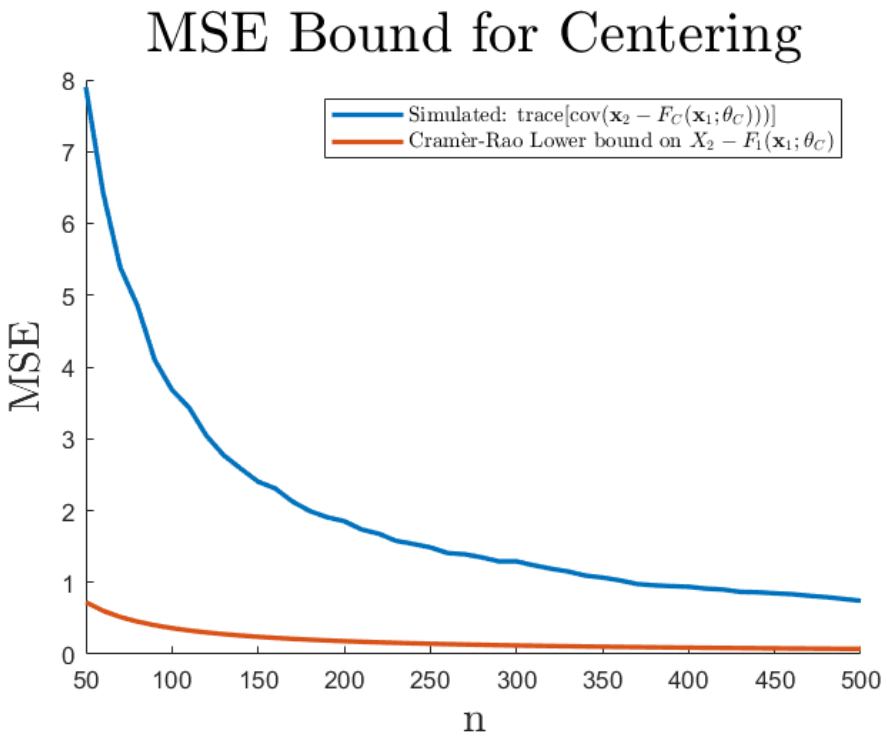


Figure 4.1: The Cramèr-Rao lower bound for the sample mean. As n increases, the error goes down and the MSE between $\mathbf{x}_2$ and $F_{\mathrm{C}}(\mathbf{x}_1; \boldsymbol{\theta}^*_{\mathrm{C}})$ approaches 0.

4.3.2 Principal Component Analysis

Let $\mathbb{X}_2$ be the centered version of $\mathbb{X}_1$ as the previous section. Let $\mathbf{\Theta_P}$ represent the space of orthonormal matrices of size $m \times m$, and denote elements as Θ. Denote the eigendecomposition of $\mathbf{\Sigma}_{XX}$ as $\mathbf{P\Lambda P}$ for orthonormal matrix $\mathbf{P}$ and diagonal matrix $\mathbf{\Lambda}$. PCA attempts to estimate $\mathbf{x}_3 = \mathbf{P}^T\mathbf{x}_2$. PCA is an MLM with Structure:

$$\mathcal{ML}_{\mathrm{P}} = (\mathbb{R}^m, \mathbb{R}^m, \Theta^T\mathbf{x}, P_{\mathbf{\Theta_P}}(\Theta_{\mathbf{P}}) = 1, \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \Theta_{\mathbf{P}}^T\Theta_{\mathbf{P}}\mathbf{x}_i)^2) \tag{4.31}$$

The optimal parameters are the eigenvalues of the sample covariance matrix, which we denote as $\Theta^*_{\mathbf{P}}$

The bayes model is $\Gamma_{X_3|X_2} = \mathbf{P}^T\mathbf{x}_2$, which again has 0 irreducible error because there is a direct functional relationship between $\mathbf{x}_3$ and $\mathbf{x}_2$. The bias is also zero:

$$\mathrm{E}_{X_2,X_3}(\mathbf{x}_3 - \Theta) = (\mathbf{P}^T - \Theta_{\mathbf{P}}^T)\mathrm{E}_{X_2}(\mathbf{x}_2) = \mathbf{0} \tag{4.32}$$

The irreducible error is

$$\mathrm{E}_{X_2}(\mathrm{var}[\mathbf{x}_3|\mathbf{x}_2]) = \mathrm{E}_{X_2}(0) = 0 \tag{4.33}$$

because the conditional variance is zero

$$\mathrm{var}[\mathbf{x}_3|\mathbf{x}_2] = \mathbf{\Lambda} - \mathbf{P}^t\mathbf{\Sigma}_{XX}\mathbf{\Sigma}_{XX}^{-1}\mathbf{\Sigma}_{XX}\mathbf{P} = \tag{4.34}$$

$$\mathbf{\Lambda} - \mathbf{P}^T\mathbf{\Sigma}_{XX}\mathbf{P} = \mathbf{\Lambda} - \mathbf{\Lambda} = \mathbf{O} \tag{4.35}$$

The derivative of the bayes model is

$$\frac{\partial \Gamma^T_{X_3|X_2}}{\partial[\boldsymbol{\alpha}_x, \alpha_y]} = \begin{bmatrix} -\mathbf{P} & \mathbf{O}_{d_x \times 1} \end{bmatrix} \tag{4.36}$$

Then by Equation 4.14 the lower bound for PCA is:

$$\mathrm{E}((\mathbf{x}_3 - F_{\mathrm{P}})(\mathbf{x}_3 - F_{\mathrm{P}})^T) \succcurlyeq 0 + 0 + \begin{bmatrix} -P & \mathbf{O}_{d_x \times 1} \end{bmatrix} \mathbf{I}(\boldsymbol{\alpha}_x, \alpha_y)^{-1} \begin{bmatrix} -\mathbf{P}^T \\ \mathbf{O}_{1 \times d_x} \end{bmatrix} = \tag{4.37}$$

$$\frac{1}{n}\boldsymbol{\Lambda} \tag{4.38}$$

4.3.3 Linear Regression

For linear regression, first assume we are working with the space $\mathbb{X}_3$ which has zero mean and covariance matrix $\boldsymbol{\Lambda}$. The joint distribution of $\mathbb{X}_3$ and $\mathbb{Y}$ is found by applying centering and PCA to original joint distribution. so

$$P(X_3, Y) \sim N(\begin{bmatrix} \mathbf{0} \\ \alpha_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Lambda} & \mathbf{P}^T\boldsymbol{\Sigma}_{XY} \\ \boldsymbol{\Sigma}^T_{XY}\mathbf{P} & \boldsymbol{\Sigma}_{YY} \end{bmatrix}) \tag{4.39}$$

then the bayes model is the conditional expectation:

$$\Gamma_{Y|3} = \alpha_y + \boldsymbol{\Sigma}^T_{XY}\mathbf{P}^T\boldsymbol{\Lambda}^{-1}\mathbf{x}_3 \tag{4.40}$$

and the derivative is:

$$\frac{\partial \Gamma^T_{Y|X_3}}{\partial[\boldsymbol{\alpha}_x, \alpha_y]} = \begin{bmatrix} -\boldsymbol{\Sigma}^T_{XY}\mathbf{P}^T\boldsymbol{\Lambda}^{-1} & 1 \end{bmatrix} \tag{4.41}$$

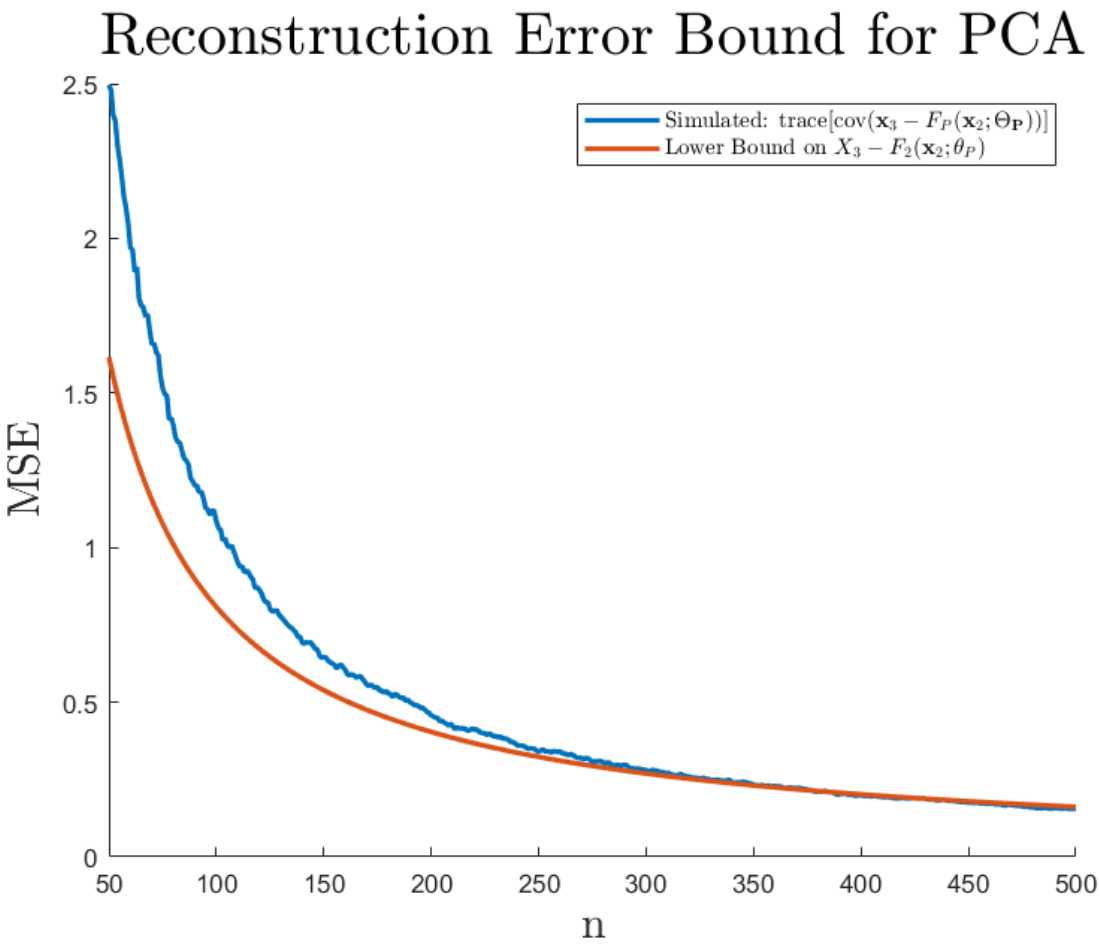


Figure 4.2: Sometimes the reconstruction error smoothly and swiftly converges to the lower bound.

The irreducible error is the conditional variance:

$$\mathrm{E}_{X_3}(\mathrm{var}[y|\mathbf{x}_3]) = \mathrm{E}_{X_3}(\boldsymbol{\Sigma}_{YY} - \boldsymbol{\Sigma}_{XY}^T\boldsymbol{\Sigma}_{XX}^{-1}\boldsymbol{\Sigma}_{XY}) = \tag{4.42}$$

$$\boldsymbol{\Sigma}_{YY} - \boldsymbol{\Sigma}_{XY}^T\boldsymbol{\Sigma}_{XX}^{-1}\boldsymbol{\Sigma}_{XY} \tag{4.43}$$

Again the bias is zero, so the bound is given as:

$$\mathrm{E}((y - F_{\mathrm{R}})(y - F_R)^T) \succcurlyeq \tag{4.44}$$

$$\boldsymbol{\Sigma}_{YY} - \boldsymbol{\Sigma}_{XY}^T\boldsymbol{\Sigma}_{XX}^{-1}\boldsymbol{\Sigma}_{XY} + 0 + \begin{bmatrix} -\boldsymbol{\Sigma}_{XY}^T\mathbf{P}^T\boldsymbol{\Lambda}^{-1} & 1 \end{bmatrix} \mathbf{I}(\boldsymbol{\alpha}_x, \alpha_y)^{-1} \begin{bmatrix} -\boldsymbol{\Lambda}^{-1}\mathbf{P}\boldsymbol{\Sigma}_{XY} \\ 1 \end{bmatrix} \tag{4.45}$$

The bound is visualized in Figure 4.3

4.4 Composition of Bounds

If we form the output composition of MLM's, $\mathcal{ML}_k \circ_O \cdots \circ_O \mathcal{ML}_2 \circ_O \mathcal{ML}_1$, it is natural to investigate how error from the first MLM propogates to the second. Define the error vector for each stage of the composition as:

$$\mathbf{e}_{i-1,i} = F_i(F_{i-2}(\cdots F_1(\mathbf{x}_1; \boldsymbol{\theta}_1))) - \mathbf{x}_i \tag{4.46}$$

Denote the Bayes MLM for each workflow as:

$$\Gamma_{i|i-1} = \mathrm{E}_{X_i|X_{i-1}}(\mathbf{x}_i|\mathbf{x}_{i-1}) \tag{4.47}$$

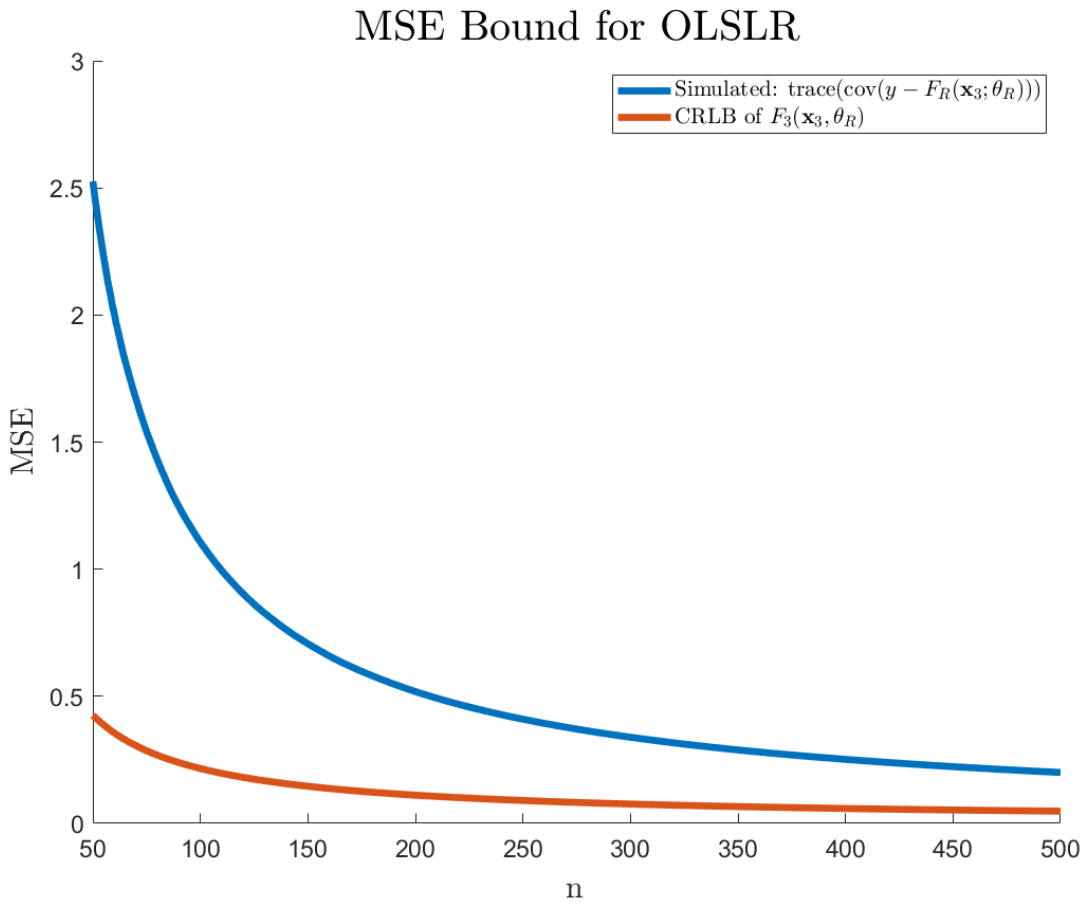


Figure 4.3: Lower bound using Equation 4.14 for OLSLR. As the number of training samples increases the MSE on the test set approaches the bound.

and the Bayes Error is $\mathbf{e}_{\Gamma_{i|i-1}} = \Gamma_{i|i-1} - \mathbf{x}_i$

We will consider the case where the error vectors and Bayes MLM's are Gaussian, and uncorrelated from step to step. Then we propose the following bound:

Proposition 1. Let $\mathcal{ML}_{\mathcal{W}} = \mathcal{ML}_k \circ_O \cdots \circ_O \mathcal{ML}_2 \circ_O \mathcal{ML}_1$ be a workflow. Assume that:

- The input spaces $\mathbb{X}_i$ for $i = 1, ..., k$ are spaces of gaussian random variables.
- The error vectors $\mathbf{e}_{i,i-1}$ and $\mathbf{e}_{j-1,j}$ are Gaussian and uncorrelated for $i \neq j$.
- The Error of the Bayes Models $\Gamma_{i|i-1}$ and $\Gamma_{j|j-1}$ are Gaussian and uncorrelated for $i \neq j$.

For $i = 2, .., k$ define:

- Lower bounds for each MLM given by Equation 4.14: $\text{cov}(\mathbf{x}_i - F_{i-1}(\mathbf{x}_{i-1}; \boldsymbol{\theta}_i^*)) \succcurlyeq \mathbf{LB}_{i-1,i}$
- $\mathbf{V}_i = \text{var}(\mathbf{x}_i | \mathbf{x}_{i-1})$.
- $\mathbf{R}_i = \text{var}(\mathbf{x}_i - F_{i-1} \circ F_{i-2} \cdots F_1) \succcurlyeq \mathbf{LB}_{i-1,i}$
- $\mathbf{E}_{F_{i-1}} = \text{E}(F_{i-1} \circ F_{i-2} \circ \cdots \circ F_1(\mathbf{x}_1; \boldsymbol{\theta}_i^*) - F_{i-1}(X_{i-1}; \alpha_{i-1}))$
- $\mathbf{D}_i = (\mathbb{I} + \frac{\partial B_{i-1,i}^T}{\partial X_i} + \frac{\partial \mathbf{E}_{F_{i-1}}^T}{\partial X_i})$
- $\mathbf{A}_i = \mathbf{V}_{i-1}^{-1} + \mathbf{D}_i \mathbf{R}_i^{-1} \mathbf{D}_i^T$
- $\mathbf{B}_{i-1} = -\text{E}(\frac{\partial \Gamma_{i|i-1}(X_{i-1})^T}{\partial X_{i-1}}) V_{i-1}^{-1}$
- $\mathbf{C}_{i-1} = \text{E}(\frac{\partial \Gamma_{i|i-1}(X_{i-1})^T}{\partial X_{i-1}}) V_{i-1}^{-1} \text{E}(\frac{\partial \Gamma_{i|i-1}(X_{i-1})^T}{\partial (X_{i-1})})^T$
- $\mathbf{I}(X_2) = \mathbf{LB}_{1,2}$

Then we can recursively bound the fisher information:

$$\mathrm{E}((F_{i-1} \circ F_{i-2} \cdots F_1 - X_i)(F_{i-1} \circ F_{i-2} \cdots F_1 - X_i)^T) \succcurlyeq\succcurlyeq \tag{4.48}$$

$$\mathbf{I}(X_i) \succcurlyeq (\mathbf{D}_i(\mathbf{L}\mathbf{B}_{i-1,i})^{-1}\mathbf{D}_i + \mathbf{V}_{i-1}^{-1} - \mathbf{B}_{i-1}^T(\mathbf{I}(X_{i-1}) + \mathbf{C}_{i-1})^{-1}\mathbf{B}_{i-1})^{-1} \tag{4.49}$$

4.4.1 Proof of Proposition

To prove this proposition, we frame a machine learning workflow as a nonlinear filtering problem. In this manner, the state is the "true" sequence of data transformations. A nonlinear filter is a system consisting of two parts:

$$X_{i+1} = G_i(X_i, \mathbf{v}_i) \tag{4.50}$$

$$Z_i = H_i(X_i, X_{i-1}, ..., X_1, Z_{i-1}, ..., Z_1, \mathbf{e}_i) \tag{4.51}$$

Here X_i is a "hidden" state, with G_i representing the process of transitioning from one state to another. Z_i is an observation of this process as a function of the current and previous states, and the current and previous observations. $\mathbf{v}_i$ and $\boldsymbol{\epsilon}_i$ are noise vectors with zero mean and known covariance matrices. To simplify this, we assume that H_i only depends on the current and directly previous state and observations:

$$H_i = H_i(X_i, X_{i-1}, Z_{i-1}, \boldsymbol{\epsilon}_i) \tag{4.52}$$

Then we can define a joint probability distribution between the state and observations as:

$$P(\mathbf{X}, \mathbf{Z}) = P(X_1)\prod_{i=2}^{k} P(Z_i|X_i, Z_{i-1}, X_{i-1})\prod_{j=2}^{k} P(X_j|X_{j-1}) \tag{4.53}$$

This distribution states that X_j only depends on the previous state, and Z_i only depends on X_i, X_{i-1}, Z_{i-1}. For $i, j = 1, ..., k$, $\mathbf{X}$ be the vector containing states X_j and $\mathbf{Z}$ containing observations Z_i. Define the fisher information as

$$\mathbf{I}(\mathbf{X}) = -\mathrm{E}(\frac{\partial^2 \log P(\mathbf{X}, \mathbf{Z})}{\partial \mathbf{X} \partial \mathbf{X}^T}) \tag{4.54}$$

Then if $\hat{\mathbf{X}}(\mathbf{Z})$ is an estimator of $\mathbf{X}$, we know that

$$\mathrm{E}((\hat{\mathbf{Z}} - \mathbf{X})(\hat{\mathbf{Z}} - \mathbf{X})^T) \succcurlyeq \mathbf{I}(\mathbf{X})^{-1} \tag{4.55}$$

Futher, we can bound individual states by finding the information submatrix which we will denote $I(X_i)^{-1}$, which is the i^th diagonal element of $\mathbf{I}(\mathbf{X})^{-1}$.

For additive gaussian noise, the system is:

$$X_{i+1} = G_i(X_i) + \mathbf{v}_i \tag{4.56}$$

$$Z_i = H_i(X_i, X_{i-1}, ..., X_1, Z_{i-1}, ..., Z_1) + \boldsymbol{\epsilon}_i \tag{4.57}$$

let $\mathbf{V}_i = \mathrm{cov}(\mathbf{v}_i)$ and $\mathbf{R}_i = \mathbf{cov}(\boldsymbol{\epsilon}_i)$ be known and invertible covariance matrices.

If we have formulated an estimator $X_i \approx \hat{X}_i(Z_1, Z_2, ..., Z_i)$ then Tichavsky et al. [95] compute the information matrix as:

$$I(X_i) = \mathbf{D}_{i-1}^{22} - \mathbf{D}_{i-1}^{12,T}(I(X_{i-1} + \mathbf{D}_{i-1}^{11})^{-1}\mathbf{D}_{i-1}^{12} \tag{4.58}$$

For matrices given by

$$\mathbf{D}_{i-1}^{11} = \mathrm{E}(\frac{\partial G_{i-1}^T}{\partial X_{i-1}})\mathbf{V}_{i-1}^{-1}\mathrm{E}(\frac{\partial G_{i-1}^T}{\partial X_{i-1}})^T \tag{4.59}$$

$$\mathbf{D}_{i-1}^{12} = -\mathrm{E}(\frac{\partial G_{i-1}^T}{\partial X_{i-1}})\mathbf{V}_{i-1}^{-1} \tag{4.60}$$

$$\mathbf{D}_{i-1}^{22} = \mathbf{V}_{i-1} + \mathrm{E}(\frac{\partial H_{i-1}^T}{\partial X_{i-1}})\mathbf{R}_i^{-1}\mathrm{E}(\frac{\partial H_{i-1}^T}{\partial X_{i-1}})^T \tag{4.61}$$

When G_i and F_i are linear, this is the classical Kalman filter [95]. If we can formulate the process of composition of MLM's as a nonlinear system of this type, then we can utilize this bound result. For the hidden state, we will use the Bayes MLM as $G_i = \Gamma_{i+1|i}$. The error term in this case is normal with mean zero and variance given by the variance of $\mathbb{X}_i$:

$$\mathbf{V}_i = \mathrm{var}(\mathbf{x}_{i+1}) \tag{4.62}$$

$$X_{i+1} = \Gamma_{i+1|i}(X_i) + \mathbf{v}_i \tag{4.63}$$

For the observations, we use the output of the composition of the learning morphism, and rewrite in terms of error. Let $\boldsymbol{\epsilon}_i = \mathbf{e}_{i-1,i} - \mathrm{E}(e_{i-1,i})$.

$$H_i = F_{i-1}(Z_{i-1}; \boldsymbol{\theta}_{i-1}) = F_{i-1}(F_{i-2}(\cdots F_1(\mathbf{Z_1}))) = \tag{4.64}$$

$$X_i + \mathbf{e}_{i-1,i} = X_i + \mathrm{E}(\mathbf{e}_{i-1,i}) + \boldsymbol{\epsilon}_i \tag{4.65}$$

Recall that the bias of an MLM was denoted as $\mathbf{B}_{i-1,i}$. Then

$$\mathrm{E}(\mathbf{e}_{i-1,i}) = \mathbf{B}_{i-1,i} + \mathrm{E}(F_{i-1}(Z_{i-1};\theta_{i-1}) - F_{i-1}(X_{i-1};\boldsymbol{\theta}_{i-1})) = \tag{4.66}$$

$$\mathbf{B}_{i-1,i} + \mathbf{E}_{F_{i-1}} \tag{4.67}$$

Then the observation process is:

$$H_i = X_i + \mathbf{B}_{i-1,i} + \mathbf{E}_{F_{i-1}} + \boldsymbol{\epsilon}_i \tag{4.68}$$

and the full nonlinear system is:

$$X_{i+1} = \Gamma_{i+1|i}(X_i) + \mathbf{v}_i \tag{4.69}$$

$$Z_i = X_i + \mathbf{B}_{i-1,i} + \mathbf{E}_{F_{i-1}} + \boldsymbol{\epsilon}_i \tag{4.70}$$

Then, the matrices defined in Proposition 1 are analogous to the matrices $\mathbf{D}_{i-1}^{11}, \mathbf{D}_{i-1}^{12}, \mathbf{D}_{i-1}^{22}$ from Tichavsky et al., and the bound follows from the assumption that $\mathbf{R}_i \succcurlyeq \mathbf{L}\mathbf{B}_{i-1,i}$. This assumption is based on the idea that compositions will induce more error via approximating the true value of a state X_i. Finally, this bound is really a bound on the fisher information matrix.

It is key to again note that we are assuming that $\boldsymbol{\epsilon}_i$ are uncorrelated Gaussian random variables. The Gaussian assumption is satisfied when the learning morphisms F_i preserve normality, for example, linear or affine transformations. Uncorrelated error occurs when the learning morphisms act as orthogonal projections. Therefore, a sequence of affine transformations on data should fulfill the assumptions of this bound.

4.4.2 Principal Component Analysis Composed with Centering

To investigate the composition of PCR, we will initialize the state $X_1 = Z_1 = \mathbf{x}_1$ as the original input data. Then for $i = 1$ the system equation is:

$$X_2 = \Gamma_{2|1}(X_1) + \mathbf{v}_1 = \mathbf{x}_1 - \boldsymbol{\alpha}_x + \mathbf{v}_1 \tag{4.71}$$

$$Z_1 = X_1 + \mathbf{B}_{1,2} + \mathbf{E}_{F_1} + \boldsymbol{\epsilon}_1 = X_1 + 0 + \boldsymbol{\epsilon}_1 \tag{4.72}$$

Since this is not yet a composition we know that the covariance of the error is simply given by the cramer-rao bound: $\mathbf{I}(X_2) = \mathbf{L}\mathbf{B}_{12} = \frac{1}{n}\Sigma_{XX}$

Then for $i = 2$ we have the system equation:

$$X_3 = \Gamma_{3|2}(X_2) + \mathbf{v}_2 = \mathbf{P}^T\mathbf{x}_2 + \mathbf{v}_2 \tag{4.73}$$

$$Z_2 = X_2 + \mathbf{B}_{2,3} + \mathbf{E}_{F_2} + \boldsymbol{\epsilon}_2 = X_2 + 0 + \boldsymbol{\epsilon}_2 \tag{4.74}$$

And we have the variance matrix

$$\mathbf{V}_2 = \mathrm{var}(\mathbf{x}_3) = \boldsymbol{\Lambda} \tag{4.75}$$

and the matrices

$$\mathbf{D}_3 = \mathbb{I} \tag{4.76}$$

$$\mathbf{B}_2 = -\mathbf{P}\mathbf{V}_2^{-1} \tag{4.77}$$

$$\mathbf{C}_2 = \mathbf{P}\mathbf{V}_2^{-1}\mathbf{P}^T \tag{4.78}$$

and bound is given as:

$$\mathbf{I}(X_3) \succcurlyeq (\mathbf{L}\mathbf{B}_{2,3}^{-1} + \mathbf{\Lambda}^{-1} - \mathbf{B}_2^T(\mathbf{I}(X_2) + \mathbf{C}_2)^{-1}\mathbf{B}_2)^{-1} \tag{4.79}$$

To see that this is less than the true information matrix, we will compute the covariance of the error directly.

$$\mathbf{R}_3 = \mathrm{cov}(\mathbf{x}_3 - \Theta_{\mathbf{P}}^T(F_1(\mathbf{x}_1; \boldsymbol{\theta}_{\mathrm{C}})) = \tag{4.80}$$

$$\mathrm{cov}(\mathbf{P}^T\mathbf{x}_2 - \Theta_{\mathbf{P}}(\mathbf{x}_2 + \mathbf{e}_{1,2})) = \tag{4.81}$$

$$\mathrm{cov}(\mathbf{e}_{2,3} - \Theta_{\mathbf{p}}\mathbf{e}_{1,2}) = \tag{4.82}$$

$$\mathbf{L}\mathbf{B}_{2,3} + \Theta_{\mathbf{p}}^T\mathbf{L}\mathbf{B}_{1,2}\Theta_{\mathbf{P}} + 2\mathrm{cov}(\mathbf{e}_{1,2}, \mathbf{e}_{2,3}) = \tag{4.83}$$

$$\mathbf{L}\mathbf{B}_{2,3} + \Theta_{\mathbf{p}}^T\mathbf{L}\mathbf{B}_{1,2}\Theta_{\mathbf{P}} + 0 \tag{4.84}$$

because the cross term is zero

$$\mathrm{cov}(\mathbf{e}_{1,2}, \mathbf{e}_{2,3}) =$$

(4.85)

$$\mathrm{E}((\mathbf{x}_2 - F_1)(\mathbf{x}_3 - F_2)^T) = \mathrm{E}(\mathbf{x}_2\mathbf{x}_3^T) - \mathrm{E}(\mathbf{x}_2\mathbf{x}_2\Theta_{\mathbf{P}}) - \mathrm{E}((\mathbf{x}_1 - \boldsymbol{\theta}_C)\mathbf{x}_3^T) + \mathrm{E}((\mathbf{x}_1 - \boldsymbol{\theta}_{\mathrm{C}})\mathbf{x}_2^T\Theta_{\mathbf{P}}) =$$

(4.86)

$$\mathbf{\Sigma}_{XX}\mathbf{P} - \mathbf{\Sigma}_{XX}\Theta_{\mathbf{P}} - \mathbf{\Sigma}_{XX}\mathbf{P} + \mathbf{0} + \mathbf{\Sigma}_{XX}\Theta_{\mathbf{P}} - \mathbf{0} =$$

(4.87)

$$\mathbf{0}$$

(4.88)

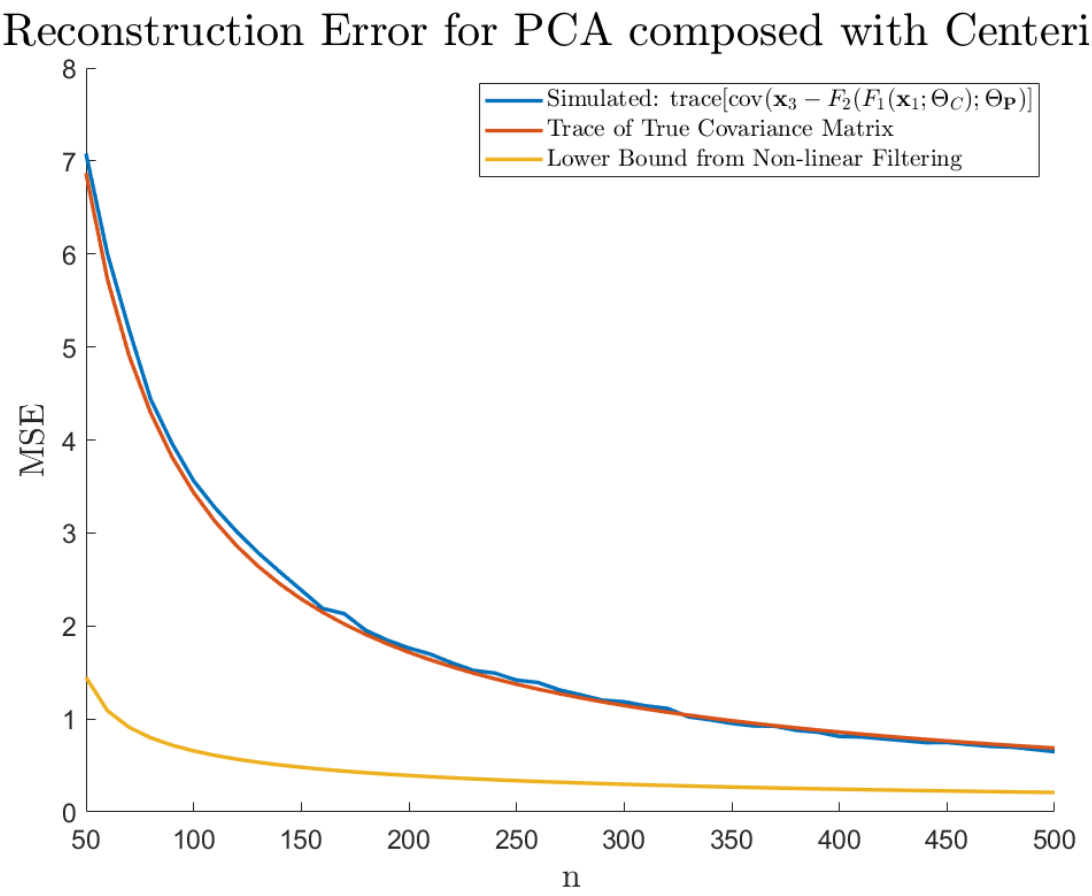


Figure 4.4: The reconstruction error of PCA composed with centering. The observed error covariance is very tight with the true error, and higher than the proposed bound. This is expected as the bound is a lower bound for the true Fisher information.

This satisfies our assumption that the two errors are uncorrelated. Then the lower bound in Equation 4.79 is a lower bound for this covariance. This is visualized in Figure 4.4. Notice that the error is larger than the error of PCA on the true centered data, and that the error is roughly additive. This is because $\Theta_\mathbf{P}$ is an orthonormal matrix, so the quadratic form $\Theta_\mathbf{p}^T \mathbf{LB}_{1,2} \Theta_\mathbf{P}$ does not change the overall magnitude of $\mathbf{LB}_{1,2}$. Given that the proposed bound is lower than the true Fisher Information, why should we use it? In many situations, we will not be able to compute the true lower bound, but this bound is realizable.

4.4.3 Bound for Principal Component Regression

Next, we examine the case of Principal Component Regression. This is the case of $i = 3$, and we have the system equation:

$$X_4 = y = \Gamma_{4|3}(X_3) + \mathbf{v}_3 = \alpha_y + \mathbf{\Sigma}_{XY}^T \mathbf{P}^T \mathbf{\Lambda}^{-1} \mathbf{x}_3 + \mathbf{v}_3 \tag{4.89}$$

$$Z_3 = X_3 + \mathbf{B}_{3,4} + \mathbf{E}_{F_3} + \boldsymbol{\epsilon}_3 = X_2 + 0 + \boldsymbol{\epsilon}_3 \tag{4.90}$$

And we have the variance matrix

$$\mathbf{V}_3 = \text{var}(\mathbf{x}_4) = \mathbf{\Sigma}_{YY} \tag{4.91}$$

and a bound on $\mathbf{R}_4 \succcurlyeq \mathbf{LB}_{3,y}$.

Then we have the matrices:

$$\mathbf{D}_4 = \mathbb{I} \tag{4.92}$$

$$\mathbf{B}_3 = -\mathbf{\Lambda} - 1\mathbf{P} * \mathbf{\Sigma}_{XY} \mathbf{V}_3^{-1} \tag{4.93}$$

$$\mathbf{C}_3 = \mathbf{\Lambda} - 1\mathbf{P} * \mathbf{\Sigma}_{XY} \mathbf{V}_3^{-1} \mathbf{\Sigma}_{XY}^T \mathbf{P}^T \mathbf{\Lambda}^{-1} \mathbf{I}(X_3) = \mathbf{LB}_{2,3} + \Theta_{\mathbf{p}}^T \mathbf{LB}_{1,2} \Theta_{\mathbf{P}} \tag{4.94}$$

and the bound:

$$\mathbf{I}(y) \succcurlyeq (\mathbf{LB}_{3,y}^{-1} + \mathbf{\Sigma}_{YY}^{-1} - \mathbf{B}_3^T (\mathbf{I}(X_3) + \mathbf{C}_3)^{-1} \mathbf{B}_3)^{-1} \tag{4.95}$$

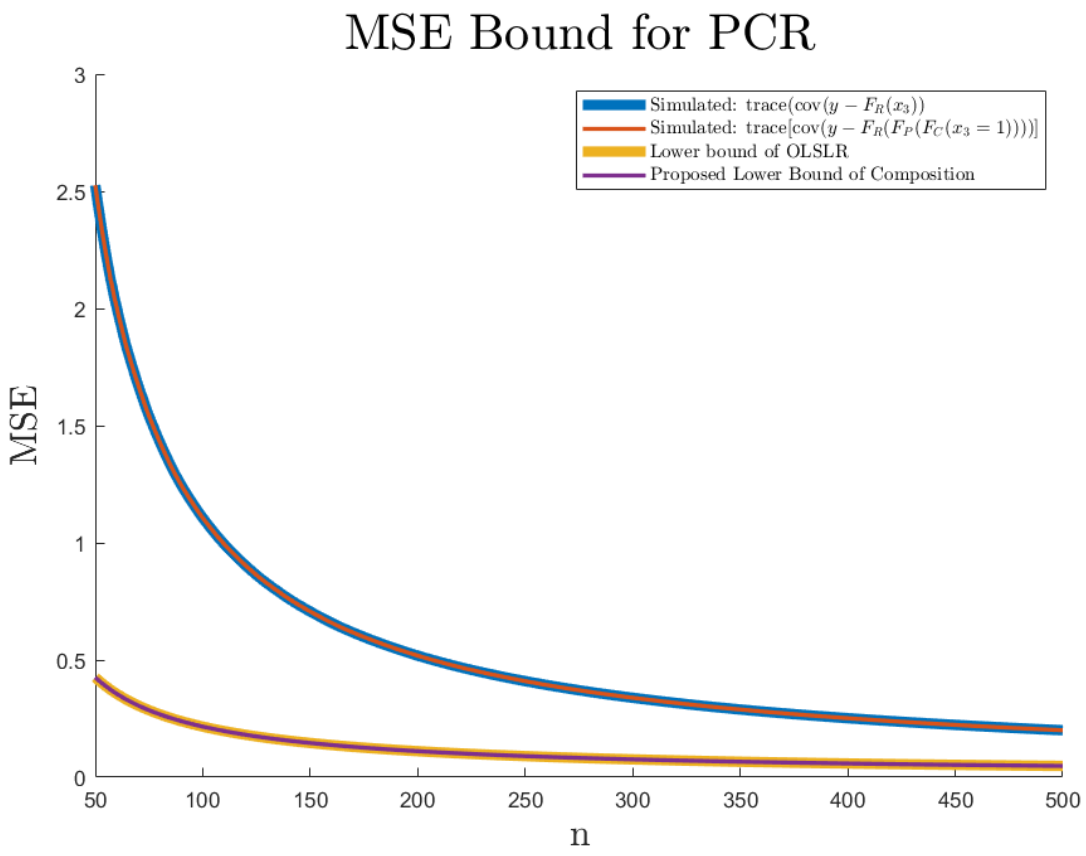


Figure 4.5: MSE and lower bound computed using Equation 4.95, overlayed with MSE and bound for OLSLR. As can be seen, they are equal in this case, because the MLM's are asymptotically equivalent.

This bound is visualized in Figure 4.5. Interestingly, in this simulation this bound is computationally equivalent to $\mathbf{LB}_{3,y}$! Further, the MSE of the composition and the linear regression on the "true" data is the same. Intuitively, this makes sense as we are not performing any dimension reduction, and therefore PCR is equivalent to Ordinary Least-Squares Linear Regression. In Chapter 3 and the appendix, we explore the idea that the regression coefficients for OLSLR and PCR multiply to the same value, which means they will naturally have equal error. In future work, we will explore the bound as dimension reduction is performed and information is lost.

4.5 Chapter Recap

In this chapter, we explored the idea of covariance bounds on the MSE of MLM's. First, we established the Bayes MLM as the best possible workflow for a given loss function. Then we defined a parameterization of the underlying distribution $P(X, Y; \boldsymbol{\alpha})$, and established a bound for MSE based on $\boldsymbol{\alpha}$, the Bayes MLM, and the bias of the MLM. For loss functions other than MSE this bound is useful as it provides a bound for the variance of the difference between an MLM and the Bayes MLM. Examples were presented for Centering, PCA, and Linear Regression.

Then we examined the bound for a composition of MLM's. To compute a bound, we presented the workflow as a nonlinear filtering system. The state was the evolution of the Bayes MLM across each stage of the composition, and the observation was the output of the workflow. By rewriting the observation we presented this problem as one with additive noise, and exploited the bound of Tichavsky et al. [95] to bound the fisher information of each stage. This is a lower bound on the fisher information, and is therefore going to be a lower bound than the CRB. We presented this bound for the stages of PCR. In the case of PCA, the bound is lower than the true covariance structure, and for full PCR the bound is equivalent to the CRB. However, in many cases we may not be able to compute the CRB for the overall workflow and it is useful to have a lower bound for workflow selection purposes.

Chapter 5

Predicting 30-Day Hospital Readmissions Using MLM's and Topological Data Analysis

In this chapter, we build a workflow using a composition of MLM's to predict 30-Day Hospital Readmissions at Barnes Jewish Hospital in St. Louis. We begin by providing a brief definition and overview of readmissions and the data under study. Our workflow utilizes the Mapper Algorithm from Topological Data analysis to cluster patients, and then trains independent models on each cluster. The cluster models are then combined into an ensemble, whereby new patients are assigned to the cluster which they are most similar to, and that model is used to predict the risk of readmission. We compared a variety of workflows for hospital readmissions, and workflows trained using Mapper significantly improves over the LACE score (most common risk prediction tool) for this patient population, as well as other models trained on the entire dataset. Finally, we examine the workflow under the lens of separability from Chapter 3.

5.1 30-Day Hospital Readmissions at Barnes Jewish

An unplanned hospital readmission occurs when a patient is discharged from the hospital but returns at a later date for reasons either related or unrelated to the previous admissions. Readmissions incur significant costs to both the hospital and the patients. The Agency for Healthcare Research and Quality estimated that in 2011 this cost amounted to an additional $41 billion dollars [46]. The emergence of the Hospital Readmissions Reduction Program resulted in a national effort to identify patients at risk and interventions to reduce these hospitalizations. However there are many complexities and challenges [52] and it remains a controversial quality metric that may inadequately account for socioeconomic status; meanwhile associations with mortality need to be further characterized as decreases in readmission have correlated with increases in mortality for heart failure and pneumonia. Disease specific and overall readmission-risk prediction modeling [28][33][70][91] and efforts to improve care delivery, especially during transition periods, have been studied extensively. The primary metric reported for predictive performance is the area under the Receiver Operating Characteristic (ROC AUC), which in medical literature is referred to as the c-Statistic. We will use c-Statistic and ROC AUC interchangeably in this chapter.

Barnes Jewish Hospital (BJH) developed an intervention for patients 65 and older diagnosed with Acute Myocardial Infarction (AMI), Congestive Heart Failure (CHF), Chronic Obstructive Pulmonary Disease (COPD), and Pneumonia (PNA). This intervention, BJH Stay Healthy Outpatient Program (SHOP), has two arms: the Stay Healthy Clinic (SHC) is a 45 minute appointment 7-10 days post discharge; and Outpatient Case Management is a multidisciplinary management approach for up to 60 days post discharge. SHC patients could be discharged home with/without Home Health services. The noted characteristics of the SHC patients are: highest readmission rates, more than 3 comorbid diseases, poor health

literacy, above average social and socioeconomic issues, poor satisfaction with previous primary physician, and living alone. A few of the SHOP goals include: increasing medication adherence, increasing provider visit attendance and decreasing hospital utilization.

Eligible patients are high risk inpatients identified using the LACE Index Scoring Tool. The LACE score, developed in Canada (with a 30 day readmission rate of 8% in the development cohort; overall cohort C-statistic 0.679) [96] is the most widely used method to quantify the risk of readmission, but only achieves a C-statistic of 0.59 in our population and in some other systems range 0.63-0.70 [61]. Yu et al. reported that "the institution specific readmission risk prediction framework is more flexible and more effective than the one-size-fit-all models like the LACE" [113]. Additionally, there is a fine balance in determining the nature and extent of variables to include into risk prediction models as Nguyen et al. reported after reviewing over 30,000 admissions that "incorporating clinically granular EHR (electronic health record) data from the full hospital stay modestly improves prediction of 30-day readmissions" [70].

This chapter presents a proof-of concept study using relevant, available and limited variables to better identify at risk patients (compared to the LACE score) at Barnes Jewish early in their hospital stay in order to enroll them into the current readmission reduction programs. A description of the cohort can be found in the next section.

5.1.1 Patient Data

The Center for Clinical Excellence at BJH determines whether a patient was readmitted. The study cohort included patients discharged from May 1, 2015 to April 30, 2016 with an index diagnosis of AMI, COPD, CHF, and PNA using discharge International Classification

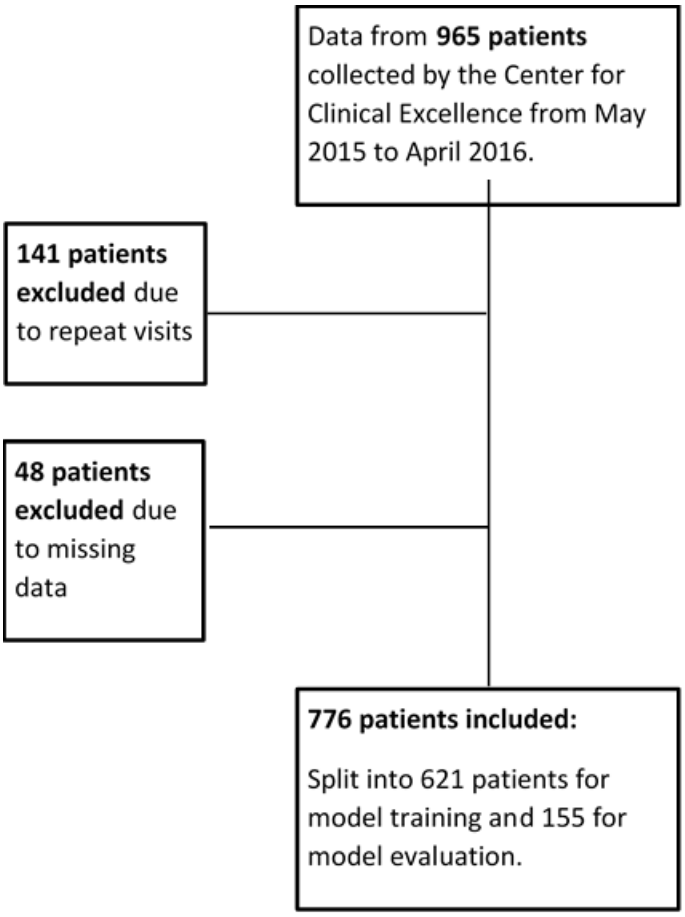


Figure 5.1: Visualization of Exclusions of data from original dataset. Most were excluded when we removed repeat visits, as we did not want to account for time dependency. The rest were excluded to missing data.

of Diseases 9 and 10 codes cross-walked (Appendix 1). 965 registrations of BJH patients age $\geq$ 65 years using the Clinical Classification Software categories published by Healthcare Cost and Utilization Project were identified, representing 824 unique patients but 776 were used in final analyses after accounting for missing data and repeat visits. A diagram of exclusions is presented in Figure 5.1.

The primary outcome we seek to predict is 30-Day Readmission. The variables collected are: LACE Risk Score, presence of Diabetes, principal diagnoses from ICD9/10 codes, gender,

ethnicity, zip code readmission rate, length of stay, age, and presence of a primary care provider. 134 (17.3%) of the patients were readmitted. In each run of the simulation, the data was divided into 621 training and 155 test patients. Some descriptions of the population under study are given in Figure 5.2.

The overall cohort readmission rate was 17.3% (134/776). Most patients had CHF (48.6%); CHF patients were also the highest percentage of readmissions, 53%. Most patients were male (54.1%) and 33.3% were 80+ years old (Figure 5.2). 18.3% of patients had diabetes and were readmitted more (24% of diabetic patients vs 17.1% of patients without diabetes). The average length of stay for all patients was 7 days but 8.3 days for readmitted patients. Most of the patients (66.4%) had a LACE score of ¿=10 and they accounted for most of the readmissions, 77.6%. (Figure 5.2). For patients who were known to have an appointment scheduled within 7 days of discharge, 18.2% were readmitted. Patients with CHF had the longest median days to readmission (Overall: 11 days, AMI: 10 days, CHF: 14 Days, COPD: 10 Days, PNA: 10 Days). The only statistically significant difference in variables between readmitted and non-readmitted patients was the LACE Score (p=.007 - LACE 5-9 and p=.002 - LACE ¿=10). There were no statistically significant differences in readmission between patients who received SHOP/SHC or had a scheduled physician appointment scheduled within 7 days post discharge prior to leaving the hospital (Figure 5.3).

The baseline model for comparison was a univariate regression with the LACE score as the predictor. Multivariate Logistic Regression (LR) Models were trained on the entire set of predictors, the predictors without LACE, the predictors without Discharge Disposition (DD), and the predictors without both LACE or DD. The removal of the LACE score was tested to identify other independent variables associated with readmission in our cohort and DD was removed because that data is not available early in the hospital course. Figure 5.4

Figure 5.2: Descriptive Table of Patient Data from Barnes Jewish Hospital, number represents number of patients, number in parentheses represents percentage of total patients in that column

Variables and Descriptors	Total Cohort (n=776)	Not Readmitted (n=642)	Number Readmitted (n=134)	p-value, Readmitted vs Not
Total Readmissions		642 (82.7%)	134 (17.3 %)	
CHF	378 (48.6%)	307 (47.7%)	71 (53.0%)	0.262
COPD	88 (11.3%)	79 (12.3%)	9 (6.7%)	0.063
AMI	198 (25.5%)	167 (26.0%)	31 (23.1%)	0.487
PNA	113 (14.6%)	90 (14.0%)	23 (17.2%)	0.348
Male	420 (54.1%)	346 (53.8%)	74 (55.2%)	0.779
Diabetes Comorbidity	142 (18.3%)	110 (17.1%)	32 (23.9%)	0.066
Average length of stay-(Days)	7.0	6.81	8.25	0.074
65-69 years old	209 (26.9%)	168 (26.2%)	41 (30.6%)	0.293
70-75 years old	174 (22.4%)	144 (22.4%)	30 (22.4%)	0.992
75-80 years old	135 (17.4%)	117 (18.2%)	18 (13.4%)	0.183
80+ years old	258 (33.3%)	213 (33.2%)	45 (33.6%)	0.928
White	472 (60.8%)	396 (61.7%)	76 (56.7%)	0.284
Black	286 (39.6%)	231 (80.8 %)	55 (41.0%)	0.267
PCP Listed at Time of Admission	604 (77.4%)	503 (78.4%)	101 (75.4%)	0.453
Discharged Home	296 (38.1%)	246 (38.3%)	50 (37.3%)	0.828
Disch. to Skilled Nursing facility	147 (18.9%)	122 (19.0%)	25 (18.7%)	0.925
Disch. with Home Health	270 (34.8%)	218 (34.0%)	52 (38.8%)	0.283
LACE Score:				
Low (0-4)	30 (3.8%)	27 (4.2%)	3 (2.2%)	0.283
Medium (5-9)	231 (30.0%)	204 (31.8)	27 (20.1%)	0.007
High (>=10)	515 (66.4%)	411 (64.0%)	104 (77.6%)	0.002

Figure 5.3: Readmission outcomes of patients receiving institution interventions.

Institution Specific Factors/Intervention	Enrolled	Readmitted and received intervention	Readmitted but did not receive intervention	p value (intervention vs no intervention)
SHOP	444	70 (15.8%)	64 (19.3%)	0.200
SHC	48	6 (12.5%)	128 (17.6%)	0.367
MD Appointment within 7d post discharge	356	65 (18.3%)	69 (16.4%)	0.502

Figure 5.4: Multivariate logistic regression, showing the odds ratio, confidence interval, and p-value of significant predictors.

Predictor	Odds ratio	95% Confidence Interval	p-value
LACE	1.22	(1.14, 1.31)	<0.001
COPD (vs CHF)	0.21	(0.11, 0.45)	<0.001
MI (vs CHF)	0.66	(0.45, 0.99)	<0.001
Disch. to Skilled Nursing Facility (vs Home)	0.50	(0.29, 0.86)	0.01
Disch. with Home Health (vs Home)	2.34	(1.57, 3.49)	<0.001
Male (vs Female)	1.97	(1.36, 2.86)	<0.001
LOS	0.97	(0.95, 0.99)	0.03
Age70-74yrs (vs 65-69 yrs)	0.69	(0.49, 0.98)	0.04
Age75-79 (vs 65-69 yrs)	1.78	(1.2, 2.6)	<0.001
Has PCP (vs no PCP)	1.77	(1.15, 2.72)	0.01

shows predictors with increased odds of readmission were LACE score (OR 1.22; 95% CI 1.14- 1.31; p¡.001); home health discharge (OR 2.34; 95% CI 1.57- 3.49; p¡.001); male sex (OR 1.97; CI 1.36- 2.86; p¡.001); age 75-79 (OR 1.78; 95% CI 1.2- 2.6; p¡.001) and having a PCP noted in the EMR (OR 1.77; 95% CI 1.5- 2.72; p¡.01).

5.1.2 Challenges

However, the c-statistic of the multivariate logistic regression model, found in Table 5.1, is 0.49, which is much worse than using only the LACE score (0.59). Using SMOTE sampling and variable selection, we improved the c-statistic to 0.64, which is still under the expected performance of the LACE score.

Developing a predictive model for readmissions is challenging for a variety of reasons. There was low data availability (< 1000 patient records were able to be collected from the databases). The predictors are limited to basic diagnostic and demographic data, and do not provide an

accurate picture of what happened during the patient's stay (which we hypothesize will have some impact on readmissions). Further, the predictors are not well correlated to readmissions, and in some cases are dependent/redundant. For example, Diabetes as a comorbidity is included in both the LACE score and as a column in our dataset. Finally, this is an imbalanced dataset, with most patients not experiencing readmissions.

Therefore, we developed a new predictive model using the MLM framework. The model works by clustering the patients into more predictive groups, and training models specifically on these groups. We addressed the class imbalance problem by exploring SMOTE and ROSE sampling, and were able to improve over the predictivity of LACE and the multivariate logistic regression. The next section provides an overview of the Mapper algorithm, a tool from Topological Data Analysis which we utilized in our workflow.

5.2 TDA Mapper as a MLM

In this section we present a brief overview of the TDA Mapper algorithm, show that it fits into the MLM framework, and build some example workflows utilizing Mapper.

5.2.1 Mapper Algorithm

Topological Data Analysis assumes that the input space $\mathbb{X}$ can be endowed with a collection of subsets $\mathbb{O}$. Elements $\mathbf{o} \in \mathbb{O}$ satisfy $\underset{i}{\cup}\mathbf{o}_i \in \mathbb{O}$ and $\overset{n<\infty}{\underset{i=1}{\cap}}\mathbf{o}_i \in \mathbb{O}$, and are called open sets. Then the ordered pair $\{\mathbb{X}, \mathbb{O}\}$, forms a *Topological Space*. TDA builds topological spaces on top of data points, and evaluates the shape of the computed spaces [106]. The critical features are closed loops in various dimensions, which are invariant to rotation or multiplicative scaling.

When $\mathbb{X}$ is also endowed with a probability space, the topological features are also endowed with a probability space [16], and can be used in machine learning. TDA has been used as a novel visualization tool in bio-medical applications [34][8], text mining [38], and remote sensing [29].

One visualization tool developed for TDA at Stanford is the Mapper Algorithm [93]. It creates a graphical representation of the data that keeps an equivalent topological structure, and has been used in a wide variety of applications [71][24]. Mapper is usually used as a method for clustering and visualization. Interesting clusters or patterns are used as a feature selection method to reduce the dimensionality of data before training learning models.

To construct the Mapper graph, first define a filtration function $A : \mathbb{X} \to \mathbb{R}$ (*note*: it isn't necessary for the range to be $\mathbb{R}$, but we're using it here for simplicity). Then define an equivalence relation $\sim_{\mathrm{A}}$ such that $\mathbf{x}_1 \sim_{\mathrm{A}} \mathbf{x}_2$ whenever $A(\mathbf{x}_1) = A(\mathbf{x}_2)$, which collapses every level set of A to a single point. The *Reeb Graph* is the quotient space of $\mathbb{X}$ under the relation $\sim_A$. Mathematically, the Mapper algorithm computationally approximates the Reeb Graph by computing the nerve of a refined pullback of an open cover, $\mathbf{O} \subset \mathbb{O}$, of $A(\mathbb{X})$. Practically, Mapper assigns datapoints to $\mathbf{O}$, and then performs clustering within each member of the open cover. Then it creates graph G with a set of nodes $\mathbf{N}_i \in \mathbf{N}$ representing the clusters, and a set of edges $\mathbf{E}$ where an edge e_{ij} means that two clusters have non-empty intersection. It has been proven to converge exactly to the Reeb graph if $\mathbf{O}$ is refined enough [18]. The full algorithm is described in [93], and rough pseudocode of computational algorithm is detailed in Algorithm 1.

In topology, an *abstract simplicial complex* is a family of non-empty finite sets that is closed when taking non-empty subsets. One of the main ideas of TDA is to create abstract simplicial complexes from sets of data [30]. The pullback operator on an open cover has a more

complicated definition that is out of the scope of this paper, but the *nerve* of an open cover is a representation of the open cover as an abstract simplicial complex. The Mapper algorithm computes the nerve of the $\mathbb{O}$ using the procedure in Algorithm 1, and the result is a graph showing the "shape" of the data [93].

Algorithm 1 Description of the TDA Mapper Algorithm by Singh, Memoli, and Carlsson et al.

Input: • Data $\mathbf{X}$, distance metric D on $\mathbb{X}$, filtration function $A : \mathbb{X} \to \mathbb{R}$

• Number of intervals k, number of bins when clustering b, percent overlap o

Output: Graph G, nodes $n_i \in \mathbf{N}$, edges $e_{ij} \in \mathbf{E}$

$\mathbf{N} \leftarrow \emptyset$
$\mathbf{E} \leftarrow \emptyset$
Compute $\mathbf{Y} = A(\mathbf{X})$
Generate an open cover of $\mathbf{Y}$ with k open intervals $\{\mathbf{I}_j\}_{j=1}^{k}$, with area a_I, such that $\mathbf{I}_j \cap \mathbf{I}_{j+1} \neq \emptyset$ and the area of each intersection is $o * a_I$
for $j = 1$ to k **do**
    Perform clustering such as k-means, using b clusters, on $\mathbf{x} \in \mathbf{X} \cap A^{-1}(\mathbf{I}_j)$
    append each cluster $\mathbf{N}_i$, for $i = 1, 2, 3...$ to $\mathbf{N}$
end for
for $i, j \in \mathbf{N}$ **do**
    if $\mathbf{N}_i \cap \mathbf{N}_j \neq \emptyset$ **then**
        Append edge e_{ij} to $\mathbf{E}$
    end if
end for
return $\mathbf{N}$, $\mathbf{E}$

5.2.2 Machine Learning Workflows with Mapper

The set of nodes, $\mathbf{N} = \{\mathbf{N}_1, ..., \mathbf{N}_w\}$, output by Mapper, represents a cover, $\{\mathbf{X}_i\}_{i=1}^{w}$ of $\mathbf{X}$. Taking Mapper as a morphism from $\mathbb{X}$ to the set of all covers of $\mathbb{X}$, the Mapper algorithm is an MLM with structure:

- Input space: the topological space $(\mathbb{X}, \mathbb{O})$

- Output space: set of all open covers of $\mathbb{X}$
- Parameter Prior: distribution over the parameters in algorithm 1 $P(D, A, k, b, o)$, representing prior knowledge or choices of the proper distance metric, filtration function, etc.
- Morphism: Nerve of the refined pullback of an open cover, $\mathbf{O}$, of $A(\mathbb{X})$
- Risk function: Graph Edit Distance [35]

Because Mapper approximates the Reeb Graph, we use the graph edit distance (GED) [35] between Mapper and the "true" Reeb Graph as the risk function for the Mapper MLM. The graph edit distance between graphs G_1 and G_2 summing up the cost of the operations necessary to transform G_1 into G_2. These operations commonly include adding/deleting edges, nodes, and changing the labels of nodes. Formally if $\mathbf{B} = [B_1, B_2, ..., B_k]$ contains the graph operations necessary to transform G_1 into G_2 and $C : \mathbf{B} \to \mathbb{R}^+$ is a cost function, then the graph edit distance is:

$$\bar{R} = \text{GED}(G_1, G_2) = \sum_{i=1}^{k} C(B_i) \tag{5.1}$$

In Section IV, we use grid search to search through the parameters of Mapper. To bring this MLM closer to the realm of statistical learning theory, future work could extend the statistical analysis from [18] to define a computationally tractable loss function and more informative parameter prior. However, in the context of the larger workflow, the choice of risk is less relevant, because the parameters are optimized over the final risk function.

We build an example machine learning workflow with Mapper and logistic regression as follows:

- The input space is $\mathbb{X}$, which has training realizations $\mathbf{X}_{\text{TR}}$, validation realizations $\mathbf{X}_{\text{V}}$, and testing realizations $\mathbf{X}_{\text{TS}}$.
- The output space is $\mathbb{Y} = \{0, 1\}$.
- $\mathcal{ML}_0$: Dummy coding the original data matrix, embeds the data into $\mathbb{R}^m$.
- $\mathcal{ML}_1$: Mapper MLM trained on realizations $\mathbf{X}_{\text{TR}}$:
  - If the first principal component is the filtration function a , then this MLM features an output composition with the PCA MLM.
  - For computational reasons down the line, we remove vertices with less than 40 data points, so the output is not a total cover of $\mathbb{X}$. When the first principal component is used this seems to have the effect of removing outliers with high/low PCA scores.
  - $\mathcal{ML}_1$ outputs a set of spaces $\{\mathbb{X}_i\}_{i=1}^k$, with training realizations separated into each group $\{\mathbf{X}_{\text{TR},i} \subset \mathbb{X}_i\}$
- $\mathcal{ML}_2 = \sum_{i=1}^w C_i(x)\mathcal{ML}_2^i$, where each $\mathcal{ML}_2^i$ has structure:
  - Input Space: $\mathbb{R}_m$,
  - Output Space: $\mathbb{Y} = [0, 1]$, representing the class probability $P(y = 1)$,
  - Morphism: Composition of:
    - Feature extraction, e.g. PCA,
    - A learning machine, e.g. logistic regression trained on $\mathbf{X}_{\text{TR},i}$
  - Parameter Prior: Gaussian priors on the regression coefficients for each node, uniform priors on the parameters of the Mapper MLM.

- Risk function: Maximum Likelihood, combined with sampling to address class imbalance, e.g. oversampling, undersampling, ROSE [66], or SMOTE [23]. Additionally, model hyperparameters can be selected with cross validation.
- $C_i : \mathbb{X} \to \mathbb{R}$ is a weighting function depending on where points lie in the input space, related to which nodes of the Mapper graph are "active" for a given data point.

- $\mathcal{ML}_3$: A decision threshold with:
  - Input space: $\mathbb{X} = [0, 1]$
  - Output space: $\mathbb{Y} = \{0, 1\}$
  - Morphism:

$$\begin{cases} y = 1 & \text{if } x \geq T \\ y = 0 & \text{else} \end{cases} \tag{5.2}$$

  - Parameter Prior: prior information of threshold $T \in [0, 1]$
  - Risk Function: Method to choose probability threshold, e.g. choosing an optimal threshold of a ROC curve over cross validation sets.

The full workflow is:

$$\mathcal{M} : \mathbb{X} \to \{0, 1\} = \mathcal{ML}_3 \circ_S (\mathcal{ML}_2 \circ_O \mathcal{ML}_1) \circ_O \mathcal{ML}_0 \tag{5.3}$$

This MLW creates and optimizes a separate workflow $\mathcal{ML}_2^i$ for each node created by the Mapper graph. The Dummy Coding is clearly an output composition, and given the mapper parameters the classifiers $\mathcal{ML}_2^i$ are trained independently, so we have an output composition

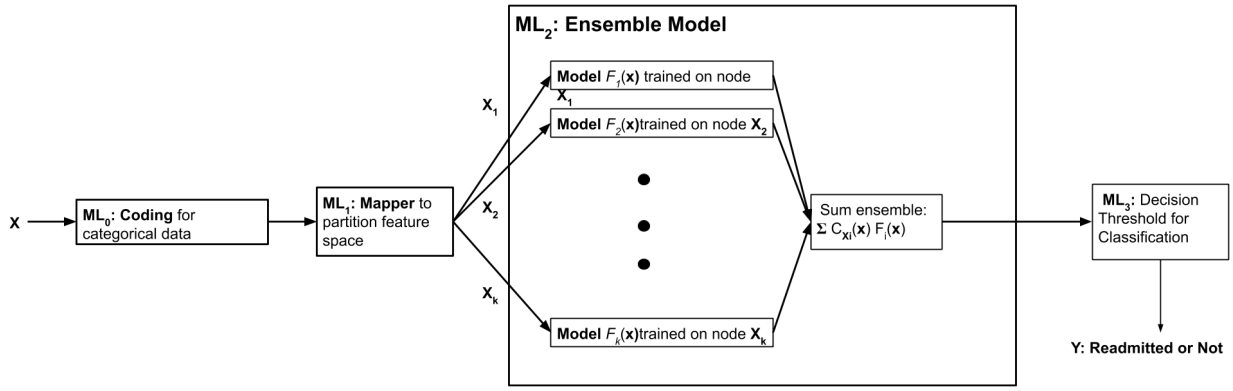


Figure 5.5: Block diagram of Eq. 5.3, showing how a workflow is created for each node. The first step is one-hot encoding the data to embed it into $\mathbb{R}^M$. The next step computes the Mapper graph of the data. Then models are trained on each node, and summed. Finally, a decision function outputs the final class prediction.

$\mathcal{ML}_2 \circ_O \mathcal{ML}_1$. However, we train the mapper parameters over the loss function of the decision threshold, and estimate that risk using ROC AUC on cross validation holdout sets. So this workflow features both output and structural compositions as defined in Chapter 2. Figure 5.5 shows a block diagram represenation of the full workflow using Mapper. Final model evaluation uses $\mathcal{M}$ as an input to an evaluation MLM using realizations from the test set $\mathbf{X}_{\text{TS}}$. The weights $C_i(x)$ in $\mathcal{ML}_2$ represent an interesting choice. Intuitively, weights should be non-zero only when a point lies in $\mathbb{X}_i$, so only a portion of the models are "active" for a given point. Because they are summing elements of a probability space, $\sum_{i=1}^{w} C_i(\mathbf{x}) = 1$ for all $\mathbf{x}$. Options for the weighting parameters include:

- assign a weight of 1 to the "closest" node and 0 to all others.
- Assign equal weight to all nodes to which the point belongs, and 0 to all others.
- Assign weight inversely proportional to the distance from the center of the interval assigned to that node.

- Assign weight proportional to the cross validation metrics of each model, i.e. models that perform better on the training data are assigned higher weights.
- Train weights within cross validation by defining a loss function based on the metric of interest.

The Mapper algorithm could be replaced with another clustering algorithm such as k-means, or any other mapping that chooses subsets of data. We chose the Mapper algorithm because the subsets it generates have some appealing properties. First, the points in one $\mathbb{X}_i$ are all "close" in the sense of the filtration function, but may have a different internal structure than another node to which they are not connected. A column of $\mathbf{X}_i$ may be positively correlated with the outcome variable, but the same column of $\mathbf{X}_j$ may be negatively correlated. When considered in a model over the entire dataset, these correlations may "compete" with each other.

Furthermore, with the proper choice of a filtration function, some nodes may have a higher incidence of the outcome variable. In previous literature this was done in order to identify subsets with high minority prevalence for further study. By training a model only on that node, we reduce some of the class imbalance on that set, theoretically increasing model performance. Finally, many clustering methods do not produce overlapping clusters, but in [83], training classifiers on overlapping cluster was shown to improve performance. The Mapper algorithm allows for datapoints that fall into multiple nodes of the Mapper graph to contribute information to each node's model.

To evaluate the workflow $\mathcal{M}$, use the workflow as input to an MLM that evaluates classifier performance over new realizations from $\mathbb{X}$ (the testing set). In the next section we focus on ROC area under the curve (AUC), Sensitivity, and Specificity as different training metrics.

5.3 Numerical Experiments

5.3.1 Comparisons

We built several versions of Eq. 5.3 across two real world datasets. The workflow breakdown is as follows:

- Realizations: Always an 80/20 training/testing split.
- $\mathcal{ML}_0$: Dummy coding performed using the default parameters from the caret package.
- $\mathcal{ML}_1$: One of [Mapper, Identity (no transformations)]. For the Mapper graph, we used a uniform prior on the number of intervals k from [5-20], the percent overlap o from [20% - 60 %], and the number of bins when clustering from [5-30]. We fixed the filtration function a as the first principal component, and the distance metric d as the gower metric, which means we used a dirac delta as the prior for these parameters.
- $\mathcal{ML}_2$: Node Models:
  - Feature Extraction: Used caret package in R [55] to perform one of: [no transformations, PCA]
  - Sampling: Used caret package in R to perform one of [no sampling, SMOTE, ROSE]
  - Learning Machines: Used caret package in R to train one of [Logistic Regression, SVM, Random Forests, AdaBoost [89]]
  - Cross Validation: Used caret package in R to generate 10-fold cross validation sets to tune model hyperparameters, such as the number of trees in the random forests.

- Weighting functions C_i: Points are assigned to nodes by computing the filtration function, assigning to appropriate intervals and then finding the closest cluster within each interval. Then weights were assigned to the one or two closest nodes. If two nodes are used, we use either equal weights or weights proportional to the ROC performance on the validation set.

- $\mathcal{ML}_3$: Decision threshold function as defined in Eq. 5.2.

Mapper graphs used the first principal component as the filtration function, and the other parameters were tuned by grid search. Preliminary investigations with other filtration functions on patient record data revealed that the first PC seemed to yield the best classifier performance, so it was fixed as the filtration function for each experiment. These graphs tended not to find any loops or interesting topological structures when using that particular filtration function. However, there was usually a good spread with respect to readmission where some nodes have a high rate and others have very low rates.

The results are grouped by the type of learning machine used in $\mathcal{ML}_2$, one of Logistic Regression, Random Forests, AdaBoost, or SVM. Each workflow uses only one of these types, and experimenting with more involved model selection on different nodes $\mathbf{X}_i$ is an interesting direction of future work. Every workflow tested was run with 10 different training/testing splits, and the resulting performance measures were averaged. The workflows are named by "Mapper/No Transformation,PCA (if applicable), Sampling method (if applicable), node weights (if applicable).

Table 5.1: Results for different workflows of logistic regression on hospital readmissions data, with (standard deviations) over n=10 runs.

LR Workflow	ROC AUC	Sensitivity	Specificity	Accuracy
No Transformation	0.49 (0.023)	0.58 (0.031)	0.48 (0.021)	0.49 (0.022)
No Transformation, SMOTE	0.64 (0.033)	0.62 (0.029)	0.67 (0.039)	0.66 (0.037)
No Transformation, ROSE	0.53 (0.041)	0.54 (0.044)	0.51 (0.045)	0.52 (0.045
PCA	0.58 (0.017)	0.68 (0.022)	0.45 (0.029)	0.49 (0.028)
PCA, SMOTE	0.49 (0.037)	0.64(0.035)	0.44 (0.034)	0.47 (0.034)
PCA, ROSE	0.45 (0.061)	0.50 (0.059)	0.55(0.065)	0.54 (0.063)
Mapper, No Transformations	0.61 (0.048)	0.62 (0.052)	0.53 (0.049)	0.55 (0.050)
Mapper, No Transformations, SMOTE	0.67 (0.066)	0.60 (0.055)	0.60 (0.064)	0.60 (0.062)
Mapper, No Transformation, ROSE	0.62 (0.073)	0.69 (0.076)	0.59 (0.078)	0.61 (0.078)
Mapper, Node PCA	0.55 (0.065)	0.62 (0.058)	0.50 (0.059)	0.52 (0.058)
Mapper, Node PCA, SMOTE	0.69(0.071)	0.62 (0.069)	0.78 (0.065)	0.75 (0.066)
Mapper, Node PCA, ROSE	0.61 (0.084)	0.58 (0.082)	0.63 (0.087)	0.62 (0.086)

5.3.2 Workflow Results on Hospital Readmissions Data

Tables 5.1-5.4 show the classification results for multiple workflows, themed by the type of classifier, averaged over 10 runs. The chosen Mapper parameters were 10 intervals, 50% overlap, and 20 bins when clustering. Typical Mapper graphs had 10 nodes, each with 40-200 patients per node. We report the ROC AUC, Sensitivity, Specificity, and Accuracy for each model tested, but note that accuracy in this case is biased heavily towards the specificity score since negatives make up 83% of the patients.

The Mapper graphs were tuned using the first principal component of the entire dataset as the filtration function, a typical graph is shown in Figure 5.6. Based off of a grid search, a bin overlap of 40-50% yielded roughly the same results, with 10 intervals as the clustering parameter. Each run produced 5-10 nodes, with readmission ranging from 5%-30%.

This dataset catalyzed the use of the Mapper graph in the ML workflow. Models trained on the entire dataset do not perform well, and we were aiming to build a workflow that

Table 5.2: Results for different workflows of SVMs for hospital readmissions data, with (standard deviations) over n=10 runs.

SVM Workflow	ROC AUC	Sensitivity	Specificity	Accuracy
No Transformation	0.63 (0.033)	0.65 (0.037)	0.6 (0.035)	0.61 (0.036)
No Transformation, SMOTE	0.59 (0.042)	0.65 (0.040))	0.49 (0.046)	0.52 (0.044)
No Transformation, ROSE	0.55 (0.083)	0.80 (0.087)	0.44 (0.091)	0.50 (0.090)
PCA	0.64 (0.039)	0.69 (0.044)	0.58 (0.038)	0.60 (0.039)
PCA, SMOTE	0.61 (0.047)	0.58 (0.043)	0.58 (0.048)	0.58 (0.046)
PCA, ROSE	0.62 (0.058)	0.62 (0.049)	0.62 (0.054)	0.62 (0.053)
Mapper, No Transformations	0.53 (0.057)	0.58 (0.075)	0.48 (0.068)	0.49 (0.070)
Mapper, No Transformations, SMOTE	0.57 (0.079)	0.54 (0.072)	0.64 (0.076)	0.62 (0.075)
Mapper, No Transformation, ROSE	0.53 (0.086)	0.50 (0.081)	0.64 (0.088)	0.61 (0.086)
Mapper, Node PCA	0.50 (0.065)	0.62 (0.073)	0.49 (0.072)	0.51 (0.072)
Mapper, Node PCA, SMOTE	0.61 (0.077)	0.73 (0.083)	0.53 (0.089)	0.56 (0.088)
Mapper, Node PCA, ROSE	0.67 (0.092)	0.77 (0.095)	0.60 (0.088)	0.63 (0.089)

Table 5.3: Results for different workflows of random forests for hospital readmissions data, with (standard deviations) over n=10 runs.

RF Workflow	ROC AUC	Sensitivity	Specificity	Accuracy
No Transformation	0.60 (0.047)	0.58(0.042)	0.57 (0.049)	0.57 (0.046)
No Transformation, SMOTE	0.52 (0.053)	0.46 (0.052)	0.75 (0.055)	0.70 (0.055)
No Transformation, ROSE	0.5 (0)	0 (0)	1(0)	0.827 (0)
PCA	0.56(0.051)	0.50 (0.066)	0.63 (0.068)	0.61 (0.068)
PCA, SMOTE	0.57 (0.053)	0.62(0.051)	0.60 (0.058)	0.60(0.056)
PCA, ROSE	0.53 (0.072)	0.54 (0.071)	0.56 (0.076)	0.56 (0.075)
Mapper, No Transformations	0.49 (0.078)	0.46 (0.084)	0.60 (0.081)	0.58 (0.082)
Mapper, No Transformations, SMOTE	0.55 (0.087)	0.58 (0.075)	0.54 (0.082)	0.55 (0.080)
Mapper, No Transformation, ROSE	0.51 (0.093)	0.54(0.116)	0.51 (0.143)	0.52 (0.137)
Mapper, Node PCA	0.57 (0.069)	0.62 (0.076)	0.62 (0.086)	0.62 (0.083)
Mapper, Node PCA, SMOTE	0.57 (0.084)	0.46 (0.095)	0.71 (0.091)	0.67 (0.092)
Mapper, Node PCA, ROSE	0.64 (0.110)	0.65 (0.099)	0.61 (0.091)	0.62 (0.097)

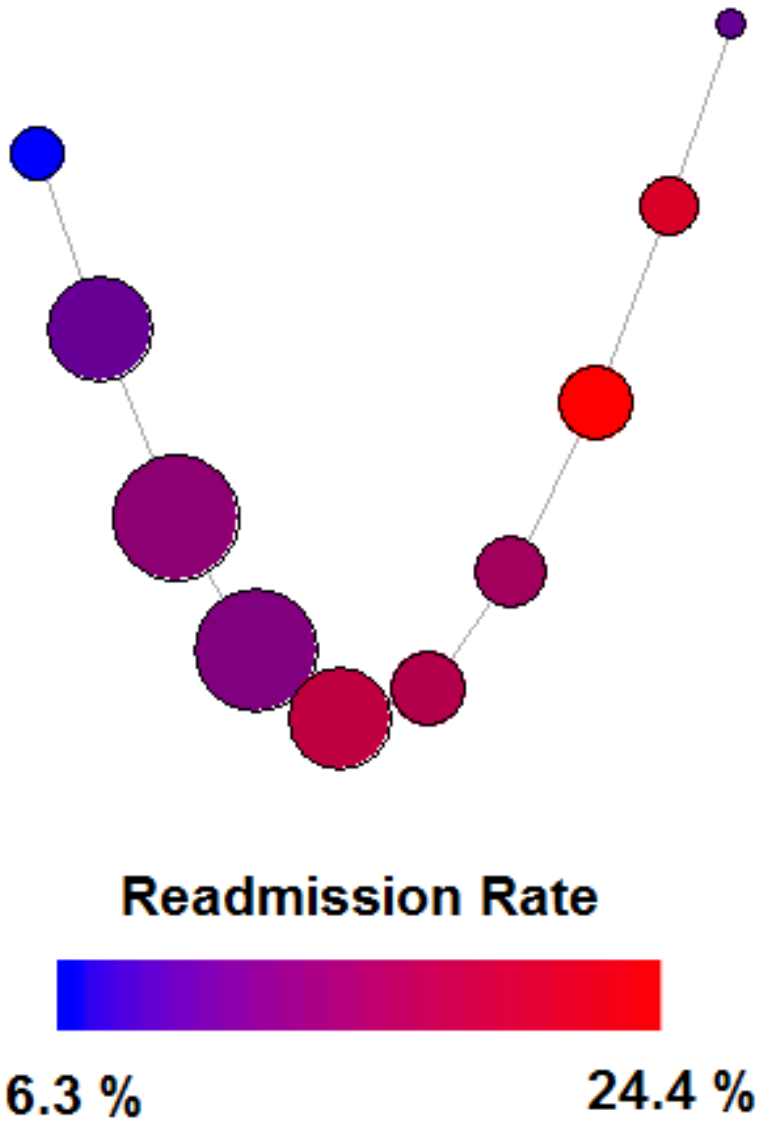


Figure 5.6: Typical Mapper graph generated from hospital readmissions data. The nodes are colored showing level of readmissions, and larger node size indicates a higher number of patients in that node.

Table 5.4: Results for different workflows of Adaboost classifiers, with (standard deviations) over n=10 runs.

AdaBoost workflow	ROC AUC	Sensitivity	Specificity	Accuracy
No Transformation	0.50 (0.043)	0.54 (0.058)	0.49 (0.049)	0.50 (0.051)
No Transformation, SMOTE	0.62 (0.056)	0.65 (0.072)	0.53 (0.070)	0.55 (0.071)
No Transformation, ROSE	0.5(0)	0 (0)	1 (0)	0.827 (0)
PCA	0.48 (0.038)	0.54 (0.044)	0.53 (0.049)	0.53 (0.048)
PCA, SMOTE	0.53 (0.051)	0.50 (0.053)	0.58 (0.057)	0.57 (0.056)
PCA, ROSE	0.69 (0.073)	0.46 (0.078)	0.74 (0.064)	0.69 (0.068)
Mapper, No Transformations	0.56 (0.079)	0.49 (0.082)	0.75 (0.086)	0.71 (0.085)
Mapper, No Transformations, SMOTE	0.63 (0.083)	0.73 (0.077)	0.63 (0.083)	0.65 (0.082)
Mapper, No Transformation, ROSE	0.54 (0.098)	0.42 (0.131)	0.67 (0.110)	0.63(0.119)
Mapper, Node PCA	0.63 (0.066)	0.69 (0.075)	0.58 (0.082)	0.60 (0.081)
Mapper, Node PCA, SMOTE	0.58 (0.088)	0.65 (0.084)	0.54 (0.091)	0.56 (0.089)
Mapper, Node PCA, ROSE	0.44 (0.141)	0.58 (0.109)	0.51 (0.092)	0.52(0.095)

improved upon the LACE score. LACE is a combination of Length of previous hospital stays, Acuity of admission (emergency/not emergencey), the Charlson Comorbidity, and number of prior Emergency Department visits, and is the most used tool to predict readmissions risk. However, our population has a huge majority of "high" risk patients, and logistic regression trained on lace results in a ROC AUC of 0.59, with the optimal sensitivity at 0.54, which only correctly predicts slightly more than half of all readmissions. Applying Mapper in to our workflow resulted in large performance increases over the LACE model.

On the Logistic Regression, the Mapper algorithm with PCA computed for individual nodes and SMOTE sampling outperforms all other workflows, with Mapper/No Transformation (NT) and NT + SMOTE also showing higher performance than others. Sensitivity is of particular interest in this problem, in order to identify as many high risk patients as possible and target them with additional resources.

The SVM classifier performed the best out of the out-of-the-box models with no sampling or other transformations, however the Sensitivity was significantly increased by the models using Mapper, Node PCA, and SMOTE/ROSE sampling. None of the models using Mapper in conjunction with Random Forests showed large improvement over training the Random Forests over the entire training set. It should be noted that the models running Random Forests with ROSE sampling in the Caret Package didn't converge, and voted "no-readmission" for every point in the test set. This issue also occured when using Adaboost classifiers with ROSE sampling in Table 5.4.

Adaboost models were improved both by sampling and in two of the Mapper workflows. One case to note is the PCA+ROSE combination, which features a "high" AUC but low Sensitivity. In this case we would throw out the model in favor of the Mapper, no transformation, SMOTE workflow which correctly identifies almost 3/4 of the readmitted patients. One reason AdaBoost models might perform better with the Mapper algorithm is that AdaBoost is often used on smaller datasets, which the Mapper workflow naturally creates.

Compared to the out of the box models trained using caret, workflows utilizing Mapper tend to have a higher variance between runs. This can be explained by additional variance introduced by assigning the testing points to different nodes of the Mapper graph. Since each run has a different testing set, different node models will predict different numbers of testing points. Additionally, variance is introduced by using SMOTE or ROSE sampling methods, since each run creates a new set of synthetic samples. Methods using ROSE Sampling and Mapper had the highest spread in metrics.

Approval

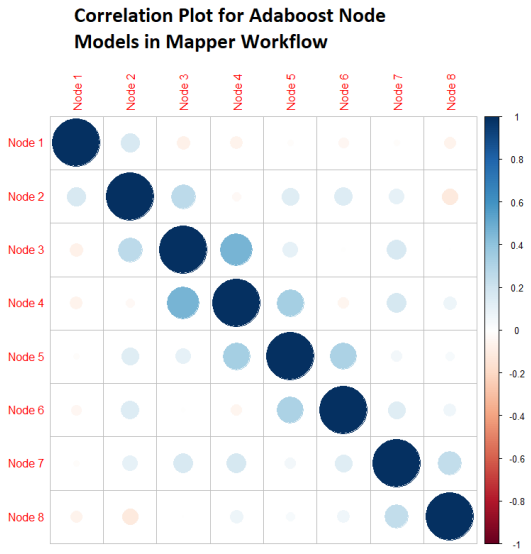
The Washington University Institutional Review Board approved the use of this data as a retrospective study. All HIPPA identifying information has been removed.

5.4 Is the Readmissions Workflow Separable?

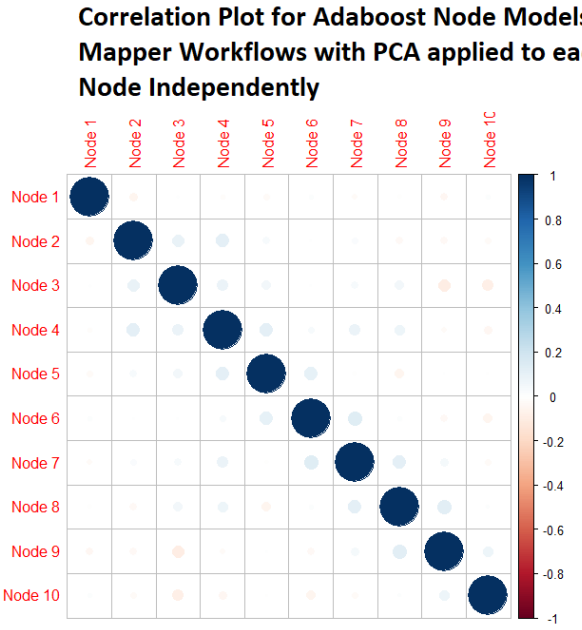
Chronologically, we developed the idea of separable MLM's after building the hospital readmissions workflow, but it is natural to ask: Can we really train the node models independently? MSE is not the loss function for this workflow, so Theorem 1 and the following results do not apply in this case. However, in classification, achieving an uncorrelated ensemble is known to improve performance [60]. Therefore, we hypothesize that uncorrelated ensemble members is one of the conditions for separability in MLM's with 0-1 loss.

In the numerical experiments we investigate the correlation between the outputs on each node. For each training, testing split we computed the mapper graph on the training data. Then we trained logistic regression, random forest, adaboost, and SVM models for each node of the mapper graph. Next, we computed the predictions for each node model on the entire test set, rather than assigning test data to the nodes. For most of the test data, the node model predictions will be highly inaccurate because they did not see that part of the data in their training set. Then, we estimate the correlation matrix between each node of the mapper graph. This is averaged over 10 training and testing splits to estimate the average node correlation.

Figure 5.7 shows the correlation of AdaBoost node models with and without PCA used for feature extraction. In both cases, the nodes have relatively low correlation. When they

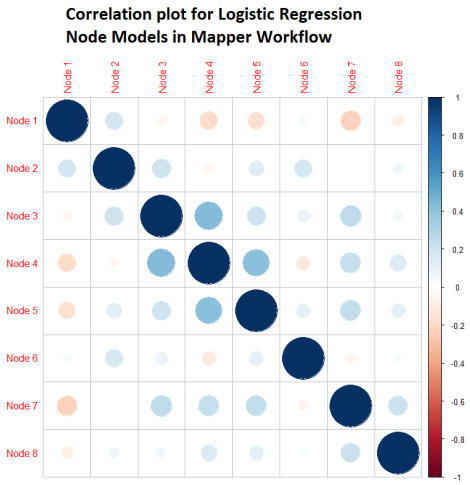


(a) Correlation plot of nodes trained with AdaBoost. The nodes are relatively uncorrelated.

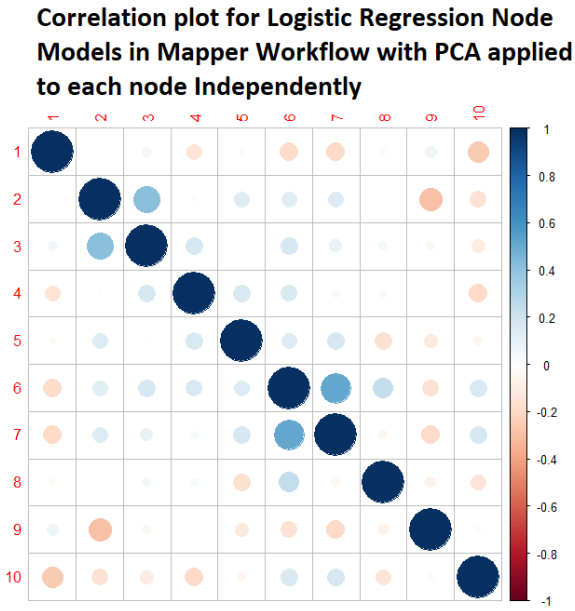


(b) Correlation plot of nodes trained with PCA for feature extraction and AdaBoost for classification. The correlation between nodes is very low, which is desired for classification.

Figure 5.7: Estimated node correlation for AdaBoost workflows with PCA(5.7b) and without PCA(5.7a). The correlation between nodes is less for PCA, which is to be expected since PCA naturally decorrelates the input data.

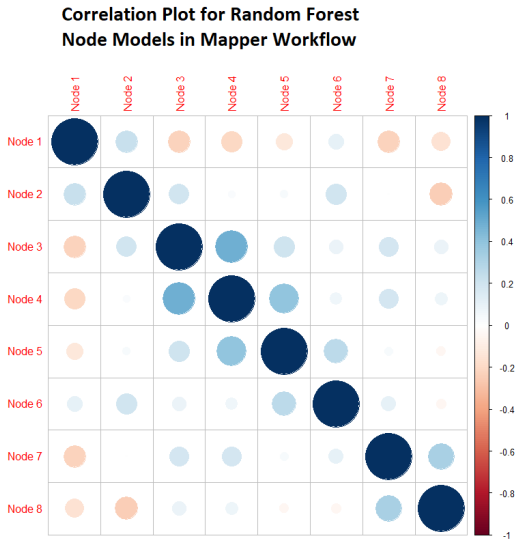


(a) Correlation plot of nodes trained with logistic regression. The nodes are more correlated than Figure 5.7a but the correlations are still fairly low.

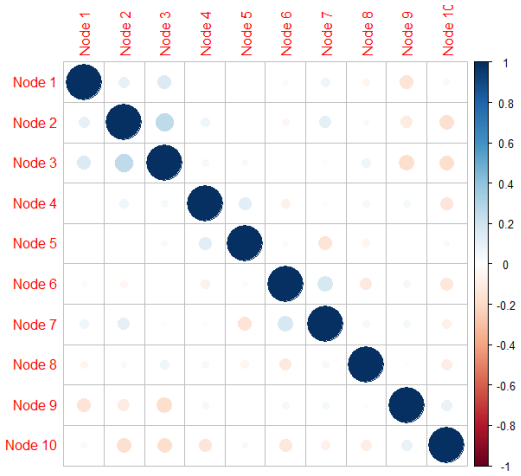


(b) Correlation plot of nodes trained with PCA for feature extraction and logistic regression for classification. PCA in this case slightly decreases the correlation between nodes but does not seem to decrease it meaningfully.

Figure 5.8: Estimated node correlation for Logistic Regression with PCA(5.8b) and without PCA(5.8a). The presence of PCA does not seem to significantly change the magnitude of correlations between nodes.



(a) Correlation plot of nodes trained with Random Forests. Significant correlations exist between nodes.



(b) Correlation plot of nodes trained with PCA for feature extraction and random forests for classification. The correlation between nodes is very low, which is desired for classification, but random forest node models with PCA still did not perform particularly well compared to other workflows.

Figure 5.9: Estimated node correlation for random forest workflows with PCA (5.9b) and without PCA (??). The correlation between nodes is less for PCA, which is to be expected since PCA naturally decorrelates the input data.

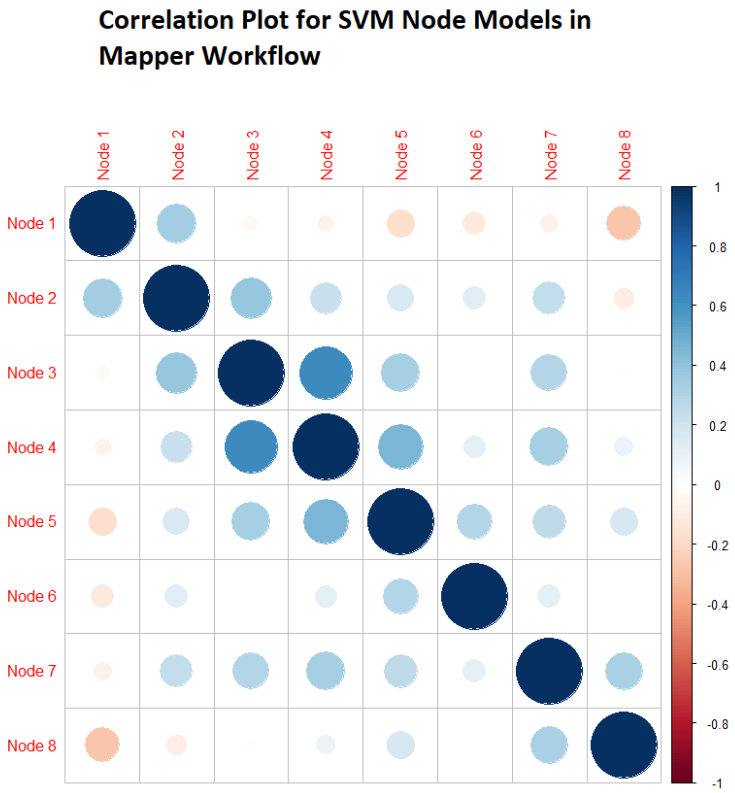


Figure 5.10: Estimated node correlation for SVM workflows without PCA. There is high positive correlation between many of the nodes.

converge, Adaboost models performed well compared to the other workflows tested. This experiment suggests the Mapper based workflow does create a set of uncorrelated workflows.

Figure 5.8 shows the correlation of Logistic Regression node Models with and without PCA. In this case the nodes are more correlated than in Figure 5.7. PCA seems to decrease the correlation between nodes slightly but not significantly. Interestingly, Node 1 tends to be negatively correlated with many of the other nodes. Node 1 is usually the node with the lowest readmission rate, which supports the hypothesis that the different groups of patients have different underlying relationships with readmissions. In general, nodes which are close on the Mapper graph are positively correlated. This makes sense as those nodes overlap in terms of training data.

Figure 5.9 shows the correlation of Random Forest Models with and without PCA. Without PCA, the nodes appear highly correlated, but with PCA most of the correlation disappears. There are still instances where the low-numbered nodes are negatively correlated to the higher numbered nodes. These nodes share no data and are on opposite sides of the general point cloud of patient data explored by Mapper. Even though the models were uncorrelated, the Random Forests with PCA workflows did not perform particularly well compared to Logistic Regression or AdaBoost workflows.

When training nodes using SVM with PCA, the algorithm does not converge. Thus we only present correlation between node models trained without PCA in Figure 5.10. There is high positive correlation between many nodes, which we hypothesize is part of the reason that workflows with Mapper and SVM performed poorly overall.

Overall, Mapper workflows using AdaBoost, Logistic Regression, and Random Forests as node models were able to create ensembles with low correlation between members. In some

cases, PCA is required to decorrelate the models, in others not. We hypothesize in these cases that these models are indeed separable, but future work is necessary to develop conditions for the separability of 0-1 Loss.

5.5 Chapter Recap

In this chapter we developed a workflow on real data to predict 30-Day hospital readmissions. This model improved over standard multivariate regression as well as the LACE score. To our knowledge, models trained directly on Mapper nodes is a novel application of the Mapper algorithm, as it is mainly used for feature selection and exploratory analysis. Some of the Mapper based workflows generated uncorrelated ensembles, which we believe is a necessary condition for separability of classification workflows.

This workflow has some limitations. For example, because there is overlap in the clusters of patients created by Mapper, it is very difficult to prove analytically that the node models are uncorrelated, and can therefore be trained independently. Additionally, we restricted node models to belong to one type of classifier. In the future we could improve this model by performing model selection (also an MLM) within each node. Our study size is limited, so our confidence intervals are higher than we would like, and in future work we would like to train this workflow on more data.

The attempt to analyze the effect of the Mapper algorithm on a workflow was the catalyst for developing the MLM framework. We needed a structure that could encompass deep topological operations while also describing the model training down the line. Further, the MLM as a structure allows us to present an equation representing our workflow when

presenting to medical audiences, which helped us present our results to the readmissions reduction committee at BJH.

Chapter 6

Conclusions and Future Work

In this dissertation, we identified a need for a systematic mathematical representation of machine learning workflows. We propose the Machine Learning Morphism as a framework for this design. MLM's are a tuple containing: (input space, output space, learning morphism, parameter prior, loss function). This contains the information necessary to implement one step in a workflow.

MLM's are endowed with the properties of asymptotic equality and separability. Equality is the foundation for comparing and building algebraic operations on MLM's. Separability is a property depending on the loss function and learning morphism that allows the parameters of an MLM to be computed by optimizing loss functions with lower dimensional parameters. MLM's are also endowed with two forms of composition. Output composition defines a sequential optimization of MLM's in a workflow, while structural composition defines a joint optimization. A workflow is then a finite sequence of compositions of MLM's.

This framework is useful because it provides a modular design where each block has the same underlying structure. Separability is a useful strategy for scalability because many MLM's are asymptotically equivalent to separable MLM's. Specifically, for MSE loss functions, linear morphisms acting on uncorrelated and centered random variables are separable. To

take advantage of this, design a workflow to engineer a set of uncorrelated, centered, variables from the input data; for example, PCR. This extends to non-linear ensembles as well, and a wide variety of MLM's can be approximated with these decompositions.

Further, we can bound the error of workflows using the MLM framework. This error depends on the underlying parameterization and represents a connection between machine learning theory and estimation theory. For MSE loss functions, this bound is a bound on the generalization error of the MLM. For non-MSE loss, we still bound the variance of an MLM around the best possible workflow, which is a useful tool for model selection. For compositions we bound the composition of MLM's, and thus a workflow, by formulating a nonlinear filtering problem. The state of the non-linear filter is the evolution of the bayes model across the workflow, and the observations are the outputs of the MLM. In the case of additive gaussian noise, the error bound is computable, and is a lower bound on the true fisher information. However, in the case of complex workflows the true fisher information is not readily available, this a useful bound.

In future work, we will continue analyzing the structure and building properties of MLM's. Specifically, I am interested in the possibility of vector spaces of workflows. Vector spaces are particularly useful because they have bases, which would represent a set of learning morphisms which span the entire space. This is naturally separable, and hypothesized to approximate many useful workflows. Future work on bounds includes investigating other types of bounds, including Baranking, Bhattacharyya , or Ziv-Zakai. Future work will also develop non-linear filtering paradigms to investigate non-gaussian, non-additive error, starting by approximating non-gaussian workflows with a gaussian nonlinear filter. Finally, we wish to build a method to automate the design of workflows using a graph based structure of MLM's.

We will conclude this dissertation with a discussion of the MLM's place in existing machine learning frameworks. Amazon and Google, as well as many others, are developing software for automated design. Packages such as PyTorch, Tensorflow, or Caret will often perform perprocessing tasks automatically. AutoML is an active area of research, as discussed in Chapter 1. So where does the MLM fit into this?

On one hand, MLM's are a *descriptive* or *explanatory* tool. There is a direct correspondence between the tasks in a software workflow and the space of MLM's. We can use MLM's to identify special properties of tasks, or design equivalent workflows and then implement them in software. We can also the measure the impact that different tasks have on the overall performance, and perform design problems on existing MLM's.

On the other hand, MLM's are a *competitive* tool to these platforms. The set of MLM's and operations of equality, structural composition, and output composition represent the beginning of an ontology of machine learning. We can construct a directed graph of MLM's, where edges represent composition of one MLM with another. Then workflows represent a path in the graph of MLM's, which can be searched according to the metric of interest. The main challenges of this graph representation are: representing MLM's with a suitable data structure and suitable optimization solvers, and directing the flow of information across the graph.

The MLM represents a data structure, which can be implemented in an object oriented way. The MLM object can be endowed with methods such as train and predict, or functions such as composition. The advantage of this structure is that every MLM has the same underlying structure, and the user simply has to specify the input/output space, learning morphism, prior, and loss function. The challenge of an arbitrary structure is that we must provide proper solvers to handle a variety of datatypes and problems.

At the end of it all, though, the MLM represents a way of thinking. What am I learning with each step in my workflow? Can I learn that objective better with a different risk function or learning morphism? Is my workflow asymptotically equivalent to the Bayes MLM? What happens if I add two workflows together? These are the questions and research that the MLM framework inspires.

Appendix A

Proof of results in Chapter 3

A.1 Proof of Theorem 1

Proof of Theorem 1. Define a set of morphisms $F_0 = \theta_0$ and $F_i : \mathbb{X}_i \to \mathbb{Y}, F_i(x_i; \theta_i) = \theta_i x_i$ for $i = 1, ..., k$. Define a corresponding set of MSE loss functions $L_0 = (y - \theta_0)^2$ $L_i = (y - F_i(x_i; \theta_i))^2 = (y - \theta_i x)^2$.

We will check the separability condition by expanding the Expected value of the MSE: Here $F_\Sigma = \theta_0 + \sum_{i=1}^{k} \theta_i^\infty x_i = F_0(\mathbf{x}; \theta_0) + \sum_{i=1}^{K} F_i(x_i; \theta_i)$. To see that we have asymptotic equality,

take the expected value of the loss function:

$$\mathrm{E}(L_\Sigma) = \mathrm{E}((y - F_0 - \sum_{i=1}^{k} F_i)^2) = \tag{A.1}$$

$$E(y^2 - 2yF_0 - 2y\sum_{i=1}^{k} F_i) + F_0^2 + 2F_0\sum_{i=1}^{k} F_i + \sum_{i=1}^{k} F_i \sum_{j=1}^{k} F_j) = \tag{A.2}$$

$$\mathrm{E}(y - 2yF_1 + F_1^2 + F_1\sum_{j\neq 1} F_j - 2y\sum_{i=2}^{k} F_i + \sum_{i=2}^{k} F_i \sum_{j=2}^{k} F_j) + \mathrm{E}(-2yF_0 + F_0^2 + 2F_0\sum_{i=1}^{k} F_i) = \tag{A.3}$$

$$\mathrm{E}(y - 2yF_1 + F_1^2) + \mathrm{E}(F_1\sum_{j\neq 1} F_j) + \mathrm{E}(-2y\sum_{i=2}^{k} F_i + \sum_{i=2}^{k} F_i \sum_{j=2}^{k} F_j) + \mathrm{E}(-2yF_0 + F_0^2) + 0 = \tag{A.4}$$

$$\mathrm{E}(y - 2yF_1 + F_1^2) + 0 + \mathrm{E}(-2y\sum_{i=2}^{k} F_i + \sum_{i=2}^{k} F_i \sum_{j=2}^{k} F_j) + \mathrm{E}(-2yF_0 + F_0^2) = \tag{A.5}$$

$$\mathrm{E}((y - F_1)^2) + \mathrm{E}(y^2 - y^2 - 2y\sum_{i=2}^{k} F_i + \sum_{i=2}^{k} F_i \sum_{j=2}^{k} F_j) + \mathrm{E}(-2yF_0 + F_0^2) = \tag{A.6}$$

$$\mathrm{E}((y - F_1)^2) + \mathrm{E}(y^2 - 2y\sum_{i=2}^{k} F_i + \sum_{i=2}^{k} F_i \sum_{j=2}^{k} F_j) - \mathrm{E}(y^2) + \mathrm{E}(-2yF_0 + F_0^2) = \tag{A.7}$$

$$\mathrm{E}((y - F_1)^2) + \mathrm{E}(y^2 - 2yF_2 + F_2^2 + F_2\sum_{i\neq 2} F_i - 2y\sum_{i=3}^{k} F_i + \sum_{i=3}^{k} F_i \sum_{j=3}^{k} F_j) -$$
$$\mathrm{E}(y^2) + \mathrm{E}(-2yF_0 + F_0^2) = \tag{A.8}$$

$$\mathrm{E}((y-F_1)^2)+\mathrm{E}(y^2-2yF_2+F_2^2)+0+$$

$$\mathrm{E}(-2y\sum_{i=3}^{k}F_i+\sum_{i=3}^{k}F_i\sum_{j=3}^{k}F_j)-\mathrm{E}(y^2)+\mathrm{E}(-2yF_0+F_0^2) \tag{A.9}$$

$$\vdots \tag{A.10}$$

$$\mathrm{E}((y-F_1)^2)+\mathrm{E}((y-F_2)^2)+\cdots+\mathrm{E}((y-F_k)^2)+\mathrm{E}((y-F_0)^2)-(k)\mathrm{E}(y^2) \tag{A.11}$$

This uses the idea that $\mathrm{E}(F_i\sum_{j\neq i}F_j)=\mathrm{E}(\theta_i x_i\sum_{j\neq i}\theta_j x_j)=0$ because x_i and x_j are uncorrelated.

Thus

$$\arg_{\Theta_0\times\Theta_1\times\cdots\times\Theta_k}\min E(L_\Sigma)= \tag{A.12}$$

$$\arg_{\Theta_0\times\Theta_1\times\cdots\times\Theta_k}\min\mathrm{E}((y-\theta_1x_1)^2)+\mathrm{E}((y-\theta_2x_2)^2)+\cdots+$$

$$\mathrm{E}((y-\theta_kx_k)^2)+\mathrm{E}((y-\theta_0)^2)-(k)\mathrm{E}(y^2)= \tag{A.13}$$

$$(\arg_{\Theta_0}\min E((y-\theta_0)^2),\arg_{\Theta_1}\min E((y-\theta_1x_1)^2),\arg_{\Theta_2}\min E((y-\theta_2x_2)^2),\ldots, \tag{A.14}$$

$$\arg_{\Theta_k}\min E((y-\theta_kx_k)^2)) \tag{A.15}$$

since $(k)E(y^2)$ does not depend on the parameters. This fulfills the definition of separability.

□

A.2 Principal Component Regression

Proof. Proof that PCR is equivalent to OLS Regression

We will start by recalling the definitions of workflow $\mathcal{W}$ and $\mathcal{ML}_3$.

Machine learning workflow is an MLM:

$$\mathcal{W} = \mathcal{ML}_2 \circ_O \mathcal{ML}_1 \circ_O \mathcal{ML}_0 = \tag{A.16}$$

$$(\mathbb{R}^k, \mathbb{R}, F_w = (\boldsymbol{\theta}_2)^T \boldsymbol{\theta}_1^{*T}(\mathbf{x} - \boldsymbol{\theta}_0 *) + \theta_{\mu_y}, P(\boldsymbol{\theta}_2)\delta(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_1^*)\delta(\boldsymbol{\theta}_0 - \boldsymbol{\theta}_0^*), L_2(y, F_w)) \tag{A.17}$$

And $\mathcal{ML}_3$ has structure:

$$\mathcal{ML}_3 : (\mathbb{R}^k, \mathbb{R}, F_3 = \bar{\boldsymbol{\theta}}^T \mathbf{x} + \bar{\theta}_0), P(\bar{\boldsymbol{\theta}}, \bar{\theta}_0) = 1, L_3 = (y - F_3)^2) \tag{A.18}$$

Let $\mathbf{X} \in \mathbb{R}^{n \times k}$ be a $n \times k$ matrix of n realizations of $\mathbb{X}$ representing the training data, and let $\mathbf{Y} \in \mathbb{R}^{n \times 1}$ be the corresponding output realizations.

We will begin by examining workflow $\mathcal{W}$. The parameters $\boldsymbol{\theta}_0$ can be estimated via the sample mean of each column:

$$\hat{\theta}_0 = \frac{1}{n} \mathbf{X}^T \mathbf{1} \tag{A.19}$$

Denote the centered variables as

$$\mathbf{X}_c = \mathbf{X} - \mathbf{1}\hat{\theta}_0^T = \mathbf{X} - \frac{1}{n} \mathbf{1}\mathbf{1}^T \mathbf{X} \tag{A.20}$$

The parameters of the principal component analysis are the eigenvectors of the sample covariance matrix:

$$\mathbf{X}_C^T \mathbf{X}_C = \hat{\boldsymbol{\Theta}}_1 \lambda \hat{\boldsymbol{\Theta}}_1 \tag{A.21}$$

We will denote the matrix $\mathbf{X}_C^T \mathbf{X}_C = \mathbf{R}$.

Note that

$$\mathbf{R} = (\mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X})^T(\mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X}) = \tag{A.22}$$

$$\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X} + \frac{1}{n^2}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{1}\mathbf{1}^T\mathbf{X} = \tag{A.23}$$

$$\mathbf{X}^T\mathbf{X} - \frac{2}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X} + \frac{n}{n^2}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X} = \tag{A.24}$$

$$\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X} = \tag{A.25}$$

$$\mathbf{X}^T(\mathbf{X} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{X}) = \tag{A.26}$$

$$\mathbf{X}^T(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T)\mathbf{X} = \tag{A.27}$$

$$\mathbf{X}^T\mathbf{X}_C \tag{A.28}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix.

To estimate the parameters $[\boldsymbol{\theta}_2\, \theta\mu_y]^T$, we define a matrix $\mathbf{Z} = \begin{bmatrix} \mathbf{X}_C\hat{\boldsymbol{\Theta}}_1 & \mathbf{1} \end{bmatrix}$

Then the parameters are given as:

$$\begin{bmatrix} \hat{\boldsymbol{\theta}}_2 \\ \hat{\theta}_{\mu_y} \end{bmatrix} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{Y} = \tag{A.29}$$

$$\left(\begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T \\ \mathbf{1}^T \end{bmatrix} \begin{bmatrix} \mathbf{X}_C\hat{\boldsymbol{\Theta}}_1 & \mathbf{1} \end{bmatrix} \right)^{-1} \begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T \\ \mathbf{1}^T \end{bmatrix} \mathbf{Y} = \tag{A.30}$$

$$\begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T\mathbf{X}_C\hat{\boldsymbol{\Theta}}_1 & \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T\mathbf{1} \\ \mathbf{1}^T\mathbf{X}_C\hat{\boldsymbol{\Theta}}_1 & n \end{bmatrix}^{-1} \begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T \\ \mathbf{1}^T \end{bmatrix} \mathbf{Y} = \tag{A.31}$$

$$\begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\hat{\boldsymbol{\Theta}}_1\Lambda\hat{\boldsymbol{\Theta}}_1^T\hat{\boldsymbol{\Theta}}_1 & \mathbf{0} \\ \mathbf{0}^T & n \end{bmatrix}^{-1} \begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T \\ \mathbf{1}^T \end{bmatrix} \mathbf{Y} = \tag{A.32}$$

$$\begin{bmatrix} \Lambda^{-1} & \mathbf{0} \\ \mathbf{0}^T & \frac{1}{n} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T \\ \mathbf{1}^T \end{bmatrix} \mathbf{Y} = \tag{A.33}$$

$$\begin{bmatrix} \Lambda^{-1}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T\mathbf{Y} \\ \frac{1}{n}\mathbf{1}^T\mathbf{Y} \end{bmatrix} \tag{A.34}$$

As a recap on $\mathcal{W}$ the parameters are given as $\hat{\boldsymbol{\theta}}_0 = \frac{1}{n}\mathbf{X}^T\mathbf{1}$, $\hat{\boldsymbol{\Theta}}_1$ are the eigenvectors of $\mathbf{R}$, $\hat{boldsymbol\theta_2} = \Lambda^{-1}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T\mathbf{Y}$, and $\hat{\theta}_{\mu_y} = \frac{1}{n}\mathbf{1}^T\mathbf{Y}$. An interesting note is that $\hat{\theta}_{\mu_y}$ is the sample mean of $\mathbf{Y}$, which is necessary because centered data $\mathbf{X}_C$ can't capture a non-zero mean.

Now we will compute the optimal parameters of $\mathcal{ML}_3$:

Let $\tilde{\mathbf{X}} = [\mathbf{X}\mathbf{1}]$, then the parameters are given by the OLS Formula:

$$\begin{bmatrix} \bar{\boldsymbol{\theta}} \\ \bar{\theta}_0 \end{bmatrix} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{Y} = \tag{A.35}$$

$$\left(\begin{bmatrix} \mathbf{X}^T \\ \mathbf{1}^T \end{bmatrix}\begin{bmatrix} \mathbf{X} & \mathbf{1} \end{bmatrix}\right)^{-1}\begin{bmatrix} \mathbf{X}^T \\ \mathbf{1}^T \end{bmatrix}\mathbf{Y} = \tag{A.36}$$

$$\begin{bmatrix} \mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{1} \\ \mathbf{1}^T\mathbf{X} & n \end{bmatrix}^{-1}\begin{bmatrix} \mathbf{X} \\ \mathbf{1}^T \end{bmatrix}\mathbf{Y} = \tag{A.37}$$

$$\begin{bmatrix} (\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X})^{-1} & -\frac{1}{n}(\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{1} \\ -\frac{1}{n}\mathbf{1}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X})^{-1} & \frac{1}{n} + \frac{1}{n^2}\mathbf{1}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{1} \end{bmatrix}\begin{bmatrix} \mathbf{X} \\ \mathbf{1}^T \end{bmatrix}\mathbf{Y} = \tag{A.38}$$

$$\begin{bmatrix} \mathbf{R}^{-1} & -\mathbf{R}^{-1}\hat{\boldsymbol{\theta}}_0 \\ -\hat{\boldsymbol{\theta}}_0^T\mathbf{R}^{-1} & \frac{1}{n} + \hat{\boldsymbol{\theta}}_0^T\mathbf{R}^{-1}\hat{\boldsymbol{\theta}}_0 \end{bmatrix}\begin{bmatrix} \mathbf{X} \\ \mathbf{1}^T \end{bmatrix}\mathbf{Y} \tag{A.39}$$

So separating the coefficient and intercept we have:

$$\bar{\boldsymbol{\theta}} = (\mathbf{R}^{-1}\mathbf{X}^T - \mathbf{R}^{-1}\hat{\boldsymbol{\theta}}_0\mathbf{1}^T)\mathbf{Y} = \tag{A.40}$$

$$(\mathbf{R}^{-1}\mathbf{X}^T - \mathbf{R}^{-1}\frac{1}{n}\mathbf{X}^T\mathbf{1}\mathbf{1}^T)\mathbf{Y} = \tag{A.41}$$

$$\mathbf{R}^{-1}\mathbf{X}_C^T\mathbf{Y} = \hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 \tag{A.42}$$

This means that the regression coefficients match up with the optimal parameters of $\mathcal{W}$ as desired.

For the intercept:

$$\bar{\theta}_0 = (-\hat{\boldsymbol{\theta}}_0^T \mathbf{R}^{-1}\mathbf{X}^T + \frac{1}{n}\mathbf{1}^T + \hat{\boldsymbol{\theta}}_0^T \mathbf{R}^{-1}\hat{\boldsymbol{\theta}}_0^T \mathbf{1}^T)\mathbf{Y} = \tag{A.43}$$

$$(-\frac{1}{n}\mathbf{1}^T\mathbf{X}\mathbf{R}^{-1}\mathbf{X}^T + \frac{1}{n}\mathbf{1}^T + \frac{1}{n^2}\mathbf{1}^T\mathbf{X}\mathbf{R}^{-1}\mathbf{X}^T\mathbf{1}\mathbf{1}^T)\mathbf{Y} = \tag{A.44}$$

$$-\frac{1}{n}\mathbf{1}^T\mathbf{X}\mathbf{R}^{-1}(\mathbf{X}^T - \mathbf{X}^T\mathbf{1}\mathbf{1}^T)\mathbf{Y} + \frac{1}{n}\mathbf{1}^T\mathbf{Y} = \tag{A.45}$$

$$-\hat{\boldsymbol{\theta}}_0^T \mathbf{R}^{-1}\mathbf{X}_C^T\mathbf{Y} + \frac{1}{n}\mathbf{1}^T\mathbf{Y} = \tag{A.46}$$

$$-\hat{\boldsymbol{\theta}}_0^T \hat{\boldsymbol{\Theta}}_1 \hat{\boldsymbol{\theta}}_2 + \hat{\theta}_{\mu_y} \tag{A.47}$$

Therefore, the machine learning workflow $\mathcal{W}$ and structural composition $\mathcal{ML}_3$ have the same output and are equivalent. □

Next, we examine the case of comparing $\mathcal{W}_2$ and $\mathcal{W}$. In this workflow, all of the parameters are optimized together, rather than fixing the centering parameters $\boldsymbol{\theta}_0$ and matrix $\boldsymbol{\Theta}_1$. To examine the optimal parameters of this matrix, let $\mathbf{X}$ be an $n \times k$ matrix of training data with corresponding $n \times 1$ output observations $\mathbf{Y}$. Then we can take the derivative of the empirical risk function,

$$\bar{R}_2 = \frac{1}{n}||\mathbf{Y} - ((\mathbf{X} - \mathbf{1}_{n\times 1}\boldsymbol{\theta}_0^T)\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 + \theta_{\mu_y})||_2^2 \tag{A.48}$$

with respect to each set of parameters:

$$\frac{\partial \bar{R}_2}{\partial \boldsymbol{\theta}_0} = \frac{2}{n}\left(\boldsymbol{\Theta}_1\boldsymbol{\theta}_2\mathbf{1}_{n\times 1}^T\mathbf{Y} - \boldsymbol{\Theta}_1\boldsymbol{\theta}_2\mathbf{1}^T X\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 + n\boldsymbol{\Theta}_1\boldsymbol{\theta}_2\boldsymbol{\theta}_2^T\boldsymbol{\Theta}_1^T\boldsymbol{\theta}_0 - n\theta_{\mu_y}\boldsymbol{\Theta}_1\boldsymbol{\theta}_2)\right) \tag{A.49}$$

$$\frac{\partial \bar{R}_2}{\partial \boldsymbol{\Theta}_1} = \frac{2}{n}\left((\boldsymbol{\theta}_0\mathbf{1}_{n\times 1}^T - \mathbf{X}_T)\mathbf{Y}^T + (\mathbf{X}^T\mathbf{X} - \boldsymbol{\theta}_0\mathbf{1}_{n\times 1}^T\mathbf{X} - \mathbf{X}^T\mathbf{1}_{n\times 1}\boldsymbol{\theta}_0^T + \boldsymbol{\theta}_0\boldsymbol{\theta}_0^T)\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 + \theta_{\mu_y}(\mathbf{X}^T\mathbf{1}_{n\times 1} - n\boldsymbol{\theta}_0)\right)\boldsymbol{\theta}_2^T \tag{A.50}$$

$$\frac{\partial \bar{R}_2}{\partial \theta_{\mu_y}} = \frac{1}{n}\left(-\mathbf{1}_{n\times 1}^T\mathbf{Y} + \mathbf{1}_{n\times 1}\mathbf{X}\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 - n\boldsymbol{\theta}_0^T\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 + n\theta_{\mu_y}\right) \tag{A.51}$$

$$\frac{\partial \bar{R}_2}{\partial \boldsymbol{\theta}_2} = \frac{2}{n}\left(-\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{Y} + \boldsymbol{\Theta}_1^T\boldsymbol{\theta}_0\mathbf{1}_{n\times 1}^T\mathbf{Y} + \boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{X}\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 - (\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\boldsymbol{\theta}_0^T\boldsymbol{\Theta}_1 + \boldsymbol{\Theta}_1^T\boldsymbol{\theta}_0\mathbf{1}_{n\times 1}^T\mathbf{X}\boldsymbol{\theta}_1)\boldsymbol{\theta}_2 + \theta_{\mu_y}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1} + n\boldsymbol{\Theta}_1^T\boldsymbol{\theta}_0\boldsymbol{\theta}_0^T\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 - n\theta_{\mu_y}\boldsymbol{\Theta}_1^T\boldsymbol{\theta}_0\right) \tag{A.52}$$

First, we will check that the estimated parameters $\hat{\boldsymbol{\theta}}_0$ from Eq. A.19, $\hat{\boldsymbol{\Theta}}_1$(Eq. A.21), $\hat{\boldsymbol{\theta}}_2$ (Eq.A.34) and $\hat{\theta}_{\mu_y}$ (Equation A.34) set these gradients to zero.

Note that when $\hat{\boldsymbol{\theta}}_0 = \frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}$ is the sample mean, Eq. A.51 simplifies to

$$-\mathbf{1}_{n\times 1}^T\mathbf{Y} + n\theta_{\mu_y} = 0 \tag{A.53}$$

which implies that $\hat{\theta}_{\mu_y} = \frac{1}{n}\mathbf{1}_{n\times 1}^T\mathbf{Y}$ as desired.

Next, looking at Equation A.49,

$$\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2(\mathbf{1}^T\mathbf{Y} - n\theta_{\mu_y}) = \tag{A.54}$$

$$\hat{\boldsymbol{\Theta}}_1\boldsymbol{\theta}_2(\mathbf{1}^T\mathbf{Y} - n\frac{1}{n}\mathbf{1}^T\mathbf{Y}) = \mathbf{0}_{k\times 1} \tag{A.55}$$

and

$$\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2(-\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 + n\hat{\boldsymbol{\theta}}_2^T\hat{\boldsymbol{\Theta}}_1^T\hat{\boldsymbol{\theta}}_0) = \tag{A.56}$$

$$\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2(-\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 + n\hat{\boldsymbol{\theta}}_0^T\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2) = \tag{A.57}$$

$$\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2(-\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 + n\frac{1}{n}\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2) = \mathbf{0}_{k\times 1} \tag{A.58}$$

So evaluating $\frac{\partial \bar{R}_2}{\partial \boldsymbol{\theta}_0}|_{\hat{\boldsymbol{\theta}}_0,\hat{\boldsymbol{\Theta}}_1,\hat{\boldsymbol{\theta}}_2,\hat{\theta}_{\mu_y}=\mathbf{0}_{k\times 1}}$ is also a zero vector.

Next we examine Equation A.50. First note that the term

$$\hat{\theta}_{\mu_y}(\mathbf{X}^T\mathbf{1}_{n\times 1} - n\hat{\boldsymbol{\theta}}_0) = \hat{\theta}_{\mu_y}(\mathbf{X}^T\mathbf{1}_{n\times 1} - n\frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}) = \mathbf{0}_{k\times 1} \tag{A.59}$$

Next, we can express the rest of Eq. A.50 in terms of the centered matrix $\mathbf{X}_C$ defined in Eq.A.20.

$$(\hat{\boldsymbol{\theta}}_0\mathbf{1}_{n\times 1}^T - \mathbf{X}^T)\mathbf{Y}^T = (\frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T - \mathbf{X}^T)\mathbf{Y}^T = -\mathbf{X}_C^T\mathbf{Y}^T \tag{A.60}$$

and

$$\mathbf{X}^T\mathbf{X} - \hat{\boldsymbol{\theta}}_0\mathbf{1}_{n\times k}^T\mathbf{X} - \mathbf{X}^T\mathbf{1}\hat{\boldsymbol{\theta}}_0^T + \hat{\boldsymbol{\theta}}_0\hat{\boldsymbol{\theta}}_0^T = \tag{A.61}$$

$$\mathbf{X}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times k}^T\mathbf{X} - \frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X} + \frac{1}{n^2}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X} = \tag{A.62}$$

$$\mathbf{X}_C^T\mathbf{X}_C = \mathbf{R} \tag{A.63}$$

where $\mathbf{R}$ is defined in Eq. A.22. Continuing

$$\mathbf{R}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 = \tag{A.64}$$

$$\hat{\boldsymbol{\Theta}}_1\Lambda\hat{\boldsymbol{\Theta}}_1^T\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 = \tag{A.65}$$

$$\hat{\boldsymbol{\Theta}}_1\Lambda\hat{\boldsymbol{\theta}}_2 = \tag{A.66}$$

$$\hat{\boldsymbol{\Theta}}_1\Lambda\Lambda^{-1}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C^T\mathbf{Y} = \tag{A.67}$$

$$\mathbf{X}_C^T\mathbf{Y} \tag{A.68}$$

Thus

$$\frac{\partial\bar{R}_2}{\partial\boldsymbol{\Theta}_1}|_{\hat{\boldsymbol{\theta}}_0,\hat{\boldsymbol{\Theta}}_1,\hat{\boldsymbol{\theta}}_2,\hat{\theta}_{\mu y}} = \mathbf{0}_{k\times 1} = (-\mathbf{X}_C^T\mathbf{Y} + \mathbf{X}_C^T\mathbf{Y})\hat{\boldsymbol{\theta}}_2 = \mathbf{0}_{k\times 1}\hat{\boldsymbol{\theta}}_2^T = \mathbf{0}_{k\times k} \tag{A.69}$$

as desired.

Finally, we examine the case of the regression coefficients in Eq. A.52.

$$\frac{\partial \bar{R}_2}{\partial \boldsymbol{\theta}_2}|_{\hat{\boldsymbol{\theta}}_0, \hat{\boldsymbol{\Theta}}_1, \hat{\boldsymbol{\theta}}_2, \hat{\theta}_{\mu_y}} =$$

(A.70)

$$\frac{2}{n}\left(-\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{Y} + \frac{1}{n}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 - \frac{2}{n}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 \right.$$
$$\left. + \hat{\theta}_{\mu_y}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1} + n\frac{1}{n^2}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 - n\frac{1}{n}\hat{\theta}_{\mu_y}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\right) =$$

(A.71)

$$\frac{2}{n}\left(-\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{Y} + \frac{1}{n}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 - \frac{1}{n}\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2\right) =$$

(A.72)

$$\frac{2}{n}\left(\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T(-\mathbf{Y} + \frac{1}{n}\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 - \frac{1}{n}\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2)\right) =$$

(A.73)

$$\frac{2}{n}\left(\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}^T(\mathbb{I}_{n\times n} - \frac{1}{n}\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T)(\mathbf{X}\hat{\boldsymbol{\Theta}}_1\hat{\boldsymbol{\theta}}_2 - \mathbf{Y})\right) =$$

(A.74)

$$\frac{2}{n}\left(\hat{\boldsymbol{\Theta}}_1^T\mathbf{X}_C(\mathbf{X}\hat{\boldsymbol{\Theta}}_1\Lambda^{-1}\hat{\boldsymbol{\Theta}}_1\mathbf{X}_C^T\mathbf{Y} - \mathbf{Y})\right) =$$

(A.75)

$$\frac{2}{n}\left(\hat{\boldsymbol{\Theta}}_1^T(\mathbf{R}^T(\mathbf{R}^{-1})^T\mathbf{X}_C^T\mathbf{Y} - \mathbf{X}_C^T\mathbf{Y})\right) = \mathbf{0}_{k\times 1}$$

(A.76)

In conclusion, the optimal parameters of workflow $\mathcal{W}$ represent a critical point of the gradient of the empirical risk function of the structural composition $\mathcal{W}_2$. If we fix $\hat{\boldsymbol{\theta}}_0 = \frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}$ and the intercept $\hat{\theta}_{\mu_y} = \mathbf{1}_{n\times 1}^T\mathbf{Y}$, Eqs. A.49 and A.51 are zero no matter what the regression coefficients and linear transformation.

Re-examining Eq. A.50 when $\boldsymbol{\Theta}_1$ and $\boldsymbol{\theta}_2$ are free for a critical point we need:

$$-\mathbf{X}_C^T\mathbf{Y} + \mathbf{X}_C^T\mathbf{X}_C\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 = \mathbf{0}_{k\times 1} \tag{A.77}$$

$$\mathbf{X}_C^T\mathbf{X}_C\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 = \mathbf{X}_C^T\mathbf{Y} \tag{A.78}$$

$$\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 = (\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y} \tag{A.79}$$

which are the coefficients of a centered linear regression. If $\boldsymbol{\Theta}_1$ is full rank, we can express $\boldsymbol{\theta}_2 = \boldsymbol{\Theta}_1^{-1}(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y}$ Then we can rexpress Eq. A.52 as:

$$-\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{Y} + \frac{1}{n}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{X}\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 - \frac{1}{n}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}\boldsymbol{\Theta}_1\boldsymbol{\theta}_2 = \tag{A.80}$$

$$-\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{Y} + \frac{1}{n}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{X}\boldsymbol{\Theta}_1\boldsymbol{\Theta}_1^{-1}(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y} - \frac{1}{n}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}\boldsymbol{\Theta}_1\boldsymbol{\Theta}_1^{-1}(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y} = \tag{A.81}$$

$$-\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{Y} + \frac{1}{n}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \boldsymbol{\Theta}_1^T\mathbf{X}_C^T\mathbf{Y} - \frac{1}{n}\boldsymbol{\Theta}_1^T\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y} = \tag{A.82}$$

$$\boldsymbol{\Theta}_1^T(-\mathbf{X}^T\mathbf{Y} + \frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{Y} + \mathbf{X}_C^T\mathbf{Y} - \frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y}) = \tag{A.83}$$

$$\boldsymbol{\Theta}_1^T(-\frac{1}{n}\mathbf{X}^T\mathbf{1}_{n\times 1}\mathbf{1}_{n\times 1}^T\mathbf{X}(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y}) = \mathbf{0}_{k\times 1} \tag{A.84}$$

The term $(\mathbf{X}_C^T\mathbf{X}_C)^{-1}\mathbf{X}_C^T\mathbf{Y}$ are the regression coefficients of a linear regression on centered data.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *The Journal of Machine Learning Research*, 15(1):1111–1133, 2014.

[3] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48, 2013.

[4] Saba Amiri, Sara Salimzadeh, and ASZ Belloum. A survey of scalable deep learning frameworks. In *2019 15th International Conference on eScience (eScience)*, pages 650–651. IEEE, 2019.

[5] Rosa I Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 616–623. IEEE, 1999.

[6] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. In *Institute for Signal and information Processing*, volume 18, pages 1–8, 1998.

[7] Yael Ben-Haim and Elad Tom-Tov. A streaming parallel decision tree algorithm. *Journal of Machine Learning Research*, 11(Feb):849–872, 2010.

[8] Paul Bendich, James S Marron, Ezra Miller, Alex Pieloch, and Sean Skwerer. Persistent homology analysis of brain artery trees. *The annals of applied statistics*, 10(1):198, 2016.

[9] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, 2001.

[10] Ian Blanes, Joan Serra-Sagrista, Michael W Marcellin, and Joan Bartrina-Rapesta. Divide-and-conquer strategies for hyperspectral image processing: A review of their benefits and advantages. *IEEE Signal Processing Magazine*, 29(3):71–81, 2012.

[11] Tomas Borovicka, Marcel Jirina Jr, Pavel Kordik, and Marcel Jirina. Selecting representative data sets. *Advances in data mining knowledge discovery and applications*, pages 43–70, 2012.

[12] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[13] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[14] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[15] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[16] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102, 2015.

[17] Andreas Buja, Dianne Cook, and Deborah F Swayne. Interactive high-dimensional data visualization. *Journal of computational and graphical statistics*, 5(1):78–99, 1996.

[18] Mathieu Carriere, Bertrand Michel, and Steve Oudot. Statistical analysis and parameter selection for mapper. *The Journal of Machine Learning Research*, 19(1):478–516, 2018.

[19] Eric Cawi, Patricio S La Rosa, and Arye Nehorai. Designing machine learning workflows with an application to topological data analysis. *PloS one*, 14(12), 2019.

[20] Eric Cawi, Patricio S. La Rosa, and Arye Nehorai. Conditions for separability of machine learning workflows. *submitted, Journal of Artificial Intelligence Research*, 2020.

[21] Sebastian Celis and David R Musicant. Weka-parallel: machine learning in parallel. In *Carleton College, CS TR*. Citeseer, 2002.

[22] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. Similarity encoding for learning with dirty categorical variables. *Machine Learning*, pages 1–18, 2018.

[23] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[24] Marc Coudriau, Abdelkader Lahmadi, and Jérôme François. Topological analysis and visualisation of network monitoring data: Darknet case study. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

[25] Michael AA Cox and Trevor F Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer, 2008.

[26] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[27] Dua Dheeru and Efi Karra Taniskidou. Uci machine learning repository, 2017.

[28] Jacques Donzé, Drahomir Aujesky, Deborah Williams, and Jeffrey L Schnipper. Potentially avoidable 30-day hospital readmissions in medical patients: derivation and validation of a prediction model. *JAMA internal medicine*, 173(8):632–638, 2013.

[29] Ludovic Duponchel. Exploring hyperspectral imaging data sets with topological data analysis. *Analytica chimica acta*, 1000:123–131, 2018.

[30] Charles Epstein, Gunnar Carlsson, and Herbert Edelsbrunner. Topological data analysis. *Inverse Problems*, 27(12):120201, 2011.

[31] Artur J Ferreira and Mário AT Figueiredo. Incremental filter and wrapper approaches for feature discretization. *Neurocomputing*, 123:60–74, 2014.

[32] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Auto-sklearn: Efficient and robust automated machine learning. In Hutter et al. [49], pages 123–143. In press, available at http://automl.org/book.

[33] Joseph Futoma, Jonathan Morris, and Joseph Lucas. A comparison of models for predicting early hospital readmissions. *Journal of biomedical informatics*, 56:229–238, 2015.

[34] Jennifer Gamble and Giseon Heo. Exploring uses of persistent homology for statistical analysis of landmark-based shape data. *Journal of Multivariate Analysis*, 101(9):2184–2199, 2010.

[35] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

[36] Vicente García, José Salvador Sánchez, and Ramón Alberto Mollineda. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25(1):13–21, 2012.

[37] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[38] Shafie Gholizadeh, Armin Seyeditabari, and Wlodek Zadrozny. Topological signature of 19th century novelists: Persistence homology in context-free text mining. 2018.

[39] Hans P Graf, Eric Cosatto, Leon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade svm. In *Advances in neural information processing systems*, pages 521–528, 2005.

[40] John W Graham, Patricio E Cumsille, and Allison E Shevock. Methods for handling missing data. *Handbook of Psychology, Second Edition*, 2, 2012.

[41] Preeti Gupta, Arun Sharma, and Rajni Jindal. Scalable machine-learning algorithms for big data analytics: a comprehensive review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(6):194–214, 2016.

[42] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

[43] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[44] David Haussler. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory, 1990.

[45] David Haussler, Michael Kearns, and Robert E Schapire. Bounds on the sample complexity of bayesian learning using information theory and the vc dimension. *Machine learning*, 14(1):83–113, 1994.

[46] Anika L Hines, Marguerite L Barrett, H Joanna, and Claudia A Steiner. Statistical brief# 172. *Agency for Healthcare Research and Quality*, 2014.

[47] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[48] Yu-Chi Ho and David L Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570, 2002.

[49] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

[50] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. pages 277–281, 2015.

[51] J-SR Jang and C-T Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE transactions on Neural Networks*, 4(1):156–159, 1993.

[52] Karen E Joynt, Ashish K Jha, et al. A path forward on medicare readmissions. *N Engl J Med*, 368(13):1175–1177, 2013.

[53] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018.

[54] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automatic model selection and hyperparameter optimization in weka. In Hutter et al. [49], pages 89–103. In press, available at http://automl.org/book.

[55] Max Kuhn. The caret package, 2009.

[56] Patricio S. LaRosa. Filtering and recognition of manuscript and printed digits in the principal component space and its orthogonal extensions. Master's thesis, University of Chile, Santiago, 5 2003.

[57] Fei-Fei Li and Jia Li. Cloud automl: Making ai accessible to every business. *Internet: https://www. blog. google/topics/google-cloud/cloud-automlmaking-ai-accessible-everybusiness*, 2018.

[58] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[59] Friedrich Liese and Klaus-J Miescke. Statistical decision theory. In *Statistical Decision Theory*, pages 1–52. Springer, 2007.

[60] Gilles Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

[61] Lian Leng Low, Kheng Hock Lee, Marcus Eng Hock Ong, Sijia Wang, Shu Yun Tan, Julian Thumboo, and Nan Liu. Predicting 30-day readmissions: performance of the lace index compared with a regression model among general medicine patients in singapore. *BioMed research international*, 2015, 2015.

[62] Nicola Lunardon, Giovanna Menardi, and Nicola Torelli. Rose: A package for binary imbalanced learning. *R journal*, 6(1), 2014.

[63] Alexander Maedche and Steffen Staab. Ontology learning. In *Handbook on ontologies*, pages 173–190. Springer, 2004.

[64] John Mandel. Use of the singular value decomposition in regession analysis. *The American Statistician*, 36(1):15–24, 1984.

[65] Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. *Elements of artificial neural networks*. MIT press, 1997.

[66] Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28(1):92–122, 2014.

[67] Klaus-J Miescke and Friedrich Liese. *Statistical Decision Theory: Estimation, Testing, and Selection*.

[68] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[69] Tapan K Nayak. Rao–cramer type inequalities for mean squared error of prediction. *The American Statistician*, 56(2):102–106, 2002.

[70] Oanh Kieu Nguyen, Anil N Makam, Christopher Clark, Song Zhang, Bin Xie, Ferdinand Velasco, Ruben Amarasingham, and Ethan A Halm. Predicting all-cause readmissions using electronic health record data from the entire hospitalization: model development and comparison. *Journal of hospital medicine*, 11(7):473–480, 2016.

[71] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, page 201102826, 2011.

[72] Beata Nowok, Gillian M. Raab, and Chris Dibben. synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software*, 74(11):1–26, 2016.

[73] Randal S. Olson and Jason H. Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In Hutter et al. [49], pages 163–173. In press, available at http://automl.org/book.

[74] Panče Panov and Sašo Džeroski. Combining bagging and random subspaces to create better ensembles. In *International Symposium on Intelligent Data Analysis*, pages 118–129. Springer, 2007.

[75] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[76] Nikita Patel and Saurabh Upadhyay. Study of various decision tree pruning methods with their empirical comparison in weka. *International journal of computer applications*, 60(12), 2012.

[77] Paul Pavlidis and William Stafford Noble. Matrix2png: a utility for visualizing matrix data. *Bioinformatics*, 19(2):295–296, 2003.

[78] Edwin PD Pednault. *Statistical learning theory*. Citeseer, 1997.

[79] Daniel Pop. Machine learning and cloud computing: Survey of distributed and saas solutions. *arXiv preprint arXiv:1603.08767*, 2016.

[80] Petr Pospichal, Jiri Jaros, and Josef Schwarz. Parallel genetic algorithm on the cuda architecture. In *European conference on the applications of evolutionary computation*, pages 442–451. Springer, 2010.

[81] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International journal of computer applications*, 175(4):7–9, 2017.

[82] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):67, 2016.

[83] Ashfaqur Rahman and Brijesh Verma. Novel layered clustering-based approach for generating ensemble of classifiers. *IEEE Transactions on Neural Networks*, 22(5):781–792, 2011.

[84] Fiana Raiber and Oren Kurland. Kullback-leibler divergence revisited. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 117–124, New York, NY, USA, 2017. ACM.

[85] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[86] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[87] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630, 2006.

[88] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[89] Raúl Rojas. Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting. *Freie University, Berlin, Tech. Rep*, 2009.

[90] Baijayanta Roy. All about categorical variable encoding, Apr 2020.

[91] Khader Shameer, Kipp W Johnson, Alexandre Yahi, Riccardo Miotto, LI Li, Doran Ricks, Jebakumar Jebakaran, Patricia Kovatch, Partho P Sengupta, Sengupta Gelijns, et al. Predictive modeling of hospital readmission rates using electronic medical record-wide machine learning: a case-study using mount sinai heart failure cohort. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 276–287. World Scientific, 2017.

[92] Shaohuai Shi, Qiang Wang, and Xiaowen Chu. Performance modeling and evaluation of distributed deep learning frameworks on gpus. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 949–957. IEEE, 2018.

[93] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.

[94] Martin Slawski et al. On principal components regression, random projections, and column subsampling. *Electronic Journal of Statistics*, 12(2):3673–3712, 2018.

[95] Petr Tichavsky, Carlos H Muravchik, and Arye Nehorai. Posterior cramér-rao bounds for discrete-time nonlinear filtering. *IEEE Transactions on signal processing*, 46(5):1386–1396, 1998.

[96] Carl van Walraven, Irfan A Dhalla, Chaim Bell, Edward Etchells, Ian G Stiell, Kelly Zarnke, Peter C Austin, and Alan J Forster. Derivation and validation of an index to predict early death or unplanned readmission after discharge from hospital to the community. *Cmaj*, 182(6):551–557, 2010.

[97] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[98] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[99] Vladimir Naumovich Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[100] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *arXiv preprint arXiv:1912.09789*, 2019.

[101] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[102] Scott I Vrieze. Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, 17(2):228, 2012.

[103] Huazhen Wang, Fan Yang, and Zhiyuan Luo. An experimental study of the intrinsic stability of random forest variable importance measures. *BMC bioinformatics*, 17(1):60, 2016.

[104] Yingxu Wang. On concept algebra: A denotational mathematical structure for knowledge and software modeling. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(2):1–19, 2008.

[105] Yingxu Wang, Yousheng Tian, and Kendal Hu. Semantic manipulations and formal ontology for machine learning based on concept algebra. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 5(3):1–29, 2011.

[106] Larry Wasserman. Topological data analysis. *Annual Review of Statistics and Its Application*, 5:501–532, 2018.

[107] Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[108] Virginia Vassilevska Williams. Multiplying matrices in o (n2. 373) time. *preprint*, 2014.

[109] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[110] Tzu-Tsung Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, 48(9):2839–2846, 2015.

[111] Dongrui Wu, Chin-Teng Lin, Jian Huang, and Zhigang Zeng. On the functional equivalence of tsk fuzzy systems to neural networks, mixture of experts, cart, and stacking ensemble regression. *IEEE Transactions on Fuzzy Systems*, 2019.

[112] Zhang Xuegong. Introduction to statistical learning theory and support vector machines. *Acta Automatica Sinica*, 26(1):32–42, 2000.

[113] Shipeng Yu, Faisal Farooq, Alexander Van Esbroeck, Glenn Fung, Vikram Anand, and Balaji Krishnapuram. Predicting readmission risk with institution-specific prediction models. *Artificial intelligence in medicine*, 65(2):89–96, 2015.

[114] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.