An end to end Machine Learning Model for Compositional Data

Sean Lamont

A thesis submitted for the degree of
Bachelor of Advanced Computing (Research and Development) (Honours)
The Australian National University

October 2019

Except where otherwise indicated, this thesis is my own original work.

Sean Lamont
3 October 2019

to my xxx, yyy (yyy is the people you want to dedicated this thesis to.)

Acknowledgments

Who do you want to thank?

Abstract

Compositional Data Analysis (CoDA) is the study of data which can be interpreted as ratios or counts. Data of this type manifests in many forms, from microbiome species counts to chemical compositions to wealth distributions. Compositional data is known to exist in a simplex, and so many standard statistical analysis methods (which assume Euclidean geometry) are unsuitable. The goal of this thesis is to expand on current dimensionality reduction research, and investigate the use of jointly optimising supervised learning loss with the addition of a reconstruction error term.

There has been some progress in the analysis of this data, most of which centres on a log transformation which embeds the simplex in Euclidean space.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis Statement

I believe A is better than B.

1.2 Introduction

Put your introduction here. You could use \fix{ABCDEFG.} to leave your comments, see the box at the left side.

You have to rewrite your thesis!!!

1.3 Thesis Outline

How many chapters you have? You may have Chapter 2, Chapter **??**, Chapter 4, Chapter 5, and Chapter 6.

1.4 Datasets

3 good example microbiome datasets here:
https://microbiome.github.io/microbiome/Data.html
Using above package and phyloseq R package to process.

1.5 Notes

Good Aitchinson ref:

In addition to techniques and mathematical analysis of the simplex, he also explains:

Hypersphere transformation to directional data is not reasonable (sphere and triangle are topologically different), and further criticises Dirichlet models

Has more small (toy) datasets, and analyses which may be interesting to compare to.

Chapter 2

Background and Related Work

In this chapter we introduce the background and motivation for this project, and related work in the area. A summary of each section is given below.

- Section 2.1: This section details the main motivation for the project, including potential areas of application.

- Section 2.2: Here we introduce Compositional Data, outlining examples and areas of application. We then present the current methods used in the analysis of this data.

- Section 2.4: This section presents the topic of dimensionality reduction. We introduce the most common algorithm, PCA, as well as methods specific to compositional data. We also present several examples to illustrate the advantages of the different approaches.

- Section 2.5: The main applications of supervised learning, regression and classification, are introduced here.

2.1 Motivation

There is an abundance of scientific data which can be considered compositional, which is to say that the components are non-negative and sum to a constant value. Any data which involves percentages or counts summing to a constant falls in this category, and so the scope of such data is very large.

Compositional data does not satisfy the standard assumptions which underlies many common methods of analysis. Despite this, it remains common practice to apply these methods to compositional data. This is particularly true for many microbiome studies, (for which we outline the necessary background in section 2.6). As explained by Gloor et al. [2017], these studies often ignore the compositional structure inherent to the data. Given the recent technological advancements in gene sequencing and the explosion of such data, appropriate analysis is crucial. Even if this structure is acknowledged, current methods for Compositional Data Analysis are severely lacking.

The importance of Compositional Data Analysis is not restricted to microbiome studies, although this is an important area of application. Economic data (e.g. GDP and wealth distribution), chemical compositions, and geology (soil or rock compositions) are just a few examples of data which are compositional. The motivation of this project is thus given by the limitations of current Compositional Data Analysis, and the large scope of applications where it is necessary.

2.2 Compositional Data

2.2.1 Definition

As stated earlier, compositional data consists of a set of non-negative vectors which sum to a constant value (this is usually normalised to be 1 so the components can be interpreted as percentages of a whole). Formally, a compositional matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ will have each row vector $x_i \in \mathbb{R}^d$ as an element of the $d-1$ dimensional simplex, which is defined as:

$$S^{d-1} = \{s \in \mathbb{R}^d : \sum_{i=1}^{d} s_i = c\}$$

for a fixed constant c (that is, each column vector x_i must sum to c). We note that the simplex is of dimension $d-1$ because we can determine the final component from the sum of the rest, given that c is a fixed constant. More precisely, the final component of a compositional vector x_i is clearly $(x_i)_d = c - \sum_{j=1}^{d-1} (x_i)_j$ and so x_i is of dimension $d-1$.

2.2.2 Example: RGB and the 2D simplex

An example which illustrates this concept well is the RGB triangle. Any colour in RGB format can be interpreted as a composition of each of the colours Red, Green and Blue. In the notation above, this corresponds to a vector in $\mathbb{R}^3$. Figure 2.1 gives a visual representation of the simplex of all such vectors.

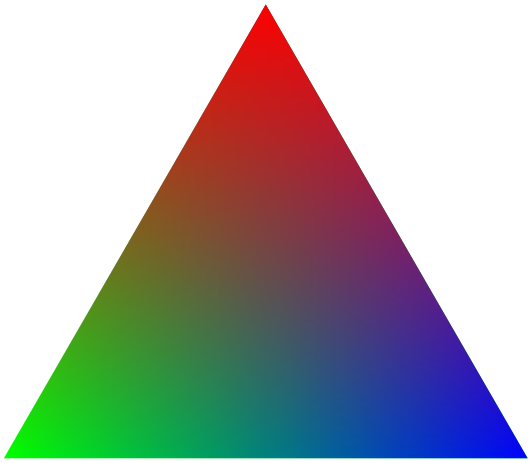
Using this example, we note several points:

- The total magnitude of the vector (R, G, B) is irrelevant to the colour produced. What is important it the relative magnitudes between the values R, G, B. This is why we can normalise compositions to 1 without losing or distorting information.

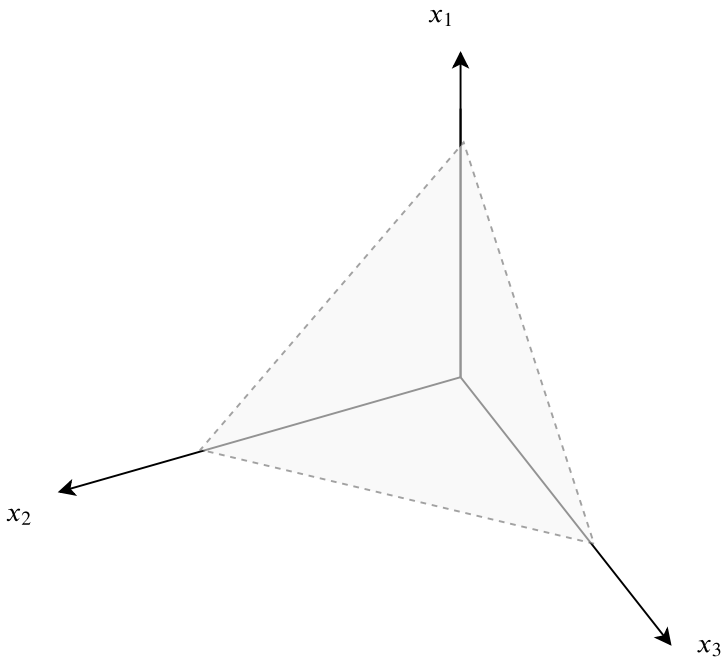
- The simplex is fully defined as a 2D subset of $\mathbb{R}^3$, as it is a triangle. In general, the simplex extends the triangle to higher dimensions.

- The maximum value any of R, G, B can take is given by each of the vertices.

- A gain in any particular R, G, B will correspond to that amount lost between the other values. This is the qualitative interpretation of the restriction that the parts must sum to a fixed value.



(a) An RGB Colour Triangle is an example of a 2D simplex in 3D space



(b) The standard 2D simplex as a subset of $\mathbb{R}^3$. The shaded region corresponds to the set $S^2 = \{\mathbf{x} \in \mathbb{R}^3 : x_1 + x_2 + x_3 = 1\}$

Figure 2.1: The 2D Simplex

2.2.3 Compositional Data Analysis (CoDA)

Given compositional data lies on the simplex, analysis of these data are complicated by the fact that they exist in a non-Euclidean mathematical space. As such, any attempt to compare distances between elements of this space cannot take place in the standard manner. The study of Compositional Data is known as Compositional Data Analysis (CoDA for short) and Aitchison [1982] was the first to treat this issue in depth. At the core of Aitchinson's approach was the idea of a log transformation. For a point on the simplex, applying these transformation embeds the point in Euclidean space facilitating standard distance comparisons. There are several of these transformations, however these all lead to similar results as they only differ by a linear transformation. The most widely used (and most applicable for this project) is the centered log ratio (clr) transformation. Using notation from Avalos et al. [2018], the clr transformation is formally defined as:

$$\mathbf{c}_{KL}(\mathbf{x}) = \log\left(\frac{x}{(\prod_{j=1}^{d} x_j)^{\frac{1}{d}}}\right)$$

We note that $\mathbf{c}_{KL} : S^{d-1} \to \mathbb{R}^d$ is a bijective map from S^{d-1} to $\mathbb{R}^d$ by observing the inverse function which is given by

$$\mathbf{c}_{KL}^{-1}(\mathbf{x}) = e^{\mathbf{x}} \prod_{i=1}^{d} e^{\frac{x_j}{1-d}}$$

...

2.3 Optimisation

Most machine learning problems can fundamentally be considered as optimisation tasks. That is, we wish to find the extrema of a function, potentially given some constraints. For example, a regression problem can be reduced to the minimisation of the l_2 loss. These functions can occasionally be solved analytically, like in the regression case. It is much more common to use gradient based optimisation algorithms, as these analytic solutions are often intractable. Many variants of these algorithms exist, but they are all based on the idea of gradien descent: Given a differentiable function $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, the gradient descent algorithm is: (calculate gradient, specify learning rate, initialise x^*, step $x^{k+1} = x^k - \eta \nabla_x f(x)$)

In certain cases, the objective (i.e. the function to be optimised) is convex and so guaranteed to have only a single minima. In practice, many problems are non-convex and optimisation through gradient based methods is not guaranteed to reach a global minima. There are several ad hoc methods which in practice improve the chances of the model converging, such as vaiable learning rates, multiple random initialisations.

2.4 Dimensionality Reduction

A concept crucial to this thesis is dimensionality reduction. Broadly, this refers to the set of techniques used to represent high dimensional data in a reduced form. Formally, we consider a data matrix $X \in \mathbb{R}^{d \times n}$ with n elements, each with dimension d. The goal of dimensionality reduction is then to find what we denote as a representation matrix $A \in \mathbb{R}^{l \times n}$, where $l < d$. Since $l < d$, each column $x_i \in \mathbb{R}^d$ of X will map to a lower dimensional representation $a_i \in \mathbb{R}^l$. This can be interpreted as projection of x_i to a new coordinate system. In practice, it can be useful to consider A as an encoding of X. We can then reconstruct an approximation to the original matrix X through a decoding matrix $V \in \mathbb{R}^{l \times d}$. Through this we can obtain a reconstruction of X of the same dimension, which is given by $V^T A \in \mathbb{R}^{d \times n}$.

Motivation? (Smaller size, noise, [Shlens, 2005]: physics example, redundancy, signal to noise ratio)

There are many different approaches to dimensionality reduction, which seek to find A based on different criteria and assumptions on the underlying data X. These approaches can be considered as minimising a distortion (generalised distance) between functions of the reconstruction $V^T A$ and the orignal matrix X. This interpretation was particularly helpful for the paper which inspired this project [Avalos et al., 2018]. Below we outline those dimensionality reduction techniques relevant to this research.

2.4.1 Principal Component Analysis (PCA)

The most well known dimensionality reduction method is Principal Component Analysis (PCA) [Mackiewicz and Ratajczak, 1993]. There are several different interpretations of PCA, including SVD, covariance, etc... An intuitive explanation is given by Shlens [2005], which we draw from. PCA seeks to find a representation matrix A where the covariance matrix of A is diagonalised. As explained by Shlens [2005], this is desirable as each basis vector for the new coordinate system is uncorrelated, which minimises the shared information between them. This representation can be...

2.4.2 Exponential Family PCA

Exponential family PCA: Extends PCA analogous to GLMs extending Linear Regression to exponential family. In the same way, allows for loss functions which are more appropriate for the parameter space, where parameters can be non Gaussian. Main example is count data i.e. Poisson which is exponential family. Situation is when data space (e.g. Poisson) is different to parameter space e.g. Reals

2.4.3 CoDA-PCA

This thesis primarily builds upon the 2018 NIPS paper on Representation Learning for Compositional Data [Avalos et al., 2018]. The main result of the paper is the algo-

rithm CoDA-PCA, which is a method for dimensionality reduction of compositional data. The standard approach to dimensionality reduction prior to this result was to apply PCA to the clr transformed data. CoDA-PCA combines clr with exponential family PCA, which allows PCA to be applied to count data. This simultaneously keeps the data centered through clr, and makes use of a loss function which considers the count nature of the data in the reconstruction error. The improvement can be seen dramatically in the ARMS example presented in the paper, where we see that CoDA-PCA and its variants are the only algorithms which faithfully represent the original data [Avalos et al., 2018], and our software can reproduce this result (Chapter 5).

(Maybe have chapter/section on the early experiments with CoDA-PCA, compared to CLR etc. on Aitchinson data)

2.5 Supervised Learning

2.5.1 Regression

2.5.1.1 Definition

One of the most widely used supervised learning methods is Linear Regression. The setup of regression is simply to estimate a target $y \in \mathbb{R}$ through a linear transformation of the features $\mathbf{x} \in \mathbb{R}^d$. For a single observation $\mathbf{x}$, this is done through the weight vector $\mathbf{w}$. The predicted value for $\mathbf{x}$ is then given by:

$$\hat{y} = \mathbf{w}^T\mathbf{x} \tag{2.1}$$

The goal is then to find $\mathbf{w}$ such that the predictions are as close as possible to the ground truth y. There is no single solution to this problem, as different formulations of distance give different results on how "close" the fit is. For example, one might prefer the properties of the l_1 distance $\sum_{i=1}^{n} |\hat{y}_i - y|$ over the l_2. The choice of distance in turn influences the optimal weights, since the weights are chosen to minimise this distance. In most practical applications, the model weights are chosen to optimise the l_2 loss, which we define as

$$\begin{aligned} l_{\text{regression}} &= ||\hat{\mathbf{y}} - \mathbf{y}||_2^2 && (2.2)\\ &= \sum_{i=1}^{n} ||\hat{y} - y||_2^2 && (2.3)\\ &= \sum_{i=1}^{n} ||\mathbf{w}^T\mathbf{x} - y||_2^2 && (2.4)\\ &= ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 && (2.5) \end{aligned}$$

This is often referred to as Ordinary Least Squares (OLS) regression, as it minimises the square l_2 loss. This is the loss function we assume for regression throughout the thesis.

2.5.1.2 Optimal Parameters

The optimal model parameters for the l_2 case can be found by minimising the loss in 2.5 with respect to the weights:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} l_{\text{regression}} = \arg\min_{\mathbf{w}} ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 \tag{2.6}$$

This can easily be solved by matrix calculus:

$$\begin{aligned} ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{y}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{w}^T\mathbf{y} - \mathbf{y}^T\mathbf{y} \end{aligned}$$

Since we are minimising with respect to $\mathbf{w}$, we can omit $\mathbf{y}^T\mathbf{y}$. Since all the terms are scalars, we may also note that $\mathbf{y}^T\mathbf{X}\mathbf{w} = (\mathbf{y}^T\mathbf{X}\mathbf{w})^T = \mathbf{X}^T\mathbf{w}^T\mathbf{y}$.

Therefore:

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{w}} l_{\text{regression}} \\ &= \arg\min_{\mathbf{w}} \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{w}^T\mathbf{y} \\ \nabla_{\mathbf{w}} l_{\text{regression}} &= 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y} \end{aligned}$$

Setting this to 0 gives us:

$$\begin{aligned} 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y} &= 0 \\ \mathbf{X}^T\mathbf{X}\mathbf{w} &= \mathbf{X}^T\mathbf{y} \\ \mathbf{w} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

This solution is a minimum of the loss function $l_{\text{regression}}$ since the Hessian $2\mathbf{X}^T\mathbf{X}$ is positive definite for all $\mathbf{X}$. Therefore the optimal set of weights for the l_2 regression loss is given by

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.7}$$

2.5.1.3 Regression Example:

Here we present an example application of regression,

2.5.1.4 Principal Component Regression

A common approach in regression problems is to apply dimensionality reduction on the features, and use the resulting representation as input for the regression. This is particularly useful for large feature spaces, and can improve model performance and efficiency. We expand on why this occurs in 2.6.2. When PCA is used, this approach is known as Principal Component Regression (PCR).

https://blogs.sas.com/content/iml/2017/10/25/principal-component-regression-drawbacks.html PCR (PCA then Regression), does not consider the response in choosing the representation. No a priori reason to use the components of maximum variance.

2.5.2 Classification

2.5.2.1 Definition

The other key supervised learning problem is classification. In a classification task, the targets $\mathbf{y}$ are now in a discrete set $\{1, ..., k\}$ which we call the classes. Given an observation $\mathbf{x} \in \mathbb{R}^d$, the goal is then to predict which class it belongs to i.e. learn a mapping $\mathbf{x} \to \mathbf{y} \in \{1, ..., k\}$. This is referred to as a multi class classification problem, in contrast to the binary case where $\mathbf{y} \in \{0, 1\}$. There are myriad ways to solve this problem in the literature. We focus on the multi class logistic regression approach, as the associated loss function will be needed for our model.

2.5.2.2 Multi class Logistic Regression

Multi class Logistic Regression approaches the classification problem through combining a softmax function with a linear transformation of the features. Formally, for a vector $\mathbf{x} \in \mathbb{R}^d$ we define softmax as a function on each component $x_i \in \mathbf{x}$:

$$S(x_i) = \frac{e^{x_i}}{\sum_{x_i \in \mathbf{x}} e^{x_i}} \tag{2.8}$$

As with Regression, we have a linear transformation of the features $\mathbf{w}^T\mathbf{x}$. The softmax is then applied to this transformation, which gives the predicted probabilities for each class:

$$\mathbb{P}(\hat{y} = j | \mathbf{w}^T\mathbf{x}) = S((\mathbf{w}^T\mathbf{x})_j) \tag{2.9}$$

$$= \frac{e^{(\mathbf{w}^T\mathbf{x})_j}}{\sum_{i=1}^{k} e^{(\mathbf{w}^T\mathbf{x})_i}} \tag{2.10}$$

The prediction is then taken as the maximum of these probabilities:

$$\hat{y} = \arg\max_j S((\mathbf{w}^T\mathbf{x})_j) \tag{2.11}$$

$$= \arg\max_j \frac{e^{(\mathbf{w}^T\mathbf{x})_j}}{\sum_{i=1}^{k} e^{(\mathbf{w}^T\mathbf{x})_i}} \tag{2.12}$$

2.5.2.3 Negative Log Likelihood

To learn the weights $\mathbf{w}$, we again must optimise a loss function. The loss function which we use is the Negative Log Likelihood. We justify the use of this loss function by considering the likelihood expression for $\mathbf{y}$, which denotes the probability of observing a particular set of targets given the parameters and features:

$$\mathcal{L}(\mathbf{y}) := \mathbb{P}(\mathbf{y}|\mathbf{w}, \mathbf{X}) \tag{2.13}$$

$$= \prod_{j=1}^{n} \mathbb{P}(y_j|\mathbf{w}^T\mathbf{x}_j) \qquad \text{(Since the observations are independent)}$$

$$= \prod_{j=1}^{n} \left(\frac{e^{(\mathbf{w}^T\mathbf{x})_j}}{\sum_{i=1}^{k} e^{(\mathbf{w}^T\mathbf{x})_i}} \right)^{k_i} \tag{2.14}$$

$$\tag{2.15}$$

(add ki for one hot encoding etc.)

To find the optimal weights, the maximum likelihood perspective is to take the weights in such a way as to maximise the above likelihood. Now we note that since log is a monotonically increasing function, the maximum x value of the log likelihood is the same as the maximum of the original likelihood. Maximising log likelihood is a common approach, since it often removes exponents and makes for simpler calculations. From our perspective, we want to find a loss function to minimise. We can express the maximisation of the log likelihood as the minimisation of the negative log likelihood. Hence through minimising the negative log likelihood, we will find the weights which give the maximum likelihood solution. We write this explicitly as:

$$-\log \mathcal{L}(\mathbf{y}) = \tag{2.16}$$

as being which is justified as it is minimising the negative (maximising the positive) of the log (monotone transformation) of the likelihood.

2.6 Microbiome

One of the main fields of interest for Compositional Data Analysis is microbiome studies. At a high level, a microbiome refers to the community of simple organisms (e.g. bacteria) which occupy a particular location on any complex organism (e.g. humans). This is a rapidly growing subfield of genomics, as it can reveal relationships which exist between the microbiome and the external environment. One of the main applications of this includes studying the human microbiome to investigate its connection to health and disease, with the American Gut Project [McDonald et al., 2018] making recent progress in this area.

2.6.1 Genomics and Gene Sequencing

Before we can understand how CoDA is relevant for Microbiome studies, we must cover what exactly the data is and how it is collected. As mentioned above, microbiome data is collected to reveal information about the distribution of smaller organisms in a particular site on another larger organism (the human gut microbiome, for example). The goal is to fully describe this microbial community by the abundance of each organism inside it. Figure xx outlines the high level pipeline of the data retrieval and processing steps to obtain this abundance data. The first step is to sequence the community of interest. There are two broad approaches to this, which are discussed in 2.6.1.1 and 2.6.1.2: Marker gene sequencing (usually 16S sequencing), and metagenomic 'shotgun' sequencing. Once this is completed, the next step is to use the resulting sequences to infer the frequencies of the organisms present in the microbiome. This is dependent on the sequencing method used, and in practice it is difficult to map from sequences to frequencies. One problem is how the sequences are mapped to individual organisms. This is again highly dependent on the sequencing method, which we detail in the following sections.

2.6.1.1 16S Sequencing

Although many species have been identified to date, there are a vast number of microorganisms which do not fit into the current taxonomy. This poses a large problem, as such organisms cannot be classified by reference to those that are currently known.One solution to the problem is to use Operational Taxonomic Units, or OTUs. This relates to one of the common gene sequencing methods, 16S. In this sequencing paradigm, only a single gene (the 'marker' gene) is sequenced for a given sample, which is usually the 16S rRNA gene. This specific gene is used since it is common to all bacteria, and its function has changed minimally over time. Variations in the 16S gene can therefore be linked to the evolution of an organism. For example, a similarity of below 97% between 16S sequences is widely considered to distinguish organisms at the species level (https://jcm.asm.org/content/jcm/45/9/2761.full.pdf). These similarities are then used to construct OTUs, grouping organisms into a single OTU if they fall within a threshold value (e.g. 97% for species). From this, an OTU table is constructed which contains counts of each OTU per sample. This data is intrinsically compositional, since the total read count of the instrument is fixed, and so the OTU counts are relative proportions [Gloor et al., 2017]. These OTUs can be constructed without reference to an existing database, however...

2.6.1.2 Metagenomic Sequencing

There has recently been significant progress in gene sequencing methods, resulting in an abundance of new microbiome data. These methods serve to provide an indication of the relative abundance of microbial species at a particular location (usually stomach or stool). (ref) It is known that microbiome data is inherently compositional, however this is often overlooked in its statistical analysis. In particular, standard PCA

is often applied, which as we have seen can lead to poor representations and therefore misleading conclusions.

Low dimensional representation of compositions good, since microbiomes can have very large feature space. Reduces space, time, avoids curse of dimensionality.

- Metagenomics vs 16S: 16S analyses sequence a single gene across organisms, whereas metagenomic sequencing maps the genome of each organism. This allows for a more detailed representation of the underlying microbiome community. "meta" since it maps all genomes of different organisms. Assembly of genomes (the entire DNA content of an organism, needed for it to "build and maintain the organism") involves piecing together whole genome from sequences of reads of base pairs (using e.g. shotgun sequencing which gets a large collection of fragments, not necessarily linked). Metagenomics is therefore prone to errors, requiring the reads to be organised and mapped to genomes of particular organisms. E.g. Chimeric contigs (contiguous sequences of DNA mapped from different species).

Useful for comparisons between organisms metagenomes, compare taxonomic content between organisms and assess for statistical difference (compositional!) https://www.ncbi.nlm.nih.g

2.6.2 Reconstruction Error

It is widely known that the use of dimensionality reduction prior to the application of supervised learning models can improve performance (see, for example, Howley et al. [2006]). There is some speculation as to exactly why this is the case, as these methods reduce the amount of information present in the data. The improvement in accuracy is mostly seen for high dimensional data, for which the widely known "curse of dimensionality" reduces model performance. The reduction of the input dimension facilitates lower computational cost and faster convergence of algorithms, and so greater performance. In practical cases of small training examples, higher dimensional data prevents many models from recognising patterns in noisy data. It is also claimed that principal components explaining less variance correspond to noise in the data. Under this view, the removal of these components would help to reduce overfitting by ensuring the model is fitted only to the useful parts of the data. Whatever the reason, supervised learning performance can be improved by dimensionality reduction on the features, and this forms the motivation for our model. Conventionally, dimensionality reduction is first performed to obtain a low level representation of X, which we denote as A. The matrix A is then used as input to a supervised learning algorithm. Such an approach constructs A with no regard to the performance of the supervised learning, only optimising based on a reconstruction loss.

2.7 Related work

2.8 Summary

Summary what you discussed in this chapter, and mention the story in next chapter. Readers should roughly understand what your thesis takes about by only reading words at the beginning and the end (Summary) of each chapter.

Chapter 3

Design and Implementation

In this chapter we introduce a novel model for Compositional Data Analysis. It combines dimensionality reduction and supervised learning into a single, end-to-end algorithm. This design is motivated by the drawbacks of traditional methods, such as PCR. Such methods obtain the low level representation independently of the training targets, and so may ignore information which can improve the predictive performance. We divide this chapter into the following sections:

- Section 3.1: Here we present a high level overview of the model, including the assmuptions, inputs and outputs.

- Section 3.2: This section describes in detail the loss function(s) which the model is optimising.

- Section 3.3: The full architecture of the model is outlined here. We also detail the computations involved in a forward pass of the model.

- Section 3.4: The implementation details of the model are presented and discussed here. We also detail the Python package for this code, which is available online.

3.1 Model Overview

The application of our model is to supervised learning problems where the features form a compositional data set. The targets are currently restricted to single real valued numbers (regression), or classification problems (either single or multi class). Formally, the model takes as input a compositional matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with each $x_i \in S^{d-1}$. The output of the model is a prediction $\hat{\mathbf{y}}$ of the true targets $\mathbf{y}$. In the regression case, the targets are $\mathbf{y} \in \mathbb{R}$, and for classification $\mathbf{y} \in \{0, ..., k\}$ where k is the number of classes. The prediction is generated from the following two steps:

- Reduce X to a representation $A \in \mathbb{R}^{n \times l}$ where $l < d$, through the mapping $\Theta(X) = A$.
- Use A as input to the prediction mapping, which we denote as $r(A) = \hat{\mathbf{y}}$.

Introduce notation. How you stack elements into vectors, vectors into matrices, bold, non-bold, etc

We can then define the end-to-end mapping from X to $\hat{\mathbf{y}}$ as the composition of the above functions:

$$f(X) = r(\Theta(X)) = r(A) = \hat{\mathbf{y}} \tag{3.1}$$

3.2 End to End Loss

The steps outlined above for the forward pass are common to many supervised learning approaches, such as PCR which also performs regression on a low level representation. The novelty of our approach is the loss function under which $f(X)$ is optimised. In typical applications, the encoding $\Theta(X)$ and the prediction $r(A)$ are separately optimised with respect to different criteria. To account for this, our model instead directly optimises the composition $f(X)$ under a single loss function.

Careful of bold vs non-bold. E.g. x_i, $\hat{\mathbf{y}}$, $\hat{y}$

To define this loss function we first recall the regression and classification loss from 2.5, and the CoDA-PCA loss from 2.4.3. For predictions $\hat{y}$, targets y, reconstruction Y and geometrically normalised data $\check{X}$ we have:

$$l_{regression} = ||\hat{y} - y||_2^2 \tag{3.2}$$

$$l_{classification} = \tag{3.3}$$

$$l_{CoDA-PCA} = \mathbf{1}^T \exp(\mathbf{Y})\mathbf{1} + \text{tr}(\check{\mathbf{X}}^T\mathbf{Y}) \tag{3.4}$$

3.3 definition

Suggest $l_{\text{regression}}$ instead of $l_{regression}$

We have mentioned that our model is an end to end solution for dimensionality reduction and supervised learning. This is done is by combining the reconstruction loss $l_{CoDA-PCA}$ with a supervised loss, $l_{regression}$ or $l_{classification}$. This combination is taken as a weighted sum of the reconstruction and supervised losses, with a scaling parameter λ. We note that all the above loss functions are differentiable, and so the weighted sum is differentiable.

Formally, given a reconstruction loss l_x, supervised loss l_y, and a scalar $\lambda \in \mathbb{R}$ we define the end-to-end loss as:

$$l_{end-to-end} = l_y + \lambda l_x \tag{3.5}$$

We note that this concept can be applied to any choice of reconstruction or supervised loss l_x or l_y, however we focus on the CoDA case with single valued regression and mulit-class classification. We can now proceed to define the specific loss functions for each case.

3.2.1 CoDA-Regress

In the regression case, we are given targets and predictions $y, \hat{y} \in \mathbb{R}$. We can then define the algorithm $CoDA - Regress$, with the corresponding loss function defined as:

$$l_{CoDA-Regress} = ||\hat{y} - y||_2^2 + \lambda \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \exp Y_{ij} + \text{tr}(\check{X}^T Y) \right) \tag{3.6}$$

3.2.2 CoDA-Cl

Similarly, for a classification problem with targets and predictions $\hat{y}, y \in \{1, ..., k\}$, we define the loss function for the algorithm $CoDA - Cl$:

$$l_{CoDA-Regress} = NLL + \lambda \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \exp Y_{ij} + \text{tr}(\check{X}^T Y) \right) \tag{3.7}$$

3.3 Model Architecture

We are now in a position to fully describe the architecture of the model. Recall the forward pass from 3.1:

$$f(X) = r(\Theta(X)) = r(A) = \hat{\mathbf{y}}$$

Both Θ and r are implemented as fully connected networks, to which we associate the weights $\mathbf{w}_\Theta$ and $\mathbf{w}_r$. We additionally define a decoder network U with weights $\mathbf{w}_U$. This network takes as input the representation $\Theta(X) = A$, and outputs a reconstruction Y of X in the original domain. This autoencoder structure was inspired by the CoDA-AE algorithm in Avalos et al. [2018]. Although not necessary for the forward pass, the reconstruction is needed for optimisation of the model weights. These model weights are given by the union of the above networks:

$$\mathbf{W}_f = \{\mathbf{w}_\Theta\} \cup \{\mathbf{w}_r\} \cup \{\mathbf{w}_U\} \tag{3.8}$$

These weights are optimised with respect to the end-to-end loss functions defined in 3.2. Given these functions are differentiable, they can be optimised by any gradient based optimisation algorithm. Algorithm 1 details the high level training cycle of the model. Figure 3.1 outlines the information flow for computing the combined

loss.

Algorithm 1: coda-to-end: An end-to-end model for CoDA

input: Compositional data matrix X, initialisation matrix I, ground truth y, scalar λ, optimiser O, epochs n
output: Trained model weights $\mathbf{W}_f$
$\mathbf{W}_f = \mathbf{I}$;
$epoch = 0$;
while $epoch < n$ **do**
  $A = \Theta(X)$;
  $\hat{y} = g(A)$;
  $Y = U(A)$;
  $l_{CoDA-PCA} = \mathbf{1}^T \exp(Y)\mathbf{1} + \text{tr}(\check{X}^T Y)$;
  $l_x = l_{CoDA-PCA}$;
  if $y \in \mathbb{R}$ **then**
    $l_y = ||\hat{y} - y||_2^2$;
  else
    $l_y = NLL$;
  end
  $l_{coda-to-end} = l_y + \lambda l_x$;
  $\mathbf{W}_f = O(\mathbf{W}_f, l_{coda-to-end})$;
  $epoch = epoch + 1$;
end

If we only optimise this network with respect to the supervised loss l_y, it leads to representations A which do not faithfully represent the data. We see this result experimentally in **??**. This can then lead to a decrease in the performance of the regression, since the representation is not accurate. This motivates the addition of the reconstruction loss l_x, which in many ways is similar to a regularisation term. By ensuring the encoding will not deviate too far from the ideal representation, the model performance can be improved.

3.4 Implementation

The model defined in 3.3 was implemented in PyTorch, with the source code available on GitHub. A high level outline of the implementation is as follows: The algorithms CoDA-Regress and CoDA-Cl form the basis for the code, and are each implemented as a separate class. The user defines the desired low level dimension, as well as the encoder and decoder dimensions. Using these parameters, an ordered dictionary is used to stack the layers for the encoder and decoder to ensure the correct ordering

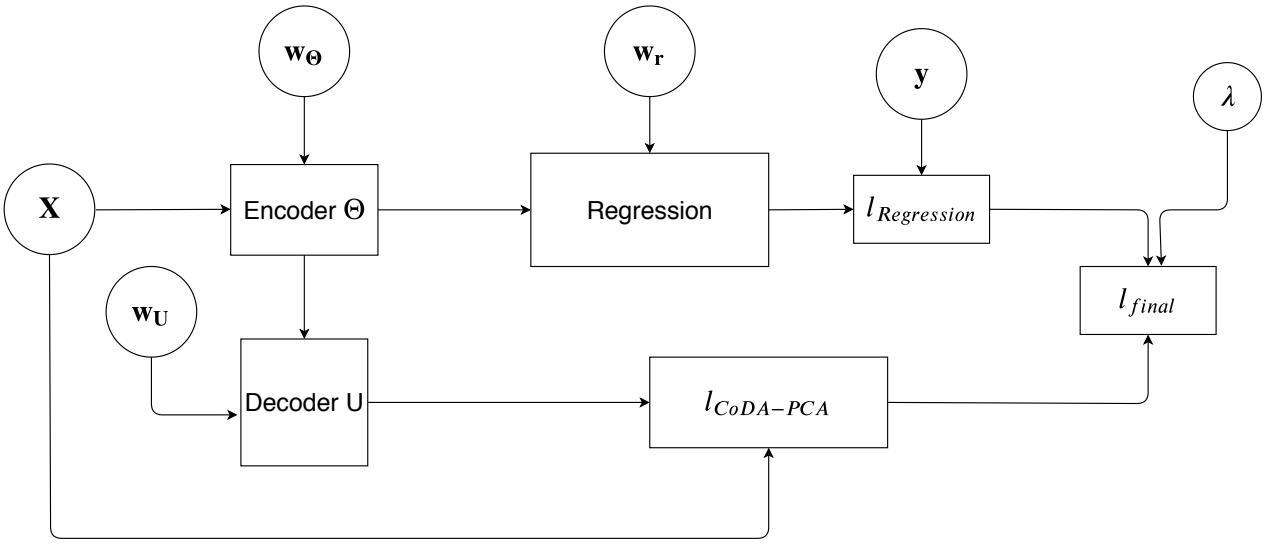


Figure 3.1: End to End ("Coda-to-end") Model Diagram

of dimensions. This includes the nonlinearities, for which we use Exponential Linear Units (ELUs). These dictionaries are then used as input to the PyTorch Sequential module to format them as PyTorch objects.

The output layer is algorithm specific. CoDA-Regress has a linear layer mapping to a single output, which is the predicted regression value. CoDA-Cl has a linear layer mapping to the number of classes followed by a log softmax layer, which gives the log probabilities for each class.

The forward pass is as described in 3.3, with the input being modified to contain both the original matrix X and $X_{ckl} = \log \check{X}$. This is done as per Avalos et al. [2018].

The optimisation is done using Adam []. Referring to algorithm 1, this takes the place of the function O. We found this to be more numerically stable during experiments in comparison to standard Stochastic Gradient Descent (SGD), in addition to faster convergence.

3.4.1 CoDA-PCA Package

We see from [Avalos et al., 2018], CoDA-PCA gives a significant improvement in dimensionality reduction over current methods. Although the code for this paper is available, it has not yet been packaged to production standards. We have reimplemented and packaged the code from [Avalos et al., 2018], along with the CoDA-Regress and CoDA-CL algorithms presented here. These are available on the Python Packaging Index (PyPI), with easy installation using pip. These algorithms can be

accessed using a simple API, which allows users to benefit from their advantages over related algorithms.

add documentation etc. of package?

3.5 Summary

We have introduced the algorithms CoDA-Regress and CoDA-Cl for supervised learning problems with compositional features. Given a feature vector $\mathbf{X}$ and target vector $\mathbf{y}$, these models fit predictions using a low dimensional representation of the original data. The final loss is computed as the sum of the supervised loss l_y and the reconstruction loss l_x, with the reconstruction loss being scaled by a real valued parameter λ.

Chapter 4

Experimental Methodology

4.0.1 Experiments

In this section we present the results from experiments on the combined model. Experiments were done for both regression and classification tasks using several data sources. We present the results of our model in comparison to several baselines on each of these sources.

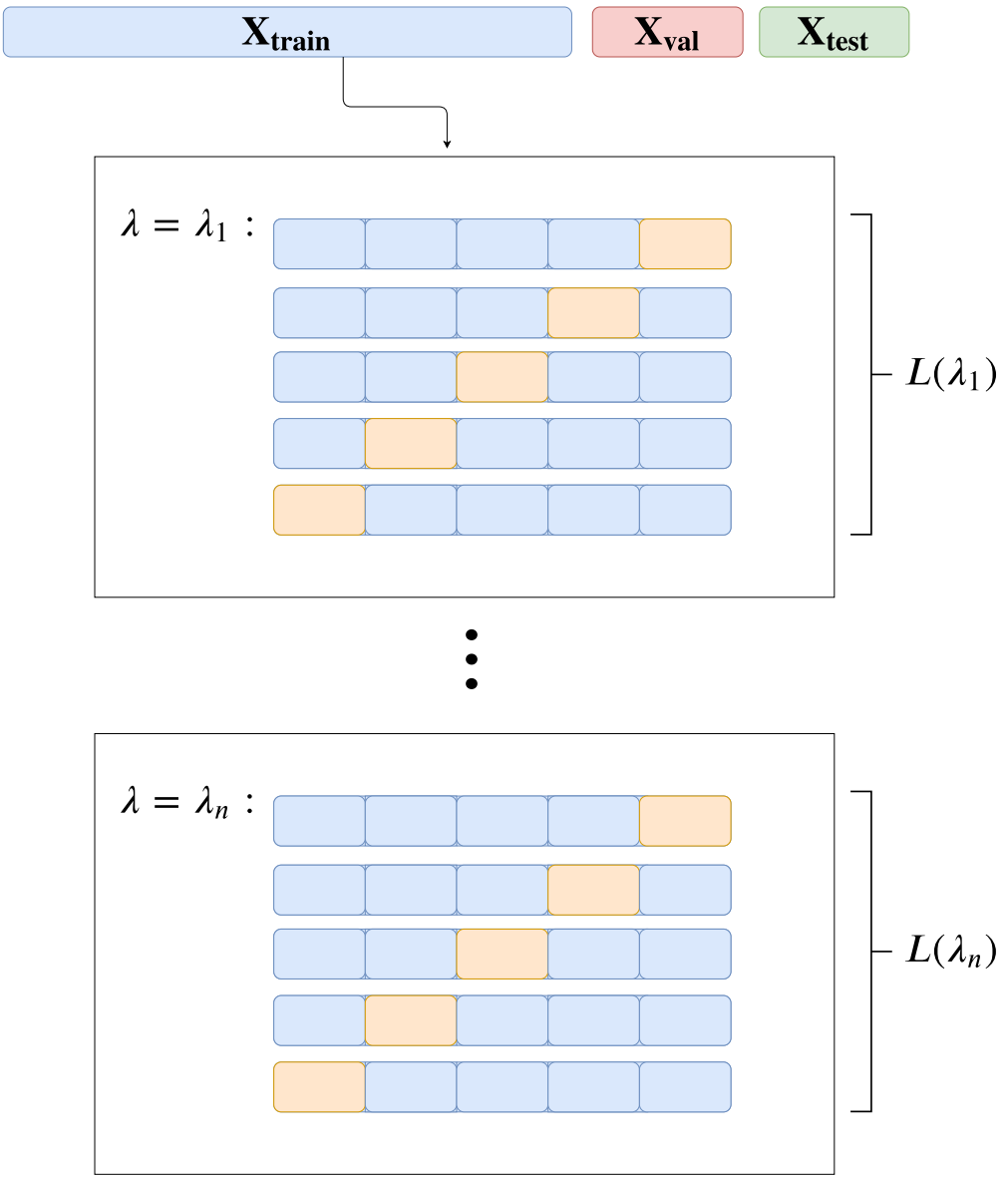
4.0.1.1 Model Evaluation

The results for the smaller Aitchinson datasets were obtained using k-fold cross validation combined with an inner hyperparameter selection loop.

K-Fold cross validation is the standard approach for model evaluation in most machine learning contexts. The procedure is as follows: Choose k, which is the number of groups to split the data into. Shuffle the data, split into k groups and take one as the test set. Train the model on the $k-1$ groups, and evaluate the performance on the test set. Repeat this procedure, changing which group is used for the test set.

This approach is highly unbiased, since the data is shuffled and so trained and tested on separate partitions of the data. In order for the model to perform well overall, it must do well on each partition of the data. This prevents a good model score being due to a lucky initial testing split. Likewise, if a model performs well on only some splits, then this suggests it does not generalise well.

Given the sensitivity of our model to the choice of λ, one solution is to use K-Fold cross validation with an additional inner loop for hyperparameter selection. The high level idea of this approach is as follows: Given a set of hyperparameters $\Lambda = \{\lambda_1, .., \lambda_n\}$. For each training partition of the initial K-Fold, X_k we do another cross validation of X_k, for each $\lambda \in \Lambda$. The value of λ which results in the best mean cross validation score is then used to train the model on X_k. Since we still do not have access to the testing data, this method will still give valid results. There is a large computational cost to this approach, with training taking $O(k|\Lambda|)$. This method, found in **?** is primarily useful for small datasets, due to the high associated computational costs. As such, it was performed on the initial testing of the model with the Aitchinson data.



$$\lambda^* = \arg\min_{\lambda_i \in \Lambda} L(\lambda_i)$$

Figure 4.1: Hyperparameter selection loop for model evaluation: $\mathbf{X}_{\text{train}}$ is partitioned (here into 5). The final λ^* is selected as the best performing λ_i on the mean inner cross validation score, $L(\lambda_i)$. λ^* is then used for training on $\mathbf{X}_{\text{train}}$, and this is repeated for each training split in the outer cross validation loop.

4.0.2 Data Sources

4.0.2.1 Aitchinson Data

The first data source that we test on is the supplementary data presented by Aitchinson [ref.]. This data is commonly used in CoDA research, and consists of 40 small sample (up to 92) compositional datasets. As part of this project, this data was digitised and uploaded to Zenodo. Zenodo is an open access research repository, and so this will allow other researchers to easily access this data in the future. The digitised data can be found here..

4.0.2.2 Metagenome Data

The next source was from a recent paper in Nature [Stewart et al., 2019], which provided the metagenome analyses from several hundred cattle rumen. The depth of each genome (which acts as a proxy for species count) is given for each of the cattle in the sample. The collection of depths per cattle is compositional due to the capacity of the sequencing device being limited, as we saw from Gloor et al. [2017]. There were 4941 genomes identified between the 283 cattle. This data provides a good opportunity for testing the model on high dimensional data, since $d >> n$ for dimension d and sample size n.

4.0.2.3 Additional Microbiome Data

4.0.3 Model Tuning

4.0.3.1 Early Stopping

Add figures and cross val results showing dip in training loss, increase in val loss and decrease in cross val performance A useful technique in the training of machine learning algorithms is early stopping. This is often employed to reduce model overfitting, as the model weights can become overly optimised on the training data if trained for too long, and so lose their ability to generalise. Early stopping is a simple solution to this; it terminates the training early if the training and validation loss diverge.

4.0.3.2 Numerical Stability

The implementation of Machine Learning algorithms on digital computers often leads to numerical instability. Given that real numbers can only be approximated to a finite precision with floating point numbers, operations on very large or small numbers can lead to numbers beyond the scope of this representation. In addition, the finite precision means that there is an inherent error associated with operations on these numbers. These errors accumulate, and can lead to results which do not align with what it would be analytically. Even when a high precision is used, it is well known that matrix operations are particularly prone to numerical instability.

Given that these form the basis for many machine learning algorithms, numerical analysis is an important consideration.

clamping, NaN, exploding gradients, vanishing gradients, learning rates, local optima

Chapter 5

Results

TODO

Experiments In this chapter we present the results of the experiments performed on the end to end model in 3.1. The goal of this section is to present the conditions under which this type of model performs well, and likewise to determine when it does not.

- Section 5.1: This section covers the application of our model to a simple toy dataset, designed to test the hypothesised advantages over traditional methods.

5.1 Toy Example: Gaussian data

The first situation where we will test our model is a synthetic dataset. We want this data constructed in such a way as to test the underlying hypothesis motivating the model in 3.1. That is, our end to end model can improve performance in cases where standard dimensionality reduction removes potentially useful information.

In order to appropriately test this, we therefore want our data to contain useful information which we expect standard methods to ignore. The approach we take to construct this is to take a d dimensional Gaussian, with mutually independent components for simplicity. The covariance matrix is then composed of the diagonal variance entries of each dimension. The important point here is to set the variance of each dimension to be largely different in magnitude. Since the covariance between dimensions is zero, we would expect the coordinate system of dimensionality reduction methods to be focused on the axes with the largest variance. We expect this coordinate system to contain little information on the lower variance dimensions. If the supervised learning target is a function of these lower dimensions, we would therefore expect to see low performance when using these as features. In contrast, the use of a combined model has the signal of supervised loss as a factor in the choice of representation. It should choose a representation which still preserves the important information in the lower variance features. So we expect to see an improvement in performance for this type of data when using our model in comparison to baseline methods.

Formally, we generate this data by taking n draws from a d dimensional Gaussian denoting the variance for each component (i.e. the diagonals of the covariance matrix) as $\sigma_1, ..., \sigma_d$. From above, we note that $\sigma_1 >> ... >> \sigma_d$. Then we define our $n \times d$ data matrix $\mathbf{X}$, with each $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \Sigma)$. Since CoDA requires non-negative values, we scale each axis by the minimum value if it is negative. Our targets will then be a function of the least variant axis, $\mathbf{y} = f(\mathbf{x}_j), j = 1, ..., d$ (where j is indexing $\mathbf{X}$ by columns i.e. axes).

Figure **??** shows an example of synthetic data constructed in such a way. We note the large difference in the spread of the data between the axes, and

Implementing this, we

From our results, we can clearly see the advantage of an end to end model over baselines when the targets are a function of the least variant axis. An interesting experiment to follow this is to invetigate how this performance differential changes as the targets become a function of more variant axes.

add results

discuss

5.1.1 Hyperparameter Grid Search

Below we present the results from the 3 trial hyperparameter grid search protocol we introduced in 4.

5.2 CoDA-PCA Microbiome data

Smallish for both dimensions, uses OTUs at genus level

Still no improvement over baseline using end to end. Even with larger sample size (1000), performance is at best the same, often much lower.

5.3 RUG

New microbiome results, using species level - Poor performance, can't do better than chance - Why?: Sheer depth of connections needed in 'fat' matrices to map them to a lower dimension. The small sample size does not allow for enough information for the network to establish meaningful relationships between the input, low level representation and the output. - Preprocessing with dim reduction works better because the dimensionality reduction algorithms are fixed, and there is no need to learn them unlike in the network. - Can see from 2D/3D plots there is not enough information in these dimensions to meaningfully distinguish classes. Also note the different representations of PCA and CoDA-PCA

A lot of modern results are of this 'fat' form, given the small sample sizes and large diversity in microbiome species. To improve performance/reduce size it may make sense to do analysis at the genus or higher level.

https://microbiomedb.org/mbio/app/ other sources if needed

5.4 Summary

Chapter 6

Conclusion

Summary your thesis and discuss what you are going to do in the future in Section 6.1.

6.1 Future Work

Good luck.

Bibliography

Aitchison, J., 1982. The statistical analysis of compositional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44, 2 (1982), 139–177. http://www.jstor.org/stable/2345821. (cited on page 7)

Avalos, M.; Nock, R.; Ong, C. S.; Rouar, J.; and Sun, K., 2018. Representation learning of compositional data. In *Advances in Neural Information Processing Systems 31* (Eds. S. Bengio; H. Wallach; H. Larochelle; K. Grauman; N. Cesa-Bianchi; and R. Garnett), 6679–6689. Curran Associates, Inc. http://papers.nips.cc/paper/7902-representation-learning-of-compositional-data.pdf. (cited on pages 7, 8, 9, 17, and 19)

Gloor, G. B.; Macklaim, J. M.; Pawlowsky-Glahn, V.; and Egozcue, J. J., 2017. Microbiome datasets are compositional: And this is not optional. *Frontiers in Microbiology*, 8 (2017), 2224. doi:10.3389/fmicb.2017.02224. https://www.frontiersin.org/article/10.3389/fmicb.2017.02224. (cited on pages 4, 13, and 23)

Howley, T.; Madden, M. G.; O'Connell, M.-L.; and Ryder, A. G., 2006. The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data. *Knowledge-Based Systems*, 19, 5 (2006), 363 – 370. doi:https://doi.org/10.1016/j.knosys.2005.11.014. http://www.sciencedirect.com/science/article/pii/S0950705106000141. AI 2005 SI. (cited on page 14)

Mackiewicz, A. and Ratajczak, W., 1993. Principal components analysis (pca). *Computers & Geosciences*, 19, 3 (1993), 303 – 342. doi:https://doi.org/10.1016/0098-3004(93)90090-R. http://www.sciencedirect.com/science/article/pii/009830049390090R. (cited on page 8)

McDonald, D.; Hyde, E.; Debelius, J. W.; Morton, J. T.; Gonzalez, A.; Ackermann, G.; Aksenov, A. A.; Behsaz, B.; Brennan, C.; Chen, Y.; DeRight Goldasich, L.; Dorrestein, P. C.; Dunn, R. R.; Fahimipour, A. K.; Gaffney, J.; Gilbert, J. A.; Gogul, G.; Green, J. L.; Hugenholtz, P.; Humphrey, G.; Huttenhower, C.; Jackson, M. A.; Janssen, S.; Jeste, D. V.; Jiang, L.; Kelley, S. T.; Knights, D.; Kosciolek, T.; Ladau, J.; Leach, J.; Marotz, C.; Meleshko, D.; Melnik, A. V.; Metcalf, J. L.; Mohimani, H.; Montassier, E.; Navas-Molina, J.; Nguyen, T. T.; Peddada, S.; Pevzner, P.; Pollard, K. S.; Rahnavard, G.; Robbins-Pianka, A.; Sangwan, N.; Shorenstein, J.; Smarr, L.; Song, S. J.; Spector, T.; Swafford, A. D.; Thackray, V. G.; Thompson, L. R.; Tripathi, A.; Vázquez-Baeza, Y.; Vrbanac, A.; Wischmeyer, P.; Wolfe, E.; Zhu, Q.; ; and Knight, R., 2018. American gut: an open platform for citizen science microbiome research. *mSystems*, 3, 3 (2018).

doi:10.1128/mSystems.00031-18. https://msystems.asm.org/content/3/3/e00031-18. (cited on page 12)

Shlens, J., 2005. A tutorial on principal component analysis. *Systems Neurobiology Laboratory, Universityof California at San Diego*, 82 (2005), 2224. https://www.cs.cmu.edu/~elaw/papers/pca.pdf. (cited on page 8)

Stewart, R. D.; Auffret, M. D.; Warr, A.; Walker, A. W.; Roehe, R.; and Watson, M., 2019. Compendium of 4,941 rumen metagenome-assembled genomes for rumen microbiome biology and enzyme discovery. *Nature Biotechnology*, 37, 8 (Aug. 2019), 953–961. https://doi.org/10.1038/s41587-019-0202-3. (cited on page 23)