FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

SPARSE METHODS FOR LATENT CLASS ANALYSIS, PRINCIPAL COMPONENT ANALYSIS AND REGRESSION WITH MISSING DATA

By

RASHAD AZIZ

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2023

Rashad Aziz defended this dissertation on August 15, 2023.

The members of the supervisory committee were:

Adrian Barbu
Professor Directing Dissertation

Anke Meyer-Baese
University Representative

Yiyuan She
Committee Member

Jinfeng Zhang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

ACKNOWLEDGMENTS

I wish to thank my advisor, committee, department, and my friend Bruce Wang.

I would also like to acknowledge the staff of the Research Computing Center for their invaluable support of research activity at FSU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Principal Component Analysis (PCA) and Latent Class Analysis (LCA) are multivariate methods commonly used for exploratory data analysis and data preprocessing in unsupervised settings. Two difficulties often encountered in data analysis are missingness and high-dimensionality. This thesis builds on the existing work for feature selection in PCA and tests new strategies for achieving sparsity in PCA estimation. The missing data problem is examined for PCA by introducing two techniques, each informed by a different paradigm of missing data methodology. It is shown that the accuracy of sparse dimension reduction in the face of missing data and irrelevant features can be improved by the techniques introduced in this work. Feature selection is also examined for binary LCA. Data experiments show that our approach to LCA feature selection is not only viable but sometimes necessary to achieve satisfactory results.

CHAPTER 1

INTRODUCTION

Principal Component Analysis (PCA) and *Latent Class Analysis (LCA)* are two widely used multivariate statistical techniques. The goal of PCA is to learn a lower-dimensional representation of continuous data, while LCA is a probabilistic approach to finite mixture modeling.[1]

Modern data challenges introduce complications that must be addressed in order to use these methods. In this thesis we will examine the use of sparsity in model estimation to handle high-dimensional data for PCA and LCA. We will also examine the problem of applying PCA and its sparse variants to data with missing values.

Given n-by-p data, both PCA and LCA involve the estimation of a p-by-k matrix $\mathbf{M}$ where k is the desired rank for PCA, or the desired number of latent classes for LCA. The elements of this matrix estimate the role of each of the data's p features in the model. In settings with large p, it is often desirable to find a model that uses a subset of $p' < p$ features that still permits model accuracy. This situation can occur if, for instance, many features are suspected to be irrelevant to the low-dimensional structure or the latent class structure of the data. The problem of choosing the best p' predictors from a total of p is *feature selection.* A benefit of feature selection is to produce interpretable models: by reducing the size of each model's $\mathbf{M}$ estimate to a much smaller p'-by-k matrix, the parameter becomes more manageable to inspect.

Feature selection in PCA can be thought of in terms of the selection of rows in the model's loading matrix $\mathbf{M}$, where row i captures the role of column i in the data. Having a row of zero's indicates that the corresponding feature is estimated to have no role in the model, and so the question of selecting features in PCA can be seen through the lens of imposing sparsity on the loading matrix. One approach to this is to truncate small elements in the loading matrix to be exactly zero, but that approach can yield misleading results (Cadima and Jolliffe, 1995). A more sophisticated approach is *Sparse Principal Component Analysis (SPCA)* (Zou et al., 2006), which uses sparse regression techniques to estimate a sparse loading matrix for PCA. This approach has

[1] We examine LCA for binary data, but the model can be defined for continuous data types. PCA is traditionally applied to continuous data and is often presented with the loose assumption of multivariate normality, but has seen extensions to other types, e.g. Zhang (2016).

been shown to be consistent under certain sparse assumptions on high-dimensional data matrices where PCA fails (Shen et al., 2013).

In this thesis, we examine sparse PCA using another sparse regression technique, *Feature Selection with Annealing (FSA)* (Barbu et al., 2017). We will propose and test a number of approaches to learning a sparse loading matrix. We will use simulated and real data to compare the performance of ordinary PCA, SPCA, a sparse PCA approach called JSPCA (Yi et al., 2017), and several sparse PCA variants we introduce. We primarily use *Principal Component Regression (PCR)* data, which allows us to compare model performance in terms of predictive accuracy.

We also examine how sparse PCA methods can be applied to data with missingness. We develop an imputation strategy that can be applied to every sparse PCA model by leveraging that model's estimate of the data's low-rank structure. The imputed data is then used in subsequent model estimation steps which require fully-observed data. We also develop an estimation approach that does not impute values, but that only learns from the data entries that are observed. These two approaches to handling missing data are both fairly common, but are also very different. By implementing both strategies in the same model, we will make a direct comparison of the performance and challenges presented by each approach.

Feature selection in the clustering setting has seen much less attention than in supervised settings. Further, feature selection for clustering commonly focuses on continuous data such as a Gaussian mixture modeling context, rather than considering categorical data. We examine the feature selection problem for LCA in a binary data setting. We will introduce a framework for joint LCA model estimation and feature selection via FSA that permits the choice of various metrics of feature relevance. We will use simulated and real data to compare the accuracy of these different approaches against each other, LCA without feature selection, and LCA using feature selection by stepwise selection. A final experiment considers the *Factor Mixture Model (FMM)* (Clark et al., 2013) in a data setting that requires simultaneous application of feature selection for LCA, sparse PCA, and the mitigation of missing values.

Setup and Notation

$\mathbf{X}, \mathbf{U}, \boldsymbol{\Gamma}$	Matrices are denoted by bold uppercase letters
$\mathbf{X}_{i\bullet}$	Row i of matrix $\mathbf{X}$
$\mathbf{X}_{\bullet j}$	Column j of matrix $\mathbf{X}$
$\mathbf{X}_{ij}$	Entry at row i and column j of matrix $\mathbf{X}$
$\mathbf{w}, \mathbf{x}, \boldsymbol{\gamma}$	Vectors are denoted by bold lowercase letters
$\mathbf{w}_i$	Entry i of vector $\mathbf{w}$
i, n, p	Scalars are denoted by lowercase letters
$\|x\|$	Absolute value of x
$\mathbf{I}$	Identity matrix
$\vec{\mathbf{0}}$	Vector of 0's
λ	Penalty parameter
λ^j	Penalty parameter corresponding to the jth index of the item penalized
$\hat{\mathbf{X}}$	Estimator for $\mathbf{X}$
$\mathbf{X}^0, \mathbf{x}^0$	Matrix $\mathbf{X}$, vector $\mathbf{x}$ with missing entries replaced with 0
$\mathbf{X}'$	Transpose of $\mathbf{X}$
$Tr\mathbf{X}$	Trace of $\mathbf{X}$
$\\|\mathbf{X}\\|_0$	The number of nonzero entries in $\mathbf{X}$
$\\|\mathbf{X}\\|_1$	$\sum_{i,j} \lvert\mathbf{X}_{ij}\rvert$
$\\|\mathbf{x}\\|_2$	$\sqrt{\sum_i \mathbf{x}_i^2}$
$\\|\mathbf{X}\\|_F$	$\sqrt{\sum_{i,j} \mathbf{X}_{ij}^2}$
$\\|\mathbf{X}\\|_{2,1}$	$\sum_i \sqrt{\sum_j \mathbf{X}_{ij}{}^2}$
x_+	$\begin{cases} x, & x > 0 \\ 0, & \text{Otherwise} \end{cases}$
$Sign(x)$	$\begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$
$\lfloor x \rfloor$	The largest integer not greater than x

CHAPTER 2

BACKGROUND

The basis for the methods discussed in this thesis is Principal Component Analysis, which we briefly introduce. This is followed by an overview of sparsity methods for regression, and how they can be embedded within PCA estimation to produce sparsity.

2.1 Principal Component Analysis

The objective of dimension reduction is to transform a matrix $\mathbf{X}$ with n instances and p features into a reduced form $\mathbf{\Gamma}$ with n instances and k features, where $k < p$, but $\mathbf{\Gamma}$ manages to retain a large amount of the information in $\mathbf{X}$. The dimension reduction approach we consider involves producing columns of $\mathbf{\Gamma}$ from linear combinations of columns of $\mathbf{X}$. Thus we could produce $\mathbf{\Gamma}$ as a product of $\mathbf{X}$ and an appropriate p-by-k loading matrix $\mathbf{B}$. The columns of $\mathbf{B}$ indicate how $\mathbf{\Gamma}$ is formed from the columns of $\mathbf{X}$.

Principal Component Analysis (PCA) (Jolliffe, 2002) estimates $\mathbf{B}$ as an orthogonal matrix with the property that the first principal component, $\mathbf{\Gamma}_{\bullet 1} = \mathbf{X}\mathbf{B}_{\bullet 1}$, maximizes its variance. Further, the k-th principal component $\mathbf{\Gamma}_{\bullet k} = \mathbf{X}\mathbf{B}_{\bullet k}$ maximizes its variance under the constraint that $\mathbf{B}$ remains orthogonal.[1] Each principal component contributes what is estimated to be the next most important axis to modeling variation across columns of $\mathbf{X}$.

Considering a matrix $\mathbf{U}$ of n samples from $\mathcal{N}(\vec{0}, \mathbf{\Sigma})$ with $\mathbf{\Sigma} = \begin{bmatrix} 1 & .95 & 0 & 0 \\ .95 & 1 & 0 & 0 \\ 0 & 0 & 1 & .95 \\ 0 & 0 & .95 & 1 \end{bmatrix}$, we expect an n-by-4 matrix with $\mathbf{U}_{\bullet 1}$ and $\mathbf{U}_{\bullet 2}$ strongly correlated, $\mathbf{U}_{\bullet 3}$ and $\mathbf{U}_{\bullet 4}$ strongly correlated, and $(\mathbf{U}_{\bullet 1}, \mathbf{U}_{\bullet 2})$ and $(\mathbf{U}_{\bullet 3}, \mathbf{U}_{\bullet 4})$ independent. Thus we expect to be able to reduce the matrix $\mathbf{U}$ into an n-by-2 matrix $\mathbf{\Gamma}$ which can still accurately reproduce most of the variation in $\mathbf{U}$. Examples of loading matrices estimated by PCA on different samples of $\mathbf{U}$ are shown in Table 2.1. To reduce $\mathbf{U}$ to two columns, we multiply $\mathbf{U}$ by the estimated $\hat{\mathbf{B}}$. Notice in Example 1, the first column $\hat{\mathbf{B}}_{\bullet 1}$ forms a weighted average of the columns of $\mathbf{U}$ using mostly $\mathbf{U}_{\bullet 1}$ and $\mathbf{U}_{\bullet 2}$, while $\hat{\mathbf{B}}_{\bullet 2}$ mostly uses $\mathbf{U}_{\bullet 3}$ and $\mathbf{U}_{\bullet 4}$. PCA has discovered the underlying structure of $\mathbf{U}$ and provided estimates of how to produce low-rank representation $\mathbf{\Gamma}$.

[1] For matrix $\mathbf{M}$, $\mathbf{M}_{ij}$ indicates the element at the *ith* row and *jth* column; $\mathbf{M}_{i\bullet}$ indicates row i; $\mathbf{M}_{\bullet j}$ indicates column j.

Example 2, however, is an equally valid and accurate PCA loading matrix for a different sample of $\mathbf{U}$. The estimate $\hat{\mathbf{B}}$ no longer illuminates the structure of $\mathbf{U}$ by selectively assigning columns of $\mathbf{U}$ to principal components, instead, every column of $\mathbf{U}$ is assigned to every principal component. It is preferable for interpretation to have as many zero or near-zero loadings in $\hat{\mathbf{B}}$ without sacrificing accuracy. PCA, however, does not produce sparse loadings.

Table 2.1: Sample Rank-2 PCA Loadings for Samples of $\mathbf{U}$

Example 1: $\hat{\mathbf{B}}$ with easily interpreted loadings	Example 2: $\hat{\mathbf{B}}$ with less interpretable loadings
$\begin{bmatrix} 0.6987 & -0.0749 \\ 0.7059 & -0.0898 \\ 0.0834 & 0.7048 \\ 0.0814 & 0.6997 \end{bmatrix}$	$\begin{bmatrix} 0.5129 & 0.4888 \\ 0.5144 & 0.4832 \\ -0.4827 & 0.5137 \\ -0.4892 & 0.5136 \end{bmatrix}$

2.2 Sparsity in Regression

Interpretability of loadings from Example 1 of Table 2.1 was a result of certain loadings being near zero. The desire for estimating a loading matrix with exact zeros in PCA is analogous to seeking exact zeros in regression coefficients to perform feature selection in predictive models. In the univariate regression model $\mathbf{y} = \mathbf{X}\mathbf{b} + \boldsymbol{\epsilon}$, we seek to estimate the $\mathbf{b}$ which minimizes the squared loss $\|\mathbf{y} - \mathbf{X}\mathbf{b}\|_2^2$. For $n > p$, $\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ is the unique minimizer. [2] [3]

As p increases to be larger than n, we face two issues. First, the minimizing $\hat{\mathbf{b}}$ is no longer unique, which breaks the stated solution. Second, the estimated model is difficult to interpret due to the large number of entries in $\hat{\mathbf{b}}$. If we suspect that many features in $\mathbf{X}$ are irrelevant to $\mathbf{y}$, the ideal estimate $\hat{\mathbf{b}}$ will have many entries exactly equal to zero. This can be achieved by sparse regression methods.

We can resolve the $p > n$ problems with the elastic net (Zou and Hastie, 2005). The elastic net regression model adds penalties on $\mathbf{b}$ to achieve stable estimates in the $p > n$ setting and to impose sparsity. Given penalty parameters λ_1 and λ_2, the objective is presented in Equation (2.1).

[2] The stated unique solution assumes the existence of $(\mathbf{X}'\mathbf{X})^{-1}$, which fails if the columns of $\mathbf{X}$ have perfect multicollinearity.
[3] $\mathbf{X}'$ is the transpose of $\mathbf{X}$.

$$\boxed{\hat{\mathbf{b}}_{EN} = \arg\min_{\mathbf{b}} \|\mathbf{y} - \mathbf{X}\mathbf{b}\|_2^2 + \lambda_1\|\mathbf{b}\|_1 + \lambda_2\|\mathbf{b}\|_2^2} \tag{2.1}$$

Elastic Net Objective

The contribution of the L_2 penalty is to produce stable, unique solutions even for $p > n$ (Marquardt and Snee, 1975), while the L_1 penalty imposes sparsity by pushing small $\hat{\mathbf{b}}$ entries to be exactly zero. Modifying λ_1 allows the practitioner to set the desired level of sparsity.[4]

2.3 Sparse Principal Component Analysis

Sparse Principal Component Analysis (SPCA) is a technique for estimating sparse PCA loadings (Zou et al., 2006). This method approximates the PCA solution but, by using the elastic net, imposes sparsity constraints on the estimated loading matrix.

The PCA loading matrix for $\mathbf{X}$ can be written as the minimizer of Equation (2.2).

$$\boxed{\hat{\mathbf{B}}_{PCA} = \arg\min_{\mathbf{B}'\mathbf{B}=\mathbf{I}} \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{B}'\|_F^2} \tag{2.2}$$

PCA Objective

$\hat{\mathbf{B}}_{PCA}$ will project $\mathbf{X}$ into a desired lower rank by constraining the number of columns of $\hat{\mathbf{B}}_{PCA}$. For instance, limiting $\hat{\mathbf{B}}_{PCA}$ to be a single column will produce the best rank 1 estimate of $\mathbf{X}$ according to the presented objective.

SPCA relaxes the orthogonality constraint and adds penalty terms to the PCA objective to form the SPCA objective.

$$\boxed{(\hat{\mathbf{A}}_{SPCA}, \hat{\mathbf{B}}_{SPCA}) = \arg\min_{\mathbf{A}'\mathbf{A}=\mathbf{I}} \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}'\|_F^2 + \lambda_1\|\mathbf{B}\|_1 + \lambda_2\|\mathbf{B}\|_F^2} \tag{2.3}$$

SPCA Objective

Due to L_1 penalty term $\lambda_1\|\mathbf{B}\|_1$, sparsity can be imposed on $\hat{\mathbf{B}}_{SPCA}$.

We observe the SPCA criterion is very similar to the elastic net[5] criterion. The similarity is promising: Zou et. al. show that for fixed $\mathbf{A}$ in the SPCA criterion, the columns of $\hat{\mathbf{B}}_{SPCA}$ can be estimated as individual elastic net problems. Given a fixed estimate $\hat{\mathbf{B}}_{SPCA}$, they also provide

[4] For more information on sparsity, see for instance Tibshirani et al. (2015)

[5] This model is introduced by Zou, Hastie as the Naïve elastic net; for Naïve elastic net estimate $\hat{\mathbf{b}}_{NEN}$, the Elastic net solution $\hat{\mathbf{b}}_{EN}$ is defined as $(1 + \lambda_2)\hat{\mathbf{b}}_{NEN}$. For simplicity, we omit the distinction in this discussion.

a method for estimating $\hat{\mathbf{A}}$. Thus the SPCA solutions can be estimated by alternating updates between $\mathbf{A}$ and $\mathbf{B}$. Note that as each column $\mathbf{B}_{\bullet j}$ of $\mathbf{B}$ is estimated separately, the L_1 penalty λ_1 can be customized for each $\mathbf{B}_{\bullet j}$. The full SPCA model thus uses L_1 penalty term $\Sigma_{j=1}^{k} \lambda_1^j \|\mathbf{B}_{\bullet j}\|_1$.

Returning to estimated loadings from Table 2.1, we compare PCA loadings and SPCA loadings in Table 2.2 on the two matrices previously sampled from $\mathcal{N}(\vec{0}, \begin{bmatrix} 1 & .95 & 0 & 0 \\ .95 & 1 & 0 & 0 \\ 0 & 0 & 1 & .95 \\ 0 & 0 & .95 & 1 \end{bmatrix})$. In Example 1, estimated loading matrices from PCA and SPCA are both suggestive of the pairwise block covariance structure, with the SPCA estimate demonstrating it more immediately by having exact zeros. In Example 2, SPCA shows the structure in a way that is obscured in the PCA estimate.

Table 2.2: Sample Rank-2 PCA and SPCA Loadings

Example 1: $\hat{\mathbf{B}}_{PCA}$	Example 1: $\hat{\mathbf{B}}_{SPCA}$
$\begin{bmatrix} 0.6987 & -0.0749 \\ 0.7059 & -0.0898 \\ 0.0834 & 0.7048 \\ 0.0814 & 0.6997 \end{bmatrix}$	$\begin{bmatrix} -0.7080 & 0 \\ -0.7062 & 0 \\ 0 & -0.7069 \\ 0 & -0.7073 \end{bmatrix}$
Example 2: $\hat{\mathbf{B}}_{PCA}$	**Example 2: $\hat{\mathbf{B}}_{SPCA}$**
$\begin{bmatrix} 0.5129 & 0.4888 \\ 0.5144 & 0.4832 \\ -0.4827 & 0.5137 \\ -0.4892 & 0.5136 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.7134 \\ 0 & -0.7007 \\ -0.7011 & 0 \\ -0.7130 & 0 \end{bmatrix}$

The fit of each low-rank estimate can be measured by the reconstruction error $\|\mathbf{X}-\mathbf{X}\hat{\mathbf{B}}_{PCA}\hat{\mathbf{B}}'_{PCA}\|_F^2$ and $\|\mathbf{X} - \mathbf{X}\hat{\mathbf{B}}_{SPCA}\hat{\mathbf{A}}'_{SPCA}\|_F^2$. In these examples, the increase in error from PCA to SPCA is negligible.

The SPCA objective can be solved by a series of regression estimates for each column of $\mathbf{B}$, where regression is performed by elastic net or thresholding to impose sparsity. The desired sparsity can be imposed on $\mathbf{B}$ by explicitly specifying the number of nonzero entries to retain in each of $\mathbf{B}$'s columns. We employ the SPCA implementation by Sjöstrand (2005) which estimates sparse $\mathbf{B}$ to minimize the SPCA objective based on desired penalty parameters λ_1 and λ_2.

Algorithm 1 Sparse Principal Component Analysis

Input:

- $\mathbf{X}$, n-by-p matrix
- k, desired number of sparse principal components
- λ_1^j, Desired number of nonzero entries for loading matrix column j if soft thresholding; desired L_1 penalties if using elastic net
- λ_2, L_2 penalty

Output: Trained model parameters $\mathbf{A}$, $\mathbf{B}$

```
 1: Initialize $\mathbf{A}$ as the first $k$ loading vectors from PCA on $\mathbf{X}$
 2: for $j = 1$ to $k$ do
 3:     while $\mathbf{B}_{\bullet j}$ has not converged do
 4:         if Soft Thresholding then
 5:             $\mathbf{B}_{\bullet j} = \mathrm{Sign}(\mathbf{A}_{\bullet j}'\mathbf{X}'\mathbf{X})(|\mathbf{A}_{\bullet j}'\mathbf{X}'\mathbf{X}| - \lambda_1^j)_+$
 6:         else if Elastic Net then
 7:             Solve $\mathbf{B}_{\bullet j} = \arg\min_{\mathbf{b}} \|\mathbf{X}\mathbf{A}_{\bullet j} - \mathbf{X}\mathbf{b}\|_2^2 + \lambda_1^j\|\mathbf{b}\|_1 + \lambda_2\|\mathbf{b}\|_2^2$ by elastic net
 8:         end if
 9:         $\mathbf{B}_{\bullet j} = \mathbf{B}_{\bullet j}/\|\mathbf{B}_{\bullet j}\|_2$
10:         $\mathbf{A}_{\bullet j} = (\mathbf{I} - \mathbf{A}_{j-1}\mathbf{A}_{j-1}')\mathbf{X}'\mathbf{X}\mathbf{B}_{\bullet j}$        ▷ Let $\mathbf{A}_c$ denote $[\mathbf{A}_{\bullet 1}, \ldots, \mathbf{A}_{\bullet c}]$, $\mathbf{A}_0 = \mathbf{A}_{\bullet 1}$
11:         $\mathbf{A}_{\bullet j} = \mathbf{A}_{\bullet j}/\|\mathbf{A}_{\bullet j}\|_2$
12:     end while
13: end for
```

2.4 Feature Selection with Annealing

Sparsity in SPCA is achieved by embedding the elastic net, and in particular the L_1 penalty, into a slightly modified PCA estimation problem. In comparable models we propose here, sparsity is achieved by embedding a different feature selection method borrowed from regression and classification: *Feature Selection with Annealing (FSA)* (Barbu et al., 2017).

For univariate regression $\mathbf{y} = \mathbf{X}\mathbf{b} + \boldsymbol{\epsilon}$, FSA estimates sparse $\mathbf{b}$ by alternating between gradient descent updates and sparsity enforcement on $\hat{\mathbf{b}}$. For a desired L_0 sparsity, FSA gradually selects entries in $\hat{\mathbf{b}}$ to be set to exactly zero, while the other values continue receiving gradient updates.

Algorithm 2 Feature Selection with Annealing for Regression

```
Input:
  • $\mathbf{X}$, $n$-by-$p$ matrix
  • $\mathbf{y}$, $n$-by-1 response vector
  • $\eta$, learning rate
  • $epochs$, number of epochs
  • $l(\mathbf{X}, \mathbf{y}, \mathbf{b})$, desired loss function
  • $m^e$, annealing schedule such that $m^e$ is the number of features kept at epoch $e$
Output: Sparse estimated model parameter $\mathbf{b}$
1: Initialize $\mathbf{b} = \vec{0}$
2: for e=1 to $epochs$ do
3:     Update $\mathbf{b} \leftarrow \mathbf{b} - \eta \frac{\partial l(\mathbf{X},\mathbf{y},\mathbf{b})}{\partial \mathbf{b}}$
4:     Keep only the $m^e$ variables with highest $|b_j|$ and renumber them $1, \ldots, m^e$
5: end for
```

The choice of annealing schedule m^e determines the rate at which features in $\mathbf{b}$ are dropped from the estimation. For n-by-p $\mathbf{X}$, $\mathbf{b}$ has p entries, but we may desire an estimate $\hat{\mathbf{b}}$ that only has $k < p$ nonzero entries. The schedule m^e is chosen to reduce the number of nonzero coefficients of the estimate $\hat{\mathbf{b}}$ from $m^0 = p$ at the beginning of processing to $m^{epochs} = k$ at the final epoch.

A slow annealing schedule allows for more estimation to occur between annealing steps, making it less likely that a relevant feature is dropped. A more aggressive schedule improves computation time as the number of features under consideration shrinks more quickly. An appealing approach from Barbu et al. (2017) is a schedule m^e that drops features aggressively in early epochs, quickly dropping the large number of readily identified irrelevant predictors. The annealing slows as estimation continues so that more difficult feature selection choices are made with more carefully estimated parameters. Their proposed annealing schedule is

$$m^e = k + (p-k)\max(0, \frac{epochs - 2e}{2e\mu + epochs}) \tag{2.4}$$

for p features, desired $\|\hat{\mathbf{b}}\|_0 = k$, non-negative annealing schedule parameter μ, and epochs $e = 1, \ldots, epochs$. The parameter μ controls how features are dropped as a function of e, with $\mu = 0$ dropping features at a linear rate and higher values dropping features quickly in earlier epochs before slowing down. Sample annealing schedules are shown in Figure 2.1.

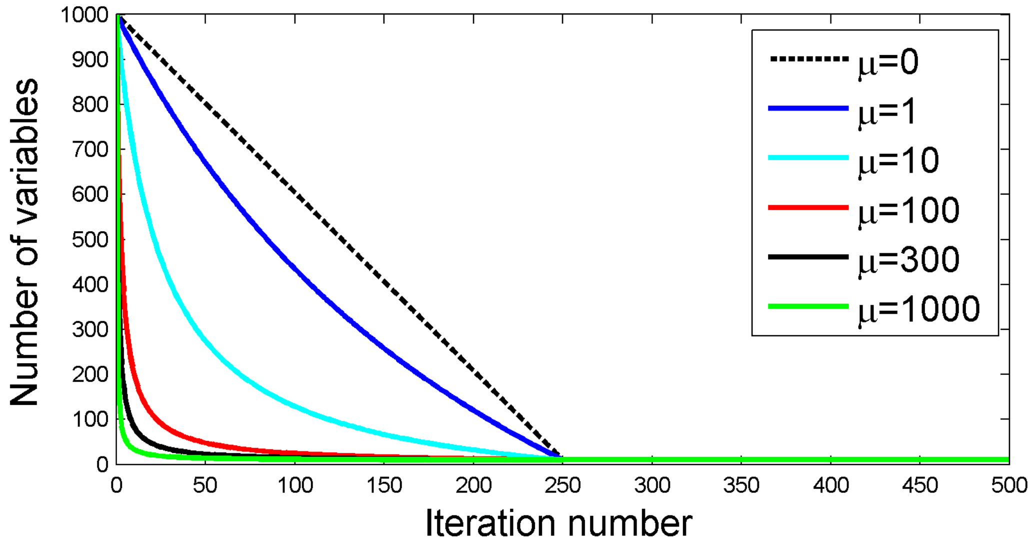


Figure 2.1: Features Remaining m^e vs. Iteration e for Various μ Values, with $p = 1000$, $k = 10$, $epochs = 500$[6]

2.5 Joint Sparse Principal Component Analysis

A recent development in sparse low-dimensional modeling is *Joint Sparse Principal Component Analysis (JSPCA)* from Yi et al. (2017), which considers

$$\boxed{(\hat{\mathbf{A}}_{JSPCA}, \hat{\mathbf{B}}_{JSPCA}) = \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\ \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}'\|_{2,1} + \lambda\|\mathbf{B}\|_{2,1}} \tag{2.5}$$

JSPCA Objective

The $L_{2,1}$ norm of a matrix is the sum of the L_2 norms of its rows:

$$\|\mathbf{M}\|_{2,1} = \sum_i \sqrt{\sum_j \mathbf{M}_{ij}^2}$$

[6] Barbu et al. (2017), reproduced with permission.

The choice of $L_{2,1}$ norm was motivated by a desire to impose row-sparsity and to improve robustness. Due to the sparsity penalty $\lambda\|\mathbf{B}\|_{2,1}$ acting on the rows of $\mathbf{B}$, entire rows are likely to be reduced by shrinkage. Because rows of $\mathbf{B}$ correspond to features in $\mathbf{X}$, this is a natural choice of penalty for feature selection. Yi et. al. suggest the $L_{2,1}$ loss term on $\mathbf{X} - \hat{\mathbf{X}}$ is more robust to outliers than the usual L_2 loss on error, improving performance when faced with data corruption and outliers.

We examine JSPCA alongside other methods in simulations and data experiments.

2.6 Sparse Reduced Rank Regression

Sparse Reduced Rank Regression (SRRR) (She, 2017) uses a low-rank representation of data $\mathbf{X}$ to predict response matrix $\mathbf{Y}$ with the following objective function

$$\boxed{(\hat{\mathbf{A}}_{SRRR}, \hat{\mathbf{B}}_{SRRR}) = \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\ \|\mathbf{Y} - \mathbf{X}\mathbf{B}\mathbf{A}'\|_F^2/(2K) + \sum_{i,j} P(|\mathbf{B}_{ij}|; \lambda^e)} \tag{2.6}$$

SRRR Objective

where (P, λ^e) define the desired sparsity-inducing penalty function, $K > 0$, and the rank of $\mathbf{XBA}'$ is controlled by the dimensions of ($\mathbf{A}$, $\mathbf{B}$). The authors introduce SRRR with an algorithm which is shown to have convergence guarantees for the appropriate selection of P and K. A thresholding method is provided allowing (2.6) to be fit with explicit sparsity restrictions on $\mathbf{B}$, such as the number of permitted nonzero entries or the number of nonzero rows. The latter constraint permits the exact specification of the number of features of $\mathbf{X}$ to use.

While SRRR is defined for multivariate regression, She (2017) also examines sparse PCA which emerges as a special case. Replacing $\mathbf{Y}$ with $\mathbf{X}$ and $\mathbf{X}$ with $\mathbf{I}$ reduces (2.6) to the SRRR-SPCA objective:

$$\boxed{(\hat{\mathbf{S}}, \hat{\mathbf{V}}) = \underset{\mathbf{V}'\mathbf{V}=\mathbf{I}}{\arg\min}\ \|\mathbf{X} - \mathbf{V}\mathbf{S}'\|_F^2 + \sum_{i,j} P(|\mathbf{S}_{ij}|; \lambda^e)} \tag{2.7}$$

SRRR-SPCA Objective

The authors note that the usual approach to sparse PCA can be viewed within the SRRR framework as a regression of $\mathbf{X}$ on itself. SPCA for instance can be generated from (2.6) by taking $\mathbf{Y} = \mathbf{X}$ and selecting appropriate P. JSPCA applies an $L_{2,1}$ norm to minimize the same self-regression setup. SRRR-SPCA, however, differs from this behavior by directly estimating low-dimension representation $\mathbf{V}$ alongside sparse $\mathbf{S}$. This approach and its differences from the SPCA objective and its derivatives will be important in our later discussion comparing various proposed sparse PCA methodologies.

2.7 Latent Class Analysis

Latent Class Analysis (LCA) is a model-based clustering method for categorical data introduced by Lazarsfeld (1950). In the LCA model, rows of a categorical data are presumed to belong to unobserved latent classes. The class to which a row belongs defines the probability distributions of its features. The goal of LCA is to use the observed distribution of features to cluster rows into likely latent classes.

Unsupervised clustering of categorical data emerges in a wide variety of fields to discover potential subgroups or validate existing categorization schemes of a population of interest. See for instance applications of LCA in the development of typologies in topics as diverse as workplace environments (Carr et al., 2022), family relationships (Barrett and Gunderson, 2021), homicidal behavior (Vaughn et al., 2009), attitudes of hunters (Ward et al., 2008), sleep patterns (Preckel et al., 2020), compulsive shopping (Challet-Bouju et al., 2020), theater attendance (Grisolía and Willis, 2012), psychosis (Kendler et al., 1998), game-playing strategies (Ghosh and Verbrugge, 2018), high school attrition (Bowers and Sprott, 2012), and protein classification techniques (Chen et al., 2007).

We will examine the problem of feature selection for LCA models.

2.7.1 LCA Model Specification

An LCA model is defined by its number of unobserved latent classes, class-conditional probabilities of response data, and the global mixture proportions of each class.

For instance, consider a latent class model for p-dimensional binary data with latent classes $K = \{1, 2, \ldots k\}$. A sample z generated by this model has p binary features which we observe, and an underlying latent class label $c \in K$ which is unobserved. Each class c has its own specification of p Bernoulli parameters which define the distributions of the p features of observations belonging to that class. This model can be specified by the following parameters:

- k, the number of latent classes

- $\boldsymbol{\pi}$, a length-k vector of probabilities where $\boldsymbol{\pi}_c$ is the probability of a sample being generated from class c. Note that $\sum_{c=1}^{k} \boldsymbol{\pi}_c = 1$.
- $\boldsymbol{\theta}$, a k-by-p matrix of class-conditional Bernoulli parameters defining the distribution of samples generated from each class. That is, for a sample $\mathbf{z}$ belonging to class c, $\boldsymbol{\theta}_{cj} = P(\mathbf{z}_j = 1)$.

2.7.2 LCA Example

As a simple example, consider realizations of three coin tosses. The three tosses within a sample are generated from the same coin, but samples are generated from a fair coin 30% of the time and a loaded coin having a 90% chance of returning heads 70% of the time. In this scenario, realizations belong to one of two latent classes (fair, unfair coin) which can be modeled as a two-class latent class model with

$$\boldsymbol{\pi} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.9 & 0.9 & 0.9 \end{bmatrix}$$

Given these parameters, an observation *(Heads, Heads, Heads)* can be shown to have a higher likelihood of having been generated from class 2 than from class 1.

2.7.3 LCA Likelihood

Given latent class model $\boldsymbol{\Lambda} = (\boldsymbol{\pi}, \boldsymbol{\theta})$ for p-dimensional binary data with latent classes $\{1, 2, \ldots, k\}$, let Z be a random variable representing p-dimensional samples distributed according to the model, T the latent class label of Z, and z a sample drawn from Z. Note that T itself is random. We summarize some useful properties of T which follow from the model specification:

- $P(T = c) = \boldsymbol{\pi}_c$
- $P(Z_j = 1 \mid T = c) = \boldsymbol{\theta}_{cj}$
- $P(Z_j = 0 \mid T = c) = 1 - \boldsymbol{\theta}_{cj}$
- $P(Z_j = z_j \mid T = c) = \begin{cases} \boldsymbol{\theta}_{cj} & z_j = 1 \\ 1 - \boldsymbol{\theta}_{cj} & z_j = 0 \end{cases}$

  $= (\boldsymbol{\theta}_{cj})^{z_j}(1 - \boldsymbol{\theta}_{cj})^{1-z_j}$
- $P(Z_j = 1) = \sum_{c=1}^{k} P(T = c)P(Z_j = 1 \mid T = c) = \sum_{c=1}^{k} \boldsymbol{\pi}_c \boldsymbol{\theta}_{cj}$

- $P(Z_1 = z_1, Z_2 = z_2, \ldots, Z_p = z_p \mid T = c)$

$$= \prod_{j=1}^{p} P(Z_j = z_j \mid T = c)$$

$$= \prod_{j=1}^{p} (\boldsymbol{\theta}_{cj})^{z_j}(1 - \boldsymbol{\theta}_{cj})^{1-z_j}$$

- $P(Z_1 = z_1, Z_2 = z_2, \ldots, Z_p = z_p)$

$$= \sum_{c=1}^{k} P(T = c) P(Z_1 = z_1, Z_2 = z_2, \ldots, Z_p = z_p \mid T = c)$$

$$= \sum_{c=1}^{k} P(T = c) \prod_{j=1}^{p} P(Z_j = z_j \mid T = c)$$

$$= \sum_{c=1}^{k} \boldsymbol{\pi}_c \prod_{j=1}^{p} (\boldsymbol{\theta}_{cj})^{z_j}(1 - \boldsymbol{\theta}_{cj})^{1-z_j}$$

The final property listed above is helpful in defining the likelihood function of the model. Given k-class LCA model $\boldsymbol{\Lambda} = (\boldsymbol{\pi}, \boldsymbol{\theta})$ for p-dimensional binary data and n-by-p binary matrix $\mathbf{Z}$, the likelihood of the model given $\mathbf{Z}$ is given by

$$\boxed{\mathscr{L}(\boldsymbol{\Lambda}; \mathbf{Z}) = \prod_{i=1}^{n} \left(\sum_{c=1}^{k} \boldsymbol{\pi}_c \prod_{j=1}^{p} (\boldsymbol{\theta}_{cj})^{\mathbf{Z}_{ij}}(1 - \boldsymbol{\theta}_{cj})^{1-\mathbf{Z}_{ij}} \right)} \tag{2.8}$$

LCA Likelihood

Latent Class Analysis model parameters are ordinarily estimated by Expectation-Maximization. Further detail can be found in Agresti (2010). In this work we will develop and test methods to apply feature selection to Latent Class Analysis using FSA.

CHAPTER 3

METHODOLOGY - LATENT CLASS ANALYSIS

3.1 Feature Selection for LCA

Latent-model categorical data with large p presents familiar challenges for modeling: stability, interpretability, and parsimony. To this end, there have been developments in regularization and feature selection for LCA. For this discussion we consider a k-class LCA model $\mathbf{\Lambda} = (\boldsymbol{\pi}, \boldsymbol{\theta})$ for p-dimensional binary data.

Penalized Methods. Latent class models on high-dimensional data require estimation of a large number of parameters, which can produce computational and numerical difficulty. One option for stabilizing LCA estimation is including a penalty term in the likelihood function (2.8) of the model. Houseman et al. (2006) apply L_1 and L_2 penalties on $\boldsymbol{\theta}$ to LCA. Their finding was that penalizing the likelihood enabled model-fitting in a setting where unpenalized estimation would diverge. This penalized estimation method has since been extended to allow stable model estimation of LCA on high-dimensional ordinal data (DeSantis et al., 2008) and to accommodate incorporation of covariates to improve model accuracy (Leoutsakos et al., 2011).

Houseman, Leoutsakos, and colleagues differentiate their likelihood penalties from approaches such as Mooijaart and Van der Heijden (1992), which assume a priori structural constraints on $\boldsymbol{\theta}$ such as equality of parameter elements having certain positional relationships. Two recent examples of structured regularization for LCA incorporate the expectation from domain knowledge that features will commonly have similar probabilities across adjacent latent classes (Chen et al., 2017) or across classes in general (Robitzsch, 2020). However in the absence of domain knowledge justifying constraint assumptions, a more general approach to penalization is appropriate (Houseman et al., 2006; Leoutsakos et al., 2011; Robitzsch, 2020).

Stepwise Methods. A completely different feature selection framework is stepwise selection. This approach fully trains a series of models, with feature selection decisions made by comparing the fit of successive models. When a convergence criterion is met, the final model is hoped to use the optimal subset of features.

Raftery and Dean (2006) introduce a stepwise selection approach for model-based clustering. They begin with a single-feature model, starting with the one showing greatest evidence of univariate clustering. The next model adds a second feature: the one that when paired with the first feature shows the greatest evidence for bivariate clustering. The addition of features continues for all features that show evidence of being relevant to the clustering. This method was originally demonstrated on continuous data but has since been applied to the categorical latent class setting with slight modifications to the feature search strategy (Dean and Raftery, 2010; Fop et al., 2017).

However, Raftery and Dean (2006) caution that the stepwise approach is not feasible in the high-dimensional setting. Penalized variable selection for model-based clustering is identified as a viable alternative to stepwise selection in high dimensional settings in a review by Bouveyron and Brunet-Saumard (2014) of high-dimensional clustering methods, and by Celeux et al. (2014) in their direct comparison of stepwise selection and regularization. Pan and Shen (2007) also find that penalized variable selection is preferable to stepwise selection methods in high dimensions, but further suggest that stepwise feature selection may be flawed in low-dimension settings as well.

3.2 Latent Class Analysis with FSA

In this section we introduce *Latent Class Analysis with FSA (LCA-FSA)*. In our approach, feature relevance will be assessed by using the model's ongoing estimate of latent class assignments. Because latent class labels are categorical and the feature space is binary, we examine candidate measures of association between categorical and binary random variables.

3.2.1 Measures of Categorical Association

The following discussion assumes discrete random variables X and Y. Probabilistic measures of association may be functions of X and Y directly, while empirical measures will operate on joint realizations $\{(x_i, y_i); x_i \sim X, y_i \sim Y\}_{i=1}^n$ which we organize into vectors $\mathbf{x} = \{x_i\}_{i=1}^n$ and $\mathbf{y} = \{y_i\}_{i=1}^n$.

Mutual Information. Mutual information is a popular and intuitive approach to measuring the association between two random variables, originating in the foundations of information theory (Shannon, 1948).

$$MI(X,Y) = \sum_{x,y} P(X=x, Y=y) log \frac{P(X=x, Y=y)}{P(X=x)P(Y=y)} \tag{3.1}$$

Note that the sum is implied to only consider joint values (x, y) where $P(X = x, Y = y) > 0$.

Mutual information is non-negative, achieving its minimum $MI(X, Y) = 0$ when X and Y are mutually independent. It is symmetric over its inputs.

A detailed introduction to mutual information will be found in general texts on information theory such as Yang (2008). A brief examination of its usage in feature selection is found in Vergara and Estévez (2014), though there is a prolific and ongoing body of work contributing novel modifications of mutual information to selection problems.

When used in LCA for feature selection with data $\mathbf{Z}$ and class label estimates $\hat{T}$, the feature relevance score for feature j is given by $MI(\mathbf{Z}_{\bullet j}, \hat{T})$.

Algorithm 3 Calculating $MI(\mathbf{x}, \mathbf{y})$

```
Input: Categorical vectors $\mathbf{x}$, $\mathbf{y}$ of length $n$
Output: Mutual Information score between $\mathbf{x}$ and $\mathbf{y}$
1: Let If be the indicator function:
       $\mathbf{If}(\texttt{statement}) = \begin{cases} 1 & \text{If True} \\ 0 & \text{Otherwise} \end{cases}$        (3.2)
2: Let $P$ and $Q$ be the sets of the distinct values taken by elements of $\mathbf{x}$ and $\mathbf{y}$, respectively
3: For $p \in P$ and $q \in Q$, define
       $C_x(p) = \frac{1}{n} \sum_t \mathbf{If}(\mathbf{x}_t = p)$        (3.3)
       $C_y(q) = \frac{1}{n} \sum_t \mathbf{If}(\mathbf{y}_t = q)$        (3.4)
       $C_{x,y}(p, q) = \frac{1}{n} \sum_t \mathbf{If}(\mathbf{x}_t = p)\mathbf{If}(\mathbf{y}_t = q)$        (3.5)
4: Calculate Mutual Information score
       $MI_{x,y} = \sum_{p \in P, q \in Q} C_{x,y}(p, q) log \frac{max(C_{x,y}(p, q), 1)}{C_x(p) C_y(q)}$        (3.6)
```

An influential association score based on mutual information is **Maximum Relevance Minimum Redundancy (mRMR)**. In a classification feature selection setting, mRMR ranks candidate features by rewarding mutual information with the response and penalizing mutual information with features that have already been selected (Peng et al., 2005).

mRMR feature relevance scores for LCA can be calculated for feature $\mathbf{Z}_{\bullet j}$ of $\mathbf{Z}$ by calculating its MI score and subtracting the average mutual information $MI(\mathbf{Z}_{\bullet j}, \mathbf{Z}_{\bullet k})$ between feature j and all other features k.

χ^2 **Test of Association.** The χ^2 test of association is a standard approach to performing inference on contingency tables (Agresti, 2010, chap. 3). While the χ^2 test is a classical inference method designed for hypothesis-testing, the statistic and derivative methods are sometimes applied to feature selection problems for classification (Forman et al., 2003; Spencer et al., 2020).

As a measure of categorical association, we define function $\chi^2(\mathbf{x}, \mathbf{y})$ between categorical vectors $\mathbf{x}$ and $\mathbf{y}$ as the statistic produced by a χ^2 test of association on those vectors when testing the null hypothesis of no association. As an LCA feature relevance score, we take $\mathbf{x}$ to be the feature being scored and $\mathbf{y}$ to be the model's current class label estimates. Calculation steps are detailed in Algorithm 4.

Algorithm 4 Calculating $\chi^2(\mathbf{x}, \mathbf{y})$

```
Input: Categorical vectors $\mathbf{x}$, $\mathbf{y}$ of length $n$
Output: Chi-square statistic $\chi^2$
1: Let If be the indicator function:
       $\mathbf{If}(\texttt{statement}) = \begin{cases} 1 & \text{If True} \\ 0 & \text{Otherwise} \end{cases}$    (3.7)
2: Let $\{P_i\}_{i=1}^p$ and $\{Q_i\}_{i=1}^q$ be the sets of the distinct values taken by elements of $\mathbf{x}$ and $\mathbf{y}$, respectively
3: Let $\mathbf{O}$ be a $p$-by-$q$ matrix where
       $\mathbf{O}_{ij} = \sum_{t=1}^{n} \mathbf{If}(\mathbf{x}_t = P_i)\mathbf{If}(\mathbf{y}_t = Q_j)$    (3.8)
4: Let $\mathbf{E}$ by a $p$-by-$q$ matrix where
       $\mathbf{E}_{ij} = \left(\sum_{t=1}^{n} \mathbf{If}(\mathbf{x}_t = P_i)\right)\left(\sum_{t=1}^{n} \mathbf{If}(\mathbf{y}_t = Q_j)\right)\left(\frac{1}{n}\right)$    (3.9)
5: Calculate chi-square statistic
       $\chi^2 = \sum_{i,j} \frac{(\mathbf{O}_{ij} - \mathbf{E}_{ij})^2}{\mathbf{E}_{ij}}$    (3.10)
```

Diff-Criterion. Javed et al. (2010) introduce the diff-criterion for measuring association between a binary response variable and binary features for the classification setting. Consider Bernoulli random variables X and Y which are thought to have some association. Framed as a classification problem, samples of X are used to predict corresponding samples of Y. The diff-criterion measures the relevance of X to Y as the absolute difference in $P(X = 1)$ given Y:

$$\mathit{DiffCrit}(Y, X) = |P(X = 1 \mid Y = 1) - P(X = 1 \mid Y = 0)| \tag{3.11}$$

The diff-criterion showed promise as a fast filtering step for feature selection in high-dimensional classification where Y is being predicted by a large number of binary features $\{X_j\}_j$. Intuitively, the criterion is meant to identify which features show little differentiation across response values. The authors suggest its potential use in a two-stage approach to feature selection, with a fast diff-criterion pass being followed by a more intensive approach.

The diff-criterion was introduced for feature selection in a supervised classification setting; we, however, apply it to unsupervised feature selection in LCA. Also, Javed et al. (2010) apply diff-criterion to settings with binary Y. We apply and test an extension of the diff-criterion to multi-class Y in a way proposed by Javed (2012) but which, to our knowledge, has not been examined in the literature.

For categorical random variable Y taking values $\{1, ..., k\}$ where $P(Y = c) = \boldsymbol{\pi}_c$, and binary random variable X,

$$\mathit{DiffCrit}(Y, X) = \sum_{i<j} \boldsymbol{\pi}_i \boldsymbol{\pi}_j |P(X = 1 \mid Y = i) - P(X = 1 \mid Y = j)| \tag{3.12}$$

When used during model training to measure feature relevance, the feature $\mathbf{Z}_{\bullet j}$ being examined is represented here as X, the model's current estimates for class labels is used here as Y, $\boldsymbol{\pi}$ is an estimated model parameter, and values of $P(X = x|Y = y)$ are estimated from observed values.

3.2.2 Likelihood-Based Feature Relevance Score

LCA feature relevance methods introduced so far have been measures of categorical association. Here, we describe a model-based approach to ranking feature importance.

In the stepwise selection methods reviewed in 3.1, feature selection progresses through comparing fit statistics of two competing, fully-trained models where the features of one model are a subset of the features of the other. We propose a similar approach which anneals features based on the effect of their removal on the model likelihood. However, this feature selection decision is embedded as part of model training.

Given binary data matrix $\mathbf{Z}$ and LCA model $\boldsymbol{\Lambda}$, the likelihood $\mathscr{L}(\boldsymbol{\Lambda}; \mathbf{Z})$ is given by (2.8). Consider that for any feature $\mathbf{Z}_{\bullet j}$, we can produce revised data $\mathbf{Z}^{(j)}$ removing that feature, and revised model $\boldsymbol{\Lambda}^{(j)}$ excluding parameter elements $\boldsymbol{\theta}_{\bullet j}$ relating to the feature. $\mathscr{L}(\boldsymbol{\Lambda}^{(j)}; \mathbf{Z}^{(j)})$ would then show the likelihood of the model while ignoring $\mathbf{Z}^{(j)}$. By calculating all such likelihoods $\{\mathscr{L}(\boldsymbol{\Lambda}^{(j)}; \mathbf{Z}^{(j)})\}_j$, features can be scored by the impact of their exclusion on the overall likelihood. Because these likelihoods will show higher values for less important features, we negate the value for use as a feature relevance score.

$$LLScore(\mathbf{Z}_{\bullet j}) = -\mathscr{L}(\boldsymbol{\Lambda}^{(j)}; \mathbf{Z}^{(j)})$$

3.2.3 Incorporating FSA into LCA

The measures of categorical association introduced above are applicable to feature selection in Latent Class Analysis. During model training, the training algorithm's current parameter estimates

($\boldsymbol{\pi}$, $\boldsymbol{\theta}$) are available, as well as data $\mathbf{Z}$ being trained on. The parameter estimates allow us to estimate the probability of a given class generating a row $\mathbf{z}$ based on its observed values, as shown in 2.7.3. Letting T be the random variable representing possible class assignments of $\mathbf{z}$, we have

$$P(\mathbf{z} \mid T = c) = \prod_{j=1}^{p} (\boldsymbol{\theta}_{cj})^{\mathbf{z}_j} (1 - \boldsymbol{\theta}_{cj})^{1-\mathbf{z}_j} \tag{3.13}$$

as well as simultaneous probability

$$P(\mathbf{z}, T = c) = \boldsymbol{\pi}_c \prod_{j=1}^{p} (\boldsymbol{\theta}_{cj})^{\mathbf{z}_j} (1 - \boldsymbol{\theta}_{cj})^{1-\mathbf{z}_j} \tag{3.14}$$

The joint probabilities between $\mathbf{z}$ and class possibilities c allow us to assign $\mathbf{z}$ a maximum likelihood estimate of its class label. These estimates for all rows of $\mathbf{Z}$ are collected in a vector of class label predictions $\hat{T}$.

Given these estimated labels, feature relevance can be ranked by calculating the association of each feature $\mathbf{Z}_{\bullet j}$ and $\hat{T}$ by any of MI, $mRMR$, $DiffCrit$, or χ^2. If using the likelihood-based approach for measuring feature relevance, scores can be obtained using the overall data likelihood in the scheme prescribed in 3.2.2. We call this model *LCA-FSA*. The process for estimating an LCA-FSA model is shown in Algorithm 5, which follows an expectation-maximization procedure for LCA detailed by Goodman (1974).

Remarks on LCA Algorithm.

Remark 1. *In practice, estimation of LCA models typically involves multiple starts.*

The quality of a trained LCA model can vary based on the random initialization of parameters. Therefore, clustering software typically suggests training a model multiple times allowing for multiple initializations. The model showing highest log-likelihood is selected. See, for instance, popular R packages for clustering (Linzer and Lewis, 2011; Scrucca et al., 2016; Maechler et al., 2022).

Remark 2. *Calculations should be designed to mitigate underflow and exact 0's in* $\boldsymbol{\theta}$.

Model estimation at times involves performing intermediate products of a high number of probabilities. This can produce underflow, where a product is returned with value exactly 0 because the actual value is beyond the floating point precision of the machine. For high-dimensional applications these calculations should be performed in terms of log probabilities rather than on probabilities directly. Further, correct but vanishingly small values in $\boldsymbol{\theta}$ can produce numerical *overflow* when

Algorithm 5 Latent Class Analysis with FSA

Input:

- $\mathbf{Z}$, n-by-p binary data matrix
- k, desired number of latent classes
- ω, desired number of features
- *epochs*, number of epochs
- m^e, FSA annealing schedule specifying number of features kept at epoch e
- A feature relevance scoring measure to be used with FSA

Output:

- Trained model parameters $\boldsymbol{\theta}$, $\boldsymbol{\pi}$
- An index of retained features $\mathbf{v} = \{j \in \{1, 2, \ldots, k\} \mid \mathbf{Z}_{\bullet j} \text{ retained by model}\}$

```
 1: Let $T$ be a vector of $n$ random variables where $T_i$ represents the unknown latent
        class of row $\mathbf{Z}_{i\bullet}$
 2: Randomly initialize estimates $\hat{T}$ of $T$ by drawing $n$ samples uniformly from $\{1, ..., k\}$
 3: Initialize $\boldsymbol{\pi}$ with $\boldsymbol{\pi}_i = Count(\hat{T} = i)/n$
 4: Initialize $\boldsymbol{\theta}$ with $\boldsymbol{\theta}_{cj}$ equal to the average value of $\{\mathbf{Z}_{ij} \mid i \in \{1, ..., n\}, \hat{T}_i = c\}$
 5: Initialize feature retention index $\mathbf{v} = [1, 2, \ldots, p]$
 6: for $e = 1$ to epochs do
 7:     Calculate $n$-by-$k$ matrix $\mathbf{C}$ of observation-class likelihoods with
            $\mathbf{C}_{ic} = P(\mathbf{Z}_{i\bullet} \mid T_i = c)$ by applying Equation (3.13)
             (See Algorithm Note in 3.2.3)
 8:     Update $\boldsymbol{\pi}$ to be the column-wise means of $\mathbf{C}$
 9:     Update $\boldsymbol{\theta}$ to $diag(\boldsymbol{\pi})^{-1}\mathbf{C}'\mathbf{Z}/n$, where $diag(\boldsymbol{\pi})$ the diagonal matrix with diagonal entries $\boldsymbol{\pi}$
10:     Apply FSA
11:         $m^e$ is the desired number of features to retain for current epoch
12:         Score all features with the chosen metric and identify those not in the top $m^e$
13:         For each column $\mathbf{Z}_{\bullet j}$ not retained:
                Remove that column from $\mathbf{Z}$
                Remove column $\boldsymbol{\theta}_{\bullet j}$ from $\boldsymbol{\theta}$
                Remove $j$ from $\mathbf{v}$
14: end for
```

such small numbers are divisors. We find it helpful in estimation to introduce a slight bias to the estimation of $\boldsymbol{\theta}$ by preventing any of its elements from becoming exactly 0.

Remark 3. *High-dimensional clustering of binary data is an ideal setting for sparse matrix programming.*

A common strategy for implementing an algorithm efficiently is to vectorize calculations. This is possible for the estimation of LCA. However in the case of high dimensions, binary data is often sparse i.e. having many features with very few positive cases. For these settings, it is worthwhile to use sparse matrix data objects and to employ operations optimized for them.

Algorithm Note. To estimate $\mathbf{C}$, calculation is performed in terms of log probabilities per Remark 2. Observe that for model $\boldsymbol{\Lambda} = (\boldsymbol{\pi}, \boldsymbol{\theta})$, row $\mathbf{z}$, and its unobserved latent class label T,

$$
\begin{aligned}
&P(T = j \mid \mathbf{z}) \\
&\quad = P(\mathbf{z} \mid T = j)P(T = j)/P(\mathbf{z}) \\
&\quad = \frac{(\prod_{i=1}^{p} |1 - \mathbf{z}_i - \boldsymbol{\theta}_{ji}|)\boldsymbol{\pi}_j}{\sum_{c=1}^{k}(\prod_{i=1}^{p} |1 - \mathbf{z}_i - \boldsymbol{\theta}_{ci}|)\boldsymbol{\pi}_c} \\
&\quad = \left(1 + \sum_{c \neq j} \frac{(\prod_{i=1}^{p} |1 - \mathbf{z}_i - \boldsymbol{\theta}_{ci}|)\boldsymbol{\pi}_c}{(\prod_{i=1}^{p} |1 - \mathbf{z}_i - \boldsymbol{\theta}_{ji}|)\boldsymbol{\pi}_j}\right)^{-1} \\
&\quad = \left(1 + \sum_{c \neq j} \exp\left(\log(\boldsymbol{\pi}_c) + \sum_{i=1}^{p} \log(|1 - \mathbf{z}_i - \boldsymbol{\theta}_{ci}|) - \log(\boldsymbol{\pi}_j) - \sum_{i=1}^{p} \log(|1 - \mathbf{z}_i - \boldsymbol{\theta}_{ji}|)\right)\right)^{-1}
\end{aligned}
$$

3.3 Methodology Comparison

In this chapter, we proposed LCA-FSA which uses annealed feature selection for training latent class analysis models. Broad comparisons between LCA feature selection approaches were made in 3.1. In particular, it was observed that regularization is a viable approach for high-dimensional data, and that nonspecific penalties are preferred to a priori structural assumptions about the parameter space without motivating domain knowledge.

Here we make more specific comparisons with other models.

Firstly, feature selection for categorical LCA is similar to the goal of feature selection in Gaussian Mixture Models. Feature selection for GMMs has seen more attention in the literature and is often a source of methodology later extended to other clustering settings. Pan and Shen (2007) performed feature selection for multivariate normal clustering using an L_1 penalty and thresholding. They remark that penalized likelihood had been thoroughly examined in the supervised setting, but not for multivariate clustering.

Dong et al. (2021) examine feature screening for a high-dimensional LCA model using joint $L_0 - L_1$ constraints. Their incorporation of L_0 allows them to achieve their goal of efficient feature filtering prior to training a classification model. In spite of the recency of this work, the authors observe that the feature selection literature continues to have little focus on the unsupervised setting. The overall method examined is comparable to LCA-FSA, with the distinction that Dong et. al. examine continuous (GMM) data.

Houseman et al. (2006) do apply penalization to the categorical LCA setting. Ridge and lasso penalties were found to be necessary to estimate an LCA model. The focus of their experiments was stability of model estimation and accuracy of predicted clustering, and so evaluating the correctness of feature selection was not relevant to their context. Further, their data experiments had low-to-moderate p.

Leoutsakos et al. (2011), closely following the work of Houseman, apply penalties to latent class regression in low-p settings. Following experiments and discussion of ridge and lasso penalties, they suggest that better results may derive from penalization that somehow promotes class separation. Specifically, they surmise a penalty that rewards features for strong association with a particular class and not others, an idea we will return to later.

There appears to be a consensus that stepwise or "best subsets" feature selection methods are inappropriate for high dimensions. In their examination of the stepwise approach to LCA feature selection, Fop et al. (2017) mention in passing that regularization is a possible approach, though they are yet to encounter the technique applied to categorical clustering. Their methodology draws heavily from the variable selection framework of Raftery and Dean (2006), which in its concluding discussion raises the question of whether their model would perform better if their method's greedy search algorithm were replaced with simulated annealing.

Riyanto et al. (2022) recently applied mutual information in a forward search method embedded into the training of LCA for categorical data. The approach was tested on data from the government of Indonesia, which classified villages into five categories of economic development based on a series of discrete measures surveyed for each village. Their model reduced the number of features needed to estimate the classification with the goal of reducing the number of features needed in village surveys, which were costly to collect. Similar to this work, we will examine mutual information for LCA-FSA, though our approach differs in several ways. Firstly, we will include simulated data for which we can objectively measure the correctness of feature selection. Secondly, we will include tests of feature selection on simulated and real data with high dimensionality, whereas the authors' experiment reduced data from 12 features to 4. Finally, we will compare multiple measures of feature relevance rather than applying only mutual information, which the authors cite as a limitation of their work.

Silvestre et. al. train categorical LCA with an embedded minimum message length (MML) criterion to perform feature selection (Silvestre et al., 2015) and the selection of the number of latent classes (Silvestre et al., 2022). Borrowing a technique from Law et al. (2004), the authors modify the likelihood to include parameters representing the probability of relevance for each feature. These parameters are estimated as part of the model and inform feature selection decisions. Experiments compare the embedded MML approach to a filtering strategy using normalized entropy and a wrapper strategy using mutual information. On synthetic data, the filter approach was unable to differentiate between relevant and irrelevant features. The wrapper approach had some success identifying relevant features, though not as well as embedded MML. On real data, embedded MML was shown to produce a model with a smaller MML than the filter and wrapper methods, though it is unclear what implication this has on model accuracy. Similarly in spirit to the embedded MML approach, LCA-FSA also applies an embedded feature selection approach to LCA. We will observe the ability to recover relevant features from synthetic data, as Silvestre et. al. do, but on data with moderate to large p. We will also test the ability to recover known labels on real-world data, allowing objective measures of model performance relative to parsimony.

3.4 LCA Simulations

In this section we present the result of a simulation study on LCA-FSA with various feature relevance metrics. The feature scoring methods are those introduced in 3.2.1: mutual information, χ^2 statistic, mRMR, diff-criterion, and a likelihood approach ("LL"). We also include results from the R package LCAvarsel (Fop and Murphy, 2017), which is a forward stepwise selection approach for LCA feature selection, called "FWD" in our experiments. [1] Finally, we include results from latent class analysis without feature selection ("No FS").

3.4.1 Simulations Setup

To compare LCA models with feature selection, we simulate data following a latent class model and observe (i) the ability to recover correct clustering into latent classes, and (ii) the ability to identify relevant features.

Simulated data samples are generated from initial specifications:

- k, the number of latent classes
- n, the number of rows (observations)
- ω, the number of relevant features
- q, the number of noisy features
- $p = \omega + q$, the total number of features

Given these, a specific data sample is generated in the following steps:

1. Generate T, a length-n vector of class labels, by selecting uniformly from $\{1, ..., k\}$
2. Generate $\boldsymbol{\theta}$ as a random k-by-p matrix with entries sampled uniformly from $(0, 1)$
3. Generate $\boldsymbol{\pi}$ as a length-k vector with values $1/k$
4. Generate n-by-ω matrix $\tilde{\mathbf{Z}}$ as a binary matrix by sampling $\tilde{\mathbf{Z}}_{ij}$ from $Bernoulli(\boldsymbol{\theta}_{cj})$ where $c{=}T_i$
5. Generate n-by-q noisy data $\mathbf{N}$ as a binary matrix by sampling $\mathbf{N}_{ij}$ from $Bernoulli(r_j)$ where $\{r_i\}_{i=1}^{q}$ is initialized as random uniform deviates
6. Generate observed data matrix $\mathbf{Z}$ with $\mathbf{Z} = \begin{bmatrix} \tilde{\mathbf{Z}} & \mathbf{N} \end{bmatrix}$

In practice, we follow the data generation steps using $2n$ rows to obtain equally-sized training and testing data matrices.

We run simulations on the following data settings

- $k \in \{2, 5, 30\}$

[1] The forward stepwise approach performs feature selection by a series of feature addition/removal steps. The model completes its feature selection when no further modification steps seem helpful, and thus choosing the total number of features to retain is part of model estimation. To fit into our discussion, we modify the forward selection algorithm to target an exact specified number of features to retain.

- $n \in \{100, 500, 1000, 5000\}$
  n.b. $n = 100$ is excluded when $k = 30$
- $\omega = 10$
- $q \in \{10, 50, 100, 500, 1000\}$

3.4.2 Scoring Metrics

Models applying feature selection are given q, i.e., they are configured to select the top ω features. Thus we are able to report the number of correctly selected features for each trained model applying feature selection. Our simulations fix $\omega = 10$, so this metric ranges from zero (all incorrectly chosen) to ten (all correctly chosen).

Estimation of a latent class model includes estimating the assignment of observations into latent classes. This is equivalent to predicting T, though not necessarily with the same labeling scheme. Thus T and predicted values $\hat{T}$ should not be thought of in terms of their label values, but as partitioning schemes on rows of $\mathbf{Z}$. To measure the accuracy of $\hat{T}$ we report the Adjusted Rand Index (Hubert and Arabie, 1985; Martinez et al., 2017) between T and $\hat{T}$ on test data.

3.4.3 Simulations Results

Results are organized by simulation setting (2-class, 5-class, and 30-class) and metric examined. These results are presented in Figures 3.1-3.3 and Tables 3.1-3.3.

For the 2-class model, we observe the LCA-FSA models' ability to recover true features in Figure 3.1. The results show slight differentiation in model performance with χ^2, diff, and MI performing well, LL slightly lagging, and mRMR underperforming. This pattern holds as q increases. The performance of mRMR worsens relative to other models with increasing q, while other models retain the ability to recover most true features. Forward selection (FWD) performs well up to $q = 100$ but suffers for higher values.

Adjusted Rand Index scores on test data for the 2-class problem show mRMR being outperformed by other models for $q <= 100$. For higher q, mRMR and FWD stand out as underperformers.

Feature selection metrics for the 5-class simulations with q=10 (Figure 3.2) show better performance for all models compared to the 2-class scenario, except FWD which seems unable to accommodate the higher number of latent classes.. The performance of mRMR continues to lag, though with less of a difference as q increases. In some cases the performance of mRMR improves with higher q, suggesting mRMR with lower q may benefit from a slower annealing schedule. Overall, all methods but FWD show an ability to recover relevant features well with larger n, and diff-criterion appears to show the best performance.

Adjusted Rand Index scores show lower performance than for the 2-class problem. Further, the performance hit that the non-FS model takes with increasing q becomes evident at 50 noisy features, where in the 2-class setting the difference appeared at $q = 500$.

In the 30-class simulation (Figure 3.3), all models except FWD are able to identify relevant features for low q and for moderate q when n is high. Performance suffers for high q, with the highest performance achieved by diff-criterion.

Rand Index scores for the 30-class setting are too low to be of utility in a practical data analysis setting. We include these results as a stress-test of models. For high q for instance, we find the point where the model without feature selection is unable to recover any signal.

A point of inquiry in these results is the isolated poor performance of mRMR at various settings. We posit that the cause may be the nature of the simulated data. Modeling data in these simulations involves recovering true features in the face of an increasing number of noisy features, a kind of needle in a haystack problem. The best-performing model will select all relevant features and drop all noisy features. mRMR however penalizes redundancy in selected features, while this data may benefit from capitalizing on discovered correlations. In later sections, we will examine the performance of mRMR in settings where the feature selection task is to select the most useful features out of many relevant ones, rather than to separate relevant from irrelevant features.

Another specific model to examine is the forward selection model. Stepwise methods are not suited for high-dimensional problems, and this becomes clear as either q or the number of latent classes increases. Further, the stepwise modeling approach takes much longer, as shown in Table 3.4.

Table 3.1: 2-Class LCA Simulations

		n Features Correctly Selected						**Test Adj. Rand Index**						
q	**n**	**LL**	**MI**	χ^2	**diff**	**FWD**	**mRMR**	**LL**	**MI**	**No FS**	χ^2	**diff**	**FWD**	**mRMR**
10	100	7.78	6.30	7.85	7.90	7.43	6.33	0.65	0.54	0.65	0.66	0.65	0.64	0.44
	500	8.40	8.68	8.85	8.70	8.60	8.03	0.76	0.77	0.77	0.77	0.77	0.77	0.61
	1000	8.65	8.93	8.98	9.00	9.08	8.38	0.77	0.77	0.78	0.78	0.78	0.78	0.62
	5000	8.93	9.28	9.35	9.35	9.53	8.63	0.77	0.78	0.78	0.78	0.78	0.78	0.62
50	100	5.40	3.08	5.78	5.98	5.43	2.63	0.53	0.26	0.46	0.55	0.61	0.61	0.18
	500	8.18	8.00	8.38	8.23	7.98	6.48	0.76	0.73	0.75	0.76	0.76	0.76	0.57
	1000	8.73	8.65	8.88	8.78	8.75	6.58	0.77	0.76	0.77	0.77	0.77	0.78	0.51
	5000	8.85	9.05	9.45	9.30	9.05	6.75	0.77	0.76	0.78	0.78	0.78	0.78	0.46
100	100	4.45	2.18	3.95	4.40	3.58	2.00	0.44	0.21	0.26	0.37	0.42	0.36	0.16
	500	7.50	5.93	7.80	7.83	7.58	4.35	0.75	0.58	0.73	0.75	0.76	0.76	0.41
	1000	7.83	7.55	8.28	8.40	8.58	4.70	0.75	0.68	0.76	0.75	0.77	0.77	0.37
	5000	8.63	9.28	9.30	9.23	8.70	5.53	0.78	0.78	0.78	0.78	0.78	0.76	0.39
500	100	0.45	0.03	0.50	0.83	0.10	0.03	0.04	0.00	0.01	0.04	0.08	0.00	0.00
	500	4.38	1.25	4.48	4.83	4.33	0.98	0.47	0.13	0.33	0.48	0.52	0.47	0.09
	1000	6.35	5.30	6.58	7.00	3.73	1.40	0.65	0.51	0.59	0.65	0.69	0.40	0.12
	5000	8.53	8.30	9.28	9.03	3.25	2.00	0.78	0.72	0.76	0.78	0.78	0.32	0.16
1000	100	0.15	0.00	0.13	0.15	0.13	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00
	500	2.00	0.00	1.78	2.30	0.75	0.05	0.24	0.00	0.02	0.22	0.29	0.09	0.00
	1000	4.50	1.45	4.55	4.78	0.67	1.35	0.52	0.16	0.25	0.48	0.52	0.09	0.15
	5000	7.85	7.30	8.88	8.85	0.15	3.00	0.72	0.62	0.73	0.76	0.76	0.04	0.29

Table 3.2: 5-Class LCA Simulations

		n Features Correctly Selected						Test Adj. Rand Index						
q	**n**	**LL**	**MI**	χ^2	**diff**	**FWD**	**mRMR**	**LL**	**MI**	**No FS**	χ^2	**diff**	**FWD**	**mRMR**
10	100	8.08	7.50	7.70	8.25	6.43	7.25	0.22	0.21	0.20	0.23	0.24	0.13	0.20
	500	9.73	9.25	9.53	9.73	6.38	8.75	0.48	0.45	0.45	0.47	0.48	0.19	0.40
	1000	9.73	9.58	9.83	9.83	5.65	8.95	0.52	0.51	0.52	0.53	0.53	0.17	0.45
	5000	9.60	9.68	9.93	9.83	4.75	9.00	0.56	0.56	0.57	0.57	0.56	0.18	0.50
50	100	4.73	3.30	4.60	5.63	3.68	3.38	0.10	0.07	0.05	0.11	0.14	0.07	0.08
	500	9.30	8.13	8.95	9.23	5.55	7.95	0.43	0.39	0.31	0.43	0.44	0.16	0.37
	1000	9.80	8.93	9.78	9.80	5.45	8.03	0.50	0.46	0.44	0.51	0.51	0.17	0.40
	5000	9.93	9.78	9.90	10.00	4.60	8.45	0.55	0.55	0.54	0.55	0.55	0.17	0.44
100	100	2.58	1.03	2.53	3.70	1.88	1.38	0.06	0.03	0.01	0.05	0.09	0.03	0.02
	500	8.73	7.08	8.13	8.78	5.05	6.73	0.39	0.32	0.18	0.38	0.39	0.16	0.30
	1000	9.75	8.95	9.35	9.70	5.18	8.08	0.48	0.44	0.35	0.47	0.49	0.17	0.39
	5000	9.83	9.95	9.90	9.95	4.65	8.25	0.56	0.57	0.54	0.57	0.57	0.17	0.44
500	100	0.25	0.00	0.35	0.53	0.23	0.05	0.00	0.00	0.00	0.00	0.01	0.00	0.00
	500	2.75	0.58	2.43	4.40	1.40	0.78	0.10	0.02	0.01	0.09	0.17	0.04	0.03
	1000	6.93	3.28	6.03	7.40	2.15	3.55	0.31	0.15	0.03	0.26	0.34	0.07	0.17
	5000	9.80	8.55	9.73	9.95	2.08	7.70	0.50	0.42	0.38	0.49	0.51	0.07	0.38
1000	100	0.08	0.00	0.15	0.23	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	500	1.05	0.20	1.05	2.00	0.08	0.18	0.03	0.00	0.00	0.03	0.06	0.00	0.00
	1000	3.38	1.40	3.23	5.25	0.18	1.48	0.14	0.06	0.00	0.14	0.20	0.00	0.06
	5000	9.48	7.03	9.38	9.83	0.13	6.65	0.48	0.36	0.20	0.46	0.49	0.00	0.33

Table 3.3: 30-Class LCA Simulations

		n Features Correctly Selected						**Test Adj. Rand Index**						
q	**n**	**LL**	**MI**	χ^2	**diff**	**FWD**	**mRMR**	**LL**	**MI**	**No FS**	χ^2	**diff**	**FWD**	**mRMR**
10	500	8.53	8.03	7.78	8.68	6.60	8.03	0.07	0.07	0.05	0.07	0.07	0.01	0.07
	1000	9.65	9.18	8.73	9.55	6.78	9.05	0.10	0.09	0.06	0.09	0.10	0.01	0.09
	5000	9.90	9.53	9.85	10.00	5.23	9.10	0.15	0.15	0.13	0.16	0.16	0.01	0.13
50	500	5.20	4.48	4.58	5.68	3.53	4.40	0.04	0.03	0.01	0.03	0.04	0.01	0.03
	1000	7.85	6.30	6.68	7.85	4.60	6.35	0.07	0.06	0.01	0.06	0.08	0.01	0.06
	5000	10.00	9.35	9.93	9.93	4.50	9.25	0.15	0.14	0.06	0.15	0.15	0.01	0.14
100	500	2.53	2.20	2.25	4.13	1.93	2.23	0.02	0.01	0.00	0.01	0.02	0.00	0.02
	1000	4.88	4.00	4.35	5.95	3.80	3.80	0.04	0.03	0.00	0.03	0.05	0.01	0.03
	5000	10.00	9.63	9.80	9.98	4.48	9.68	0.14	0.13	0.03	0.14	0.14	0.01	0.13
500	500	0.40	0.08	0.33	0.73	0.30	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1000	0.63	0.13	0.55	1.55	0.35	0.33	0.00	0.00	0.00	0.00	0.01	0.00	0.00
	5000	6.70	5.60	6.05	9.13	1.46	5.15	0.07	0.06	0.00	0.06	0.10	0.00	0.05
1000	500	0.25	0.05	0.18	0.38	0.13	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	1000	0.20	0.03	0.20	0.48	0.00	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	5000	2.28	1.63	2.48	6.90	0.22	2.10	0.02	0.01	0.00	0.02	0.07	0.00	0.02

Table 3.4: LCA Simulations Average Model Estimation Time (s)

n	**noisy**	**LL**	**MI**	**No FS**	χ^2	**diff**	**FWD**	**mRMR**
100	**10**	0.20	0.23	0.35	0.53	0.29	33.20	0.22
	50	0.41	0.30	0.19	1.17	0.24	126.46	0.46
	100	0.55	0.31	0.51	2.95	0.42	267.78	0.84
	500	1.54	0.83	0.48	6.98	0.52	1466.94	5.51
	1000	2.76	0.88	0.73	12.78	0.79	2591.24	17.42
500	**10**	0.82	1.01	0.69	1.24	0.83	77.40	0.78
	50	2.04	1.06	1.22	2.49	1.11	262.82	1.28
	100	4.00	1.38	1.60	4.95	1.45	562.85	1.97
	500	14.23	3.62	5.07	12.88	3.30	4448.61	9.75
	1000	24.52	5.96	9.13	22.84	5.57	9415.28	27.90
1000	**10**	1.29	1.27	1.17	1.42	1.31	130.38	1.23
	50	3.21	1.68	2.15	3.40	1.74	486.92	1.84
	100	6.85	2.67	2.73	6.24	2.23	1014.46	2.80
	500	24.41	5.94	9.67	18.17	6.07	8413.09	13.94
	1000	42.57	9.82	15.80	32.34	9.71	25430.59	38.91
5000	**10**	4.56	4.28	4.29	4.55	3.89	682.18	4.38
	50	11.53	6.35	7.98	11.42	6.19	2104.82	6.26
	100	26.75	8.37	11.03	19.15	7.82	4594.74	9.19
	500	108.28	24.57	44.93	62.62	23.50	66734.08	49.51
	1000	224.49	49.27	93.54	120.43	49.63	118644.04	131.82

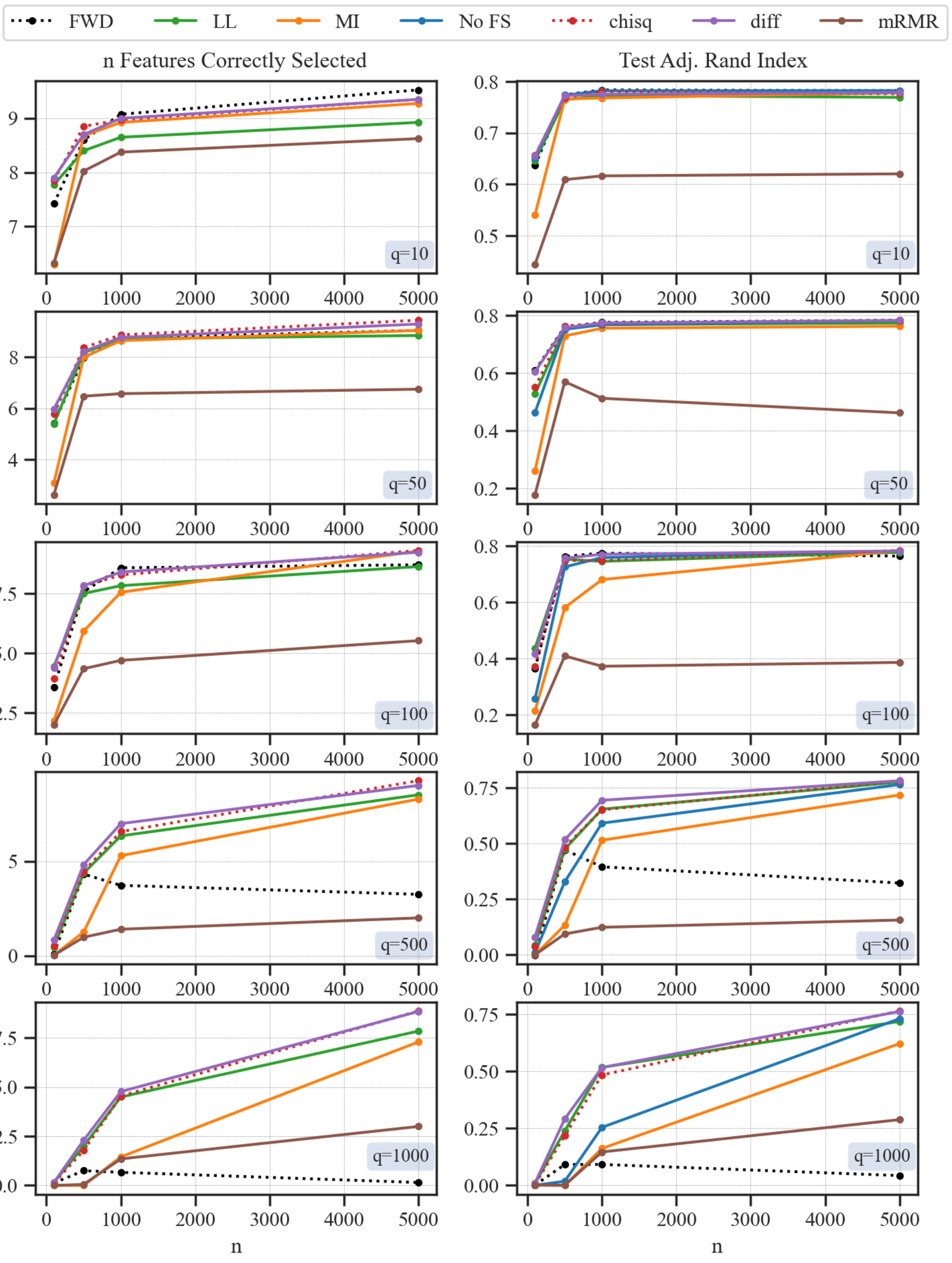


Figure 3.1: LCA Simulation Scores for 2-Class Simulations with varying n, q

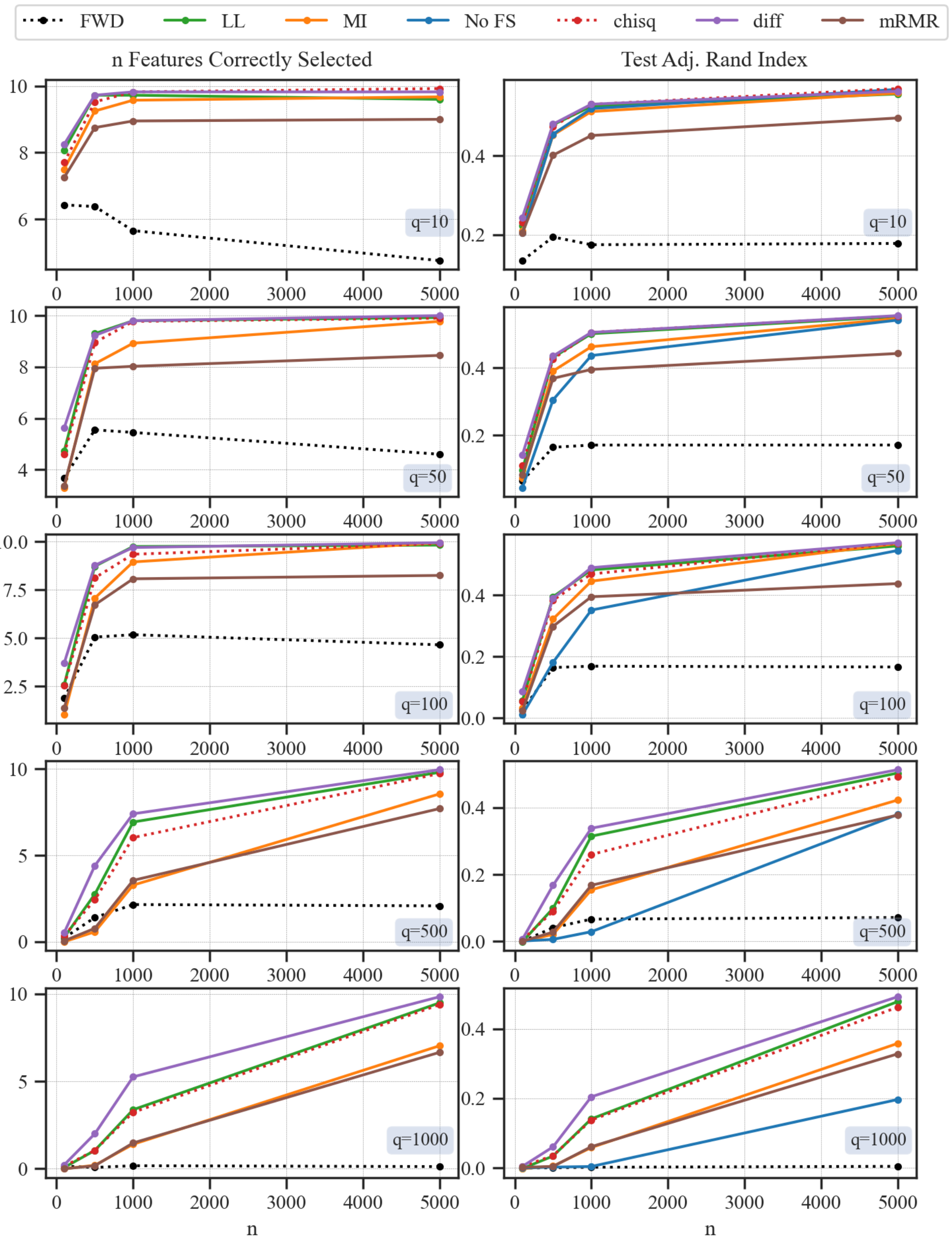


Figure 3.2: LCA Simulation Scores for 5-Class Simulations with varying n, q

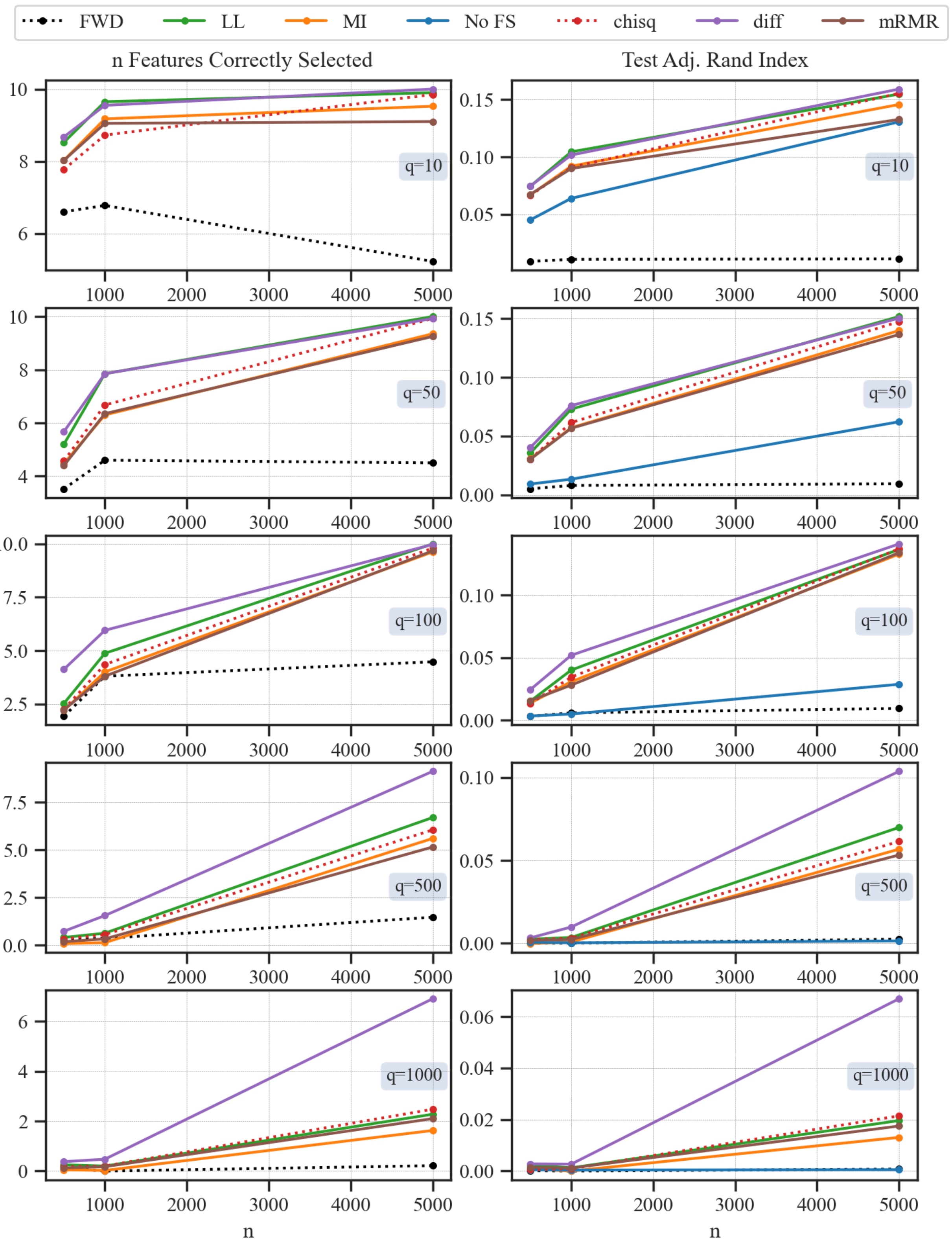


Figure 3.3: LCA Simulation Scores for 30-Class Simulations with varying n, q

3.5 LCA Experiment - Classification of School Types

In this section, we test LCA-FSA by attempting to recover underlying class labels of real data for which the true class labels are known. Comparable to a classification setting, we use knowledge of the true labeling to measure the accuracy of class predictions.

3.5.1 Experiment Data and Setup

Data for this experiment is derived from public data on K-12 schools published by the Florida Department of Education (2016). For the 2015-2016 academic year, we collect FLDOE's reported classification of each Florida public school as an elementary, middle, or high school. We combine this with FLDOE records on enrollment figures in individual classes for each public school for that time period. The resulting data matrix $\mathbf{Z}$ includes a row for each school with binary features indicating whether each class had any enrollment at that school. The LCA models will attempt to recover class labels y, which indicate school level.

The data includes 2,909 schools which collectively offer 2,586 classes, producing $\mathbf{Z}$ with n=2,909 and p=2,586. The true mixing proportions of classes are 63.42% elementary school, 20.04% middle school, and 16.53% high school. The data is saturated with 0's, with an overall sparsity rate of 97.1%. The observed cumulative distribution of sparseness of features is shown in Figure 3.4, showing that over 90% of features have fewer than 10% nonsparse observations.

It would be fair to question the difficulty of the task of differentiating between elementary, middle, and high schools. One could imagine the ability to achieve accurate clustering with as few as two features. We offer the following observations:

1. Not all schools of a particular type offer the same classes. For instance, Advanced Placement classes, International Baccalaureate classes, and trade classes exist in high schools, but no one of those categories of coursework can successfully identify all high schools.

2. Schools of different types can overlap in course offerings. For instance a college preparatory middle school can offer coursework usually encountered in high school, and a remedial or second chance high school can offer coursework typically found in middle schools.

3. The problem is unsupervised. Relationships between features and latent classes which would be easy to discover in a supervised setting may prove elusive in the unsupervised case. In particular, the maximum amount of parsimony which still permits a performant model is not evident a priori.

4. Finally, the value of examining this data is demonstrated by the experimental results, which show marked performance differences of competing FSA approaches at varying levels of feature retention.

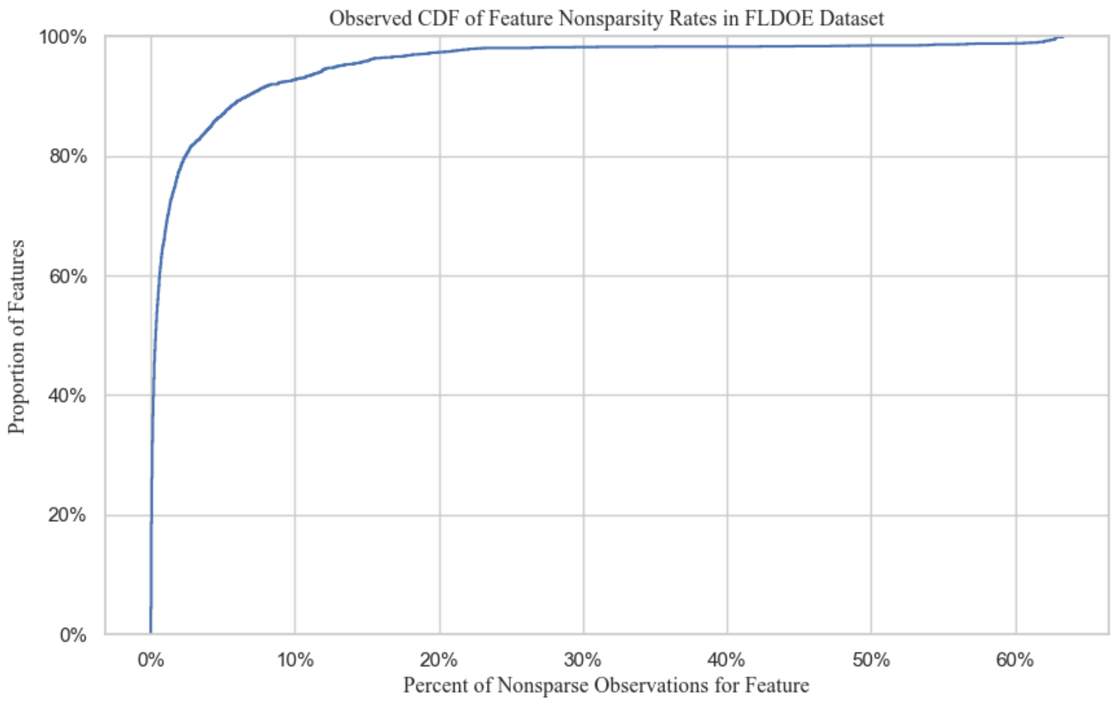


Figure 3.4: Observed Cumulative Distribution of Percent of Positive Cases for FLDOE Data Features

Because there is no analogue of selecting a "true" number of features, we train LCA models at varying numbers of retained features, ω. To measure performance, we split the data into training and testing sets and report the average Adjusted Rand Index for each model over 100 train/test replicates.

3.5.2 Experiment Results

Results are shown in Figure 3.5 and Table 3.5.

The non-FS LCA model does not perform feature selection, but is included as a reference line in the chart for clarity. Forward selection for LCA is excluded because the software implementation we had used for that model in simulations was unable to accommodate data with such a large number of features. From these results it is observed that feature selection methods show unequal sensitivity to choice of ω. mRMR and χ^2 show performance that decreases steadily with lower ω.

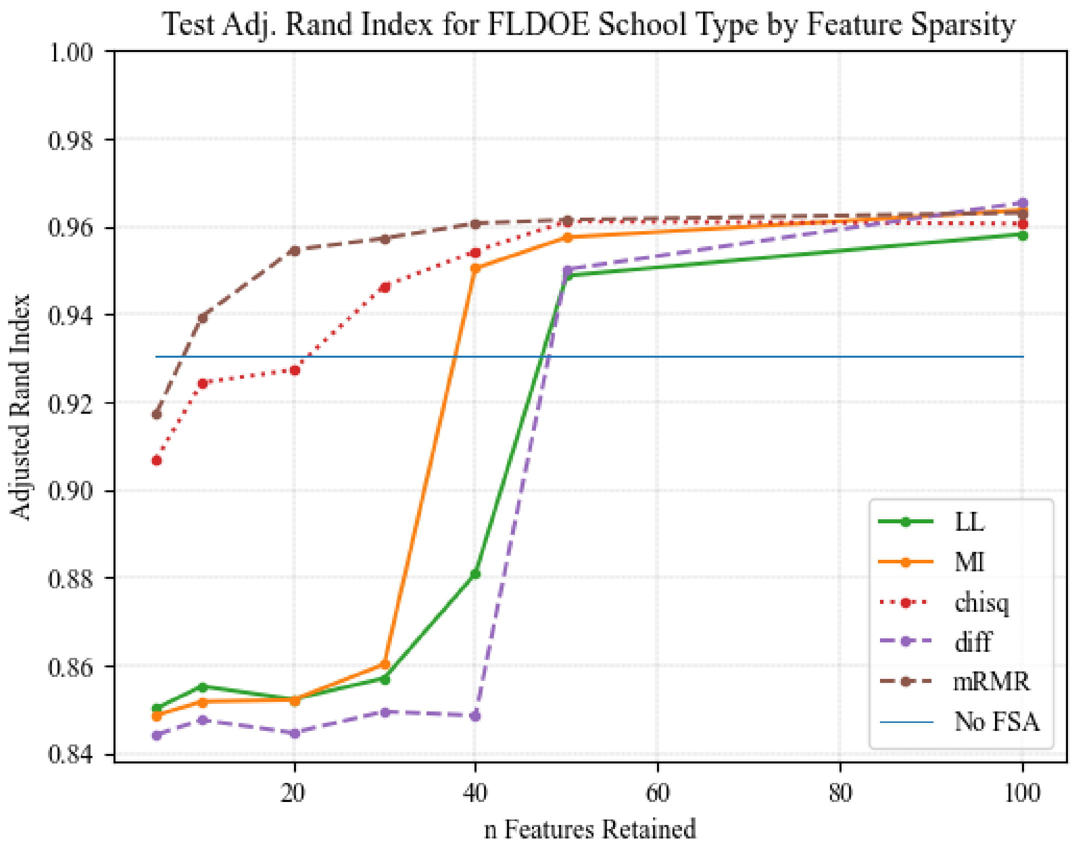


Figure 3.5: Recovering Latent Classes from FLDOE School Type Data. Average Test Adj. Rand Index over 100 Train/Test Splits per Model.

MI, LL, and diff, however, show a sharp performance decrease between the 30-feature models and 50-feature models. With the upper range of feature retention (ω=100), the different models appear to converge to comparable performance.

A second observation is that with all LCA-FSA models outperform ordinary LCA without feature selection at $\omega \geq 50$. LCA-FSA with mRMR in particular achieved a higher average accuracy with a parsimonious ω=10 model than the full LCA model using all 2,586 features. Table 3.6 shows some details of one of these 10-feature mRMR models, listing the features (courses) the model selected, the actual observed proportions of those courses occurring within each school type, and the corresponding model estimates $\hat{\boldsymbol{\theta}}$.

Table 3.5: Test Adj. Rand Index for FLDOE School Type by ω

ω	**No FS**	**LL**	**MI**	χ^2	**diff**	**mRMR**
5	0.93	0.85	0.85	0.91	0.84	0.92
10		0.86	0.85	0.92	0.85	0.94
20		0.85	0.85	0.93	0.84	0.95
30		0.86	0.86	0.95	0.85	0.96
40		0.88	0.95	0.95	0.85	0.96
50		0.95	0.96	0.96	0.95	0.96
100		0.96	0.96	0.96	0.97	0.96

Table 3.6: Selected Features, Actual Feature Proportions, and $\hat{\boldsymbol{\theta}}$ of a Sample 10-Feature LCA-FSA Model on FLDOE Data

Course	**Elementary School Proportion**	**Middle School Proportion**	**High School Proportion**	**Estimated Elementary School Parameter**	**Estimated Middle School Parameter**	**Estimated High School Parameter**
LANG ARTS GRADE 3	0.998	0.000	0.002	0.999	0.000	0.000
MATH GRADE THREE	0.998	0.000	0.002	0.999	0.000	0.000
PHYSICAL EDUCATION 3	0.997	0.000	0.002	0.999	0.000	0.000
SCIENCE GRADE THREE	0.997	0.000	0.002	0.998	0.000	0.000
PHYSICAL EDUCATION 2	0.991	0.000	0.002	0.992	0.000	0.000
M/J GRADE 7 MATH	0.001	0.957	0.010	0.002	0.986	0.011
M/J LANG ARTS 2	0.001	0.952	0.012	0.002	0.974	0.014
M/J GRADE 8 PRE-ALG	0.001	0.962	0.015	0.002	0.982	0.018
ENG 2	0.000	0.000	0.919	0.001	0.000	0.894
ENG 3	0.000	0.000	0.909	0.001	0.000	0.883

3.6 LCA Experiment - Text Classification

In this section we test LCA-FSA on a text classification task. Abstract text from published thesis work are collected from three departments of an academic institution. The modeling task is to accurately cluster these text documents, recovering their grouping into departments.

3.6.1 Experiment Data and Setup

A data matrix is prepared of thesis abstract text and the thesis's department of origin. This data is collected from open-access metadata records hosted by an academic institution (Florida State University Libraries, 2018). Published abstracts are collected from three academic departments: Psychology (n=641), English (n=634), and Earth, Ocean and Atmospheric Sciences (n=628). These three were selected as they are well-differentiated from each other in terms of their domains of knowledge, and because they each have a similar number of documents.

The final data matrix has n=1,903 and p=27,332, and we intend to model it as a three-class latent model. What follows is an overview of the process used to generate the data.

3.6.2 Natural Language Processing

In order to apply LCA, we must convert a corpus of text data into a binary matrix with one record per document.

A text document is first *tokenized*, that is, converted to a list of words it contains. During tokenization, several preprocessing steps are used to improve the usefulness of the data. Common steps performed are

- The removal of punctuation marks.
- The removal of *stop-words*, which are words so commonly used that they are not helpful in differentiating different types of text content, such as "a", "the", "and".
- *Lemmatization*, which is the transformation of a word to a common, basic form independent of its grammatical usage. For instance, lemmatization would reduce "automobiles" to "automobile".
- Part-of-speech tagging. Parsing the part of speech for a token (e.g. noun, verb, adjective) can enhance the quality of lemmatization.

For instance, tokenization of the first line of the Gettysburg Address:

```
Four score and seven years ago our fathers brought forth on this continent,
a new nation, conceived in Liberty, and dedicated to the proposition that
all men are created equal.
```

produces list of tokens

```
four, score, seven, year, ago, father, bring, forth, continent, new, nation,
conceive, liberty, dedicate, proposition, men, create, equal
```

The final step to produce a data matrix is to convert the lists of tokens for each document into a binary indicator matrix with documents as rows, tokens as features, and values indicating whether a token occurred in a document. This step, commonly called the *bag of words* approach, converts tokens to vectors in a vector space.

Remark 4. *Populating a document-token matrix with binary indicators is not the most powerful approach to creating a bag of words matrix of data. Instead, the matrix can contain the frequency of each token's occurrence in each document, that frequency scaled by how commonly the token occurs within the entire corpus, or other variations. These approaches produce continuous vectors that can more sensitively capture differences between documents. However, to examine LCA, we utilize the binary approach.*

Detailed coverage of this and other NLP techniques used here can be found in Zong et al. (2021).

We apply tokenization, stop-word removal, part-of-speech tagging, and lemmatization with the Natural Language Toolkit software library (Bird et al., 2009). Processing into bag of words is performed using the Gensim software library (Řehůřek and Sojka, 2010).

The final data matrix has 1,903 observations and 27,332 features. The large p is a reflection of how many unique tokens were extracted from the documents. For clustering purposes, we anticipate that most of these features are irrelevant. Further, most features will be extremely sparse, occurring in very few documents.

3.6.3 Experiment Results

LCA and LCA-FSA are applied to the generated data and the resulting clustering is compared to the known labeling of abstracts into departments using the Adjusted Rand Index. Average test scores over 100 train/test splits are shown in Table 3.8 and Figure 3.6.

The results show a striking pattern of clustering accuracy as a function of the number of features retained, ω. Figure 3.6 includes a constant line showing the score of latent class analysis with no

Table 3.7: Summary Statistics on Text Documents in Corpus

	Minimum	Maximum	Mean	Median	Standard Deviation
Number of Sentences	1	94	12.1	10	8.2
Number of Tokens Extracted	1	1,530	184.3	158	121.7

feature selection. For this data, any amount of feature selection outperformed LCA without feature selection.

The highest-performing choice of ω for all feature selection strategies was to subset 500 features from the original 27,332. The diff-criterion showed the best performance, and mRMR performed the worst out of LCA-FSA models. Forward selection results are not included as the relevant software package was unable to handle data with this number of features.

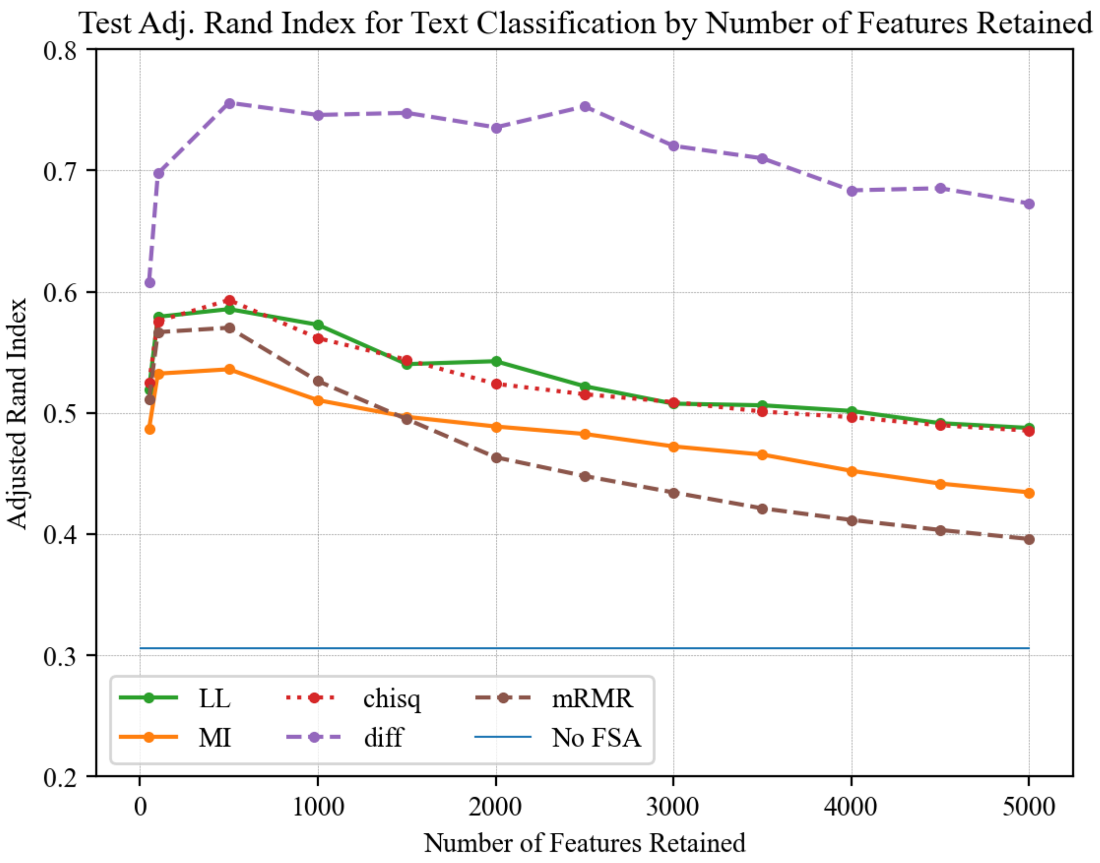


Figure 3.6: Average Test Adjusted Rand Index on Text Classification Data

Table 3.8: Test Adj. Rand Index for Text Classification by ω

ω	**No FS**	**LL**	**MI**	χ^2	**diff**	**mRMR**
50	0.31	0.52	0.49	0.52	0.61	0.51
100		0.58	0.53	0.58	0.70	0.57
500		0.59	0.54	0.59	0.76	0.57
1000		0.57	0.51	0.56	0.75	0.53
1500		0.54	0.50	0.54	0.75	0.49
2000		0.54	0.49	0.52	0.74	0.46
2500		0.52	0.48	0.52	0.75	0.45
3000		0.51	0.47	0.51	0.72	0.43
3500		0.51	0.47	0.50	0.71	0.42
4000		0.50	0.45	0.50	0.68	0.41
4500		0.49	0.44	0.49	0.69	0.40
5000		0.49	0.43	0.49	0.67	0.40

A sample LCA-FSA model trained with diff-criterion at ω=50 produces an estimate $\hat{\boldsymbol{\theta}}$ that is a manageable size for interpretation. The 50 features selected correspond to tokens:

> `analysis, analyze, area, behavior, climate, collection, compare, condition, control, current, data, determine, due, effect, examine, factor, finding, high, however, increase, indicate, large, life, low, may, measure, model, ocean, participant, region, result, results, sample, show, significant, story, study, suggest, support, surface, task, temperature, test, time, use, variability, water, wind, work, write`

$\hat{\boldsymbol{\theta}}$ can take practical meaning by observing which tokens take the highest values for each class. This is shown in Tables 3.9-3.11. In each table, $\hat{\boldsymbol{\theta}}$ is subsetted and reordered for clarity in understanding each class.

Classes can be interpreted by the tokens which are probable in one class and improbable in others. From Table 3.9 it can be seen that Class 1 has several tokens overlapping with Class 3, such as `study` and `use`. However Class 1 also has differentiating tokens, including: `surface, water, temperature, region, area, analyze, ocean, wind, climate, variability`. By observing groundtruth labels, we confirm that this cluster represents abstracts from the Department of Earth, Ocean and Atmospheric Sciences.

Class 2 captures the Department of English. Table 3.10 shows differentiating tokens `story, life, write, collection`. Differentiating tokens for Class 3, Psychology, are shown in Table 3.11 to be `finding, behavior, participant, task`.

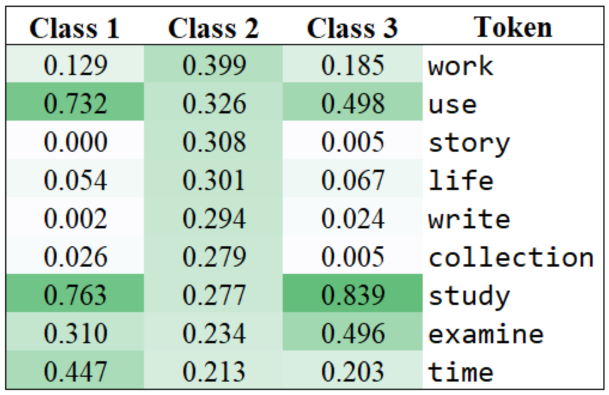
Table 3.9: Top Values for Tokens in Sample $\hat{\boldsymbol{\theta}}$ for Class 1

$\hat{\boldsymbol{\theta}}$ Subsetted and Reordered for Clarity

Class 1	Class 2	Class 3	Token
0.763	0.277	0.839	study
0.732	0.326	0.498	use
0.593	0.035	0.275	data
0.592	0.141	0.260	show
0.559	0.119	0.433	result
0.512	0.071	0.315	model
0.452	0.063	0.290	high
0.447	0.213	0.203	time
0.443	0.127	0.270	analysis
0.393	0.107	0.119	large
0.380	0.144	0.409	however
0.376	0.020	0.161	determine
0.371	0.018	0.005	surface
0.365	0.026	0.189	compare
0.354	0.012	0.192	low
0.345	0.039	0.325	increase
0.345	0.010	0.016	water
0.343	0.000	0.007	temperature
0.340	0.026	0.043	region
0.328	0.024	0.104	due
0.311	0.082	0.431	suggest
0.310	0.234	0.496	examine
0.306	0.099	0.394	may
0.306	0.023	0.319	indicate
0.306	0.037	0.079	area
0.293	0.037	0.205	condition
0.293	0.040	0.233	significant
0.281	0.102	0.037	analyze
0.277	0.005	0.000	ocean
0.271	0.080	0.421	effect
0.261	0.002	0.000	wind
0.248	0.000	0.004	climate
0.247	0.000	0.031	variability
0.237	0.013	0.286	sample

Table 3.10: Top Values for Tokens in Sample $\hat{\boldsymbol{\theta}}$ for Class 2

$\hat{\boldsymbol{\theta}}$ Subsetted and Reordered for Clarity



Class 1	Class 2	Class 3	Token
0.129	0.399	0.185	`work`
0.732	0.326	0.498	`use`
0.000	0.308	0.005	`story`
0.054	0.301	0.067	`life`
0.002	0.294	0.024	`write`
0.026	0.279	0.005	`collection`
0.763	0.277	0.839	`study`
0.310	0.234	0.496	`examine`
0.447	0.213	0.203	`time`

Table 3.11: Top Values For Tokens in Sample $\hat{\boldsymbol{\theta}}$ for Class 3

$\hat{\boldsymbol{\theta}}$ Subsetted and Reordered for Clarity

Class 1	Class 2	Class 3	Token
0.763	0.277	0.839	study
0.732	0.326	0.498	use
0.310	0.234	0.496	examine
0.559	0.119	0.433	result
0.311	0.082	0.431	suggest
0.271	0.080	0.421	effect
0.380	0.144	0.409	however
0.306	0.099	0.394	may
0.174	0.017	0.381	measure
0.163	0.068	0.357	current
0.080	0.039	0.344	finding
0.187	0.017	0.340	test
0.345	0.039	0.325	increase
0.159	0.077	0.323	control
0.056	0.027	0.320	behavior
0.306	0.023	0.319	indicate
0.512	0.071	0.315	model
0.158	0.004	0.312	results
0.190	0.026	0.310	factor
0.000	0.032	0.309	participant
0.133	0.049	0.294	support
0.452	0.063	0.290	high
0.237	0.013	0.286	sample
0.593	0.035	0.275	data
0.443	0.127	0.270	analysis
0.592	0.141	0.260	show
0.009	0.013	0.250	task
0.293	0.040	0.233	significant
0.293	0.037	0.205	condition
0.447	0.213	0.203	time

3.7 Discussion

Simulation and experiment results show a range of outcomes for competing feature relevance metrics used with LCA-FSA.

An initial observation is that there can be marked differences in performance between feature selection methods, even when they are conceptually similar. mRMR for instance has a close relationship to MI, but shows very different behavior on simulated data. These performance differences highlight the need to investigate comparative performance of different feature selection criteria.

At the same time, it is also observed that no single feature selection approach dominates the others in accuracy across data settings. For instance, the best performer on both simulated data and text classification data was diff-criterion and the worst was mRMR; this situation is reversed for school classification data. Thus it is noted not only that FSA criteria ought to be compared in terms of performance, but that performance in one data setting is not necessarily an indicator of performance in other settings.

A pattern shared by all tested FSA methods is improved accuracy on simulated data with increasing n, which is to be expected. The same rule however does not follow for ω, the number of features retained in a model. For the school type data, model accuracy was approximately monotonically increasing against ω. For the text classification data, however, accuracy was monotonically *decreasing* against ω for all but the most extreme levels.

A final observation on these results is that, with few exceptions, FSA was able to improve the performance of LCA. LCA without FSA showed competitive performance on simulated data in settings with a low number of latent classes and a low number of noisy features. For simulations with a high number of latent classes or high dimensionality, however, it was outperformed by all FSA methods, including mRMR. In the most extreme case ordinary LCA was not able to learn at all. In experiment data, it is shown that a wide range of ω values can simultaneously achieve parsimony and improved performance.

CHAPTER 4

METHODOLOGY - SPARSE PRINCIPAL COMPONENT ANALYSIS

In this chapter we examine various approaches to imposing sparsity on PCA models. Extensions of the original SPCA (2.3) model are introduced which apply feature selection with annealing (2.4) to impose sparsity on model parameters during estimation. These methods include

1. ASPCA (4.1), which follows the SPCA algorithm but uses FSA for imposing sparsity
2. GSPCA (4.2), which imposes sparsity by applying FSA "globally" to the entire model loading matrix
3. WPCA (4.4), which incorporates feature weights into PCA
4. SELF (4.5), which directly estimates a low-rank representation of data $\mathbf{X}$ as its own entity, $\mathbf{\Gamma}$, rather than the usual approach of reducing data using a loading matrix as $\mathbf{XB}$.

In (4.3) we examine an imputation strategy for these models. SELF is also tested with additional strategies for handling missingness and incorporating feature weights. Algorithms for each method will be provided. These methods are tested on simulated and real data in a Principal Component Regression setting.

4.1 Annealed Sparse Principal Component Analysis

SPCA estimates a sparse PCA representation of data by estimating the columns of loading matrix $\mathbf{B}$ with sparse regression methods. We consider a sparse PCA model with sparsity produced by FSA (2.4) rather than the elastic net. FSA produces sparse regression estimates by annealing, so the L_1 penalty of SPCA is replaced by the desired annealing schedule.

Having a customizable annealing schedule by parameter μ, as those shown in Figure 2.1, may allow more control over the sparsity estimation. We test *Annealed Sparse Principal Component Analysis (ASPCA)* to see if the choice of FSA or elastic net for sparsity affects performance.

$$\boxed{(\hat{\mathbf{A}}_{ASPCA}, \hat{\mathbf{B}}_{ASPCA}) = \underset{\substack{\|\mathbf{B}\|_0 \le m \\ \mathbf{A}'\mathbf{A}=\mathbf{I}}}{\arg\min}\ \|\mathbf{X} - \mathbf{XBA}'\|_F^2 + \lambda\|\mathbf{B}\|_F^2} \tag{4.1}$$

ASPCA Objective

Algorithm 6 Annealed Sparse Principal Component Analysis

Input:

- $\mathbf{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- λ, L_2 penalty
- *epochs*, number of epochs
- m^{ej}, FSA annealing schedule specifying number of nonzero entries of $\mathbf{B}_{\bullet j}$ kept at epoch e

Output: $\mathbf{A}, \mathbf{B}$ solutions of Eq. (4.1)

```
1: Initialize $\mathbf{A}$ as the first $k$ loading vectors from PCA on $\mathbf{X}$
2: for $j = 1$ to $k$ do
3:     while $\mathbf{B}_{\bullet j}$ has not converged do
4:         Solve $\mathbf{B}_{\bullet j} = \arg\min_{\mathbf{b}} \|\mathbf{XA}_{\bullet j} - \mathbf{Xb}\|_2^2 + \lambda\|\mathbf{b}\|_2^2$ with desired sparsity by FSA
5:         $\mathbf{B}_{\bullet j} = \mathbf{B}_{\bullet j}/\|\mathbf{B}_{\bullet j}\|_2$
6:         $\mathbf{A}_{\bullet j} = (\mathbf{I} - \mathbf{A}_{j-1}\mathbf{A}_{j-1}')\mathbf{X}'\mathbf{XB}_{\bullet j}$          ▷ Let $\mathbf{A}_c$ denote $[\mathbf{A}_{\bullet 1}, \ldots, \mathbf{A}_{\bullet c}]$, $\mathbf{A}_0 = \mathbf{A}_{\bullet 1}$
7:         $\mathbf{A}_{\bullet j} = \mathbf{A}_{\bullet j}/\|\mathbf{A}_{\bullet j}\|_2$
8:     end while
9: end for
```

4.2 Globally Annealed Sparse Principal Component Analysis

Sjöstrand's sequential SPCA implementation, which forms the basis for Algorithm 6, enjoys some similarities to PCA. The loading matrix $\mathbf{B}$ is estimated one column at a time. Thus once $\mathbf{B}_{\bullet 1}$ is estimated, its values are fixed when estimation continues to $\mathbf{B}_{\bullet 2}$. An estimate $\hat{\mathbf{B}}_k$ from a rank k model fitted on $\mathbf{X}$ will have the same first k columns as higher-rank estimate $\hat{\mathbf{B}}_{k+i}$. PCA loadings have the same property: $\mathbf{B}_{\bullet 1}$ is the loading column producing the best rank 1 estimate of $\mathbf{X}$, and will be the same regardless of how many more columns of $\mathbf{B}$ are estimated.

The sequential approach that SPCA and ASPCA apply to estimating $\mathbf{B}$ carries some limitations. She (2017) notes that separate estimation of sparse loading vectors, while a common approach in sparse PCA models, does not necessarily achieve an optimal estimate when these individual estimates are composed into $\mathbf{B}$. Better models may result from jointly estimating all columns of $\mathbf{B}$, as is done in 2.6.

In Algorithm 1 for SPCA and Algorithm 6 for ASPCA, the desired sparsity for $\mathbf{B}$ is manually chosen for each of the k univariate regression problems solved. For data experiments, in order to estimate a rank-k loading matrix $\hat{\mathbf{B}}_{SPCA}$ or $\hat{\mathbf{B}}_{ASPCA}$ with q nonsparse entries, we specify the desired sparsity at each column as $\lfloor q/k \rfloor$. Prespecifying the sparsity structure of $\mathbf{B}$ in this way is somewhat arbitrary, but for the sequential approach we have little choice.

We test a more flexible approach to estimating a sparse $\mathbf{B}$ where the desired overall sparsity is applied globally to $\mathbf{B}$ rather than per column. Recall that in univariate regression, FSA performs feature selection by annealing parameter entries between the parameter's gradient updates. According to the annealing schedule, a number of entries with the lowest absolute magnitudes are dropped from the model. By estimating the loading matrix $\mathbf{B}$ in sparse PCA by a gradient method, we can apply FSA to perform "loading entry selection" on the entire matrix.

The general approach is to initialize $\mathbf{A}$ and $\mathbf{B}$ and update their estimates for a certain number of epochs. For certain epochs, the annealing schedule imposes sparsity on $\mathbf{B}$, so that it has fewer nonzero entries as estimation continues. Estimation progresses until annealing is complete and the model has converged.

Over training epochs, entries in $\mathbf{B}$ are dropped by being forced to 0. When this happens, subsequent gradient updates allow remaining entries of $\mathbf{B}$ to adjust to accommodate the change before the next annealing occurs.

Consider the ASPCA objective in Equation (4.1) with $\mathbf{A}'\mathbf{A} = \mathbf{I}$:

$$l(\mathbf{A}, \mathbf{B}) = \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}'\|_F^2 + \lambda \|\mathbf{B}\|_F^2 \tag{4.2}$$

Proposition 1. *The gradient of* $l(\mathbf{A}, \mathbf{B})$ *w.r.t.* $\mathbf{B}$ *is*

$$\nabla_{\mathbf{B}} l(\mathbf{A}, \mathbf{B}) = 2\mathbf{X}'\mathbf{X}(\mathbf{B} - \mathbf{A}) + 2\lambda \mathbf{B}.$$

Proof. Taking the derivative of (4.2),

$$\begin{aligned}
\frac{\partial l}{\partial \mathbf{B}} &= \frac{\partial}{\partial \mathbf{B}} Tr(\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}')'(\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}') + \lambda \mathbf{B}'\mathbf{B} \\
&= \frac{\partial}{\partial \mathbf{B}} Tr \mathbf{X}'\mathbf{X} - \mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}' - \mathbf{A}\mathbf{B}'\mathbf{X}'\mathbf{X} + \mathbf{A}\mathbf{B}'\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}' + \lambda \mathbf{B}'\mathbf{B} \\
&= \frac{\partial}{\partial \mathbf{B}} Tr(-2\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}') + \frac{\partial}{\partial \mathbf{B}} Tr(\mathbf{B}'\mathbf{X}'\mathbf{X}\mathbf{B}) + \frac{\partial}{\partial \mathbf{B}} Tr(\lambda \mathbf{B}'\mathbf{B})
\end{aligned}$$

$$= -2\mathbf{X}'\mathbf{X}\mathbf{A} + 2\mathbf{X}'\mathbf{X}\mathbf{B} + 2\lambda\mathbf{B}$$
$$= 2\mathbf{X}'\mathbf{X}(\mathbf{B} - \mathbf{A}) + 2\lambda\mathbf{B} \quad \square$$

Now fixing $\mathbf{B} = \hat{\mathbf{B}}$ we return to the loss to optimize over $\mathbf{A}$.

Proposition 2. *Objective (4.2) is minimized w.r.t.* $\mathbf{A}$ *by* $\hat{\mathbf{A}} = \mathbf{U}\mathbf{V}'$ *where* $\mathbf{U}\mathbf{D}\mathbf{V}'$ *is the singular value decomposition of* $\mathbf{X}'\mathbf{X}\mathbf{B}$.

Proof.

$$\begin{aligned}
\hat{\mathbf{A}} &= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\ Tr(\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}')'(\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}') + \lambda\mathbf{B}'\mathbf{B} \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\ Tr(-2\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}') + Tr(\mathbf{A}\mathbf{B}'\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}') + Tr(\lambda\mathbf{B}'\mathbf{B}) \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\ -2Tr(\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}') + Tr(\mathbf{B}'\mathbf{X}'\mathbf{X}\mathbf{B}) + \lambda Tr(\mathbf{B}'\mathbf{B}) \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\ Tr(\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}') \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\ \|\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{A}'\|_F^2
\end{aligned}$$

The solution to the final optimization is known to be $\hat{\mathbf{A}} = \mathbf{U}\mathbf{V}'$ where $\mathbf{X}'\mathbf{X}\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}'$ by SVD (Zou et al., 2006). $\square$

These updating methods for $\mathbf{A}$ and $\mathbf{B}$ are used in conjunction with FSA to produce *Globally Annealed Sparse Principal Component Analysis (GSPCA)*, presented in Algorithm 7.

For a toy example, consider a data matrix $\mathbf{U}$ of n samples from $\mathcal{N}(\vec{\mathbf{0}}, \mathbf{\Sigma})$ with

$$\mathbf{\Sigma} = \begin{bmatrix}
1 & 0.95 & 0.95 & 0.95 & 0.95 & 0.95 & 0 & 0 \\
0.95 & 1 & 0.95 & 0.95 & 0.95 & 0.95 & 0 & 0 \\
0.95 & 0.95 & 1 & 0.95 & 0.95 & 0.95 & 0 & 0 \\
0.95 & 0.95 & 0.95 & 1 & 0.95 & 0.95 & 0 & 0 \\
0.95 & 0.95 & 0.95 & 0.95 & 1 & 0.95 & 0 & 0 \\
0.95 & 0.95 & 0.95 & 0.95 & 0.95 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.95 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.95 & 1
\end{bmatrix}$$

We should be able to represent $\mathbf{U}$ with two principal components, one drawing from the first six columns of $\mathbf{U}$, the second from the final two columns. This would yield an ideal 8-by-2 loading

Algorithm 7 Globally Annealed Sparse Principal Component Analysis (GSPCA)

Input:

- $\mathbf{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- η, learning rate
- *epochs*, number of epochs
- λ, L_2 penalty
- m^e, FSA annealing schedule specifying the number of nonzero entries of $\mathbf{B}$ kept at epoch e

Output: Estimated model parameters $\mathbf{A}$, $\mathbf{B}$

```
1: Initialize both $\mathbf{A}$ and $\mathbf{B}$ as the first $k$ loading vectors from PCA on $\mathbf{X}$
2: for $e = 1$ to $epochs$ do
3:     $\mathbf{A} = \mathbf{U}\mathbf{V}'$ with $\mathbf{X}'\mathbf{X}\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}'$ by SVD
4:     $\nabla_{\mathbf{B}} = 2\mathbf{X}'\mathbf{X}(\mathbf{B} - \mathbf{A}) + 2\lambda\mathbf{B}$
5:     $\mathbf{B} = \mathbf{B} - \eta\nabla_{\mathbf{B}}/n$
6:     Apply FSA to $\mathbf{B}$
7:         $m^e$ is the desired number of nonzero elements of $\mathbf{B}$ for current epoch
8:         Retain the top $m^e$ entries of $\mathbf{B}$ with the highest absolute values, setting the
               remaining entries to 0
9:         Normalize columns of $\mathbf{B}$ to have length 1
10: end for
```

matrix $\mathbf{B}$ with 6 nonzero entries in the first column and 2 nonzero entries in the second column. However, recall that SPCA and ASPCA require prespecification of sparsity per column of $\mathbf{B}$. Without manually testing different possibilities, we implement these models on data by applying sparsity evenly across principal components. In this case, $\hat{\mathbf{B}}_{SPCA}$ would then have 4 nonzero entries in each column. GSPCA on the other hand automatically selects which entries to anneal freely from the entire matrix, so that $\hat{\mathbf{B}}_{GSPCA}$ can have varying amounts of sparsity per column without manual adjustments. PCA, of course, does not impose any sparsity.

Sample loading matrices on $\mathbf{U}$ are shown in Table 4.1.

Table 4.1: Estimated Loading Matrices for Samples of $\mathbf{U}$

Sample $\hat{\mathbf{B}}_{PCA}$	Sample $\hat{\mathbf{B}}_{SPCA}$	Sample $\hat{\mathbf{B}}_{GSPCA}$
$\begin{bmatrix} 0.4072 & -0.0153 \\ 0.4117 & -0.0090 \\ 0.4058 & -0.0065 \\ 0.4089 & -0.0155 \\ 0.4075 & -0.0152 \\ 0.4070 & -0.0201 \\ 0.0219 & 0.7054 \\ 0.0251 & 0.7079 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.4876 & 0 \\ 0.3844 & -0.0004 \\ 0.3471 & 0 \\ 0.7029 & -0.0115 \\ 0 & 0.7105 \\ 0 & 0.7036 \end{bmatrix}$	$\begin{bmatrix} 0.4074 & 0 \\ 0.4081 & 0 \\ 0.4085 & 0 \\ 0.4084 & 0 \\ 0.4084 & 0 \\ 0.4086 & 0 \\ 0 & 0.7072 \\ 0 & 0.7070 \end{bmatrix}$

The observed $\hat{\mathbf{B}}_{SPCA}$ is able to identify structure in the sampled $\mathbf{U}$, but does not identify an ideal sparse representation due to the limitation of having to prespecify sparsity in each loading column. While estimating principal components sequentially is similar to PCA, it reduces the method's flexibility. $\hat{\mathbf{B}}_{GSPCA}$ on the other hand is able to apply differing levels of sparsity to each principal component when given a total desired sparsity of 8 nonzero entries for the entire matrix.

4.3 Handling Missing Data by Imputation

Knowledge of a data matrix's structure can inform an imputation strategy for partially-observed data. We present an imputation approach based on sparse PCA model estimation and integrate the approach with the models presented.

Sparse PCA models estimate $\mathbf{X}$ as the low-rank $\mathbf{XBA'}$.[1] Fixing current estimates of $\mathbf{A}$ and $\mathbf{B}$, consider row $\mathbf{x}$ of $\mathbf{X}$ having missing values.

$$\mathbf{x} = (x_1 \quad x_2 \quad \bullet \quad x_4 \quad \bullet \quad x_6)$$

Let $\mathbf{w} = (w_1 \quad w_2)$ represent the unobserved missing entries $(x_3 \quad x_5)$ in $\mathbf{x}$. Take $\mathbf{x}^0$ as 0-imputed $\mathbf{x}$ and define function $f(\mathbf{w})$ that places the values of $\mathbf{w}$ into corresponding positions of a data row and places 0's elsewhere. For the shown example, $f(\mathbf{w}) = (0 \quad 0 \quad w_1 \quad 0 \quad w_2 \quad 0)$.

We can now consider fully-observed representation $\tilde{\mathbf{x}} = \mathbf{x}^0 + f(\mathbf{w}) = (x_1 \quad x_2 \quad w_1 \quad x_4 \quad w_2 \quad x_6)$ which we use to simultaneously estimate model parameters ($\mathbf{A}$, $\mathbf{B}$) and imputed values $\mathbf{w}$.

The low-rank reconstruction of $\tilde{\mathbf{x}}$ is then $(\mathbf{x}^0 + f(\mathbf{w}))\mathbf{BA'}$ and its error can be calculated as

$$l(\mathbf{A}, \mathbf{B}) = \|\mathbf{x}^0 + f(\mathbf{w}) - (\mathbf{x}^0 + f(\mathbf{w}))\mathbf{BA'}\|_2^2 \tag{4.3}$$

We estimate $\mathbf{w}$ to minimize the loss given $\mathbf{A}$ and $\mathbf{B}$. This imputation step is placed alongside estimation steps for $\mathbf{A}$ and $\mathbf{B}$, which rely on fixing the current imputation estimates.

To minimize l, recognize that $f(\mathbf{w}) = \mathbf{wP}$ for appropriate placement matrix $\mathbf{P}$. For instance,

$$\begin{pmatrix} w_1 & w_2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & w_1 & 0 & w_2 & 0 \end{pmatrix}$$

In the general case, if $\mathbf{x}$ has p entries with m missing, $\mathbf{P}$ is m-by-p with

$$\mathbf{P}_{ij} = \begin{cases} 1 & \mathbf{w}_i \text{ is the missing value corresponding to } \mathbf{x}_j \\ 0 & \text{Otherwise} \end{cases}$$

Proposition 3. *The minimum of the loss* (4.3) *w.r.t.* $\mathbf{w}$ *can be obtained analytically as:*

$$\mathbf{w} = -\mathbf{x}^0\mathbf{D'P'}(\mathbf{PDD'P'})^{-1}.$$

Proof. Revisiting the row loss,

$$\begin{aligned} l(\mathbf{A}, \mathbf{B}) &= \|\mathbf{x}^0 + f(\mathbf{w}) - (\mathbf{x}^0 + f(\mathbf{w}))\mathbf{BA'}\|_2^2 \\ &= \|\mathbf{x}^0 + \mathbf{wP} - (\mathbf{x}^0 + \mathbf{wP})\mathbf{BA'}\|_2^2 \\ &= \|\mathbf{x}^0 - \mathbf{x}^0\mathbf{BA'} + \mathbf{wP} - \mathbf{wPBA'}\|_2^2 \end{aligned}$$

[1] JSPCA (2.5) reduces the dimension of data $\mathbf{X}$ in a manner compatible with this imputation approach, and we incorporate imputation into experiments with JSPCA on data with missingness. The authors' estimation algorithm (Yi et al., 2017) acts on $\mathbf{X}'$, so a slight reparametrization is applied to retain consistency within our discussion.

$$
\begin{aligned}
&= \|\mathbf{x}^0(\mathbf{I} - \mathbf{BA}') + \mathbf{wP}(\mathbf{I} - \mathbf{BA}')\|_2^2 \\
&= \|\mathbf{C} + \mathbf{wPD}\|_2^2 \text{ for } \mathbf{D} = (\mathbf{I} - \mathbf{BA}') \text{ and } \mathbf{C} = \mathbf{x}^0\mathbf{D} \\
&= (\mathbf{C} + \mathbf{wPD})(\mathbf{C} + \mathbf{wPD})' \\
&= \mathbf{CC}' + 2\mathbf{wPDC}' + \mathbf{wPDD}'\mathbf{P}'\mathbf{w}'
\end{aligned}
$$

Minimizing the loss for $\mathbf{w}$,

$$\frac{\partial l}{\partial \mathbf{w}} = 2\mathbf{CDP}' + 2\mathbf{wPDD}'\mathbf{P}' = 0 \implies \mathbf{w} = -\mathbf{x}^0\mathbf{D}'\mathbf{P}'(\mathbf{PDD}'\mathbf{P}')^{-1}. \quad \square$$

Remark 5. *High missingness can produce singularity or near-singularity in* $\mathbf{PDD}'\mathbf{P}'$*, which needs to be inverted. The inversion problem can be stabilized by adding a ridge parameter (Foucart, 1999; Jensen and Ramirez, 2012; Hoerl and Kennard, 1970).*

Thus for a single row $\mathbf{x}$, corresponding placement matrix $\mathbf{P}$, fixed estimates $\mathbf{A}$ and $\mathbf{B}$, ridge parameter λ_r, and $\mathbf{D}$ defined as above, $\hat{\mathbf{w}} = -\mathbf{x}^0\mathbf{D}'\mathbf{P}'(\mathbf{PDD}'\mathbf{P}' + \lambda_r\mathbf{I})^{-1}$. All methods introduced thus far use a low-rank estimate of $\mathbf{X}$ taking form $\mathbf{XM}_1\mathbf{M}_2$ where matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ are estimated to have desirable properties such as semi-orthogonality or sparsity. Thus the imputation method presented here can be applied to each model. The general imputation approach is presented in Algorithm 8. Extensions of SPCA, ASPCA, and GSPCA to incorporate imputation for missingness are shown in Algorithms 9, 10, and 11.

Algorithm 8 Imputation by Low-Rank Completion

```
Input:
  • $\mathbf{X}$, $n$-by-$p$ data matrix
  • $\mathbf{A}$, $\mathbf{B}$ such that $\mathbf{X} \approx \mathbf{XBA}'$
Output: Fully-imputed data $\mathbf{X}$
1: for each row $\mathbf{x}$ of $\mathbf{X}$ containing missing values do
2:     Generate $m$-by-$p$ placement matrix $\mathbf{P}$ as
           $\mathbf{P}_{ij} = \begin{cases} 1 & x_i \text{ is the } j^{\text{th}} \text{ missing value of } \mathbf{x} \\ 0 & \text{Otherwise} \end{cases}$
       where $m$ is the number of missing entries in $\mathbf{x}$
3:     $\mathbf{w} = -\mathbf{x}^0\mathbf{DD}'\mathbf{P}'(\mathbf{PDD}'\mathbf{P}' + \lambda_r\mathbf{I})^{-1}$ with $\mathbf{D} = \mathbf{I} - \mathbf{BA}'$ and $\lambda_r$ an optional
         imputation ridge parameter
4:     Replace corresponding row in $\mathbf{X}$ with $\mathbf{x}^0 + \mathbf{wP}$
5: end for
```

Algorithm 9 Sparse Principal Component Analysis with Imputation

Input:

- $\mathcal{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- λ_1^j, Desired number of nonzero entries for loading matrix column j if soft thresholding; desired L_1 penalties if using elastic net
- λ_2, L_2 penalty

Output: Estimated model parameters $\mathbf{A}$, $\mathbf{B}$ and low-rank estimate $\hat{\mathbf{X}}$ of $\mathcal{X}$

```
1:  Set $\mathbf{X} = \mathcal{X}^0$                                   ▷ Let $\mathbf{M}^0$ denote 0-imputed $\mathbf{M}$
2:  Initialize $\mathbf{A}$ as the first $k$ loading vectors from PCA on $\mathbf{X}$
3:  for $j = 1$ to $k$ do
4:      while $\mathbf{B}_{\bullet j}$ has not converged do
5:          if Soft Thresholding then
6:              $\mathbf{B}_{\bullet j} = \text{Sign}(\mathbf{A}'_{\bullet j}\mathbf{X}'\mathbf{X})(|\mathbf{A}_{\bullet j}{}'\mathbf{X}'\mathbf{X}| - \lambda_1^j/2)_+$
7:          else if Elastic Net then
8:              Solve $\mathbf{B}_{\bullet j} = \arg\min_{\mathbf{b}} \|\mathbf{X}\mathbf{A}_{\bullet j} - \mathbf{X}\mathbf{b}\|_2^2 + \lambda_1^j\|\mathbf{b}\|_1 + \lambda_2\|\mathbf{b}\|_2^2$ with desired sparsity
                  by elastic net
9:          end if
10:         $\mathbf{B}_{\bullet j} = \mathbf{B}_{\bullet j}/\|\mathbf{B}_{\bullet j}\|_2$
11:         $\mathbf{A}_{\bullet j} = (\mathbf{I} - \mathbf{A}_{j-1}\mathbf{A}_{j-1}{}')\mathbf{X}'\mathbf{X}\mathbf{B}_{\bullet j}$      ▷ Let $\mathbf{A}_c$ denote the first $c$ columns of $\mathbf{A}$, $\mathbf{A}_0 = \mathbf{A}_{\bullet 1}$
12:         $\mathbf{A}_{\bullet j} = \mathbf{A}_{\bullet j}/\|\mathbf{A}_{\bullet j}\|_2$
13:         Update imputation with new ($\mathbf{A}$, $\mathbf{B}$) as shown in Algorithm 8
14:     end while
15: end for
16: Set $\hat{\mathbf{X}} = \mathbf{X}\mathbf{B}\mathbf{A}'$
```

Algorithm 10 Annealed Sparse Principal Component Analysis with Imputation

Input:

- $\mathcal{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- λ, L_2 penalty
- m^{ej}, FSA annealing schedule specifying number of nonzero entries of $\mathbf{B}_{\bullet j}$ kept at epoch e

Output: Estimated model parameters $\mathbf{A}$, $\mathbf{B}$ and low-rank estimate $\hat{\mathbf{X}}$ of $\mathcal{X}$

```
1:  Set $\mathbf{X} = \mathcal{X}^0$                                   ▷ Let $\mathbf{M}^0$ denote 0-imputed $\mathbf{M}$
2:  Initialize $\mathbf{A}$ as the first $k$ loading vectors from PCA on $\mathbf{X}$
3:  for $j = 1$ to $k$ do
4:      while $\mathbf{B}_{\bullet j}$ has not converged do
5:          Solve $\mathbf{B}_{\bullet j} = \arg\min_{\mathbf{b}} \|\mathbf{X}\mathbf{A}_{\bullet j} - \mathbf{X}\mathbf{b}\|_2^2 + \lambda\|\mathbf{b}\|_2^2$ with desired sparsity by FSA
6:          $\mathbf{B}_{\bullet j} = \mathbf{B}_{\bullet j}/\|\mathbf{B}_{\bullet j}\|_2$
7:          $\mathbf{A}_{\bullet j} = (\mathbf{I} - \mathbf{A}_{j-1}\mathbf{A}_{j-1}{}')\mathbf{X}'\mathbf{X}\mathbf{B}_{\bullet j}$      ▷ Let $\mathbf{A}_c$ denote the first $c$ columns of $\mathbf{A}$, $\mathbf{A}_0 = \mathbf{A}_{\bullet 1}$
8:          $\mathbf{A}_{\bullet j} = \mathbf{A}_{\bullet j}/\|\mathbf{A}_{\bullet j}\|_2$
9:          Update imputation with new ($\mathbf{A}$, $\mathbf{B}$) as shown in Algorithm 8
10:     end while
11: end for
12: Set $\hat{\mathbf{X}} = \mathbf{X}\mathbf{B}\mathbf{A}'$
```

Algorithm 11 Globally Annealed Sparse Principal Component Analysis with Imputation

Input:

- $\mathcal{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- η, learning rate
- *epochs*, number of epochs
- λ, L_2 penalty
- m^e, FSA annealing schedule specifying the number of nonzero entries of $\mathbf{B}$ kept at epoch e

Output: Estimated model parameters $\mathbf{A}$, $\mathbf{B}$ and low-rank estimate $\hat{\mathbf{X}}$ of $\mathcal{X}$

```
1: Set $\mathbf{X} = \mathcal{X}^0$    ▷ Let $\mathbf{M}^0$ denote 0-imputed $\mathbf{M}$
2: Initialize both $\mathbf{A}$ and $\mathbf{B}$ as the first $k$ loading vectors from PCA on $\mathbf{X}$
3: for $e = 1$ to epochs do
4:     $\mathbf{A} = \mathbf{UV}'$ with $\mathbf{X}'\mathbf{XB} = \mathbf{UDV}'$ by SVD
5:     $\nabla_{\mathbf{B}} = 2\mathbf{X}'\mathbf{X}(\mathbf{B} - \mathbf{A}) + 2\lambda\mathbf{B}$
6:     $\mathbf{B} = \mathbf{B} - \eta\nabla_{\mathbf{B}}/n$
7:     Apply FSA to $\mathbf{B}$
8:         $m^e$ is the desired number of nonzero elements of $\mathbf{B}$ for current epoch
9:         Retain the top $m^e$ entries of $\mathbf{B}$ with the highest absolute values, setting the
             remaining entries to 0
10:        Normalize columns of $\mathbf{B}$ to have length 1
11:    Update imputation with new ($\mathbf{A}$, $\mathbf{B}$) as shown in Algorithm 8
12: end for
13: Set $\hat{\mathbf{X}} = \mathbf{XBA}'$
```

4.4 Feature-Weighted PCA

Here we examine feature weights for PCA by incorporating a parameter to explicitly govern the influence of features in the model. While not achieving genuine sparsity, the goal of feature weights is to reduce the influence of less useful features.

Recall PCA objective (2.2) finds orthogonal $\mathbf{B}$ to minimize low-rank estimate $\mathbf{XBB}'$ of $\mathbf{X}$ with $\hat{\mathbf{B}}_{PCA} = \underset{\mathbf{B}'\mathbf{B}=\mathbf{I}}{\arg\min} \|\mathbf{X} - \mathbf{XBB}'\|_F^2$.

Consider diagonal matrix $\mathbf{W}$ with positive diagonal entries. The *Weighted PCA (WPCA)* model estimates a low-rank representation of matrix $\mathbf{X}$ where the effect of feature $\mathbf{X}_{\bullet i}$ is weighted by $\mathbf{W}_{ii}$.

$$\boxed{\mathbf{B}_{WPCA} = \underset{\mathbf{B'WB=I}}{\arg\min} \|(\mathbf{X} - \mathbf{XBB'})\sqrt{\mathbf{W}}\|_F^2} \tag{4.4}$$

Weighted PCA Objective

The objective can be motivated by an equivalent formulation of loss where $\hat{\mathbf{X}}$ is low-rank estimate $\mathbf{XBB'}$ of $\mathbf{X}$:

$$\sum_{i,j} (\mathbf{X}_{ij} - \hat{\mathbf{X}}_{ij})^2 \mathbf{W}_{jj}$$

This expression shows how diagonal entries in $\mathbf{W}$ weight the effect of $\mathbf{X}$ in a columnwise fashion. Estimating PCA with feature weights is a special case of the "Generalized SVD" model of (Greenacre, 1984, Appendix A.1), in which the simultaneous application of row and feature weights is shown to be reducible to an ordinary SVD problem.

To accommodate the incorporation of feature weights into the model, we first approach optimizing for $\mathbf{B}$.

Proposition 4. *Given n-by-p matrix* $\mathbf{X}$ *and p-by-p positive diagonal matrix* $\mathbf{W}$*, let*

$$\mathbf{M} = 2\mathbf{W}^{1/2}\mathbf{X'XW}^{-1/2} - \mathbf{W}^{-1/2}\mathbf{X'XW}^{-1/2}$$

and let $\tilde{\mathbf{A}}$ *be a p-by-k matrix satisfying*

$$\tilde{\mathbf{A}} = \underset{\mathbf{A'A=I}}{\arg\max}\, Tr\mathbf{A'MA}$$

Then the WPCA Objective (4.4) *is satisfied by* $\tilde{\mathbf{B}} = \mathbf{W}^{-1/2}\mathbf{A}$.

Proof. The objective is first modified to a more tractable form.

$$\begin{aligned}
\tilde{\mathbf{B}} &= \underset{\mathbf{B'WB=I}}{\arg\min} \sum_{i,j} (\mathbf{X}_{ij} - \hat{\mathbf{X}}_{ij})^2 \mathbf{W}_{jj} \\
&= \underset{\mathbf{B'WB=I}}{\arg\min} \|(\mathbf{X} - \mathbf{XBB'})\sqrt{\mathbf{W}}\|_F^2 \\
&= \underset{\mathbf{B'WB=I}}{\arg\min}\, Tr((\mathbf{X} - \mathbf{XBB'})\sqrt{\mathbf{W}})'(\mathbf{X} - \mathbf{XBB'})\sqrt{\mathbf{W}} \\
&= \underset{\mathbf{B'WB=I}}{\arg\min}\, Tr\sqrt{\mathbf{W}}'(\mathbf{X} - \mathbf{XBB'})'(\mathbf{X} - \mathbf{XBB'})\sqrt{\mathbf{W}} \\
&= \underset{\mathbf{B'WB=I}}{\arg\min}\, Tr\mathbf{W}(\mathbf{X'X} - 2\mathbf{X'XBB'} + \mathbf{BB'X'XBB'}) \\
&= \underset{\mathbf{B'WB=I}}{\arg\min}\, Tr - 2\mathbf{WX'XBB'} + \mathbf{WBB'X'XBB'}
\end{aligned}$$

Let $\tilde{\mathbf{A}} = \sqrt{\mathbf{W}}\tilde{\mathbf{B}}$. We then have $\tilde{\mathbf{B}} = \mathbf{W}^{-1/2}\tilde{\mathbf{A}}$. In particular, $\tilde{\mathbf{B}}'\mathbf{W}\tilde{\mathbf{B}} = \tilde{\mathbf{A}}'\mathbf{W}^{-1/2}\mathbf{W}\mathbf{W}^{-1/2}\tilde{\mathbf{A}} = \tilde{\mathbf{A}}'\tilde{\mathbf{A}}$ establishes the equivalency $\tilde{\mathbf{B}}'\mathbf{W}\tilde{\mathbf{B}} = \mathbf{I} \iff \tilde{\mathbf{A}}'\tilde{\mathbf{A}} = \mathbf{I}$ which allows us to continue the derivation in terms of $\tilde{\mathbf{A}}$.

$$\begin{aligned}
\tilde{\mathbf{B}} &= \underset{\mathbf{B}'\mathbf{W}\mathbf{B}=\mathbf{I}}{\arg\min}\, Tr - 2\mathbf{W}\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{B}' + \mathbf{W}\mathbf{B}\mathbf{B}'\mathbf{X}'\mathbf{X}\mathbf{B}\mathbf{B}' \\
&= \mathbf{W}^{-1/2}\tilde{\mathbf{A}}, \text{ where } \tilde{\mathbf{A}} \text{ satisfies}
\end{aligned}$$

$$\begin{aligned}
\tilde{\mathbf{A}} &= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\, Tr - 2\mathbf{W}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2}\mathbf{A}\mathbf{A}'\mathbf{W}^{-1/2} + \mathbf{W}\mathbf{W}^{-1/2}\mathbf{A}\mathbf{A}'\mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2}\mathbf{A}\mathbf{A}'\mathbf{W}^{-1/2} \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\, Tr - 2\mathbf{W}^{1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2}\mathbf{A}\mathbf{A}' + \mathbf{A}\mathbf{A}'\mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2}\mathbf{A}\mathbf{A}' \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\, Tr\mathbf{A}'(-2\mathbf{W}^{1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2})\mathbf{A} + \mathbf{A}'(\mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2})\mathbf{A} \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\min}\, Tr\mathbf{A}'(-2\mathbf{W}^{1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2} + \mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2})\mathbf{A} \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\, Tr\mathbf{A}'(2\mathbf{W}^{1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2} - \mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2})\mathbf{A}
\end{aligned}$$

Taking $\mathbf{M} = 2\mathbf{W}^{1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2} - \mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2}$ completes the proof. □

Proposition 5. *Defining $\tilde{\mathbf{A}}$, $\mathbf{M}$ as in Proposition 4, the columns of $\tilde{\mathbf{A}}$ are the orthonormal eigenvectors of $\mathbf{M} + \mathbf{M}'$ corresponding to the largest eigenvalues.*

Proof.

$$\begin{aligned}
\tilde{\mathbf{A}} &= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\, Tr\mathbf{A}'\mathbf{M}\mathbf{A} \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\, \frac{1}{2}Tr(\mathbf{A}'\mathbf{M}\mathbf{A}) + \frac{1}{2}Tr(\mathbf{A}'\mathbf{M}'\mathbf{A}) \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\, Tr\mathbf{A}'\mathbf{M}\mathbf{A} + \mathbf{A}'\mathbf{M}'\mathbf{A} \\
&= \underset{\mathbf{A}'\mathbf{A}=\mathbf{I}}{\arg\max}\, Tr\mathbf{A}'(\mathbf{M} + \mathbf{M}')\mathbf{A}
\end{aligned}$$

With orthogonal $\mathbf{A}$ and symmetric $\mathbf{M} + \mathbf{M}'$, the optimization is solved by the following result from (Harville, 1997, Thm. 21.12.5). □

Proposition 6. *Given p-by-p symmetric $\mathbf{S}$ with eigenvalues $d_1 \geq d_2 \geq \ldots \geq d_p$ and p-by-k $\mathbf{Q}$ satisfying $\mathbf{Q}'\mathbf{Q} = \mathbf{I}$ (where $k \leq p$),*

$$tr\mathbf{Q}'\mathbf{S}\mathbf{Q} \leq \sum_{i=1}^{k} d_i$$

and equality is achieved when the columns of $\mathbf{Q}$ are orthonormal eigenvectors of $\mathbf{S}$ corresponding to d_1, d_2, $\ldots$, d_k.

Proposition 6 provides the necessary information to estimate $\tilde{\mathbf{A}}$ given $\mathbf{M}$. Estimation is completed by taking $\mathbf{B}_{WPCA} = \mathbf{W}^{-1/2}\tilde{\mathbf{A}}$.

Algorithm 12 Weighted PCA

```
Input:
  • $\mathbf{X}$, $n$-by-$p$ data matrix
  • $k$, model rank
  • $epochs$, number of epochs
Output:
  • Loading matrix $\mathbf{B}$, solution of Equation (4.4)
  • $\mathbf{W}$, diagonal matrix of feature weights
1: Obtain $\hat{\mathbf{X}} = \mathbf{X}\mathbf{B}_{PCA}\mathbf{B}'_{PCA}$ where $\mathbf{B}_{PCA}$ is obtained from rank-$k$ PCA on $\mathbf{X}$
2: Set $\mathbf{w}$ as variances of columns of $\hat{\mathbf{X}} - \mathbf{X}$
3: Create diagonal matrix $\mathbf{W}$ with $\mathbf{W}_{jj} = \max(\mathbf{w}_j, .1)^{-1}$
4: Set $\mathbf{M} = 2\mathbf{W}^{1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2} - \mathbf{W}^{-1/2}\mathbf{X}'\mathbf{X}\mathbf{W}^{-1/2}$
5: Set the columns of $\mathbf{A}$ as eigenvectors of $\mathbf{M} + \mathbf{M}'$ corresponding to the $k$ largest eigenvalues
6: Obtain estimate $\mathbf{B} = \mathbf{W}^{-1/2}\mathbf{A}$
```

In simulations and experiments, WPCA is implemented by calculating feature weights as in Algorithm 12 and applying them in Matlab's implementation of feature-weighted PCA (The MathWorks, Inc., 2023).

4.5 Sparse Estimation of Latent Factors

SPCA, ASPCA, and GSPCA all follow the self-regression formulation to produce a low-rank estimate of $\mathbf{X}$ as $\mathbf{X} \approx \mathbf{XBA}'$ with sparse $\mathbf{B}$ reducing $\mathbf{X}$ and semi-orthogonal $\mathbf{A}$ remapping it to the original dimension.

To introduce our final approach, recall the alternative to self-regression proposed by SRRR-SPCA (2.7), where $\mathbf{X}$ is instead modeled as $\mathbf{X} \approx \mathbf{VS}'$ with semi-orthogonal $\mathbf{V}$ and sparsity constraints imposed on $\mathbf{S}$. Model estimation directly estimates low-dimensional representation $\mathbf{V}$.

We pursue the same idea, estimating reduced representation $\mathbf{\Gamma}$ of $\mathbf{X}$, and a matrix $\mathbf{A}$, such that $\mathbf{X} \approx \mathbf{\Gamma A}'$. However we make a slight deviation from SRRR-SPCA and consider the case of maintaining sparsity *and* semi-orthogonality in the same parameter $\mathbf{A}$. This yields our objective for *Sparse Estimation of Latent Factors (SELF)*:

$$\boxed{(\hat{\mathbf{A}}_{SELF}, \hat{\mathbf{\Gamma}}_{SELF}) = \underset{\substack{\|\mathbf{A}\|_0 \le m \\ \mathbf{A}'\mathbf{A}=\mathbf{I}}}{\arg\min} \|\mathbf{X} - \mathbf{\Gamma A}'\|_F^2} \tag{4.5}$$

SELF Objective

A more general form of the SELF Objective incorporates feature weights in a similar style as Weighted PCA in 4.4. Let $\mathbf{W}$ be a positive diagonal matrix where $\mathbf{W}_{ii}$ is the feature weight we wish to apply to feature $\mathbf{X}_{\bullet i}$.

$$\boxed{(\hat{\mathbf{A}}_{SELF-W}, \hat{\mathbf{\Gamma}}_{SELF-W}) = \underset{\substack{\|\mathbf{A}\|_0 \le m \\ \mathbf{A}'\mathbf{W}\mathbf{A}=\mathbf{I}}}{\arg\min} \|(\mathbf{X} - \mathbf{\Gamma A}')\sqrt{\mathbf{W}}\|_F^2} \tag{4.6}$$

Weighted SELF Objective

In practice, sparse PCA models do not pursue the goal of estimating $\mathbf{A}$ to satisfy both $\mathbf{A}'\mathbf{A} = \mathbf{I}$ and $\|\mathbf{A}\|_0 \le m$. While nonsparse PCA produces estimate $\mathbf{X} \approx \mathbf{XAA}'$ with $\mathbf{A}'\mathbf{A} = \mathbf{I}$, SPCA achieves sparsity by separating the desired parameter properties across semi-orthogonal $\mathbf{A}$ and sparse (but not semi-orthogonal) $\mathbf{B}$ to yield $\mathbf{X} \approx \mathbf{XBA}'$. We, too, must accept this relaxation on orthogonality when imposing sparsity on $\mathbf{A}$. We begin by deriving some helpful results in calculating parameter estimates for SELF, but in the case of estimating $\mathbf{A}$ we consider only the nonsparse case. In the algorithm to estimate the SELF model presented further, $\mathbf{A}$ is estimated to be semi-orthogonal but deviates from that property as sparsity is imposed upon it by FSA.

Proposition 7. *The Weighted SELF Objective (4.6) is minimized w.r.t* $\mathbf{\Gamma}$ *by* $\hat{\mathbf{\Gamma}} = \mathbf{XWA}(\mathbf{A}'\mathbf{WA})^{-1}$.

Proof. Let $l = \|(\mathbf{X} - \mathbf{\Gamma A}')\sqrt{\mathbf{W}}\|_F^2$. Then

$$\begin{aligned}
\arg\min_{\mathbf{\Gamma}} l &= \arg\min_{\mathbf{\Gamma}} \|\mathbf{X}\sqrt{\mathbf{W}} - \mathbf{\Gamma A}'\sqrt{\mathbf{W}}\|_F^2 \\
&= \arg\min_{\mathbf{\Gamma}} Tr(\mathbf{X}\sqrt{\mathbf{W}} - \mathbf{\Gamma A}'\sqrt{\mathbf{W}})'(\mathbf{X}\sqrt{\mathbf{W}} - \mathbf{\Gamma A}'\sqrt{\mathbf{W}}) \\
&= \arg\min_{\mathbf{\Gamma}} Tr\sqrt{\mathbf{W}}\mathbf{X}'\mathbf{X}\sqrt{\mathbf{W}} - 2\sqrt{\mathbf{W}}\mathbf{X}'\mathbf{\Gamma A}'\sqrt{\mathbf{W}} + \sqrt{\mathbf{W}}\mathbf{A\Gamma}'\mathbf{\Gamma A}'\sqrt{\mathbf{W}} \\
&= \arg\min_{\mathbf{\Gamma}} Tr\mathbf{WX}'\mathbf{X} - 2\mathbf{WX}'\mathbf{\Gamma A}' + \mathbf{WA\Gamma}'\mathbf{\Gamma A}' \\
&= \arg\min_{\mathbf{\Gamma}} Tr - 2\mathbf{WX}'\mathbf{\Gamma A}' + \mathbf{WA\Gamma}'\mathbf{\Gamma A}' \\
&= \arg\min_{\mathbf{\Gamma}} Tr - 2\mathbf{A}'\mathbf{WX}'\mathbf{\Gamma} + \mathbf{\Gamma A}'\mathbf{WA\Gamma}'
\end{aligned}$$

$$\begin{aligned}
&\frac{\partial l}{\partial \mathbf{\Gamma}} = \frac{\partial}{\partial \mathbf{\Gamma}} Tr - 2\mathbf{A}'\mathbf{WX}'\mathbf{\Gamma} + \mathbf{\Gamma A}'\mathbf{WA\Gamma}' = -2\mathbf{XWA} + 2\mathbf{\Gamma A}'\mathbf{WA} \\
&\frac{\partial l}{\partial \mathbf{\Gamma}} = 0 \implies \mathbf{\Gamma} = \mathbf{XWA}(\mathbf{A}'\mathbf{WA})^{-1}. \quad \square
\end{aligned}$$

Proposition 8. *The SELF Objective (4.5) is minimized w.r.t* $\mathbf{\Gamma}$ *by* $\hat{\mathbf{\Gamma}} = \mathbf{XA}(\mathbf{A}'\mathbf{A})^{-1}$.

Proof. This follows from Proposition 7 by taking $\mathbf{W} = \mathbf{I}$. $\square$

Proposition 9. *The SELF Objective (4.5) without sparsity is minimized w.r.t* $\mathbf{A}$ *by* $\hat{\mathbf{A}} = \mathbf{UV}'$ *where* $\mathbf{UDV}'$ *is the singular value decomposition of* $\mathbf{X}'\mathbf{\Gamma}$.

Proof.

$$\begin{aligned}
\hat{\mathbf{A}} &= \arg\min_{\mathbf{A}'\mathbf{A}=\mathbf{I}} \|\mathbf{X} - \mathbf{\Gamma A}'\|_F^2 \\
&= \arg\min_{\mathbf{A}'\mathbf{A}=\mathbf{I}} Tr(\mathbf{X} - \mathbf{\Gamma A}')'(\mathbf{X} - \mathbf{\Gamma A}') \\
&= \arg\min_{\mathbf{A}'\mathbf{A}=\mathbf{I}} Tr(-2\mathbf{A\Gamma}'\mathbf{X}) + Tr(\mathbf{A\Gamma}'\mathbf{\Gamma A}') \\
&= \arg\max_{\mathbf{A}'\mathbf{A}=\mathbf{I}} Tr(\mathbf{A\Gamma}'\mathbf{X}) \\
&= \arg\max_{\mathbf{A}'\mathbf{A}=\mathbf{I}} Tr(\mathbf{X}'\mathbf{\Gamma A}')
\end{aligned}$$

which is known to be maximized w.r.t $\mathbf{A}$ by $\hat{\mathbf{A}} = \mathbf{UV}'$ with $\mathbf{X}'\mathbf{\Gamma} = \mathbf{UDV}'$ (Zou et al., 2006). $\square$

Proposition 10. *The Weighted SELF Objective (4.6) without sparsity is minimized w.r.t* $\mathbf{A}$ *by* $\hat{\mathbf{A}} = \mathbf{W}^{-1/2}\mathbf{UV}'$ *where* $\mathbf{UDV}'$ *is the singular value decomposition of* $\sqrt{\mathbf{W}}\mathbf{X}'\mathbf{\Gamma}$.

Proof.

Define $\hat{\mathbf{A}}$, $\mathbf{Z}$, and $\mathbf{B}$ as follows:

$$\hat{\mathbf{A}} = \underset{\mathbf{A}'\mathbf{W}\mathbf{A}=\mathbf{I}}{\arg\min} \|\mathbf{X}\sqrt{\mathbf{W}} - \mathbf{\Gamma}\mathbf{A}'\sqrt{\mathbf{W}}\|_F^2$$
$$\mathbf{Z} = \mathbf{X}\sqrt{\mathbf{W}}$$
$$\mathbf{B} = \sqrt{\mathbf{W}}\mathbf{A}$$

Observing that $\mathbf{A} = \mathbf{W}^{-1/2}\mathbf{B}$, we have

$$\mathbf{A}'\mathbf{W}\mathbf{A} = \mathbf{I} \iff (\sqrt{\mathbf{W}}\mathbf{A})'(\sqrt{\mathbf{W}}\mathbf{A}) = \mathbf{I} \iff \mathbf{B}'\mathbf{B} = \mathbf{I} \tag{4.7}$$

Therefore we can express $\hat{\mathbf{A}}$ in terms of $\mathbf{B}$ and $\mathbf{Z}$:

$$\begin{aligned}\hat{\mathbf{A}} &= \underset{\mathbf{A}'\mathbf{W}\mathbf{A}=\mathbf{I}}{\arg\min} \|\mathbf{X}\sqrt{\mathbf{W}} - \mathbf{\Gamma}\mathbf{A}'\sqrt{\mathbf{W}}\|_F^2 \\ &= \mathbf{W}^{-1/2}\hat{\mathbf{B}}\end{aligned}$$

where

$$\hat{\mathbf{B}} = \underset{\mathbf{B}'\mathbf{B}=\mathbf{I}}{\arg\min} \|\mathbf{Z} - \mathbf{\Gamma}\mathbf{B}'\|_F^2 \tag{4.8}$$

which is given by Proposition 9. □

4.5.1 SELF With Missing Data

Sparse PCA models thus far have accommodated missing data by using the estimated low-rank structure of the data to impute values (4.3). In simple terms, the model predicts the values of missing data based on observed values from the same row. This imputation strategy is applied to the SELF model in 4.10.1.

Here we examine an alternative to imputation of missing values, instead using only what data is observed to estimate the model.

Non-Imputed Estimation of A. In the non-missing case, SELF estimates $\mathbf{A}$ by taking the SVD of $\mathbf{X}'\mathbf{\Gamma}$ (Algorithm 13 Line 4). This is clearly not doable if there is missingness in $\mathbf{X}$, and the prior missingness approach imputed values to have a full data matrix to work with.

However, consider the role each row of $\mathbf{A}$ plays in the model. SELF approximates $\mathbf{X}$ with $\hat{\mathbf{X}} = \mathbf{\Gamma}\mathbf{A}'$. Rows of $\mathbf{A}$ determine how to produce columns of $\hat{\mathbf{X}}$ from values in $\mathbf{\Gamma}$, so that $\hat{\mathbf{X}}_{\bullet j} = \mathbf{\Gamma}\mathbf{A}'_{j\bullet}$.

Algorithm 13 Sparse Estimation of Latent Factors

Input:

- $\mathbf{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- *epochs*, number of epochs
- m^e, FSA annealing schedule specifying the number of nonzero entries of $\mathbf{A}$ kept at epoch e

Output: Estimated latent factors $\mathbf{\Gamma}$ of $\mathcal{X}$ and estimated loading matrix $\mathbf{A}$

```
1: Initialize $\mathbf{\Gamma}$ as the first $k$ principal components and $\mathbf{A}$ as the first $k$ loading
     vectors from PCA on $\mathbf{X}$
2: for $e = 1$ to epochs do
3:     $\mathbf{\Gamma} = \mathbf{X}\mathbf{A}(\mathbf{A}'\mathbf{A})^{-1}$
4:     $\mathbf{A} = \mathbf{U}\mathbf{V}'$ with $\mathbf{X}'\mathbf{\Gamma} = \mathbf{U}\mathbf{D}\mathbf{V}'$ by SVD
5:     Apply FSA to $\mathbf{A}$
6:         $m^e$ is the desired number of nonzero elements of $\mathbf{A}$ for current epoch
7:         Retain the top $m^e$ entries of $\mathbf{A}$ with the highest absolute values, setting the
              remaining entries to 0
8: end for
```

Suppose we wished to estimate each row $\mathbf{A}_{j\bullet}$ individually. Rather than regarding the entire objective we use a column-specific objective for SELF:

$$l = \|\mathbf{X}_{\bullet j} - \mathbf{\Gamma}\mathbf{A}'_{j\bullet}\|_F^2$$

Proposition 11. *The Columnar SELF Objective* $\|\mathbf{X}_{\bullet j} - \mathbf{\Gamma}\mathbf{A}'_{j\bullet}\|_F^2$ *is minimized w.r.t* $\mathbf{A}'_{j\bullet}$ *by* $\hat{\mathbf{A}}'_{\mathbf{j}\bullet} = (\mathbf{\Gamma}'\mathbf{\Gamma})^{-1}\mathbf{\Gamma}'\mathbf{X}_{\bullet j}$.

Proof.

$$\begin{aligned}
&\arg\min_{\mathbf{a}} \|\mathbf{X}_{\bullet j} - \mathbf{\Gamma}\mathbf{a}\|_F^2 \\
&= \arg\min_{\mathbf{a}} (\mathbf{X}_{\bullet j} - \mathbf{\Gamma}\mathbf{a})'(\mathbf{X}_{\bullet j} - \mathbf{\Gamma}\mathbf{a}) \\
&= \arg\min_{\mathbf{a}} \mathbf{X}'_{\bullet j}\mathbf{X}_{\bullet j} - 2\mathbf{X}'_{\bullet j}\mathbf{\Gamma}\mathbf{a} + \mathbf{a}'\mathbf{\Gamma}'\mathbf{\Gamma}\mathbf{a} \\
&= \arg\min_{\mathbf{a}} -2\mathbf{X}'_{\bullet j}\mathbf{\Gamma}\mathbf{a} + \mathbf{a}'\mathbf{\Gamma}'\mathbf{\Gamma}\mathbf{a}
\end{aligned}$$

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{a}} - 2\mathbf{X}'_{\bullet j}\mathbf{\Gamma}\mathbf{a} + \mathbf{a}'\mathbf{\Gamma}'\mathbf{\Gamma}\mathbf{a} = 0 \\
&\quad\implies -2\mathbf{\Gamma}'\mathbf{X}_{\bullet j} + 2\mathbf{\Gamma}'\mathbf{\Gamma}\mathbf{a} = 0 \\
&\quad\implies \mathbf{a} = (\mathbf{\Gamma}'\mathbf{\Gamma})^{-1}\mathbf{\Gamma}'\mathbf{X}_{\bullet j}. \quad \square
\end{aligned}$$

Estimation of a row $\mathbf{A}_{j\bullet}$ separately from the rest of $\mathbf{A}$ presents an opportunity to navigate missing values. Because the estimate of $\mathbf{A}_{j\bullet}$ relies only on column $\mathbf{X}_{\bullet j}$ of $\mathbf{X}$, only the missing values of $\mathbf{X}_{\bullet j}$ are relevant. Therefore, consider $\tilde{\mathbf{X}}_{\bullet j}$ to be the subset of $\mathbf{X}_{\bullet j}$ which excludes its missing values. Let $\tilde{\mathbf{\Gamma}}$ be the subset of $\mathbf{\Gamma}$ excluding the same rows as $\tilde{\mathbf{X}}_{\bullet j}$.

Per Proposition 11, we estimate $\mathbf{A}'_{j\bullet} = (\tilde{\mathbf{\Gamma}}'\tilde{\mathbf{\Gamma}})^{-1}\tilde{\mathbf{\Gamma}}'\tilde{\mathbf{X}}_{\bullet j}$. Composing these estimates together for each j, we have a candidate estimator for $\mathbf{A}$.

Note that this deviation from SVD means $\mathbf{A}$ is no longer orthogonal.

Non-Imputed Estimation of Gamma. Estimation of $\mathbf{\Gamma}$ with missingness can also be approached without imputation. We examine the technique of estimating each row of $\mathbf{\Gamma}$ using only non-missing values of the corresponding row of $\mathbf{X}$.

Consider a row $\boldsymbol{\gamma}$ of $\mathbf{\Gamma}$ and corresponding data row $\mathbf{x}$. Per Proposition 7, $\boldsymbol{\gamma}$ can be estimated from the values of $\mathbf{x}$ and $\mathbf{A}$ as $\boldsymbol{\gamma} = \mathbf{xWA}(\mathbf{A}'\mathbf{WA})^{-1}$.

In the case of missing values, subset $\tilde{\mathbf{x}}$ is taken as a row of values of $\mathbf{x}$ for which data is not missing. Further, $\tilde{\mathbf{A}}$ is the row-subset of $\mathbf{A}$ which corresponds to the columns retained in $\tilde{\mathbf{x}}$, and $\tilde{\mathbf{W}}$ the same subset of $\mathbf{W}$. These subsets allow us to circumvent missing data in $\mathbf{x}$ to estimate the corresponding row of $\mathbf{\Gamma}$. Incorporating a ridge bias per Remark 5 yields

$$\boldsymbol{\gamma} = \tilde{\mathbf{x}}\tilde{\mathbf{W}}\tilde{\mathbf{A}}(\tilde{\mathbf{A}}'\tilde{\mathbf{W}}\tilde{\mathbf{A}} + \lambda_r\mathbf{I})^{-1}$$

Ill-Condition Filter. An additional technique for mitigating the effects of missingness on model estimation concerns the estimation of $\mathbf{A}$. Within each epoch, $\mathbf{A}$ is updated to minimize the SELF objective based on the values of $\mathbf{X}$ and $\mathbf{\Gamma}$, as shown in Proposition 9. $\mathbf{\Gamma}$ itself is estimated row-wise as $\boldsymbol{\gamma} = \tilde{\mathbf{x}}'\tilde{\mathbf{A}}(\tilde{\mathbf{A}}'\tilde{\mathbf{W}}\tilde{\mathbf{A}} + \lambda_r\mathbf{I})^{-1}$ as shown above.

Recall that a ridge parameter λ_r is used to alleviate potential instability of the inversion problem $(\tilde{\mathbf{A}}'\tilde{\mathbf{W}}\tilde{\mathbf{A}} + \lambda_r\mathbf{I})^{-1}$. In cases of high missingness, however, the inversion calculation can still be unstable. We store this information by calculating, for each row of the $\mathbf{\Gamma}$ update step, the reciprocal condition number[2] of the matrix $\tilde{\mathbf{A}}'\tilde{\mathbf{W}}\tilde{\mathbf{A}}$ used to produce it. This gives us a measurement of which rows of $\mathbf{\Gamma}$ risk being the least reliably estimated.

Combining this information with a desired threshold λ_m, $\mathbf{A}$ can be updated using only rows from $\mathbf{\Gamma}$ and $\mathbf{X}$ with satisfactory reciprocal condition numbers. This is implemented in Algorithm 14, Line 29.

[2] Details on conditioning can be found in Higham (2002). We use reciprocal 1-norm condition numbers as calculated by LAPACK (Anderson et al., 1999, chap. 4).

4.5.2 Feature Weights

In the presence of noisy or irrelevant features, model accuracy may be improved by downweighting their effect on model training. Here we explore downweighting features based on their unexplained error, as estimated by deviation from observed and estimated values for each feature.

Take $\mathbf{X} = \mathbf{\Gamma}\mathbf{A}' + \mathbf{E}$ where $\mathbf{A}$ is fixed, $\mathbf{\Gamma} \sim \mathcal{N}(\vec{\mathbf{0}}, \mathbf{I})$, and $\mathbf{E} \sim \mathcal{N}(\vec{\mathbf{0}}, \mathbf{\Psi})$ with $\mathbf{\Psi}$ a diagonal matrix having diagonal entries $\mathbf{\Psi}_{jj} = \sigma_j^2$.

During model training, $\mathbf{X}$ is modeled as a low-rank matrix $\hat{\mathbf{X}}$. For each column $\mathbf{X}_{\bullet j}$ of $\mathbf{X}$, error variance σ_j^2 can be estimated as $\|\hat{\mathbf{X}}_{\bullet j} - \mathbf{X}_{\bullet j}\|_2^2$, where rows with missing values for $\hat{\mathbf{X}}_{\bullet j}$ are ignored.

We use estimated reciprocal standard deviations $\hat{\sigma_j^2}^{-1/2}$ as feature weights during estimation.

The additional steps for estimation with weights occur in each epoch as follows:

- Estimate data matrix $\mathbf{X}$ as $\hat{\mathbf{X}} = \mathbf{\Gamma}\mathbf{A}'$ based on current model estimates of $\mathbf{\Gamma}$ and $\mathbf{A}$.
- For each column $\mathbf{X}_{\bullet j}$ and current estimate $\hat{\mathbf{X}}_{\bullet j}$, calculate estimated variance $\|\hat{\mathbf{X}}_{\bullet j} - \mathbf{X}_{\bullet j}\|_2^2$ by ignoring rows with missing values. Collect these variances in vector $\mathbf{v}$.
- Threshold values of $\mathbf{v}$ to take a minimum value of .1. Produce weights matrix $\mathbf{W}$ with values as the inverse values in $\mathbf{v}$.

The calculation steps for weighted SELF are included in Algorithm 14, Line 18.

4.5.3 Weighted Feature Selection

Weights matrix $\mathbf{W}$ from 4.5.2 may be informative in feature selection.

In the SELF model, feature selection acts through sparsification of $\mathbf{A}$. Weights can be incorporated in feature selection by imposing FSA on $\mathbf{W}\mathbf{A}$ and imposing the resulting sparsity on $\mathbf{A}$. This modification encourages dropping features which are estimated through $\mathbf{W}$ to be less relevant. Weighted feature selection is implemented in Algorithm 14, Line 35.

Algorithm 14 Sparse Estimation of Latent Factors with Missing Data

Input:

- $\mathcal{X}$, n-by-p data matrix
- k, desired number of sparse principal components
- *epochs*, number of epochs
- m^e, FSA annealing schedule specifying the number of nonzero entries of $\mathbf{A}$ kept at epoch e
- λ_r, ridge parameter to stabilize matrix inversion
- λ_m, ill-condition threshold

Output: Estimated latent factors $\mathbf{\Gamma}$ of $\mathcal{X}$ and estimated loading matrix $\mathbf{A}$

```
 1: Set $\mathbf{X} = \mathcal{X}^0$
 2: Initialize feature weights matrix $\mathbf{W}$ as $\mathbf{I}_p$
 3: Initialize row reciprocal condition vector $\mathbf{r}$ as $n$-dimensional $\vec{\mathbf{0}}$
 4: Initialize $\mathbf{\Gamma}$ as the first $k$ principal components and $\mathbf{A}$ as the first $k$ loading
      vectors from PCA on $\mathbf{X}$
 5: for $e = 1$ to epochs do
 6:     Update $\mathbf{\Gamma}$:
 7:     for each row $\mathbf{x}$ of $\mathbf{X}$ do
 8:         if data row is fully-observed then
 9:             Update corresponding row $\boldsymbol{\gamma}$ of $\mathbf{\Gamma}$ with $\boldsymbol{\gamma} = \mathbf{xWA}(\mathbf{A'WA})^{-1}$
10:         else if data row has missingness then
11:             Set $\tilde{\mathbf{x}}$ as the column subset of $\mathbf{x}$ for which its values are non-missing in original
                  data row $\mathcal{X}_{i\bullet}$ where $i$ is the current row number
12:             Set $\tilde{\mathbf{A}}$ as column subset of $\mathbf{A}$ corresponding to $\tilde{\mathbf{x}}$
13:             Set $\tilde{\mathbf{W}}$ as row and column subset of $\mathbf{W}$ corresponding to $\tilde{\mathbf{x}}$
14:             Update corresponding row $\boldsymbol{\gamma}$ of $\mathbf{\Gamma}$ with $\boldsymbol{\gamma} = \tilde{\mathbf{x}}\tilde{\mathbf{W}}\tilde{\mathbf{A}}(\tilde{\mathbf{A}}'\tilde{\mathbf{W}}\tilde{\mathbf{A}} + \lambda_r\mathbf{I})^{-1}$
15:             Update $\mathbf{r}_i$ to be the reciprocal condition number of $\tilde{\mathbf{A}}'\tilde{\mathbf{W}}\tilde{\mathbf{A}}$        ▷ See Section 4.5.1
16:         end if
17:     end for
18:     if using feature weights then
19:         Set $\hat{\mathbf{X}} = \mathbf{\Gamma A'}$
20:         Estimate column error variances $\mathbf{v}$ with $\mathbf{v}_j$ the observed variance of $\hat{\mathbf{X}}_{\bullet j}$ −
              $\mathbf{X}_{\bullet j}$ where for each $j$, rows with missing values $\mathcal{X}_{ij}$ in the original data are
              excluded from the calculation
21:         Update diagonal values of $\mathbf{W}$ with $\mathbf{W}_{jj} = \max(\mathbf{v}_j, .1)^{-1}$
22:     end if
```

Algorithm 14 Sparse Estimation of Latent Factors with Missing Data (continued)

```
23:     Update $\mathbf{A}$:
24:     if $\mathcal{X}$ has no missing data then
25:         Set $\mathbf{A} = \mathbf{U}\mathbf{V}'$ with $\mathbf{X}'\mathbf{\Gamma} = \mathbf{U}\mathbf{D}\mathbf{V}'$ by SVD
26:     else
27:         for $j = 1, \ldots, p$ do
28:             Let $F_1$ be an index on rows of $\mathcal{X}$ having $F_1 = \{1 \leq i \leq n \mid \mathcal{X}_{ij}$ is non-missing$\}$
29:             Let $F_2$ be an index on rows of $\mathcal{X}$ having $F_2 = \{1 \leq i \leq n \mid \mathbf{r}_i \geq \lambda_m\}$
30:             Produce $\tilde{\mathbf{\Gamma}}$ by subsetting rows of $\mathbf{\Gamma}$ to $F_1 \cap F_2$
31:             Produce $\tilde{\mathbf{X}}_{\bullet j}$ by subsetting $\mathcal{X}$ to column $j$ and rows in $F_1 \cap F_2$
32:             Update row $j$ of $\mathbf{A}$ with $\mathbf{A}_{j\bullet} = (\tilde{\mathbf{\Gamma}}'\tilde{\mathbf{\Gamma}} + \lambda_r \mathbf{I})^{-1}\tilde{\mathbf{\Gamma}}'\tilde{\mathbf{X}}_{\bullet j}$.    ▷ If $F_1 \cap F_2 = \varnothing$, set $\mathbf{A}_{j\bullet} = \vec{\mathbf{0}}$
33:         end for
34:     end if
35:     Apply FSA to $\mathbf{A}$
36:         $m^e$ is the desired number of nonzero elements of $\mathbf{A}$ for current epoch
37:         Identify the top $m^e$ entries of $\mathbf{WA}$ with the highest absolute values. Retain the
               corresponding entries of $\mathbf{A}$, setting the remaining entries to 0
38: end for
39: $\mathbf{\Gamma}\mathbf{A}'$ is the low-rank estimate of $\mathcal{X}$
```

4.6 Methodology Comparison

We make some comparisons between models.

The PCA objective (2.2) estimates $\mathbf{X}$ as $\mathbf{XBB}'$ where $\mathbf{B}'\mathbf{B} = \mathbf{I}$. WPCA incorporates feature weights into the PCA objective.

Of the sparse PCA models, the first distinction to make is those models that follow the "self-regression" (She, 2017) setup of SPCA. That is, SPCA, ASPCA, and GSPCA all approximate $\mathbf{X} \approx \mathbf{XBA}'$ with $\mathbf{B}$ sparse and $\mathbf{A}'\mathbf{A} = \mathbf{I}$. SELF, however, follows the SRRR-SPCA setup where $\mathbf{X} \approx \mathbf{\Gamma A}'$, with $\mathbf{A}$ sparse. Yet SELF does share a characteristic of SPCA that semi-orthogonality is lost when sparsity is imposed on $\mathbf{A}$.

Another difference worth mentioning is that while SELF and SRRR-SPCA both directly estimate low-dimensional representations of $\mathbf{X}$, the representation $\mathbf{V}$ in SRRR-SPCA satisfies $\mathbf{V}'\mathbf{V} = \mathbf{I}$ while SELF's $\mathbf{\Gamma}$ does not. Our motivation for allowing this relaxation on both SELF parameters $\mathbf{\Gamma}$ and $\mathbf{A}$ is to test the model with the non-imputation strategy on missing data. To perform this approach, rows of $\mathbf{\Gamma}$ are estimated individually, which allows us to accommodate the particular missingness pattern observed in each row of $\mathbf{X}$. Likewise, $\mathbf{A}$ estimation handles missingness in $\mathbf{X}$ by estimating the rows of $\mathbf{A}$ individually, accommodating the particular missingness pattern observed in each column of $\mathbf{X}$. For both $\mathbf{\Gamma}$ and $\mathbf{A}$, the piecewise estimation strategy renders the parameters non-orthogonal.

Another characteristic of these sparse PCA models concerns the nature of how the sparse parameter is estimated. In particular, the implementations of SPCA and ASPCA that we examine estimate their loading matrix sequentially. In contrast, SRRR-SPCA estimates its sparse parameter's columns jointly. GSPCA follows this approach over the sequential strategy, applying FSA on its entire loading matrix. SELF follows this method of imposing sparsity as well. Note that while SELF's non-imputation method does use a kind of sequential approach to estimating the rows of $\mathbf{A}$, it differs from the sequential nature of SPCA and ASPCA in an important way. The estimation of $\mathbf{A}$ in SELF is a self-contained step which is repeated within training epochs alongside the update step of $\mathbf{\Gamma}$. As a result, not only is the full-column estimate of $\mathbf{A}$ available during every epoch of model training, but across epochs all of $\mathbf{A}$ is able to receive updates. Potential advantages of the sequential strategy applied to SPCA is discussed in Sjöstrand (2005).

GSPCA updates its sparse parameter via gradient descent, a departure from SPCA, ASPCA, and SELF. Therefore, its sparse parameter requires a learning rate to be applied. The choice of learning rate may need to be considered alongside annealing rate μ. That is, a more aggressive annealing schedule may benefit from a learning rate that is able to update quickly in response to features being dropped. On the other hand, a slower μ and lower learning rate, perhaps combined with momentum[3], may encourage smoother updates to GSPCA. Thus GSPCA's sparse parameter might be expected to show less dramatic parameter updates compared to a sequential approach, but is also less fixed in the sense that the entire sparse parameter receives updates in every epoch.

In contrast to this, SELF recreates its estimates $\mathbf{\Gamma}$ and $\mathbf{A}$ based on each other and on $\mathbf{X}$, from scratch, within each epoch.

The imputation method (4.3) is available to all examined models. This approach uses each row's observed values to estimate its unobserved values. The relationship between observed and unobserved features is taken from what has been learned of the data's low-rank structure, as captured in model parameters. Estimating missing values by regressing them on observed values is a common approach to imputation. An early example has Buck (1960) imputing missing values by generating a series of linear regressions to predict unobserved values by observed values. Beale and Little (1975) refine this approach by repeating iterations of imputation until the estimated values converge. Many modern imputation strategies, including ours, involve estimating and iteratively updating these estimates.

The basis of our estimates for missing data is learning low-rank structure. While the imputation strategy estimates a low-rank representation by using data completed with imputed values, the non-imputation approach (4.5.1) only uses observed values in its calculations. This general tactic for estimation has been called *available-case analysis* (Little and Rubin, 2019, chap. 3). A noted drawback of the available-case approach, however, is that the resulting estimate may not satisfy the desirable properties expected of that estimate in the non-missing case. This situation emerges in the non-imputed estimation of the SELF model, where the componentwise calculation of $\mathbf{A}$ does not guarantee $\mathbf{A}'\mathbf{A} = \mathbf{I}$; this is not an issue in the imputation case. If desired, $\mathbf{\Gamma}$ and $\mathbf{A}$ can be rotated by multiplication with an appropriate square matrix to produce an orthogonal $\mathbf{A}$. A sparse PCA variant that follows a non-imputation strategy but retains orthogonality its loading matrix parameter can be found in Zhang (2016).

A complication of the imputation strategy we employ concerns the sequentially-updating nature of SPCA and ASPCA as examined here. In these models, the model's understanding of low-rank

[3] *Momentum* is a gradient descent technique where each applied gradient update is a weighted average between current and previous gradient estimates (Sutskever et al., 2013)

structure in the data is limited by which loading matrix column step the estimation process has reached. That is, when these models begin estimation of the first loading matrix column, only that column is available for producing imputation estimates, and so imputation in this step assumes a rank-1 structure. Likewise, at the step estimating loading matrix column k, only the first k columns are available to estimate imputed values. Further, at the point when the final column is being estimated, any improved ability to model the missing data does not affect the estimates of loading columns that have already been estimated.

Methodologies for handling missingness in low-rank settings have seen enormous attention recently due to their applicability in matrix factorization for recommender systems. It is worth mentioning some notable contributions in modeling incomplete low-rank data. Candes and Recht (2012) show that, with some probability, many incomplete low-rank matrices can have their missing values recovered exactly; they also provide a convex algorithm for doing so. Mazumder et al. (2010) soften the goal of exact recovery and instead seek to learn low-rank structure by minimizing a loss function which ignores missing values. Regularization is a frequent technique used to combat estimation problems introduced by higher levels of missingness; see for instance Josse et al. (2009). Many variations of both imputation and non-imputation strategies have emerged in the literature. For an overview, techniques with specific relevance to PCA are briefly outlined in (Jolliffe, 2002, chap. 13.6). Low-rank methods for matrix completion were recently surveyed by Nguyen et al. (2019). A broader overview of matrix completion methods is given by Dax (2014).

SELF includes additional modeling options including the incorporation of feature weights in model estimation and feature selection and the ability to avoid data rows showing too much missingness. These options, and imputation and non-imputation strategies, are compared for SELF in an ablation study in Section 4.10.

4.7 Sparse PCA Simulations

We test models on simulated principal component regression (PCR) data. These models perform an unsupervised dimension reduction task and a supervised regression task. Having both tasks allows us to measure the accuracy of feature selection for dimension reduction and the subsequent accuracy of the overall regression model. This is tested over varying sample sizes and numbers of noisy features.

A simulated sample of $(\mathbf{\Gamma}, \mathbf{X}, \mathbf{y})$ begins with generating an n-by-3 matrix of NIID latent factors $\mathbf{\Gamma} \sim \mathcal{N}(\vec{\mathbf{0}}, \mathbf{I}_3)$.

The response vector $\mathbf{y}$ is generated as the sum of each row of $\mathbf{\Gamma}$.

$$\mathbf{y} = \mathbf{\Gamma} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

We then generate n-by-10 data matrix $\mathbf{X}$ from $\mathbf{\Gamma}$. $\mathbf{A}$ is generated as a 10-by-3 matrix of independent standard normal deviates. Following this,

1. Relevant features of data are generated as $\tilde{\mathbf{X}} = \mathbf{\Gamma}\mathbf{A}' + \mathbf{E}$ where $\mathbf{E} \sim \mathcal{N}(\vec{\mathbf{0}}, .3\mathbf{I}_{10})$
2. q uninformative (noisy) features are generated as n-by-q matrix $\mathbf{Q} \sim \mathcal{N}(\vec{\mathbf{0}}, \mathbf{I}_q)$
3. Final data matrix $\mathbf{X}$ is produced as the composite of relevant and irrelevant features: $\mathbf{X} = \begin{bmatrix} \tilde{\mathbf{X}} & \mathbf{Q} \end{bmatrix}$
4. Missingness is imposed on entries of $\mathbf{X}$ randomly at a rate of 50%.

Simulations are run on varying n and q. Model accuracy is reported by the following measures. First, each PCR model's loading matrix used for dimension reduction has sparsity that is meant to reduce the impact of irrelevant features. Not all models impose exact sparsity, but row-norms of the loading matrix can be used to see how many of the ten relevant features were selected as the most important. For the regression component of the model, we report R^2 on test data simulated alongside training data, as well as R^2 on test data without missingness imposed. Note that in this latter case, the models were still trained on data with missingness. Resulting scores are averages of 100 replicates of each data configuration (n, q).

The results show SELF having the best performance in test R^2. PCA shows a slight performance edge over WPCA, though they are close. Beyond the lowest values of n, model performance does not seem to improve as a function of sample size.

We highlight two observations from these results. First, Table 4.2 shows the correctness of model "feature selection". Because model sparsity is not imposed in a way that explicitly selects a fixed number of features, we use loading matrix row-norms to count how many of the 10 true features occupy the 10 "largest" rows. By this metric, PCA and WPCA appear to do very well. Although we considered this a natural approach to measuring whether models were focusing on the right features, in this setting the metric did not reliably translate into model accuracy. For instance, PCA and WPCA often outperformed SELF by this metric, but are not comparable to SELF performance in test R^2.

A second observation the striking behavior of SPCA, which shows dramatically decreasing performance against increasing n. We examine this counter-intuitive behavior further.

Table 4.2: Number of Features Correctly Selected of PCR Models with varying n, q

n	**q**	**PCA**	**WPCA**	**SPCA**	**ASPCA**	**JSPCA**	**GSPCA**	**SELF**[4]
100	**10**	9.2	9.3	8.4	9.0	8.7	9.2	9.3
100	**50**	8.7	8.6	7.5	8.5	7.7	8.5	8.9
100	**100**	8.4	8.4	7.1	8.2	6.9	8.2	8.3
500	**10**	9.9	9.9	9.5	9.6	9.3	9.8	9.8
500	**50**	9.8	9.8	9.2	9.4	9.0	9.8	9.7
500	**100**	9.8	9.8	9.1	9.4	8.8	9.7	9.7
1,000	**10**	9.9	10.0	9.6	9.7	9.5	9.9	9.9
1,000	**50**	9.9	9.9	9.5	9.6	9.3	9.9	9.8
1,000	**100**	9.8	9.9	9.5	9.6	9.3	9.8	9.8
2,000	**10**	10.0	10.0	9.8	9.8	9.6	10.0	9.9
2,000	**50**	9.9	10.0	9.7	9.7	9.4	9.9	9.9
2,000	**100**	9.9	10.0	9.7	9.7	9.5	9.9	9.8
5,000	**10**	10.0	10.0	9.9	9.9	9.6	10.0	9.9
5,000	**50**	10.0	10.0	9.8	9.8	9.6	10.0	9.9
5,000	**100**	10.0	10.0	9.8	9.8	9.7	10.0	9.9
10,000	**10**	10.0	10.0	9.9	9.9	9.6	10.0	9.9
10,000	**50**	10.0	10.0	9.9	9.9	9.7	10.0	9.9
10,000	**100**	10.0	10.0	9.9	9.9	9.7	10.0	9.9

[4] The SELF model used in these results follows Algorithm 14 with the feature weights option, and is identified as variant S-NAWFS in 4.10.2.

Table 4.3: Test R^2 of PCR Models with varying n, q

n	**q**	**PCA**	**WPCA**	**SPCA**	**ASPCA**	**JSPCA**	**GSPCA**	**SELF**
100	**0**	0.59	0.57	0.39	0.49	0.53	0.73	0.88
100	**10**	0.58	0.57	0.48	0.54	0.49	0.54	0.83
100	**50**	0.56	0.55	0.60	0.57	0.48	0.57	0.80
100	**100**	0.54	0.54	0.60	0.56	0.46	0.56	0.77
500	**0**	0.61	0.58	0.51	0.52	0.52	0.77	0.89
500	**10**	0.61	0.58	0.29	0.56	0.47	0.46	0.87
500	**50**	0.61	0.58	0.56	0.62	0.48	0.49	0.86
500	**100**	0.60	0.58	0.62	0.62	0.50	0.51	0.86
1,000	**0**	0.62	0.59	0.54	0.52	0.50	0.78	0.89
1,000	**10**	0.61	0.59	0.22	0.55	0.47	0.42	0.88
1,000	**50**	0.61	0.59	0.48	0.60	0.47	0.47	0.87
1,000	**100**	0.61	0.59	0.56	0.62	0.50	0.46	0.87
2,000	**0**	0.62	0.59	0.55	0.55	0.49	0.78	0.89
2,000	**10**	0.62	0.59	0.14	0.52	0.45	0.44	0.88
2,000	**50**	0.62	0.59	0.39	0.59	0.46	0.45	0.87
2,000	**100**	0.62	0.59	0.49	0.61	0.48	0.47	0.87
5,000	**0**	0.62	0.59	0.55	0.55	0.44	0.77	0.89
5,000	**10**	0.62	0.59	0.08	0.51	0.44	0.44	0.88
5,000	**50**	0.62	0.59	0.25	0.56	0.44	0.45	0.88
5,000	**100**	0.62	0.59	0.34	0.59	0.45	0.45	0.88
10,000	**0**	0.62	0.59	0.54	0.55	0.41	0.77	0.89
10,000	**10**	0.62	0.59	0.05	0.50	0.42	0.44	0.89
10,000	**50**	0.62	0.59	0.15	0.54	0.41	0.46	0.88
10,000	**100**	0.62	0.59	0.24	0.56	0.42	0.46	0.88

Table 4.4: Non-Missing Test R^2 of PCR Models with varying n, q

n	**q**	**PCA**	**WPCA**	**SPCA**	**ASPCA**	**JSPCA**	**GSPCA**	**SELF**
100	**0**	0.89	0.87	0.81	0.74	0.86	0.89	0.98
100	**10**	0.87	0.86	0.83	0.77	0.81	0.83	0.95
100	**50**	0.84	0.84	0.83	0.79	0.76	0.83	0.91
100	**100**	0.81	0.81	0.81	0.78	0.71	0.82	0.90
500	**0**	0.91	0.89	0.83	0.78	0.82	0.94	0.98
500	**10**	0.91	0.89	0.83	0.79	0.78	0.86	0.98
500	**50**	0.90	0.88	0.89	0.86	0.78	0.88	0.98
500	**100**	0.89	0.87	0.90	0.85	0.83	0.88	0.97
1,000	**0**	0.91	0.89	0.84	0.76	0.82	0.94	0.98
1,000	**10**	0.91	0.89	0.83	0.81	0.77	0.86	0.98
1,000	**50**	0.91	0.89	0.89	0.85	0.77	0.88	0.98
1,000	**100**	0.90	0.88	0.89	0.86	0.82	0.90	0.98
2,000	**0**	0.92	0.89	0.86	0.75	0.79	0.95	0.98
2,000	**10**	0.92	0.89	0.84	0.77	0.74	0.87	0.98
2,000	**50**	0.91	0.89	0.87	0.84	0.78	0.89	0.98
2,000	**100**	0.91	0.89	0.89	0.86	0.79	0.91	0.98
5,000	**0**	0.92	0.90	0.86	0.75	0.74	0.95	0.98
5,000	**10**	0.92	0.90	0.82	0.71	0.72	0.89	0.98
5,000	**50**	0.92	0.90	0.85	0.80	0.75	0.89	0.98
5,000	**100**	0.92	0.90	0.87	0.83	0.78	0.89	0.98
10,000	**0**	0.92	0.89	0.86	0.74	0.67	0.95	0.98
10,000	**10**	0.92	0.89	0.81	0.71	0.70	0.89	0.98
10,000	**50**	0.92	0.89	0.84	0.77	0.73	0.89	0.98
10,000	**100**	0.92	0.89	0.85	0.81	0.72	0.90	0.98

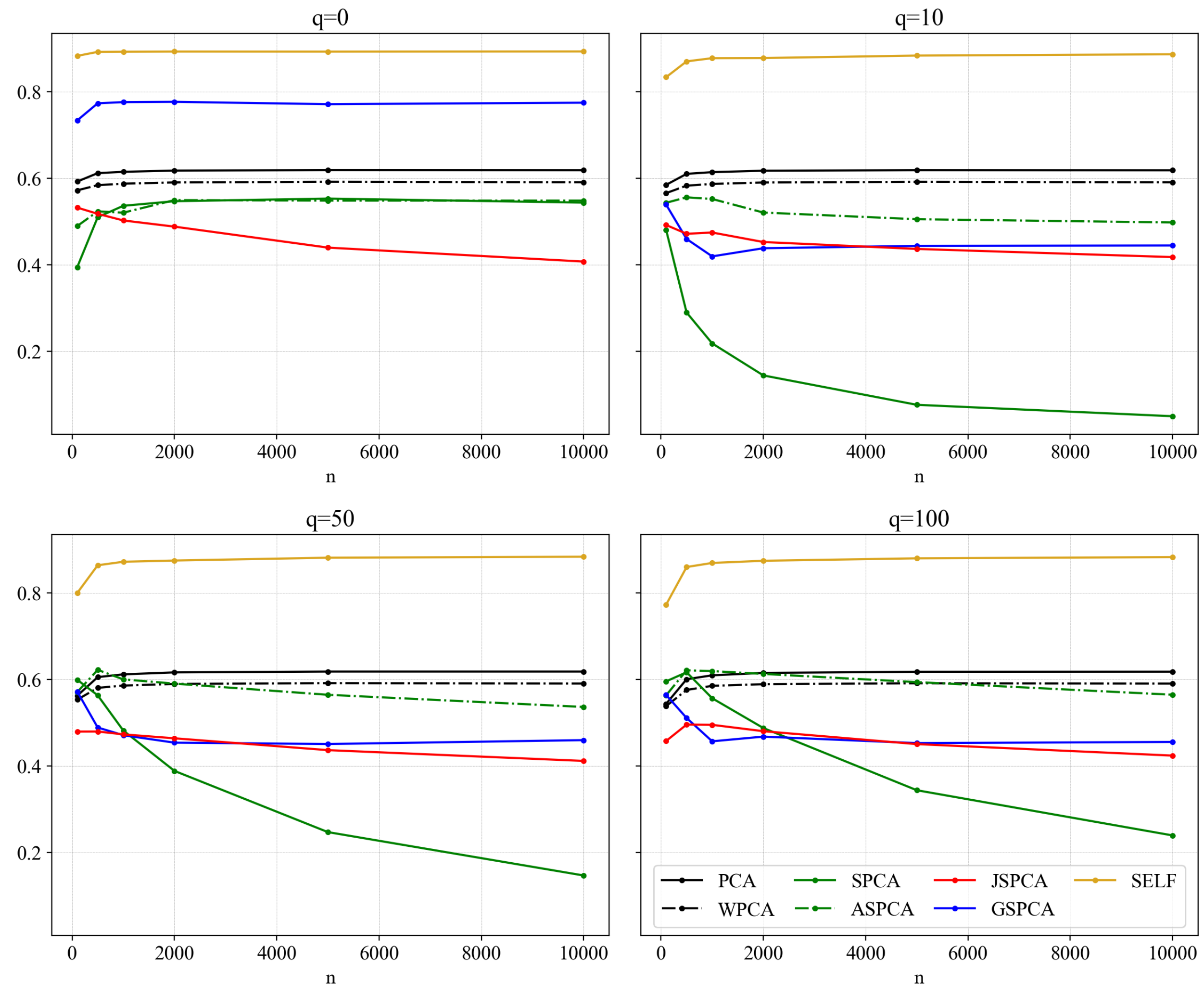


Figure 4.1: Test R^2 of PCR Models with varying n, q

4.7.1 SPCA Performance with Missing Data

We focus on the $q = 10$ data setting, which should present an easy feature selection problem given ample n.

Training and test R^2 plummet as training sample size n increases. However, when these models are applied to test data absent missing values, R^2 shows good values, indicating that the issue may be in how missingness is handled.

Table 4.5: SPCA PCR Model Performance with $q = 10$

n	**Number of True Features Selected**	**Training R^2**	**Test R^2**	**Non-Missing Test R^2**
100	8.45	0.55	0.48	0.83
500	9.47	0.34	0.29	0.83
1,000	9.63	0.24	0.22	0.83
2,000	9.79	0.15	0.14	0.84
5,000	9.88	0.08	0.08	0.82
10,000	9.93	0.05	0.05	0.81

We surmise two potential mechanisms for this issue which are byproducts of the imputation strategy rather than a reflection of the accuracy of the SPCA model. The first observation regards the implementation of the imputation strategy. Imputation is applied by using low-rank structure as learned during model training. This strategy is applied to all sparse PCA models[5] under consideration. However, it is not possible to apply imputation in exactly the same way. Consider that GSPCA, JSPCA, and SELF iteratively update their loading matrix in its entirety. Thus at any stage of model training, the current estimate of the entire loading matrix is applied to the imputation problem. As noted in (4.6) however, SPCA and ASPCA estimate their loading matrix *sequentially*. The consequence of this is that during estimation of the first sparse loading matrix column, the model is only able to capitalize on a rank-1 understanding of the data to apply towards imputation. When estimation proceeds to the next sparse loading column, a rank-2 estimate is applied. Eventually estimation arrives at the final loading matrix column permitting imputation estimation by the full rank of the model, but the enhanced accuracy of imputation is unable to impact the loading matrix columns that have already been estimated. This limitation applies to both SPCA and its close derivative, ASPCA. While ASPCA may show less drastic of an effect, it shares the pathology of suffering from higher training n. We do not dismiss the possibility that a modified imputation strategy may better accommodate sequential sparse PCA models, or that some modification of the sequential approach may circumvent this limitation; however, we do not pursue these directions in this work.

[5] A non-imputation strategy has been developed for SELF, but the imputation strategy is still implemented and tested for that model

The second drawback to the imputation strategy which may explain SPCA's behavior regards the instability of the imputation calculation in cases of high missingness. Missing values are imputed row by row, and higher levels of missingness can produce ill-posed calculations. Simply put, given the same missingness rate, an n = 10,000 sample is more likely to have rows presenting ill-posed problems compared to an n = 100 sample. Considering the case of outliers in data analysis, it is easy to imagine that a small number of extreme occurrences can interfere with model estimation. To observe this, we take a close look at a single simulation instance.

PCR simulation results are averages over 100 replicates, meaning models are trained against each data setting (n, q) generated from 100 random seeds. However, there is some consistent behavior within replicates.

First, given a seed and sample size n, training data is drawn as the first n rows of the same sample generated by that seed. Thus for a given seed value, the training data produced in the n=100 case is the same as the first 100 rows of any training data produced by that seed with $n > 100$. Second, whatever the training sample size n is chosen for training a model, the size of test data in these simulations is always $n = 1,000$. This means that within a seed, the only difference in training data for different n is how many rows of the same data sample are used for model training, and there is no difference in samples used to calculate test scores. The final result is that by examining SPCA performance within a specific seed against different n, all factors are held constant except sample size.

Taking a specific seed as an example, SPCA models trained at every level of n are all tested on the exact same test data. The performance of SPCA against data generated from this specific replicate unsurprisingly follows the overall observed pattern of performance.

Table 4.6: Sample SPCA PCR Simulation Replicate

Training n	**Number of True Features Selected**	**Training R^2**	**Test R^2**	**Non-Missing Test R^2**
100	10	0.92	0.76	0.99
500	10	0.59	0.56	0.86
1,000	10	0.53	0.42	0.99
2,000	10	0.20	0.21	0.99
5,000	10	0.06	0.04	0.99
10,000	10	0.06	0.04	0.95

However, focusing on a single simulation replicate allows us to examine model behavior more closely. Consider the model trained at $n = 100$, with 100-by-p training data $\mathbf{X}$. By the imputation strategy used with SPCA and other models, part of model estimation involves using model parameter estimates ($\mathbf{A}$, $\mathbf{B}$) to impute values of data under consideration. Thus a trained SPCA model in this framework has the implicit ability to impute values for data sampled from the same low-rank structure the model was trained on.

All SPCA models trained in this replicate use training data that contains $\mathbf{X}$ as its first 100 rows. Training data is created with a 50% missingness rate. Because the data is simulated, however, we can recover the true data values that were obfuscated before imposing missingness. Further, we can compare these true values against imputation estimates produced by this replicate's SPCA models trained on various n. By applying each model's imputation method to the same $\mathbf{X}$, we can compare imputed values against true and exact underlying values. This is shown in Figure 4.2.

The observed pattern shows accurate imputation for the SPCA model trained on n=100. As training sample size increases, however, imputed estimates take on increasingly extreme values. Simulated data naturally lies within a range of approximately -5 to +5. At the n=10,000 case, however, imputed values show values as extreme as 100.

The data for the instance we are examining has 10 true features and q=10 noisy features. Taking the extreme imputed value as an example, we identify the data row in $\mathbf{X}$ for which that value was generated. The training data row, $\mathbf{x}$, with missing values, is found in this instance to be

$$(\bullet \quad \bullet \quad \bullet \quad \bullet \quad -1.4 \quad -0.2 \quad \bullet \quad \bullet \quad \bullet \quad 2.0 \quad 0.8 \quad 1.5 \quad 1.3 \quad 0.8 \quad \bullet \quad \bullet \quad -0.5 \quad -0.3 \quad \bullet \quad 0.8)$$

Given estimated model parameters ($\mathbf{A}$, $\mathbf{B}$) from the $n = 10{,}000$ SPCA model from this replicate, we apply the imputation strategy outlined in Algorithm 8. Following the instructions produces placement matrix $\mathbf{P}$ and takes $\mathbf{D} = \mathbf{I} - \mathbf{B}\mathbf{A}'$. With ridge parameter λ_r and 0-imputed row $\mathbf{x}^0$, the estimated imputed values are $\mathbf{w} = -\mathbf{x}^0\mathbf{D}\mathbf{D}'\mathbf{P}'(\mathbf{P}\mathbf{D}\mathbf{D}'\mathbf{P}' + \lambda_r\mathbf{I})^{-1}$.

$$\mathbf{A} = \begin{pmatrix} 0.115 & -0.204 & 0.110 \\ -0.084 & 0.297 & -0.311 \\ 0.596 & 0.092 & 0.174 \\ 0.159 & 0.497 & 0.752 \\ -0.181 & -0.486 & 0.399 \\ -0.137 & -0.187 & 0.190 \\ 0.303 & -0.468 & -0.006 \\ 0.39 & 0.091 & -0.235 \\ -0.365 & -0.109 & 0.211 \\ -0.412 & 0.322 & -0.015 \\ 0.001 & 0.005 & 0.007 \\ 0.000 & 0.004 & -0.003 \\ 0.004 & -0.001 & 0.006 \\ 0.003 & -0.001 & 0.001 \\ 0.001 & -0.002 & 0.000 \\ 0.002 & 0.007 & 0.010 \\ -0.001 & 0.000 & -0.002 \\ -0.002 & 0.000 & -0.001 \\ 0.001 & 0.000 & 0.000 \\ -0.005 & -0.001 & 0.000 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0.112 & -0.200 & 0.102 \\ -0.082 & 0.297 & -0.312 \\ 0.597 & 0.092 & 0.172 \\ 0.159 & 0.498 & 0.754 \\ -0.180 & -0.488 & 0.400 \\ -0.133 & -0.185 & 0.186 \\ 0.303 & -0.469 & 0.000 \\ 0.391 & 0.090 & -0.234 \\ -0.365 & -0.108 & 0.210 \\ -0.413 & 0.321 & -0.012 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Taking $\lambda_r = 10^{-5}$, estimated missing values are $\mathbf{w} = -\mathbf{x}^0\mathbf{D}\mathbf{D}'\mathbf{P}'\mathbf{M}^{-1}$ where $\mathbf{M} = \mathbf{P}\mathbf{D}\mathbf{D}'\mathbf{P}' + \lambda_r\mathbf{I}$, yielding imputation estimates

$$\mathbf{w} = \begin{pmatrix} 1.73 & -12.09 & 42.33 & 98.05 & -16.29 & 0.11 & -2.28 & -0.05 & 1.28 & 0.03 \end{pmatrix}$$

which is seen to include extreme and problematic imputation estimates.

More importantly, the critical matrix $\mathbf{M}$ in this instance has a condition number of 6,516.3, indicating that the inversion problem is sensitive to perturbation. The instability of $\mathbf{M}$ is not particular to the 100-row sample $\mathbf{X}$, which was satisfactorily modeled by SPCA with imputation applied to that $n = 100$ case. The matrix $\mathbf{M}$ is created not only from the missingness of its corresponding row, but from the observed values of that row, the present model estimates ($\mathbf{A}$, $\mathbf{B}$), and the mitigating effect of λ_r.

The numerical challenges which emerge in this imputation setup are no different from those present in other settings, such as solutions for underdetermined systems. There are a number of possible responses. An easy approach is examining the tradeoff of introducing more bias by increasing λ_r. More deliberate constraints may be applied to imputation estimates to keep them within control, such as requiring they be sampled from an explicit probability distribution or incorporating a penalty on their magnitude. Or, the relative effect of potentially unstable imputed estimates can be reduced by a weighting strategy. However, we forego more advanced imputation strategies in favor of the non-imputation direction introduced in (4.5.1) and tested against imputation in (4.10).

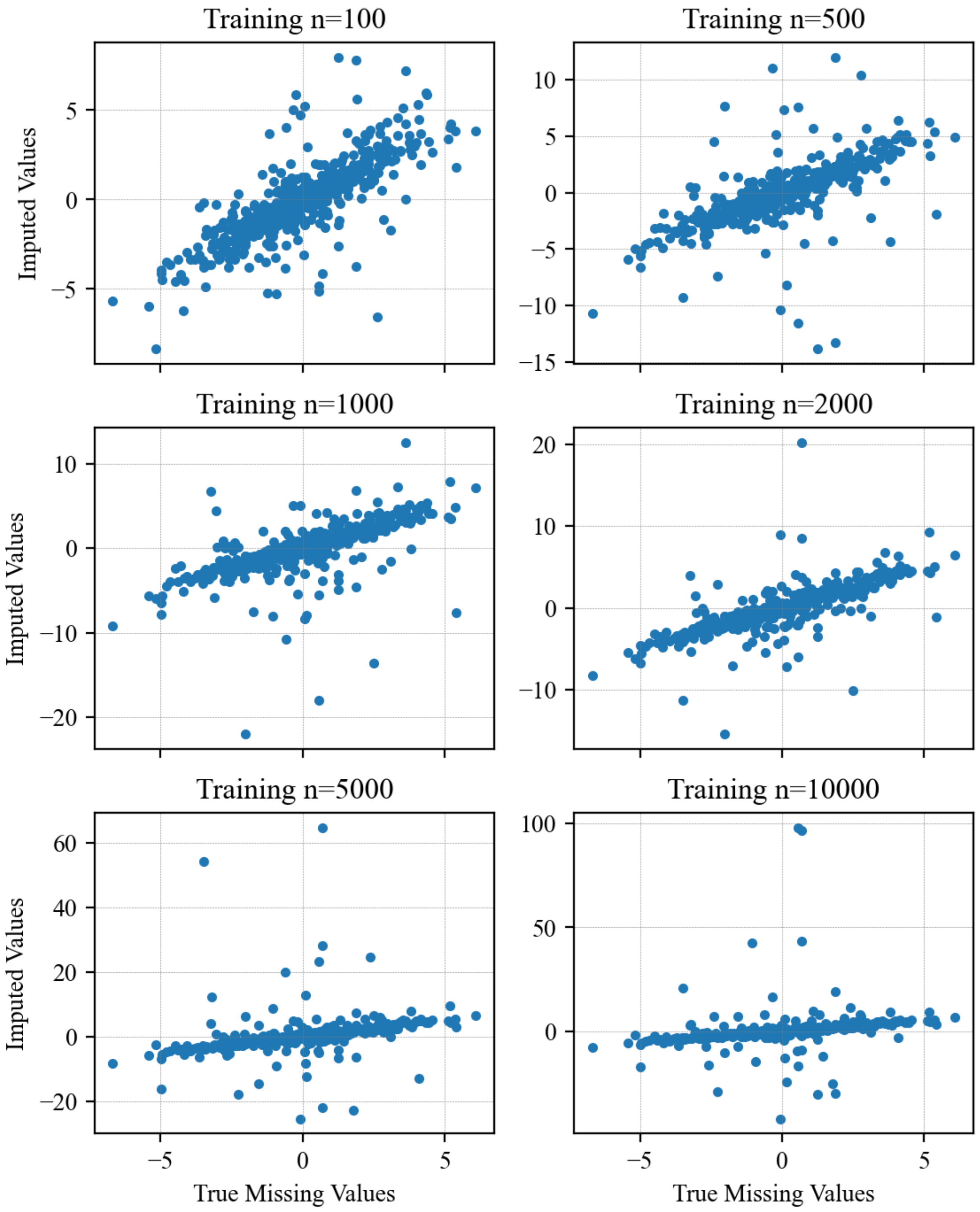


Figure 4.2: True vs. Imputed Missing Values for a Single Replicate of SPCA PCR Models

4.8 Sparse PCA Experiment - Postsecondary Institution Characteristics

Here we apply sparse PCA models with regression on a real dataset compiled of academic institution characteristics.

A cross-section of aggregate institutional metrics is collected for 1,197 public and private North American colleges and universities for the 2016-2017 academic year. Metrics include demographics of the student body like percentages by race and gender, figures on enrollments and degrees awarded, academic indicators of admits such as average ACT and SAT scores, aggregate financial aid characteristics and cost of attendance, graduation rates, and faculty characteristics. In total, there are 316 features, with `Endowment ($)` separated as the response to predict.

Not all metrics are found for all institutions, which generates missingness in the data. The overall missingness rate of all predictors is 44.0%. Predictors are centered and scaled to have observed mean 0 and standard deviation 1. As is often the case with monetary data, the response `Endowment ($)` shows skew which is resolved by taking a log transform (see Figure 4.3).

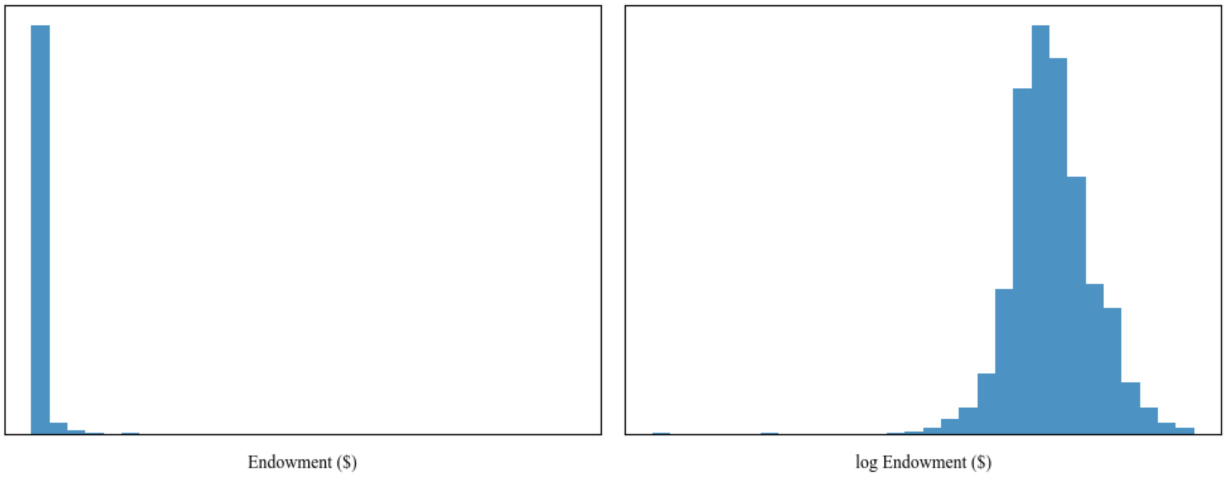


Figure 4.3: Deskewing Response `Endowment ($)`

Models tested include SPCA, ASPCA, GSPCA, and JSPCA with the imputation strategy described in Algorithm 8; the SELF variant identified as S-NAWFS in 4.10.2; 0-imputed PCA and WPCA. These are used as the PCA component of PCR models, so that each model's reduced-dimension estimate of the data is used as the predictors of an OLS model predicting the response. We include, for comparison, OLS applied directly on 0-imputed data.

PCR models estimate a rank-2 representation of data. PCR models imposing sparsity (SPCA, ASPCA, GSPCA, JSPCA, and SELF) are configured to allow on average 30 nonzero values in each column of their estimated loading matrix. Data is split into 70% for training and 30% for testing. Model performance is scored by observing the average R^2 on training and testing data over 100 random train/test splits.

Table 4.7: PCR Model Performance on Postsecondary Institutions Data

PCR Model	**Training R^2**	**Test R^2**
0-Imputed OLS	0.79	0.38
ASPCA + OLS	0.60	0.59
GSPCA + OLS	0.59	0.58
PCA + OLS	0.56	0.55
SELF + OLS	0.60	0.59
SPCA + OLS	0.60	0.59
WPCA + OLS	0.57	0.55
JSPCA + OLS	0.54	0.52

Results show the worst performance for OLS, though the model showed the highest average training R^2 indicating overfitting. Better performance is achieved by PCA, Weighted PCA, and JSPCA. The best-performing models are GSPCA, SPCA, ASPCA, and SELF.

Figure 4.8 shows a sample loading matrix estimated from a sparse model, partitioned into three tables. The sparsity imposed allows us to exclude most of the 315 predictors. Additionally, the sparsity shows an interpretable factor structure, allowing us to isolate "factor 1" predictors and "factor 2" predictors, shown below.

Factor 1 predictors: `Early action applicants enrolled, Final cohort (exclusions), Initial FTIC Cohort, Female freshman enrollment, Total enrollment, Bachelor's degrees awarded, Male freshman enrollment, FTIC Enrollment, Early action applicants accepted, Full-time enrollment, Undergraduate alumni of record, Number of applicants accepted, Male applicants accepted, Female applicants accepted, Final Cohort (Pell Grant), Initial FTIC Cohort (Pell Grant), Full-time faculty, ACT Science: % of students scoring below`

6, Number of registered clubs and organizations, Early decision applicants enrolled, Early action applicants, Total number of undergraduate students, Number of applicants, Female applicants, Male applicants, Early decision applicants accepted.

Factor 2 predictors: Average FTIC financial aid package FTIC, Average financial aid package, Average freshmen aid, Average aid, Average freshman retention rate, % of freshmen in top quarter of high school class, ACT Reading: % of students scoring 30-36, SAT 25th percentile, SAT 75th percentile, SAT Writing average score, 6-year graduation rate (average), SAT Math: % of students scoring 400-499, 6-year graduation rate (single cohort), 4-year graduation rate, 5-year graduation rate, SAT Math average score, Predicted graduation rate, SAT Verbal average score, ACT Composite average score, ACT 75th percentile, ACT 25th percentile.

It is readily observed that factor 1 predictors are a measurement of institution size, focusing on enrollment figures, while factor 2 predictors focus on academic strength. The inclusion of financial aid predictors in factor 2 suggest that those may also be a proxy for academic strength, or that they also capture variation and may call for a rank-3 model.

Table 4.8: Sample Estimated Loading Matrix for Automotive Data

Loadings 1	Loadings 2
-0.60	0
-0.58	0
-0.58	0
-0.58	0
-0.57	0
-0.57	0
-0.56	0
-0.59	0.03
-0.55	0
-0.59	0.05
-0.54	0
-0.53	0
-0.52	0
-0.51	0
-0.51	0
-0.51	0

Loadings 1	Loadings 2
-0.50	0
-0.50	0
-0.49	0
-0.49	0
-0.49	0
-0.58	0.11
-0.45	0
-0.45	0
-0.44	0
-0.41	0
0	-0.59
0	-0.60
0	-0.61
0	-0.62
0	-0.65
0	-0.66

Loadings 1	Loadings 2
0	-0.66
0	-0.68
0	-0.68
0	-0.68
0	-0.70
0	0.71
0	-0.71
0	-0.71
0	-0.72
0	-0.72
0	-0.73
0	-0.74
0	-0.74
0	-0.76
0	-0.76

Matrix partitioned into three submatrices

Excludes features with both loadings 0

4.9 Factor Mixture Model Experiment - Automotive Dataset

In this section we test a Factor Mixture Model (FMM) on a dataset of automobile sales records. FMM is a model that applies LCA on data to produce clusters and then applies separate factor analysis models to each cluster. We apply LCAFSA with sparse PCA methods on this data, followed by OLS regression, to predict the asking price of vehicles based on their listed characteristics.

4.9.1 Large Car Dataset

The Large Car Dataset (Competitive Intelligence Solutions, 2021) is a public dataset of vehicle listing collected from automotive dealers in Illinois from 2018-2020. Each record indicates a vehicle, the listed characteristics of that vehicle, and the listing's asking price. The data includes a mixture of categorical features, continuous features, and missingness.

The stated asking price of a listing is isolated as the target of prediction.

Listings include information about vehicles that serve as continuous predictors, such as a vehicle's mileage, top speed, and wheel size. There are also categorical features, mostly indicating the presence of automotive amenities such as adaptive cruise control, entertainment systems, and lane keep systems.

Missingness is present in the data for two reasons. First, listings were collected from over one-thousand dealerships and so will have inconsistent coverage of vehicle information. Second, many features are conditional on vehicle type. For instance, information about charger levels and charger power are only relevant to electric vehicles, while the number of engine cylinders does not apply to electric vehicles and would be shown as missing rather than having a value 0. Likewise, the number of windows would be missing for motorcycles.

To prepare data for analysis, we randomly subset listings of 10,000 vehicles from the source data. The sample is stratified to include a cross-section of vehicle types. Of the features present in the data, we retain for analysis those which capture a generic characteristic of the vehicle. For instance, we include mileage as a potential predictor of asking price, but we exclude information on a vehicle's make and model from which prices can essentially be memorized by a model. Other features are removed for being direct proxies of asking price such as MSRP and Base Price.

The final data includes 35 continuous predictors. Categorical predictors include binary attributes present in vehicle listing information. We enhance these binary predictors to also include missingness indicators for each source feature used. For instance, while only electric vehicles will have observed data for a feature `vf_chargerpowerkw`, all records will have a value for the corresponding derived binary indicator feature reporting whether that feature is observed. This yields a total of 113 binary predictors.

4.9.2 Factor Mixture Models

The Factor Mixture Model (FMM) is a contribution by Muthén (2006) in unsupervised analysis of data featuring continuous and categorical indicators. Instead of making a decision between taking a categorical approach via LCA or a continuous dimensional approach via *Factor Analysis (FA)*, several hybrid methods are described allowing both approaches to be used in the same model. We apply a method closely following one examined in Clark et al. (2013)[6] which applies LCA to the categorical features to partition the data into clusters, and then applies class-conditional FA to each cluster. In our case, we will apply LCA and LCA-FSA on our binary features, followed by class-conditional sparse PCR models.

To use FMM, the practitioner must decide on the number of latent classes for LCA and the rank of the low-dimensional FA/PCA models. Clark et al. (2013) outlines an approach which increments the number of latent classes and ranks so that a variety of possible combinations is examined. To this end, we test FMMs for varying configurations and observe the resulting predictive accuracy.

4.9.3 Experimental Setup

In this experiment, we will train FMMs to predict the asking price of automotive sales listings based on vehicle characteristics.

Each FMM has an LCA or LCA-FSA component assigning records to estimated classes. Each class has its own PCA-type model performing dimension reduction, and each of those models feeds into an OLS model to make final predictions. The steps are as follows.

1. Train an LCA or LCA-FSA model on the categorical features present in the dataset. In our experiment we have 113 binary predictors to be used for this step. We also have various feature selection approaches (listed in 3.2) which can be applied with LCA-FSA.

2. Apply the trained model on the data, producing a partition of data into clusters. Apply a dimension reduction technique to the continuous features corresponding to records within each cluster. In our experiment we have 35 continuous features which will be used in dimension reduction. We also have various sparse PCA approaches we can apply, as well as varying sparsity levels we can examine.

[6] In Clark et al. (2013), the relevant model variant is referred to as FMM-4.

3. For each cluster's PCA-reduced continuous data, train an OLS model to predict the target outcome.

The final trained model is applied in the same way on new data: first, the trained LCA model is applied to the data's categorical features to assign cluster labels for the observations; second, each cluster applies its specific PCA model to its partition of continuous features; finally, each partition's reduced features are fed into that cluster's trained OLS model.

The LCA-FSA models we test use feature selection approaches (mutual information, χ^2 statistic, mRMR, diff-criterion, LL) set to select ω=25 categorical features. LCA without feature selection is also included. We test these with 2, 3, and 5 latent classes.

Dimension reduction models examined include PCA, SPCA, WPCA, ASPCA, GSPCA, and SELF. In the case of missingness, PCA and WPCA begin by 0-imputing continuous features. Because the number of source continuous features is low (35), we introduce a feature selection problem by appending continuous features with 50 additional noisy features generated as $NIID(0, 1)$. Sparse PCA methods are tested at various sparsity levels: targeting the most relevant 10, 20, or 35 features. These models are tested at ranks 2, 3, and 5.

Prior to applying models, continuous features are standardized to have observed mean 0 and variance 1. The prediction target, asking price, is log-transformed. FMMs are created by producing each possible combination of an LCA model and PCA model, and attaching an OLS model to each class-conditional PCR model. These models are trained on random splits of training and testing data at varying sample sizes. The resulting test R^2 of each model is reported as an average of 100 train/test splits.

4.9.4 FMM Results

Testing various types of LCA models attached to various PCA models produces many combinations to examine. However, the results offer some convenient ways to reduce the number of cases which need to be examined.

First, we do not examine every combination of sparse or nonsparse PCA models and LCA models. Within each data setting (n, number of latent classes, rank of PCA model), each PCA model is tested against every LCA model. In reporting results, however, we allow each candidate sparse PCA model to be attached to whatever LCA configuration produced the best average results for its corresponding PCR model at that data setting. These selections did not reveal a particular

LCA feature selection approach as the dominant one in terms of how frequently it was selected; this suggests that the choice was not as critical to model accuracy.

A second source of complexity of examining results is the choice of sparsity imposed within sparse PCA models. We test sparse PCA models at sparsity levels of 10, 20, and 35 average nonzero entries per loading matrix column. Recall that continuous predictors in this experiment include 35 continuous features measuring automotive characteristics plus the 50 noisy features we add. This produces three sparsity levels to examine for each sparse PCA method. However, the results present us with a simpler view. Across the various configurations of number of LCA classes, PCA ranks, and n, each sparse PCA approach showed a clear affinity for a single loading matrix sparsity level based on R^2. We present each sparse PCA model with its best-performing sparsity level.

First, we show baseline results from trying simpler models on the data. By 0-imputing data, we can test using OLS by itself for prediction. We also examine 0-imputed PCR using ordinary PCA and Weighted PCA, both without LCA. The results are shown in Table 4.9 and Figure 4.4, which report test R^2 over 100 train/test splits at varying sample sizes. Note that OLS results are repeated in the charts which show PCR at varying ranks.

OLS and Rank-2 PCR models show unstable behavior as a function of sample size. It is suspected that a combination of missingness issues and an assumption of homogeneous data causes issues for these models. This effect seems to alleviate for PCR models at higher rank, with WPCR outperforming PCR. The best observed test R^2 here is .243.

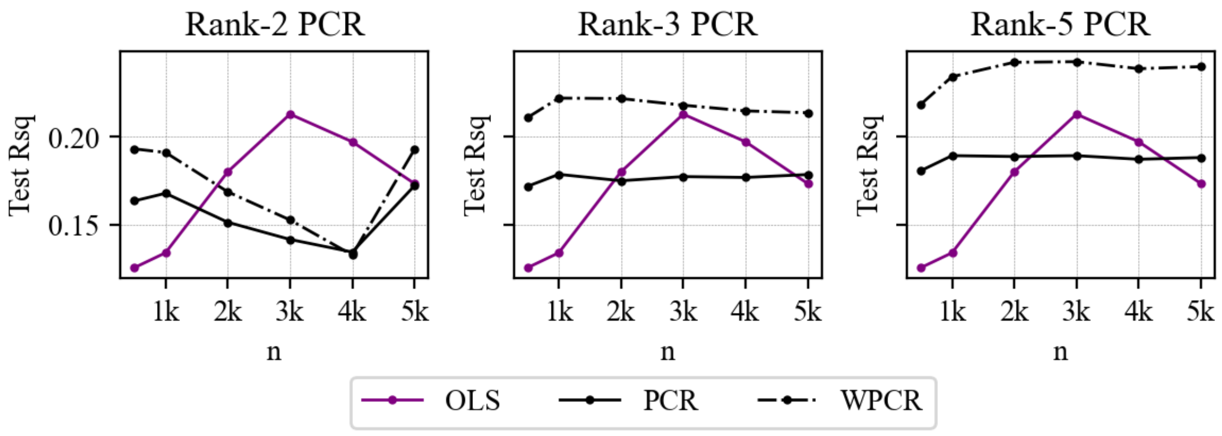


Figure 4.4: OLS and PCR Test R^2 on Automotive Data

Table 4.9: OLS, PCR, and WPCR Test R^2 on Automotive Data

n	**OLS**	**Rank-2 PCR**	**Rank-3 PCR**	**Rank-5 PCR**	**Rank-2 WPCR**	**Rank-3 WPCR**	**Rank-5 WPCR**
500	0.126	0.164	0.172	0.181	0.193	0.211	0.219
1,000	0.134	0.168	0.179	0.189	0.191	0.222	0.234
2,000	0.180	0.151	0.175	0.189	0.169	0.222	0.242
3,000	0.213	0.142	0.177	0.189	0.153	0.218	0.243
4,000	0.197	0.134	0.177	0.187	0.133	0.215	0.239
5,000	0.173	0.172	0.179	0.188	0.193	0.214	0.240

FMM models are trained at varying numbers of latent classes and PCA ranks. Test R^2 from these models are shown in Tables 4.10-4.12 and Figure 4.5. A comparison of all model types at their best settings is shown in Table 4.13 and Figure 4.6.

An immediate observation is that the incorporation of LCA as a step prior to PCA modeling produces better accuracy than PCR on its own or OLS on its own, validating the FMM premise. Overall, SELF appears to show the best performance. All models seem to struggle with the Rank-2 setting, showing better results at higher ranks. The highest average accuracy overall is achieved by the 5-class SELF model at Rank-3 with training $n = 5,000$.

Table 4.10: 2-Class FMM Test R^2 on Automotive Data

LCA Classes	PCA Rank	n	ASPCA	GSPCA	PCA	SELF	SPCA	WPCA	JSPCA
2	2	500	0.341	0.316	0.312	0.350	0.371	0.325	0.345
		1000	0.379	0.351	0.342	0.385	0.393	0.359	0.353
		2000	0.388	0.373	0.362	0.411	0.390	0.385	0.355
		3000	0.380	0.373	0.362	0.403	0.375	0.387	0.355
		4000	0.375	0.375	0.359	0.397	0.375	0.378	0.350
		5000	0.372	0.376	0.358	0.395	0.376	0.375	0.350
	3	500	0.360	0.340	0.326	0.386	0.392	0.352	0.352
		1000	0.395	0.374	0.366	0.422	0.414	0.390	0.379
		2000	0.421	0.405	0.397	0.445	0.423	0.418	0.388
		3000	0.424	0.414	0.407	0.444	0.420	0.427	0.386
		4000	0.426	0.418	0.411	0.441	0.423	0.427	0.383
		5000	0.427	0.421	0.415	0.440	0.424	0.427	0.384
	5	500	0.370	0.354	0.341	0.405	0.397	0.371	0.377
		1000	0.404	0.386	0.375	0.430	0.423	0.406	0.403
		2000	0.427	0.414	0.403	0.454	0.433	0.431	0.419
		3000	0.431	0.420	0.413	0.456	0.432	0.433	0.420
		4000	0.433	0.424	0.416	0.456	0.433	0.433	0.418
		5000	0.434	0.425	0.419	0.457	0.434	0.434	0.420

Table 4.11: 3-Class FMM Test R^2 on Automotive Data

LCA Classes	**PCA Rank**	**n**	**ASPCA**	**GSPCA**	**PCA**	**SELF**	**SPCA**	**WPCA**	**JSPCA**
3	**2**	**500**	0.349	0.322	0.312	0.372	0.375	0.330	0.340
		1000	0.384	0.355	0.348	0.397	0.401	0.364	0.355
		2000	0.397	0.381	0.371	0.416	0.399	0.391	0.358
		3000	0.390	0.382	0.371	0.410	0.388	0.393	0.349
		4000	0.387	0.385	0.371	0.408	0.389	0.389	0.346
		5000	0.383	0.385	0.369	0.405	0.387	0.385	0.346
	3	**500**	0.365	0.343	0.331	0.399	0.394	0.357	0.359
		1000	0.400	0.381	0.372	0.433	0.422	0.396	0.384
		2000	0.429	0.411	0.404	0.457	0.430	0.428	0.396
		3000	0.433	0.422	0.415	0.461	0.432	0.435	0.391
		4000	0.436	0.426	0.419	0.458	0.433	0.435	0.391
		5000	0.440	0.431	0.424	0.464	0.436	0.437	0.388
	5	**500**	0.374	0.355	0.344	0.418	0.398	0.376	0.387
		1000	0.408	0.391	0.382	0.441	0.425	0.414	0.414
		2000	0.435	0.421	0.410	0.465	0.438	0.438	0.433
		3000	0.439	0.428	0.420	0.465	0.439	0.442	0.428
		4000	0.441	0.431	0.423	0.462	0.440	0.441	0.430
		5000	0.442	0.432	0.426	0.462	0.439	0.443	0.429

Table 4.12: 5-Class FMM Test R^2 on Automotive Data

LCA Classes	**PCA Rank**	**n**	**ASPCA**	**GSPCA**	**PCA**	**SELF**	**SPCA**	**WPCA**	**JSPCA**
5	**2**	500	0.361	0.335	0.320	0.379	0.380	0.340	0.350
		1000	0.397	0.374	0.366	0.416	0.413	0.379	0.357
		2000	0.418	0.403	0.396	0.440	0.425	0.407	0.363
		3000	0.416	0.408	0.399	0.438	0.416	0.411	0.359
		4000	0.417	0.414	0.403	0.440	0.421	0.416	0.355
		5000	0.419	0.416	0.406	0.437	0.419	0.414	0.354
	3	500	0.374	0.352	0.338	0.407	0.393	0.363	0.375
		1000	0.410	0.394	0.384	0.446	0.431	0.406	0.398
		2000	0.442	0.425	0.416	0.476	0.450	0.439	0.409
		3000	0.448	0.436	0.427	0.480	0.451	0.446	0.411
		4000	0.455	0.444	0.434	0.482	0.457	0.451	0.401
		5000	0.456	0.445	0.437	0.477	0.456	0.452	0.404
	5	500	0.379	0.351	0.342	0.407	0.396	0.371	0.386
		1000	0.415	0.397	0.391	0.448	0.431	0.423	0.425
		2000	0.444	0.430	0.420	0.474	0.453	0.449	0.448
		3000	0.450	0.437	0.429	0.477	0.453	0.454	0.448
		4000	0.456	0.445	0.436	0.480	0.458	0.457	0.450
		5000	0.456	0.446	0.439	0.476	0.457	0.463	0.454

Table 4.13: Test R^2 of All Models at Best Settings on Automotive Data

				Test Rsq					
				n					
	PCA Rank	**LCA Classes**	**PCA Sparsity**[7]	**500**	**1000**	**2000**	**3000**	**4000**	**5000**
OLS				0.126	0.134	0.180	0.213	0.197	0.173
PCR - PCA	**5**			0.181	0.189	0.189	0.189	0.187	0.188
PCR - WPCA	**5**			0.219	0.234	0.242	0.243	0.239	0.240
FMM - PCA	**5**	**5**		0.342	0.391	0.420	0.429	0.436	0.439
FMM - WPCA	**5**	**5**		0.371	0.423	0.449	0.454	0.457	0.463
FMM - JSPCA	**5**	**5**		0.386	0.425	0.448	0.448	0.450	0.454
FMM - ASPCA	**5**	**5**	**10**	0.379	0.415	0.444	0.450	0.456	0.456
FMM - SPCA	**5**	**5**	**10**	0.396	0.431	0.453	0.453	0.458	0.457
FMM - GSPCA	**5**	**5**	**20**	0.351	0.397	0.430	0.437	0.445	0.446
FMM - SELF	**3**	**5**	**20**	0.407	0.446	0.476	0.480	0.482	0.477

Table 4.14 shows the non-missing rates for select continuous predictors used by PCA in a 3-class FMM model which showed high performance. It is observed that for at least certain features, the assignment of records to classes appears to organize data by what features are observed. It is presumed that Class 1 data, which shows higher missingness rates for the select predictors, will have a PCA model that has the opportunity to emphasize other features.

[7] *PCA Sparsity* denotes the average number of nonzero entries per estimated loading matrix column out of the number of continuous features in the data, p=35+50

Table 4.14: Class-Conditional Non-missing Rates of Select Predictors in Sample FMM Model

Predictor	**Overall Non-missing Rate**	**Class 1 Non-missing Rate**	**Class 2 Non-missing Rate**	**Class 3 Non-missing Rate**
`wheels`	36.3%	1.4%	99.8%	97.1%
`seatrows`	31.7%	0.0%	95.1%	81.6%
`seats`	34.0%	0.9%	92.4%	92.9%
`wheelbaseshort`	41.8%	14.0%	96.9%	86.1%
`wheelsizerear`	28.4%	0.8%	81.7%	73.5%
`wheelsizefront`	28.5%	0.9%	81.7%	73.5%
`transmissionspeeds`	20.5%	4.4%	44.4%	53.7%

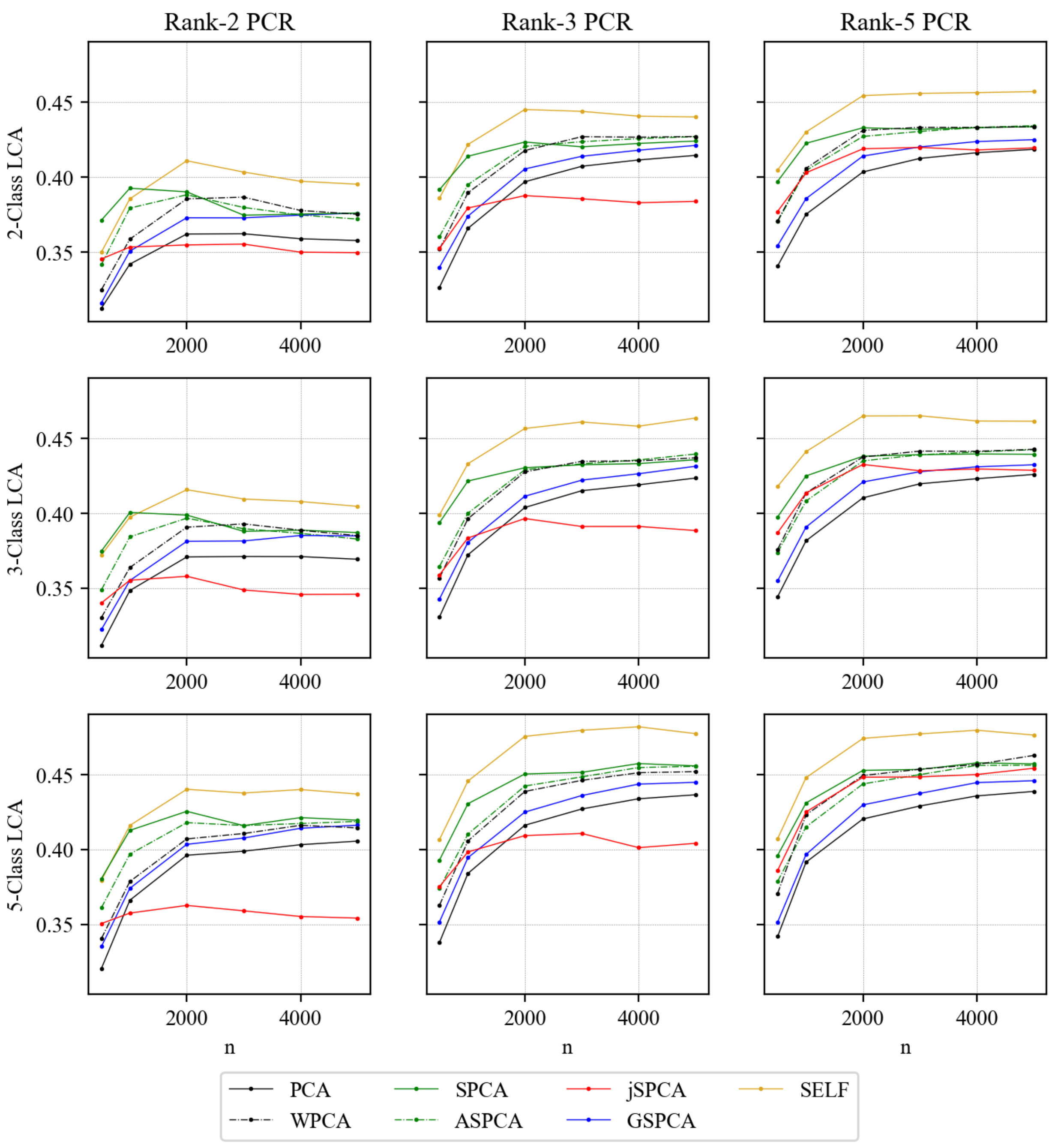


Figure 4.5: FMM Test R^2 on Automotive Data

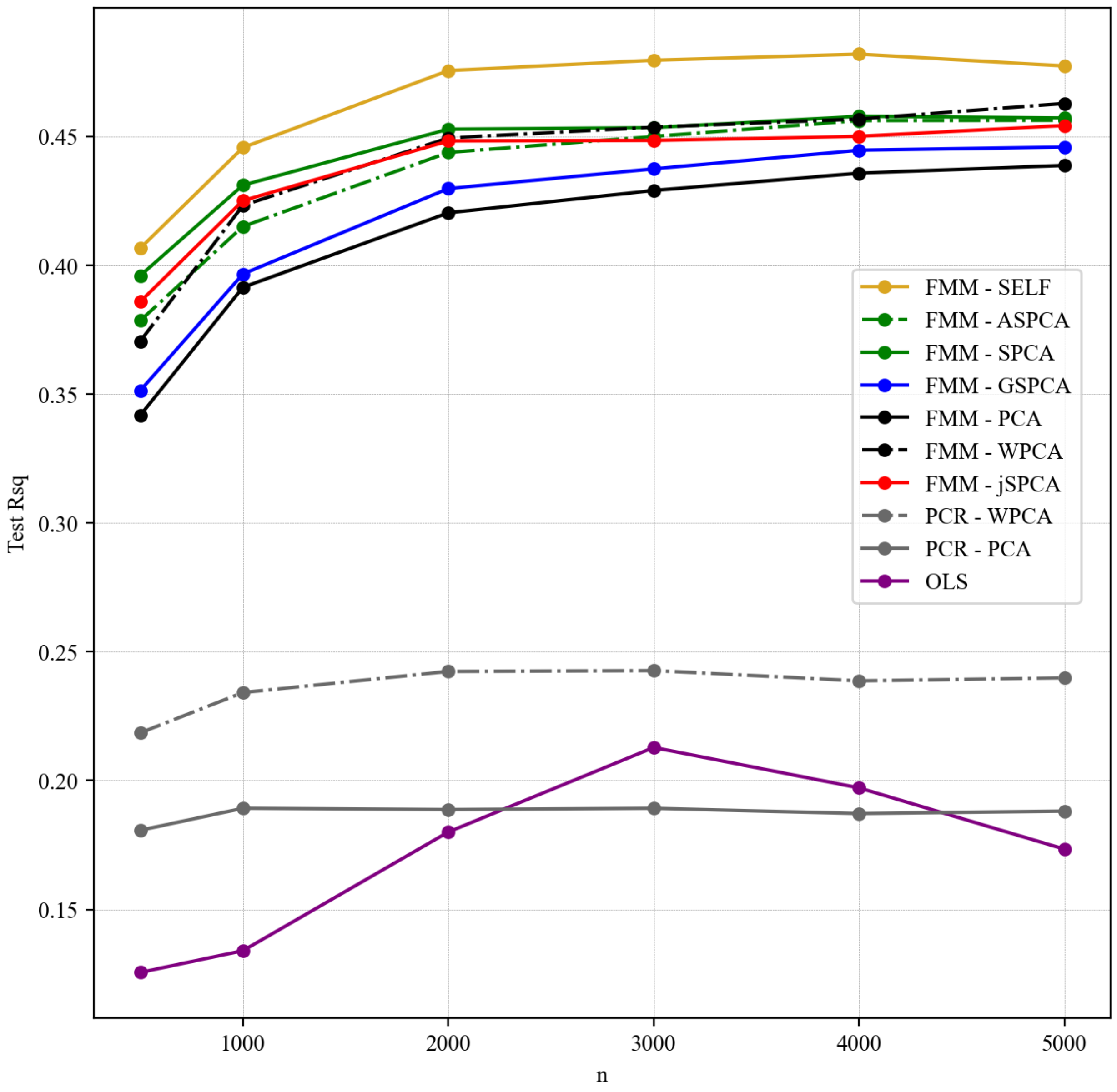


Figure 4.6: Test R^2 of All Models at Best Settings on Automotive Data

4.10 SELF Ablation Study

In this section we examine the motivation and performance impact of various sparse PCA enhancements tested with the SELF model. The following discussion concerns the estimation of matrices $\mathbf{\Gamma}$ and $\mathbf{A}$ in the service of modeling data matrix $\mathbf{X}$ based on the SELF Objective (4.5).

4.10.1 SELF with Imputation

We first show a formulation of SELF using the imputation method of 4.3. In this approach, missing entries of a row are estimated as $\mathbf{w}$. Applying a placement function $f(\mathbf{w})$ produces a vector of the same size as a data row. $f(\mathbf{w})$ takes value 0 where row $\mathbf{x}$ is observed and has an imputed estimate in positions corresponding to $\mathbf{x}$'s missing values. This suggests loss $l = \|\mathbf{x}^0 + f(\mathbf{w}) - \hat{\mathbf{x}}\|_2^2$ where $\hat{\mathbf{x}} = \boldsymbol{\gamma}\mathbf{A}'$. Holding $\mathbf{A}$, we optimize over $\mathbf{\Gamma}$ and $\mathbf{w}$.

For individual row $\mathbf{x}$ of $\mathbf{X}$, consider corresponding latent estimate $\boldsymbol{\gamma}$ and imputed value vector w.

Proposition 12. *The row loss function*

$$l = \|\mathbf{x}^0 + f(\mathbf{w}) - \boldsymbol{\gamma}\mathbf{A}'\|_2^2 \tag{4.9}$$

with data row $\mathbf{x}$ *and corresponding missingness placement matrix* $\mathbf{P}$ *can be minimized w.r.t* $(\boldsymbol{\gamma}, \mathbf{w})$ *by*

$$\begin{bmatrix} \gamma & \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^0\mathbf{A} & \mathbf{x}^0\mathbf{P}' \end{bmatrix} \begin{bmatrix} \mathbf{A}'\mathbf{A} & \mathbf{A}'\mathbf{P}' \\ -\mathbf{PA} & \mathbf{PP}' \end{bmatrix}^{-1}. \tag{4.10}$$

Proof.

$$\begin{aligned}
l &= \|\mathbf{x}^0 + f(\mathbf{w}) - \boldsymbol{\gamma}\mathbf{A}'\|_2^2 \\
&= (\mathbf{x}^0 + f(\mathbf{w}) - \boldsymbol{\gamma}\mathbf{A}')(\mathbf{x}^0 + f(\mathbf{w}) - \boldsymbol{\gamma}\mathbf{A}')' \\
&= \mathbf{x}^0\mathbf{x}^{0\prime} + 2\mathbf{x}^0 f(\mathbf{w})' - 2\mathbf{x}^0\mathbf{A}\boldsymbol{\gamma}' - 2f(\mathbf{w})\mathbf{A}\boldsymbol{\gamma}' + f(\mathbf{w})f(\mathbf{w})' + \boldsymbol{\gamma}\mathbf{A}'\mathbf{A}\boldsymbol{\gamma}' \\
\frac{\partial l}{\partial \boldsymbol{\gamma}} &= 0 \implies \boldsymbol{\gamma}\mathbf{A}'\mathbf{A} - f(\mathbf{w})\mathbf{A} = \mathbf{x}^0\mathbf{A} \implies \boldsymbol{\gamma}\mathbf{A}'\mathbf{A} - \mathbf{wPA} = \mathbf{x}^0\mathbf{A} \\
\frac{\partial l}{\partial \mathbf{w}} &= 0 \implies \boldsymbol{\gamma}\mathbf{A}'\mathbf{P}' - \mathbf{wPP}' = \mathbf{x}^0\mathbf{P}',
\end{aligned}$$

where $\mathbf{P}$ is the appropriate placement matrix for $\mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{wP}$, as constructed in Algorithm 8.

$\boldsymbol{\gamma}$ and $\mathbf{w}$ then form a system:

$$\boldsymbol{\gamma}(\mathbf{A}'\mathbf{A}) + \mathbf{w}(-\mathbf{PA}) = \mathbf{x}^0\mathbf{A}$$
$$\boldsymbol{\gamma}(\mathbf{A}'\mathbf{P}') + \mathbf{w}(\mathbf{PP}') = \mathbf{x}^0\mathbf{P}'.$$

Or in block matrix notation,

$$\begin{bmatrix} \boldsymbol{\gamma} & \mathbf{w} \end{bmatrix} \begin{bmatrix} \mathbf{A}'\mathbf{A} & \mathbf{A}'\mathbf{P}' \\ -\mathbf{PA} & \mathbf{PP}' \end{bmatrix} = \begin{bmatrix} \mathbf{x}^0\mathbf{A} & \mathbf{x}^0\mathbf{P}' \end{bmatrix}$$

yielding

$$\begin{bmatrix} \boldsymbol{\gamma} & \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^0\mathbf{A} & \mathbf{x}^0\mathbf{P}' \end{bmatrix} \begin{bmatrix} \mathbf{A}'\mathbf{A} & \mathbf{A}'\mathbf{P}' \\ -\mathbf{PA} & \mathbf{PP}' \end{bmatrix}^{-1}. \tag{4.11}$$

□

We can thus estimate the latent representation $\boldsymbol{\gamma}$ and missing entries $\mathbf{w}$ simultaneously for each row of $\mathbf{X}$.

With estimated imputed values for a data matrix, and estimated $\boldsymbol{\Gamma}$, $\mathbf{A}$ can be estimated per Proposition 9.

Per Remark 5, we add a ridge parameter to alleviate matrix inversion instability:

$$\begin{bmatrix} \hat{\boldsymbol{\gamma}} & \hat{\mathbf{w}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^0\mathbf{A} & \mathbf{x}^0\mathbf{P}' \end{bmatrix} (\begin{bmatrix} \mathbf{A}'\mathbf{A} & \mathbf{A}'\mathbf{P}' \\ -\mathbf{PA} & \mathbf{PP}' \end{bmatrix} + \lambda_r \mathbf{I})^{-1}. \tag{4.12}$$

The method is presented in Algorithm 15.

Algorithm 15 Sparse Estimation of Latent Factors with Imputation

```
Input:
  • $\mathcal{X}$, $n$-by-$p$ data matrix
  • $k$, desired number of sparse principal components
  • $epochs$, number of epochs
  • $\lambda_r$, ridge parameter to stabilize matrix inversion
  • $m^e$, FSA annealing schedule specifying the number of nonzero entries of $\mathbf{A}$ kept at epoch $e$
Output: Estimated latent factors $\mathbf{\Gamma}$ of $\mathcal{X}$ and estimated loading matrix $\mathbf{A}$
 1: Set $\mathbf{X} = \mathcal{X}^0$
 2: Initialize $\mathbf{\Gamma}$ as the first $k$ principal components and $\mathbf{A}$ as the first $k$ loading
      vectors from PCA on $\mathbf{X}$
 3: for $e = 1$ to $epochs$ do
 4:     for each row $\mathbf{x}$ of $\mathbf{X}$ do
 5:         if data row is fully-observed then
 6:             Update corresponding row of $\mathbf{\Gamma}$ with $\boldsymbol{\gamma} = \mathbf{x}'\mathbf{A}(\mathbf{A}'\mathbf{A})^{-1}$
 7:         else if data row has missingness then
 8:             Set $\mathbf{P}$ as appropriate placement matrix for row $\mathbf{x}$ as in Algorithm 8
 9:             $\begin{bmatrix}\boldsymbol{\gamma} & \mathbf{w}\end{bmatrix} = \begin{bmatrix}\mathbf{x}^0\mathbf{A} & \mathbf{x}^0\mathbf{P}'\end{bmatrix} \left( \begin{bmatrix}\mathbf{A}'\mathbf{A} & \mathbf{A}'\mathbf{P}' \\ -\mathbf{P}\mathbf{A} & \mathbf{P}\mathbf{P}'\end{bmatrix} + \lambda_r \mathbf{I}\right)^{-1}$
10:             Update corresponding row of $\mathbf{\Gamma}$ with $\boldsymbol{\gamma}$
11:             Update corresponding row of $\mathbf{X}$ with $\mathbf{x}^0 + \mathbf{w}\mathbf{P}$
12:         end if
13:     end for
14:     $\mathbf{A} = \mathbf{U}\mathbf{V}'$ with $\mathbf{X}'\mathbf{\Gamma} = \mathbf{U}\mathbf{D}\mathbf{V}'$ by SVD
15:     Apply FSA to $\mathbf{A}$
16:          $m^e$ is the desired number of nonzero elements of $\mathbf{A}$ for current epoch
17:          Retain the top $m^e$ entries of $\mathbf{A}$ with the highest absolute values, setting the
                 remaining entries to 0
18: end for
19: $\mathbf{\Gamma}\mathbf{A}'$ is the low-rank estimate of $\mathcal{X}$.
```

4.10.2 Simulations on SELF Variants

Here we report performance of components of the SELF model using the simulations setup of 4.7. These simulations use sparse PCA models in a Principal Component Regression setting. Thus, we score model performance based on accuracy of feature selection and accuracy of regression.

Variations of SELF models are abbreviated:

- S-I: SELF with Imputation (4.10.1)

- S-IA: SELF with Imputation and **A** condition filter (4.5.1)
- S-NA: SELF with Non-imputation (4.5.1) and **A** condition filter
- S-NAW: SELF with Non-imputation, **A** condition filter, and weighted estimation (4.5.2)
- S-NAWFS: SELF with Non-imputation, **A** condition filter, weighted estimation, and weighted feature selection (4.5.3)

Table 4.15: Number of Features Correctly Selected by SELF PCR Models with varying n, q

n	**q**	**S-I**	**S-IA**	**S-NA**	**S-NAW**	**S-NAWFS**
100	**10**	8.9	8.8	8.3	9.2	9.3
100	**50**	7.8	8.0	7.5	8.6	8.9
100	**100**	7.6	7.5	7.2	7.7	8.3
500	**10**	9.4	9.4	8.9	9.8	9.8
500	**50**	8.6	8.5	8.7	9.7	9.7
500	**100**	8.4	8.4	8.6	9.6	9.7
1,000	**10**	9.5	9.6	9.0	9.9	9.9
1,000	**50**	8.8	8.5	9.0	9.8	9.8
1,000	**100**	8.5	8.6	8.9	9.8	9.8
2,000	**10**	9.6	9.6	9.1	9.9	9.9
2,000	**50**	8.9	8.7	9.1	9.8	9.9
2,000	**100**	8.3	8.3	9.1	9.8	9.8
5,000	**10**	9.6	9.6	9.1	9.9	9.9
5,000	**50**	8.8	8.8	9.2	9.9	9.9
5,000	**100**	8.6	8.4	9.2	9.9	9.9
10,000	**10**	9.7	9.8	9.2	10.0	9.9
10,000	**50**	8.9	9.0	9.2	10.0	9.9
10,000	**100**	8.2	8.2	9.2	10.0	9.9

Table 4.16: Test R^2 of SELF PCR Models with varying n, q

n	**q**	**PCA**	**S-I**	**S-IA**	**S-NA**	**S-NAW**	**S-NAWFS**
100	**0**	0.59	0.20	0.22	0.86	0.88	0.88
100	**10**	0.58	0.21	0.21	0.68	0.81	0.83
100	**50**	0.56	0.15	0.16	0.62	0.76	0.80
100	**100**	0.54	0.13	0.12	0.61	0.72	0.77
500	**0**	0.61	0.30	0.27	0.86	0.89	0.89
500	**10**	0.61	0.28	0.26	0.73	0.86	0.87
500	**50**	0.61	0.26	0.27	0.71	0.85	0.86
500	**100**	0.60	0.26	0.25	0.70	0.84	0.86
1,000	**0**	0.62	0.32	0.29	0.86	0.89	0.89
1,000	**10**	0.61	0.27	0.29	0.72	0.86	0.88
1,000	**50**	0.61	0.29	0.27	0.72	0.85	0.87
1,000	**100**	0.61	0.30	0.28	0.72	0.85	0.87
2,000	**0**	0.62	0.31	0.30	0.86	0.89	0.89
2,000	**10**	0.62	0.29	0.31	0.73	0.87	0.88
2,000	**50**	0.62	0.29	0.28	0.73	0.86	0.87
2,000	**100**	0.62	0.29	0.28	0.73	0.86	0.87
5,000	**0**	0.62	0.31	0.31	0.86	0.89	0.89
5,000	**10**	0.62	0.30	0.29	0.73	0.88	0.88
5,000	**50**	0.62	0.30	0.29	0.73	0.87	0.88
5,000	**100**	0.62	0.30	0.31	0.73	0.87	0.88
10,000	**0**	0.62	0.31	0.32	0.86	0.89	0.89
10,000	**10**	0.62	0.31	0.30	0.73	0.88	0.89
10,000	**50**	0.62	0.31	0.35	0.73	0.87	0.88
10,000	**100**	0.62	0.30	0.29	0.73	0.87	0.88

4.10.3 Postsecondary Institutions Characteristics

Here we use the PCR setup of (4.8) to compare the performance of SELF variants combined with OLS. As in that section, the data is split into 70% training and 30% test data. SELF models are trained at rank 2. Sparsity is imposed on loading matrices at a rate of 30 nonsparse entries per loading matrix column on average. R^2 is reported over 100 train/test splits.

SELF variants are abbreviated:

- S-I: SELF with Imputation (4.10.1)
- S-IA: SELF with Imputation and **A** condition filter (4.5.1)
- S-NA: SELF with Non-imputation (4.5.1) and **A** condition filter
- S-NAW: SELF with Non-imputation, **A** condition filter, and weighted estimation (4.5.2)
- S-NAWFS: SELF with Non-imputation, **A** condition filter, weighted estimation, and weighted feature selection (4.5.3)

Result show that imputation models struggle with this data, with a large jump in accuracy coming from the non-imputation strategy. Beyond that, weights and weighted feature selection offer modest performance gains.

Table 4.17: SELF Variant Performance on Postsecondary Institutions Data

PCR Model	**Training** R^2	**Test** R^2
S-I	0.101	0.092
S-IA	0.102	0.092
S-NA	0.591	0.573
S-NAW	0.593	0.576
S-NAWFS	0.602	0.585

4.10.4 Automotive Dataset

We examine the performance of SELF variants in the factor mixture setup of (4.9). Following the prior setup, a number of LCA models are applied to binary indicators from that data. Class-conditional SELF models are trained on continuous features of the resulting clusters of these LCA models, with low-rank representations feeding into final OLS models predicting the target outcome. Test R^2 is observed on the target outcome, the log asking price of vehicles.

LCA models are trained at 5 latent classes, with and without feature selection at $\omega = 25$. SELF variants are trained at rank 3 and a sparsity level yielding an average of 20 nonzero entries per loading matrix column. These are the settings which showed the best results for the full SELF model in Table 4.13; here we apply those settings to all SELF variants. As before, each variant is paired with the LCA setup which produced the best overall results. Average test R^2 is reported over 100 train/test splits. SELF variant names are abbreviated as in prior ablation sections.

As previously observed, the major performance difference comes from the using the non-imputation strategy over imputation. The application of weights in estimation appears to help, while weighted feature selection show some performance increases except for the lowest case of n.

Table 4.18: SELF Variant Performance on Automotive Data

n	**S-I**	**S-IA**	**S-NA**	**S-NAW**	**S-NAWFS**
500	0.162	0.171	0.398	0.419	0.407
1,000	0.186	0.185	0.432	0.445	0.446
2,000	0.196	0.193	0.458	0.471	0.476
3,000	0.188	0.203	0.465	0.474	0.480
4,000	0.196	0.200	0.462	0.477	0.482
5,000	0.215	0.212	0.468	0.473	0.477

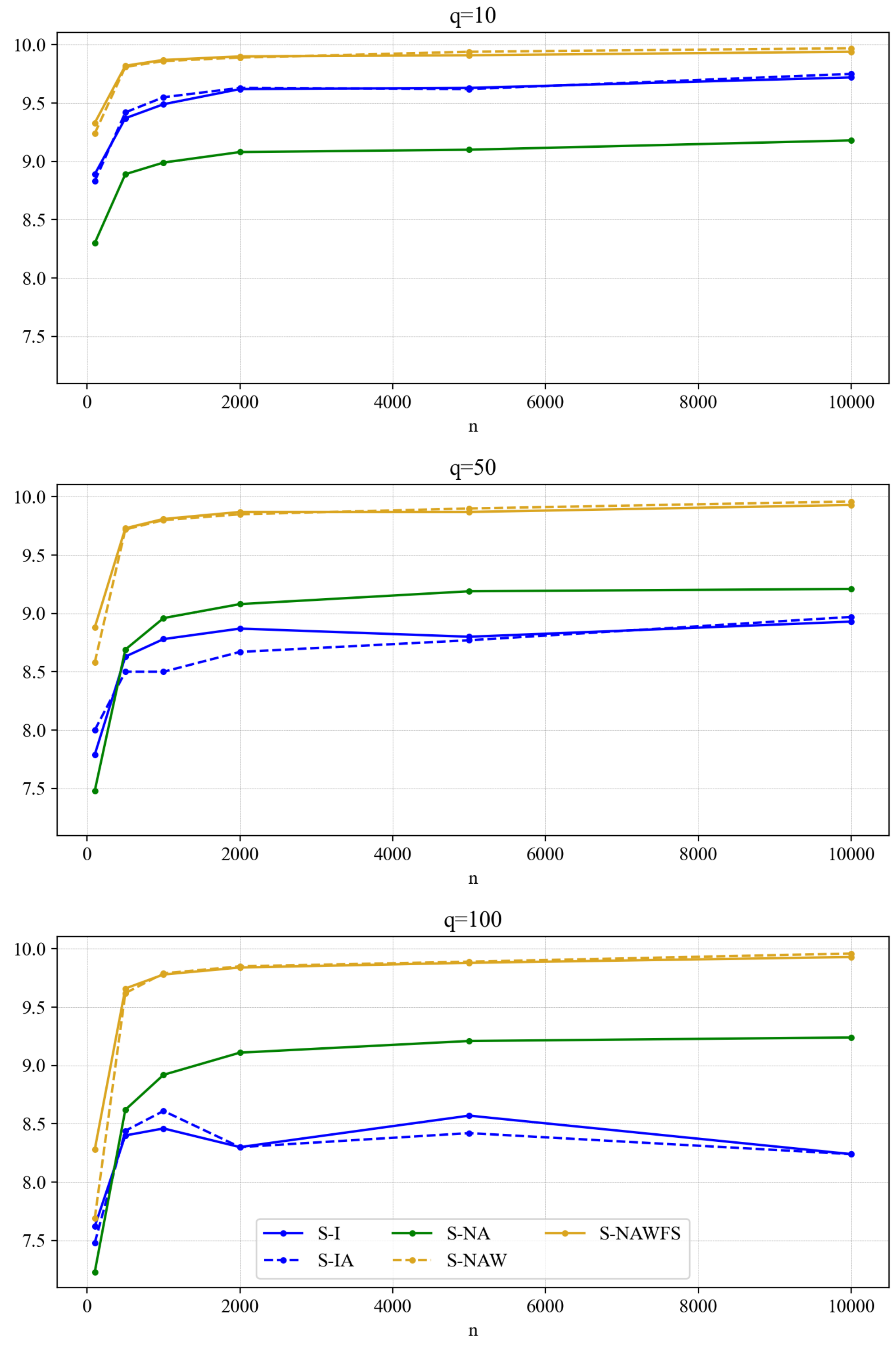


Figure 4.7: Number of Features Correctly Selected by SELF PCR Models with varying n, q

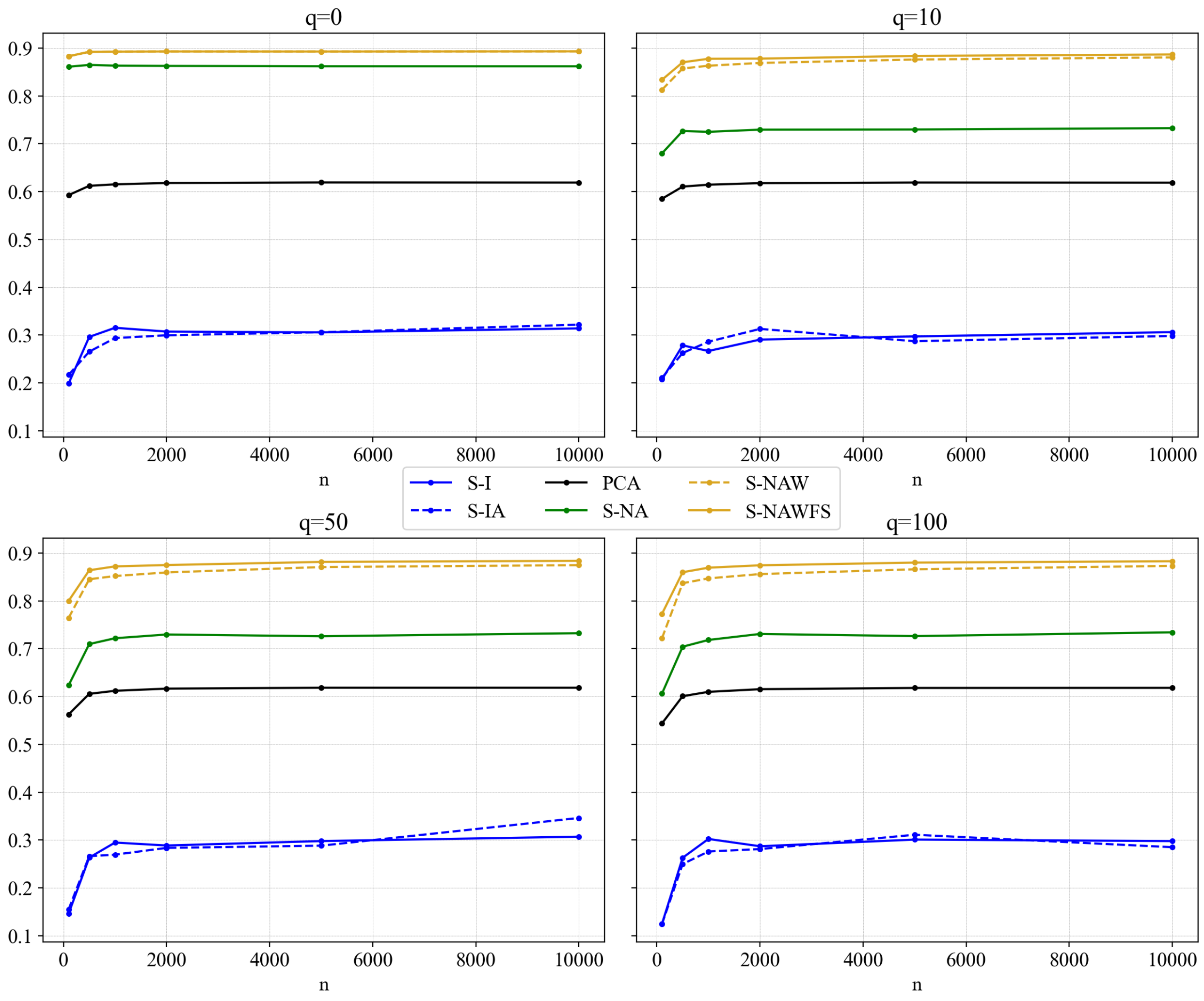


Figure 4.8: Test R^2 of SELF PCR Models with varying n, q

CHAPTER 5

CONCLUSION

Here we will summarize the contributions and findings of this work.

5.1 Feature Selection for Latent Class Analysis

We began with a review of the brief history of feature selection for unsupervised clustering of categorical data, noting the absence of methods examined in high-dimensional settings. We introduce LCA-FSA, an estimation framework where feature relevance metrics can be used to perform feature selection in a hybrid nature alongside model estimation rather than as a pre-processing or post-processing step. The performance of a number of feature relevance measures is tested, including some familiar approaches like mutual information, a measure using the chi-square statistic for association, and mRMR. We also examine a recent contribution, the diff-criterion, originally introduced by Javed et al. (2010) as a fast screening step for feature selection in binary classification problems. Finally, we examine a likelihood-based approach for measuring feature importance, and an existing forward selection procedure for feature selection in LCA.

5.1.1 LCA Simulations

These LCA feature selection methods were tested against simulated data, allowing us to compare their performance in terms of feature selection and latent class recovery. Results show that in the simplest setting (2-class, 10 noisy features), all methods are capable of recovering relevant features to some extent. Further, all models- including LCA with no feature selection- are able to produce satisfactory recovery of latent classes in this setting, except mRMR. As problem complexity increases with a higher number of latent classes and more noisy features, the forward selection procedure takes a major hit in feature selection and clustering accuracy. This merely confirms what is already stated in the literature: stepwise methods are not ideal for high-dimensional settings.

However, increasing problem complexity affects the accuracy of *all* models, and we observe that some approaches are more robust to this complexity than others. After forward selection, models using mRMR, mutual information, and LCA without feature selection suffer the largest relative performance decreases.

Not all simulated settings showed a clear feature selection approach dominating others in performance. However the diff-criterion was always in the group of top performers, and in other cases was the identifiable top performer. We find this to be a satisfying result due to diff-criterion's intuitive and computationally simple approach.

5.1.2 LCA Experiments

LCA-FSA methods were tested on real data, which sometimes showed a different story than simulations.

An unsupervised problem attempting to recover school types (elementary, middle, or high school) used p=2,586 binary features indicating what classes had enrollment at each of p=2,909 Florida schools. Intuition suggests that the data should have a natural class-conditional distribution, leading to the unsurprising result that LCA-FSA models are able to recover underlying classes with high accuracy. However, mRMR is shown to be the standout model- the reverse of our findings in simulations, with chi-square as a close second. Diff-criterion, mutual information, and the likelihood method show an apparent critical point, where model performance at $\omega \leq 30$ is considerably worse than $\omega \geq 50$; this weakness is not shared by mRMR and chi-square. Note that the $\omega = 50$ level of feature selection means a selection of 50 features to use out of the original 2,586. At that level of ω and higher, LCA-FSA with any feature relevance metric is seen to outperform LCA without feature selection.

Another unsupervised 3-class experiment is tested with bag of words text data having p=27,332 and n=1,903, though the sample size available for model training is less due to splitting data into training and testing sets. In this setting, diff-criterion returns as the dominant model. A striking finding shows that any amount of moderate-to-extreme feature selection ($50 \leq \omega \leq 5,000$) outperforms LCA without feature selection, with the aggressive $\omega = 500$ setting showing the best results for all feature relevance metrics. At this setting, diff-criterion showed an average test Adjusted Rand Index of .76, compared to nonsparse LCA's .31.

Overall it is shown that LCA-FSA is not only a viable technique for training sparse models, but can be crucial to the accuracy of latent class models on high-dimensional data. Further, the parsimony enabled by feature selection models in these two experimental settings permitted readily-interpretable model parameters, which would otherwise be a challenge given such high p. It is observed that the choice of best feature selection metric may be dataset-dependent, and that it is important to test such models at varying levels of ω.

5.2 Sparse PCA

We also examined sparse PCA models, including the well-known SPCA (Zou et al., 2006), the recently introduced JSPCA (Yi et al., 2017), and introduced some new approaches, with the goal of handling missing data. The ASPCA model is developed to follow SPCA but to use FSA for to impose sparsity rather than the usual elastic net. GSPCA applies FSA to its loading matrix elementwise, rather than by the usual sequential training approach for sparse PCA methods. The SELF model deviates from the common setup inspired by SPCA with $\mathbf{X} \approx \mathbf{XBA}'$, and instead directly estimates latent data representation $\mathbf{\Gamma}$ and its corresponding loading matrix, with $\mathbf{X} \approx \mathbf{\Gamma A}'$, more in the style of SRRR-SPCA (2.6).

We adapt the above sparse PCA models to handle missing data. Two missingness strategies are developed, tested and compared: using low-rank structure to impute data allowing estimation to continue on full data, and a relaxed estimation approach that mimics SVD but avoids missing values. The former approach is common to the domain of matrix completion problems, while the general tactic of modifying estimation to circumvent missing entries is also a common approach.

A number of additional sparse PCA modeling tactics are developed and examined. We modify PCA and SELF to incorporate optional feature weights with the goal of downweighting features which the model estimates to have a greater noise component. An additional missingness mitigation strategy is tested with SELF, where rows of data are excluded from calculations if their missingness pattern is shown to yield an unstable estimation problem.

5.2.1 Sparse PCA Simulations

Data is simulated to test sparse PCA models in a PCR setting with missingness (50%) and irrelevant features. In these simulations, SELF with non-imputation is shown to perform well. Other models show worse performance, decreasing in average test R^2 with increasing n. PCA and WPCA, which are tested with 0-imputation as their missingness strategy, are immune to this effect. Therefore, it is suspected that the imputation strategy may be prone to instability, even with a basic attempt at mitigating these issues with a ridge parameter.

The imputation and non-imputation methods are more directly compared in an ablation study where variants of the SELF model and its optional components are individually tested. In our PCR simulations setup, it is readily observed that SELF with the imputation strategy is vastly outperformed by 0-imputed nonsparse PCA in average test R^2. SELF estimated with non-imputation, however, outperforms PCA. The incorporation of weights in non-imputed SELF estimation further increases average predictive accuracy, particularly in the case of data simulated with noisy features.

It is concluded that a dominant factor in our simulation results is the missingness strategy. In our initial testing, imputation by low-rank completion was found to be effective at lower missingness rates. The accuracy of the imputation strategy is also seen in the low-n setting in Figure 4.2. The stability of naive 0-imputed PCA results relative to imputation methods in our simulations demonstrates that in certain missingness settings, imputing values by estimation from surrounding data values may be more of a hindrance than an enhancement.

There are undoubtedly robustification approaches that can be applied to our imputation strategy to resolve the issues we observed. However, we conclude our examination of missingness strategies by highlighting non-imputation, as implemented in the SELF model, as a viable alternative. Further, it is observed that the incorporation of weights, as in SELF and WPCA, can improve unsupervised dimension reduction in the face of irrelevant features.

5.2.2 Sparse PCA Experiments

A regression dataset is compiled which seeks to predict the endowments of institutions of higher education. The predictors include $p = 315$ continuous measures of various institutional characteristics. Because the predictors feature many overlapping measures, it is hypothesized that they can be accurately estimated by dimension reduction; because there is missingness (44%) due to inconsistent reporting across schools, it is further hypothesized that dimension reduction may be *necessary* to accurately model the data. Finally, due to the predictors capturing a vast array of institutional traits, it is presumed that sparse PCA will enable us to ignore the majority of our predictors.

Our suspicions are confirmed in average test R^2 results, showing that 0-imputed OLS (training $R^2 = .79$, test $R^2 = .38$) is readily outperformed by 0-imputed PCR using PCA (training $R^2 = .56$, test $R^2 = .55$). Further, PCR using sparse PCA methods SPCA, ASPCA, GSPCA, and SELF show higher performance, with test R^2 in the range of .58-.59.

The Large Car Dataset presents us with the task of predicting the listed asking price of a car based on its recorded characteristics. The data features similar characteristics to the postsecondary institution data: continuous predictors covering a variety of unrelated traits (including our addition of noisy features), missingness, and our hypothesis of a low-rank structure. However, due to different vehicle types, the data allows us to test factor mixture models (Muthén, 2006) where LCA with an unknown number of classes is to be paired with class-conditional PCA with an unknown rank. The method suggested by Muthén follows a sequential examination of each combination of LCA model and PCA model. In our case, we combine LCA models of varying number of latent classes and feature selection approaches with PCR models with varying ranks, sparsity, and missingness approaches.

The results, reported as average test R^2 values, first validate the FMM setup by showing that class-conditional PCR models working alongside LCA easily outperform PCR or OLS without LCA. Further, at varying choices of number of LCA classes and number of sparse PCA model ranks, all LCA models paired with sparse PCA models outperform LCA paired with PCA, with the sole exception of JSPCA. The FMM model using 0-imputed weighted PCA, though not sparse, also outperforms the FMM with ordinary 0-imputed PCA. For the majority of FMM settings, SELF is the dominant performer.

Similar to the LCA experiment setting, the feature selection employed by FSA-LCA models ($\omega = 25$) enabled easier interpretation of parameter estimates relative to nonsparse LCA whose parameters would reflect all of the 113 binary features in the original data.

5.3 Closing Remarks

Our examination of latent class analysis and principal component analysis covers some territory that has seen much attention, such as sparse estimation of PCA, as well as less-examined topics, such as unsupervised feature selection for binary clustering. Analysis of continuous data by PCA was exacerbated by the presence of irrelevant features, missingness, in some cases high-dimensionality, and in one case an underlying latent class structure. LCA data also featured irrelevant features and high dimensionality. For each category of data challenges faced, we presented multiple competing strategies. These strategies included known methods as well as new contributions.

We hope the outcomes of our analysis are found to be interesting and useful. As we observe the rapid and exciting development of powerful AI models with unprecedented size and complexity, classical methods like PCA and LCA continue to be cornerstones of quantitative methodology for many fields of study. Model interpretability is particularly essential to many scientific settings, where analysis results are evaluated in the context of existing theory and domain knowledge. Our aspiration in this work is to support the incorporation of modern techniques into statistical methods that serve as the traditional tools of analysis for many practitioners.

BIBLIOGRAPHY

Agresti, A. (2010). *Analysis of ordinal categorical data*, volume 656. John Wiley & Sons. 14, 18

Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., et al. (1999). *LAPACK users' guide*. SIAM. 67

Barbu, A., She, Y., Ding, L., and Gramajo, G. (2017). Feature selection with annealing for computer vision and big data learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2). 2, 9, 10

Barrett, A. E. and Gunderson, J. (2021). Grandparent–adult grandchild relationships: Development of a typology using latent class analysis. *The Gerontologist*, 61(5):724–734. 12

Beale, E. M. and Little, R. J. (1975). Missing values in multivariate analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 37(1):129–145. 72

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc. 41

Bouveyron, C. and Brunet-Saumard, C. (2014). Model-based clustering of high-dimensional data: A review. *Computational Statistics & Data Analysis*, 71:52–78. 16

Bowers, A. J. and Sprott, R. (2012). Why tenth graders fail to finish high school: A dropout typology latent class analysis. *Journal of Education for Students Placed at Risk (JESPAR)*, 17(3):129–148. 12

Buck, S. F. (1960). A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society: Series B (Methodological)*, 22(2):302–306. 72

Cadima, J. and Jolliffe, I. T. (1995). Loadings and correlations in the interpretation of principal components. *Journal of Applied Statistics*, 22(2). 1

Candes, E. and Recht, B. (2012). Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119. 73

Carr, D. C., Jason, K., Taylor, M., and Washington, T. R. (2022). A brief report on older working caregivers: developing a typology of work environments. *The Journals of Gerontology: Series B*, 77(7):1263–1268. 12

Celeux, G., Martin-Magniette, M.-L., Maugis-Rabusseau, C., and Raftery, A. E. (2014). Comparing model selection and regularization approaches to variable selection in model-based clustering. *Journal de la Societe francaise de statistique*, 155(2):57–71. 16

Challet-Bouju, G., Mariez, J., Perrot, B., Grall-Bronnec, M., and Chauchard, E. (2020). A typology of buyers grounded in psychological risk factors for compulsive buying (impulsivity, self-esteem, and buying motives): latent class analysis approach ina community sample. *Frontiers in Psychiatry*, 11:277. 12

Chen, F., Mackey, A. J., Vermunt, J. K., and Roos, D. S. (2007). Assessing performance of orthology detection strategies applied to eukaryotic genomes. *PloS one*, 2(4):e383. 12

Chen, Y., Li, X., Liu, J., and Ying, Z. (2017). Regularized latent class analysis with application in cognitive diagnosis. *Psychometrika*, 82:660–692. 15

Clark, S. L., Muthén, B., Kaprio, J., D'Onofrio, B. M., Viken, R., and Rose, R. J. (2013). Models and strategies for factor mixture analysis: An example concerning the structure underlying psychological disorders. *Structural equation modeling: a multidisciplinary journal*, 20(4):681–703. 2, 91

Competitive Intelligence Solutions (2021). Large car dataset. `https://www.kaggle.com/datasets/cisautomotiveapi/large-car-dataset`. Accessed: 2023-7-22. 90

Dax, A. (2014). Imputing missing entries of a data matrix: a review. *Journal of Advanced Computing*, 3(3):98–222. 73

Dean, N. and Raftery, A. E. (2010). Latent class analysis variable selection. *Annals of the Institute of Statistical Mathematics*, 62:11–35. 16

DeSantis, S. M., Houseman, E. A., Coull, B. A., Stemmer-Rachamimov, A., and Betensky, R. A. (2008). A penalized latent class model for ordinal data. *Biostatistics*, 9(2):249–262. 15

Dong, W., Li, X., Xu, C., and Tang, N. (2021). Hybrid hard-soft screening for high dimensional latent class analysis. Statistica Sinica Preprint No: SS-2021-0061. 24

Florida Department of Education (2015-2016). Florida School Grades Archives: 2015-16. `https://www.fldoe.org/accountability/accountability-reporting/school-grades/archives.stml`. 36

Florida State University Libraries (2018). Diginole Harvest Policy. `https://www.lib.fsu.edu/sites/default/files/sites/default/files/upload/diginoleharvestpolicy.v1-1.pdf`. 40

Fop, M. and Murphy, T. B. (2017). *LCAvarsel: Variable selection for latent class analysis.* 26

Fop, M., Smart, K. M., and Murphy, T. B. (2017). Variable selection for latent class analysis with application to low back pain diagnosis. *The Annals of Applied Statistics*, pages 2080–2110. 16, 24

Forman, G. et al. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3(Mar):1289–1305. 18

Foucart, T. (1999). Stability of the inverse correlation matrix. Partial ridge regression. *Journal of Statistical Planning and Inference*, 77. 56

Ghosh, S. and Verbrugge, R. (2018). Studying strategies and types of players: Experiments, logics and cognitive models. *Synthese*, 195:4265–4307. 12

Goodman, L. A. (1974). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61(2):215–231. 21

Greenacre, M. J. (1984). *Theory and Applications of Correspondence Analysis*. Academic Press. 60

Grisolía, J. M. and Willis, K. G. (2012). A latent class model of theatre demand. *Journal of Cultural Economics*, 36:113–139. 12

Harville, D. A. (1997). *Matrix Algebra From a Statistician's Perspective*. Springer Science & Business Media. 61

Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*. SIAM. 67

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1). 56

Houseman, E. A., Coull, B. A., and Betensky, R. A. (2006). Feature-specific penalized latent class analysis for genomic data. *Biometrics*, 62(4):1062–1070. 15, 24

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2:193–218. 27

Javed, K. (2012). *Development of feature selection algorithms for high-dimensional binary data*. PhD thesis, University of Engineering and Technology Lahore. 20

Javed, K., Babri, H. A., and Saeed, M. (2010). Feature selection based on class-dependent densities for high-dimensional binary data. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):465–477. 19, 20, 111

Jensen, D. R. and Ramirez, D. E. (2012). Variations on ridge traces in regression. *Communications in Statistics - Simulation and Computation*, 41(2). 56

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer-Verlag New York, second edition. 4, 73

Josse, J., Husson, F., et al. (2009). Gestion des données manquantes en analyse en composantes principales. *Journal de la société française de statistique*, 150(2):28–51. 73

Kendler, K. S., Karkowski, L. M., and Walsh, D. (1998). The structure of psychosis: latent class analysis of probands from the roscommon family study. *Archives of general psychiatry*, 55(6):492–499. 12

Law, M. H., Figueiredo, M. A., and Jain, A. K. (2004). Simultaneous feature selection and clustering using mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1154–1166. 25

Lazarsfeld, P. F. (1950). The logical and mathematical foundation of latent structure analysis. *Studies in Social Psychology in World War II Vol. IV: Measurement and Prediction*, pages 362–412. 12

Leoutsakos, J.-M. S., Bandeen-Roche, K., Garrett-Mayer, E., and Zandi, P. P. (2011). Incorporating scientific knowledge into phenotype development: penalized latent class regression. *Statistics in medicine*, 30(7):784–798. 15, 24

Linzer, D. A. and Lewis, J. B. (2011). poLCA: An R package for polytomous variable latent class analysis. *Journal of Statistical Software*, 42(10):1–29. 21

Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data*, volume 793. John Wiley & Sons. 72

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2022). *cluster: Cluster Analysis Basics and Extensions.* R package version 2.1.4 — For new features, see the 'Changelog' file (in the package source). 21

Marquardt, D. and Snee, R. D. (1975). Ridge regression in practice. *The American Statistician*, 29(1). 6

Martinez, W. L., Martinez, A. R., and Solka, J. (2017). *Exploratory data analysis with MATLAB.* Crc Press. 27

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322. 73

Mooijaart, A. and Van der Heijden, P. G. (1992). The em algorithm for latent class analysis with equality constraints. *Psychometrika*, 57:261–269. 15

Muthén, B. (2006). Should substance use disorders be considered as categorical or dimensional? *Addiction*, 101:6–16. 91, 114

Nguyen, L. T., Kim, J., and Shim, B. (2019). Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237. 73

Pan, W. and Shen, X. (2007). Penalized model-based clustering with application to variable selection. *Journal of machine learning research*, 8(5). 16, 24

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238. 18

Preckel, F., Fischbach, A., Scherrer, V., Brunner, M., Ugen, S., Lipnevich, A. A., and Roberts, R. D. (2020). Circadian preference as a typology: Latent-class analysis of adolescents' morningness/eveningness, relation with sleep behavior, and with academic outcomes. *Learning and Individual Differences*, 78:101725. 12

Raftery, A. E. and Dean, N. (2006). Variable selection for model-based clustering. *Journal of the American Statistical Association*, 101(473):168–178. 15, 16, 24

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`. 41

Riyanto, A., Kuswanto, H., and Prastyo, D. D. (2022). Mutual information-based variable selection on latent class cluster analysis. *Symmetry*, 14(5). 24

Robitzsch, A. (2020). Regularized latent class analysis for polytomous item responses: An application to spm-ls data. *Journal of Intelligence*, 8(3):30. 15

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289–317. 21

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423. 16

She, Y. (2017). Selective factor extraction in high dimensions. *Biometrika*, 104(1):97–110. 11, 50, 71

Shen, D., Shen, H., and Marron, J. S. (2013). Consistency of sparse PCA in high dimension, low sample size contexts. *Journal of Multivariate Analysis*, 115. 2

Silvestre, C., Cardoso, M., and Figueiredo, M. (2022). An mml embedded approach for estimating the number of clusters. In *17th Conference of the IFCS 2022–International Federation of Classification Societies: Classification and Data Science in the Digital Age*. CLAD-Associação Portuguesa de Classificação e Análise de Dados. 25

Silvestre, C., Cardoso, M. G., and Figueiredo, M. (2015). Feature selection for clustering categorical data with an embedded modelling approach. *Expert systems*, 32(3):444–453. 25

Sjöstrand, K. (2005). Matlab implementation of LASSO, LARS, the elastic net and SPCA. *Informatics and Mathematical Modelling, Technical University of Denmark, DTU*. 7, 71

Spencer, R., Thabtah, F., Abdelhamid, N., and Thompson, M. (2020). Exploring feature selection and classification methods for predicting heart disease. *Digital health*, 6:2055207620914777. 18

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR. 72

The MathWorks, Inc. (2023). *Statistics and Machine Learning Toolbox™ User's Guide.* The MathWorks, Inc. 62

Tibshirani, R., Wainwright, M., and Hastie, T. (2015). *Statistical Learning with Sparsity.* Chapman and Hall/CRC. 6

Vaughn, M. G., DeLisi, M., Beaver, K. M., and Howard, M. O. (2009). Multiple murder and criminal careers: A latent class analysis of multiple homicide offenders. *Forensic Science International*, 183(1-3):67–73. 12

Vergara, J. R. and Estévez, P. A. (2014). A review of feature selection methods based on mutual information. *Neural computing and applications*, 24:175–186. 17

Ward, K. J., Stedman, R. C., Luloff, A., Shortle, J. S., and Finley, J. C. (2008). Categorizing deer hunters by typologies useful to game managers: A latent-class model. *Society and Natural Resources*, 21(3):215–229. 12

Yang, Y. (2008). Elements of information theory. 17

Yi, S., Lai, Z., He, Z., Cheung, Y., and Liu, Y. (2017). Joint sparse principal component analysis. *Pattern Recognition*, 61. 2, 10, 55, 113

Zhang, Q. (2016). *Sparse generalized PCA and dependency learning for large-scale applications beyond Gaussianity.* PhD thesis, The Florida State University. 1, 72

Zong, C., Xia, R., and Zhang, J. (2021). *Text data mining*, volume 711. Springer. 41

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistics Society*, 67(2). 5

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2). 1, 6, 52, 64, 113

BIOGRAPHICAL SKETCH

Rashad Aziz is a Florida resident and serves as Data Scientist for the Florida State University Campus Reimagined Initiative.