Sparse Generalised Principal Component Analysis

Luke Smallman[a,*], Andreas Artemiou[a], Jennifer Morgan[a]

[a] *Cardiff University, School of Mathematics, Senghennydd Road, Cardiff, CF24 4AG*

Abstract

In this paper, we develop a sparse method for unsupervised dimension reduction for data from an exponential-family distribution. Our idea extends previous work on Generalised Principal Component Analysis by adding L^1 and SCAD penalties to introduce sparsity. We demonstrate the significance and advantages of our method with synthetic and real data examples. We focus on the application to text data which is high-dimensional and non-Gaussian by nature and discuss the potential advantages of our methodology in achieving dimension reduction.

Keywords: dimension reduction, PCA, text mining, exponential family
2010 MSC: 62H25, 62-09, 62J07, 68T50

1. Introduction

Dimension reduction tools are now a common-place solution to two of the major difficulties incurred by high-dimensional big data: computational expense and a breakdown of classical statistical methodology. The former stems from both the difficulty of storage and access of huge datasets, especially when they are not sparse, and from the time costs of running many popular algorithms on such data. The latter arises as most classical methodology for inference was introduced for low-dimensional data, and often cannot generalise beyond a few dimensions. As such, dimension reduction techniques are used to extract a lower-dimensional manifold which describes the structure of the original data as well as possible. Dimension reduction methodology can be broadly split into two categories: methods for feature selection and methods for feature extraction. The former chooses a subset of the original features, whilst the latter finds a (low-dimensional) vector-valued function of the original features. Many examples of feature extraction are referred to as "linear", as the extracted features are simply a linear combination of the original features. In this paper, we will combine two well-known feature selection techniques with a generalised principal component analysis algorithm (a well-known feature extraction method) to achieve sparse dimension reduction for non-Gaussian data. We will focus on the application of this method to Poisson-distributed data, which has received increased attention lately due to the application of Poisson-based techniques in the analysis of text data. PCA (and the variants thereof which we will discuss in this

*Corresponding author
Email address: smallmanl@cardiff.ac.uk (Luke Smallman)

article) are methods for unsupervised dimension reduction. That is, they do not require or use any response associated with the data which may be available, in contrast to supervised methods which do require response information and make use of it to inform the dimension reduction.

Principal Component Analysis (PCA) has been extensively used in the literature since its introduction by Pearson (1901) and more importantly by Hotelling (1933). It has mainly been used for dimension reduction through feature extraction. The main objective of PCA is to sequentially extract orthogonal features which maximize the variability of our data. One of PCA's most useful features is that it produces a linear transformation of the data, calculated with a simple matrix multiplication. This is very advantageous for dimension reduction scenarios, when calculating a complicated function of the entire dataset would be infeasibly expensive in computation time. Instead, the loadings matrix (the matrix which performs the dimension reduction) can be calculated using a smaller, representative dataset, and then the transformation quickly calculated for the rest of the data. PCA was introduced for Gaussian distributed data and has been extended significantly over the years to address different research problems (see for example Tipping and Bishop (1999), Collins et al. (2002), Ding and He (2004), de Leeuw (2006) and Diederichs et al. (2013) among others). The need to develop a generalised version of PCA to address application to the exponential family of distributions (which encompasses a wide variety of models for real-world data) was recognised in Landgraf and Lee (2015) who developed Generalised Principal Component Analysis (GPCA).

In this work, we focus on the application of PCA algorithms to text data which is naturally high-dimensional and non-Gaussian. Text data is usually transformed into numerical data so that classical statistical tools can be applied to it. One of the more common ways to do so is to construct a "document-term matrix" where each row is an observation and each unique word/term within the collection is a column. Then the ij^{th} entry is an integer denoting the number of times the i^{th} document contains the j^{th} term. This is an inherently high-dimensional representation, and the number of terms p typically grows with the number of documents n. However, we expect that many of these terms will be uninformative for many purposes. In general, given a wide variety of documents we often expect that there will be a significant number of terms which are introduced solely because of the form of the writing and which will be common across a number of the documents, even if they are very different in content. For instance, in a collection of letters we would expect to find salutations, well-wishings, addresses, from-lines and the like; such inclusions would likely be common across most letters but would be unlikely to convey any meaning of interest to the text miner. Further, grammar and structure of language dictates many words, such as pronouns, "and", "the", etc., which are important to a human reader but not to a computer. These all expand the dimension of the dataset. Hence, we would like to apply some method of dimension reduction which can provide a much smaller representation of the data. We also desire that our transformed representation does not use terms which do not provide any useful meaning (such as "and", "Yours sincerely"). In order to achieve this, we will require our transformation to have sparse loadings; including a term must improve the model more than some penalty. This also has the advantage of increasing the interpretability of our loadings. Without sparsity, most loading components are typically non-zero; with sparsity, we can often reduce the number of non-zero components to a manageable number to inspect. In the context of text data, this means we can find a list of which

terms are considered important for understanding a document. To apply GPCA and our extension, we require an exponential family model for the data. Given the form of the document-term matrix as a count of term occurrences, we prescribe a simple model for the data where the number of occurrences of term j across each document is given by a Poisson distribution with some mean λ_j.

In Section 2 we will discuss previous work, focusing on extensions to PCA and methods for dimension reduction of text data. In Section 3 we will define GPCA. We will then extend this to Sparse Generalised Principal Component Analysis (SGPCA) in Section 4.1, and present an efficient way to estimate it. We will compare their performances on both synthetic datasets and two healthcare datasets in Section 5 and Section 6 respectively, before discussing the implications of this work and plans for future work in Section 7. To improve readability, detailed derivations will be relegated to the appendices.

2. Previous Work

In this section, we will discuss some related previous work, focusing both on extensions to the usual Gaussian PCA and on dimension reduction methods for text data.

2.1. PCA Extensions

We begin in this section by discussing PCA extensions, first defining Gaussian PCA, then Sparse PCA, Joint Sparse PCA and Robust PCA.

2.1.1. PCA Summary

Let $\mathrm{X} \in \mathbb{R}^{n \times p}$ be a data matrix formed of n observations of a p-dimensional random variable $\boldsymbol{\mathcal{X}}$, that is $\mathrm{X} = [\mathbf{x}_1 | \mathbf{x}_2 | \cdots | \mathbf{x}_n]^{\mathrm{T}}$ with $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^{\mathrm{T}}$ for $i = 1, \ldots, n$. Then the usual definition of (Gaussian) PCA is the (k-dimensional) orthogonal linear projection such that the first component of the projected data is in the direction with the most variance, the second component is in the direction with the second most variance, and so on. It is also well-established in the literature that the PCA approximation to X minimises the squared reconstruction error (the difference between the lower-rank approximation to X and the true X).

2.1.2. Sparse PCA

Sparse PCA (SPCA) from Zou et al. (2006) is based around their "SPCA criterion" which finds an approximation to X of the form $\mathrm{XBA}^{\mathrm{T}}$ ($\mathrm{A}, \mathrm{B} \in \mathbb{R}^{p \times k}$) which minimises

$$\|\mathrm{X} - \mathrm{XBA}^{\mathrm{T}}\|_2^2 + \lambda \sum_{i=1}^{k} \|\boldsymbol{\beta}_i\|_2^2 + \sum_{j=1}^{k} \lambda_{1,j} \|\boldsymbol{\beta}_j\|_1,$$

subject to $\mathrm{A}^{\mathrm{T}}\mathrm{A} = \mathrm{I}_k$ and where $\mathrm{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k]$. In this case, the matrix B gives the loadings, and has both L^1 and L^2 penalties imposed on its entries, inducing sparsity. The combination of these penalties is often known as the elastic net (due to Zou and Hastie (2005)).

2.1.3. Sparse PCA via Rotation and Truncation

This method, developed in Hu et al. (2016), provides an alternative method for sparse PCA to that of Zou et al. (2006). The essential idea is that any rotation of the PCA loadings provides an orthogonal basis spanning the same subspace. Thus, the authors propose a method to find a rotation matrix and a sparse basis which approximates the PCA loadings after rotation. They offer four methods of truncation, which is used to find the approximating sparse basis, each of which offers slightly different performance and control.

2.1.4. Robust PCA

In Kwak (2008), Kwak formulated Robust PCA (RPCA) in terms of the projection variance maximisation formulation of PCA. That is, viewing PCA as finding the projection matrix $\mathrm{U} \in \mathbb{R}^{p \times k}$ which maximises $\mathrm{Tr}\left(\mathrm{U}^{\mathrm{T}}\mathrm{C}\mathrm{U}\right)$ subject to $\mathrm{U}^{\mathrm{T}}\mathrm{U} = \mathrm{I}_k$, where C is the covariance matrix of X. This problem can be reformulated as maximising $\sum_{i=1}^{n} \left\|\mathrm{U}^{\mathrm{T}}\mathbf{x}_i\right\|_2^2$ subject to the same orthogonality constraint. They propose modifying the norm used in this formulation to yield the problem of maximising $\sum_{i=1}^{n} \left\|\mathrm{U}^{\mathrm{T}}\mathbf{x}_i\right\|_1$ subject to $\mathrm{U}^{\mathrm{T}}\mathrm{U} = \mathrm{I}_k$. This increases the robustness of the problem to outliers.

2.1.5. Sparse Probabilistic Principal Component Analysis

Sparse Probabilistic Principal Component Analysis (SPPCA), formulated in Guan and Dy (2009) is an extension of Probabilistic PCA (Tipping and Bishop (1999)) and SPCA (Zou et al. (2006)), essentially working to combine the two. They specify a probabilistic model for PCA which incorporates a sparsity inducing prior on the loadings. In the paper they investigate three different priors: a Laplacian prior, an inverse-Gaussian prior and a Jeffrey's prior. In this work we consider the Laplacian prior case which corresponds to an L^1 penalty on the components.

2.1.6. Sparse Exponential Family Principal Component Analysis

Lu et al. (2016) proposed Sparse Exponential Family Principal Component Analysis (SEPCA) roughly as an extension to SPCA (Zou et al. (2006)) to the exponential family. Given data X, their problem is specified by

$$\min_{\mathrm{Z}:\mathrm{Z}^{\mathrm{T}}\mathrm{Z}=\mathrm{I},\mathrm{W},\mathbf{b}} \sum_{n} A\left(\mathrm{W}^{\mathrm{T}}\mathbf{z}_n + \mathbf{b}\right) - \mathrm{Tr}\left(\left(\mathrm{Z}\mathrm{W} + \mathbf{1}\mathbf{b}^{\mathrm{T}}\right)\mathrm{X}^{\mathrm{T}}\right) + P\left(\mathrm{W}, \mathbf{b}\right)$$

where A is a function specified by the exponential distribution and $P(\cdot,\cdot)$ is a sparsifying penalty. In particular,

$$P\left(\mathrm{W}, \mathbf{b}\right) = \lambda_0 \left\|\mathrm{Z}\mathrm{W} + \mathbf{1}\mathbf{b}^{\mathrm{T}}\right\|_2^2 + \sum_{i=1}^{k} \lambda_i \left|\mathbf{W}_i\right|$$

which the authors claim enables reconstruction of the principal components in degenerate cases and induces sparsity in the loading vectors.

2.2. Text Data Dimension Reduction Methods

In this section we will briefly discuss three methods for text dimension reduction: Multinomial Inverse Regression, Nonnegative Matrix Factorisation and Latent Dirichlet Allocation.

2.2.1. Multinomial Inverse Regression

Multinomial Inverse Regression (MNIR), introduced in Taddy (2013) and further in Taddy (2015), is a dimension reduction method for text data based around a multinomial model for text data. The method is supervised, intended to provide a reduced dimension projection of the data suitable for use in a classification algorithm or similar. As such, it requires a response variable for each observation, which it uses to determine informative terms. As such, our comparison with it will only be performed for the classed synthetic dataset in Section 5.2. We will not include it in our comparison on the healthcare dataset in Section 6, as it is only capable of producing a projection of the same dimension as the response variable; as our response in this case is univariate, but all the other methods will be used to provide a 3-dimensional representation.

2.2.2. Nonnegative Matrix Factorisation

Nonnegative Matrix Factorisations (for an overview, see Gillis (2014)) aims to factorise a nonnegative matrix X in the form X = WH, where W and H are both also nonnegative. In the framework of text mining, the basis vectors formed by W can be interpreted as topics in term-space and H can be interpreted as giving the importance of each topic for each document.

2.2.3. Latent Dirichlet Allocation

Latent Dirichlet Allocation, from Blei et al. (2003), is a (generative) probabilistic model for text. Roughly, it assumes each document draws a vector of topic probabilities, then repeatedly draws a topic according to those probabilities, drawing a word each time it draws a topic. For full details of how these quantities are distributed (and how the underlying probabilities are estimated), consult Blei et al. (2003), but for our purposes the most important aspect of the model is that it is used to estimate a matrix $\mathrm{B} = [\boldsymbol{\beta}_1 | \ldots | \boldsymbol{\beta}_k]^{\mathrm{T}}$, where each vector $\boldsymbol{\beta}_i$ gives (multinomial) probabilities of drawing each of the words in the vocabulary given that the word is drawn according to topic i. It is important to note that the choice of the word "topics" is to assist interpretation; this method is not supervised, the topics are driven only by the observed word counts and the number of topics is a user-chosen parameter.

3. Generalised PCA

In this section, we introduce the Generalised PCA of Landgraf and Lee (2015) for exponential family distributions; for completeness and clarity, we begin by discussing exponential families and some of their properties.

3.1. Exponential Family Distributions

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^p$ be a random vector from a distribution in the exponential family of distributions with parameter $\boldsymbol{\theta}$. The probability density function for $\mathbf{x}$ has the form $f(\mathbf{x}|\boldsymbol{\theta}) = h(\mathbf{x}) \exp\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} - b(\boldsymbol{\theta})\right)$, where $h(\mathbf{x})$ and $b(\boldsymbol{\theta})$ are both scalar-valued functions, with the former serving to normalise the distribution. A straightforward calculation shows that $\mathbb{E}(\boldsymbol{\mathcal{X}}) = b'(\boldsymbol{\theta})$. The *canonical link function* g is defined as the left inverse of b', i.e. $g(b'(\boldsymbol{\theta})) = \boldsymbol{\theta}$. The model where the expected value is set to the observed data is known as the *saturated model*, its parameters will be denoted by $\tilde{\boldsymbol{\theta}}$, and it will be equal to $g(\mathbf{x})$.

3.2. GPCA Definition

In Section 2.1.1 we gave the usual formulation of PCA. An equivalent formulation of PCA, suitable for the definition of GPCA, is to find the optimum $\mathrm{U} \in \mathbb{R}^{p\times k}$ and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)^{\mathrm{T}} \in \mathbb{R}^p$ which minimise

$$\sum_{i=1}^{n} \left\| \mathbf{x}_i - \boldsymbol{\mu} - \mathrm{U}\mathrm{U}^{\mathrm{T}} \left(\mathbf{x}_i - \boldsymbol{\mu} \right) \right\|_2^2,$$

where U is the projection matrix. With a Gaussian distribution (with known variance), the canonical link function is the identity function, so the saturated model parameters are the data themselves, the natural parameter is the mean and the deviance is proportional to squared error loss. We can see then that, under the assumption of a Gaussian distribution, the above formulation of PCA is equivalent to finding the deviance-optimal approximation to the natural parameters of the form $\boldsymbol{\mu} + \mathrm{U}\mathrm{U}^{\mathrm{T}} \left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right)$. This can be readily extended to any exponential family distribution; we find that the deviance has the following form (details of the derivation can be found in Appendix A1):

$$D\left(\mathrm{U}, \boldsymbol{\mu}\right) = \sum_{i=1}^{n} \sum_{j=1}^{p} \left\{ b_j \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right) \right]_j \right) - x_{ij} \left\{ \mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right) \right]_j \right\} \right\} = \sum_{i,j} D_{ij} \quad (1)$$

where the b_j functions are subscripted to allow for the possibility of each component of our random vector coming from a different exponential family distribution (although we will proceed to examine only cases where all variables are from the same distribution, albeit with different natural parameters). We denote the coordinate-observation-wise deviance D_{ij} and low-rank natural parameter estimate θ_{ij}, defined by

$$D_{ij} = b_j\left(\theta_{ij}\right) - x_{ij}\theta_{ij}, \quad \theta_{ij} = \mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j$$

Thus we can now define the GPCA of Landgraf and Lee (2015) as the projection given by the U that minimises the deviance. Formally, let

$$\mathrm{U}^*, \boldsymbol{\mu}^* = \underset{\mathrm{U}\in\mathbb{R}^{p\times k}, \boldsymbol{\mu}\in\mathbb{R}^p}{\arg\min} \; D\left(\mathrm{U}, \boldsymbol{\mu}\right),$$

then $\left(g\left(\mathrm{X}\right) - \mathbb{1}(\boldsymbol{\mu}^*)^{\mathrm{T}} \right) \mathrm{U}^*$ is the GPCA projection.

4. Sparse GPCA

In this section we will present the definition of SGPCA and an algorithm for approximating it.

4.1. Definition

There is a wide literature using penalties on regression coefficients to induce sparsity in statistical procedures, primarily focusing on the L^1 (LASSO-type penalty, see Tibshirani (1996)) and L^2 (ridge-type penalty, see Hoerl and Kennard (1970)) penalties as well as combination of the two (elastic net, see Zou and Hastie (2005)). Another popular penalty with desirable features is the Smoothly Clipped Absolute Deviation (SCAD) penalty of Fan and Li (2001). The SCAD penalty is usually defined by its derivative

$$P'_S(\theta;\lambda,a) = \lambda I\left(\theta \leq \lambda\right) + \frac{(a\lambda-\theta)_+}{a-1} I\left(\theta > \lambda\right) \tag{2}$$

where $x_+ = \max(x,0)$. The penalty has two parameters, λ and a. The former is usually chosen on a problem-by-problem basis, often with some form of cross-validation, while the latter is often taken as 3.7 due to an argument in Fan and Li (2001). Integrating, as in Appendix A4, shows that the penalty has the form

$$P_S(\theta;\lambda,a) = \begin{cases} \lambda\theta & 0 < \theta \leq \lambda \\ -\frac{\theta^2 - 2a\lambda\theta + \lambda^2}{2(a-1)} & \lambda < \theta \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} & a\lambda < \theta \end{cases} \tag{3}$$

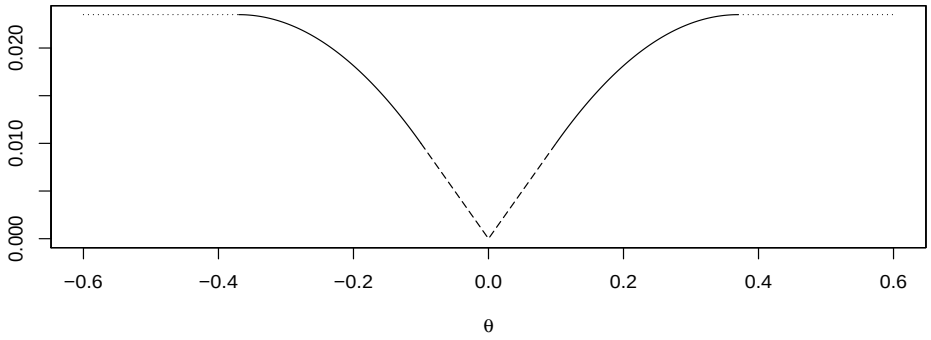
That is, "near" 0 the penalty is linear and decreases towards the origin, "far" from 0 the penalty is a positive constant, and between the two areas the penalty is quadratic, again decreasing towards the origin. This can also be seen in Figure 4.1a, where the linear section is dashed, the constant section is dotted, and the quadratic section is solid. Note that the figure shows the symmetrised SCAD penalty (that is, it shows the penalty on $|\theta|$). In Figure 4.1b we also show the penalty for three different values of λ (while holding a fixed at 3.7) to demonstrate the behaviour of the penalty as λ changes.

In contrast, the L^1 penalty is much simpler, being only the absolute value of the component. Clearly, the L^1 penalty differs only by a multiplicative factor from the SCAD penalty near 0, but unlike the SCAD penalty it does not change in form across its domain. From a computational point of view, this means that the L^1 penalty is slightly less computationally expensive, not requiring the evaluation of any conditional operators. Although the evaluation of these operators is not particularly expensive, the estimation algorithm we will discuss later requires very frequent evaluation of the penalty function.

The motivation behind the L^1 penalty is to ensure that non-zero coefficients will only be added to the model if they reduce the unpenalised objective function by an amount proportional to their magnitude. "Large" coefficients therefore can be interpreted as coefficients which make the model significantly better. In our method, λ_L is the coefficient of proportionality which determines how much a unit increase in a coefficients distance from 0 must decrease the objective function for it to be a permissible change. The SCAD penalty performs a similar function to the L^1 penalty; the primary difference is that once a coefficient is sufficiently large in absolute value, the penalty does not increase (i.e. the linear region in Figure 4.1a). Ideally, this should ensure that once a coefficient is determined to be significant, its magnitude is decided by the objective function without the penalty function artificially reducing it.

For this particular problem, we would like to penalise only U by imposing upon it either an L^1 penalty, the SCAD penalty, or a combination of both. Specifically, we define

(a) The (symmetrised) SCAD penalty, with $\lambda = 0.1$ and $a = 3.7$



(b) The (symmetrised) SCAD penalty for $\lambda = 0.1$ (solid), $\lambda = 0.15$ (dashed), and $\lambda = 0.2$ (dotted)

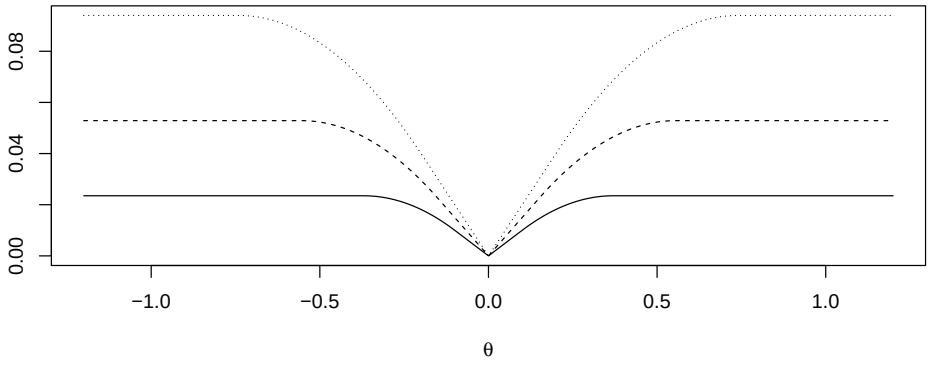


Figure 4.1: SCAD Penalty

the penalty function

$$P(\mathrm{U}, \lambda_L, \lambda, a, \lambda_S) = \sum_{i,j} \left\{ \lambda_L \left| U_{ij} \right| + \lambda_S P_S \left(\left| \mathrm{U}_{ij} \right| ; \lambda, a \right) \right\} \tag{4}$$

The coefficient $\lambda_S \geq 0$ of the SCAD penalty, like λ_L, is used to control the weighting of SCAD against the L^1 penalty and the deviance. We introduce this, as whilst changing λ changes the magnitude of the SCAD penalty, it also changes the characteristics significantly by changing the knots of the quadratic splines. Notice that we place the SCAD penalty on the absolute value of each component, not on the value itself, as the SCAD penalty is only defined for positive values. We do not penalise $\boldsymbol{\mu}$ as we desire it to capture (in some sense) the mean of each of the component natural parameters. Non-zero entries in $\boldsymbol{\mu}$ are to be expected, and allow us to centre the natural parameters. We can now define our SGPCA objective function

$$S(\mathrm{U}, \boldsymbol{\mu}; \lambda_L, \lambda, a, \lambda_S) \coloneqq D(\mathrm{U}, \boldsymbol{\mu}) + P(\mathrm{U}; \lambda_L, \lambda, a, \lambda_S) \tag{5}$$

For the remainder of this paper, we will not explicitly notate the dependence on λ_L, λ_S, or a in order to simplify notation.

Remark 1. Here we emphasize that in this problem it does not make sense to apply an L^2 penalty to U, since the usual definition of the L^2 penalty for matrices (also known as the Frobenius norm) is equivalent to $\mathrm{Tr}\left(\mathrm{U}^{\mathrm{T}}\mathrm{U}\right)$ which, by the semi-orthogonality of U, is $\mathrm{Tr}\left(\mathrm{I}_p\right) = p$. Consequently, for the purposes of this article we use only the L^1 and SCAD penalties imposed on U. Other penalties will be explored in the future.

Remark 2. In the rest of this article we will consider three SGPCA formulations: the L^1 penalty, the SCAD penalty and a linear combination. We have no a priori reason to believe that the combined penalty ought to be superior to either, but our specification of the problem lends itself well to considering it; as such it would be remiss not to investigate its performance.

4.2. Estimation Algorithm

In general, (5) is non-convex in U and $\boldsymbol{\mu}$. This, combined with the non-differentiability of the penalty function at 0 and the semi-orthogonality constraint upon U, make finding the optimum values of U and $\boldsymbol{\mu}$ difficult. One way to deal with the semi-orthogonality condition would be to use a Lagrangian method; in Appendix A2 we use a Lagrange multiplier augmented objective function to derive first order optimality conditions. However, the Lagrangian still has the difficulties of non-differentiability. Instead, in Section 4.2.1 we will derive a perturbed majoriser of the objective function to be used in a Majorise-Minimise (MM) algorithm using the results of Hunter and Li (2005). Crucially, this perturbed majoriser will be differentiable, allowing us to use gradient-based optimisation. In Section 4.2.2, we will demonstrate the method of Wen and Yin (2013) to perform such optimisation whilst preserving the semi-orthogonality constraints.

4.2.1. Majorisation

The crux of our majorisation procedure is the method of Hunter and Li (2005), who developed theory for majorising penalty functions of the form $P(\boldsymbol{\theta}) = \sum_{i=1}^{p} \lambda_i p_i \left(|\theta_i| \right)$,

where the functions $p_i(\cdot)$ are non-negative. Both the L^1 and SCAD penalties can be written in this form, allowing us to use their majorisation procedure.

To illustrate, let us consider the ij^{th} component of U. For notational simplicity, let us denote this component by u. Then the contribution of u to the penalty function $P(\mathrm{U})$ is $\lambda_L |u| + \lambda_S P_S(|u|; \lambda, a)$

The MM algorithm we will construct is iterative, so at time-step t we have a previous value for each component, which we will denote $u^{(t-1)}$. We will first construct the majoriser and perturbed majoriser for the L^1 penalty. We can approximate $|u|$ by

$$|u| \approx \left|u^{(t-1)}\right| + \frac{u^2 - \left(u^{(t-1)}\right)^2}{2\left|u^{(t-1)}\right|} \tag{6}$$

In Hunter and Li (2005), the authors showed that this function majorises the exact component L^1 penalty function, justifying its use in a majorisation of the entire objective function. We can then construct the perturbed approximate L^1 penalty function by modifying the denominator

$$|u| \approx \left|u^{(t-1)}\right| + \frac{u^2 - \left(u^{(t-1)}\right)^2}{2\left(\varepsilon + \left|u^{(t-1)}\right|\right)} \tag{7}$$

where $\varepsilon > 0$. This function is now well-defined for all values of u and $u^{(t-1)}$, and furthermore is differentiable in both too. It is clear that as $\varepsilon \downarrow 0$ (where $\downarrow$ indicates approaching from above), this perturbed penalty tends to (6).

We now consider the case of the SCAD penalty (ignoring for now the multiplicative factor λ_S), which can be majorised using the formulation of Hunter and Li (2005) by

$$P_S\left(\left|u^{(t-1)}\right|; \lambda, a\right) + \frac{\left(u^2 - \left(u^{(t-1)}\right)^2\right) P_S'\left(\left|u^{(t-1)}\right| +; \lambda, a\right)}{2\left|u^{(t-1)}\right|} \tag{8}$$

where the $+$ denotes the right-hand limit. The only pertinent case is when $u^{(t-1)} = 0$, in which case $P_S'(0+) = \lambda$. As before, we construct the perturbed approximation by replacing the denominator

$$P_S\left(\left|u^{(t-1)}\right|; \lambda, a\right) + \frac{\left(u^2 - \left(u^{(t-1)}\right)^2\right) P_S'\left(\left|u^{(t-1)}\right| +; \lambda, a\right)}{2\left(\varepsilon + \left|u^{(t-1)}\right|\right)} \tag{9}$$

Like before, (9) tends to (8) as $\varepsilon \downarrow 0$. Consequently, we might expect that the optimum using (7) and (9) would be close to the optimum using (6) and (8) for sufficiently small ε, while allowing us to use gradient-based optimisation methods.

Remark 3. It is worth noting that neither (7) or (9) majorise the L^1 penalty or the SCAD penalty respectively. Instead, Hunter and Li (2005) show that they majorise a perturbed version of the respective penalties. The interested reader is referred to their paper for the details. From now, we will refer to them as majorisers, with the understanding that this only holds exactly in the limit as $\varepsilon \downarrow 0$.

In the original formulation of GPCA, the authors use a quadratic approximation to the deviance which they then majorise in order to construct an MM algorithm for their problem. The particular majorisation they use allows them to find a closed form minimiser over $\boldsymbol{\mu}$ and reduces to an orthogonal Procrustes problem in U for each iteration (for more details on the orthogonal Procrustes problem, see Schönemann (1966)). The introduction of the penalty term we use does not allow the use of the same technique here. Instead, we majorise only the penalty term, leaving the deviance unchanged. This gives us our majorised objective function:

$$M\left(\mathrm{U}, \boldsymbol{\mu} \middle| \mathrm{U}^{(t-1)}\right) \coloneqq D(\mathrm{U}, \boldsymbol{\mu}) + M_P^{\varepsilon}\left(\mathrm{U} \middle| \mathrm{U}^{(t-1)}\right) \tag{10}$$

where M_P^{ε} is the perturbed majoriser of the combined L^1 and SCAD penalties, given by:

$$M_P^{\varepsilon}\left(\mathrm{U} \middle| \mathrm{U}^{(t-1)}\right) = \sum_{i=1}^{n}\sum_{j=1}^{p}\left\{\lambda_L\left|\mathrm{U}_{ij}^{(t-1)}\right| + \lambda_S P_S\left(\left|\mathrm{U}_{ij}^{(t-1)}\right|; \lambda, a\right) + \frac{\left(\mathrm{U}_{ij}^2 - \left(\mathrm{U}_{ij}^{(t-1)}\right)^2\right)\left(\lambda_L + \lambda_S P_S'\left(\left|\mathrm{U}_{ij}^{(t-1)}\right| +; \lambda, a\right)\right)}{2\left(\varepsilon + \left|\mathrm{U}_{ij}^{(t-1)}\right|\right)}\right\} \tag{11}$$

4.2.2. Gradient-Based Optimisation

In Wen and Yin (2013), the authors derive a method for gradient-based optimisation which preserves orthogonality. Their scheme works in generality, for all differentiable optimisation problems with orthogonality constraints, but we will illustrate it only in application to solving our perturbed majoriser problem.

Given a current estimate of U (denoted $\mathrm{U}^{(t-1)}$) and the gradient of the objective function at that point (denoted G), we construct the skew-symmetric matrix $\mathrm{A} \coloneqq \mathrm{G}\left(\mathrm{U}^{(t-1)}\right)^{\mathrm{T}} - \mathrm{U}^{(t-1)}\mathrm{G}^{\mathrm{T}}$. Wen and Yin showed that the function

$$\mathrm{Y}(\tau) \coloneqq \left(\mathrm{I} + \frac{\tau}{2}\mathrm{A}\right)^{-1}\left(\mathrm{I} - \frac{\tau}{2}\mathrm{A}\right)\mathrm{U}^{(t-1)}$$

defines a descent direction for $\tau \geq 0$. Additionally, $\mathrm{Y}(\tau)$ is smooth in τ and $\mathrm{Y}(\tau)^{\mathrm{T}}\mathrm{Y}(\tau) = \left(\mathrm{U}^{(t-1)}\right)^{\mathrm{T}}\mathrm{U}^{(t-1)} = \mathrm{I}$, so each value of τ produces a feasible point preserving the semi-orthogonality condition on U. Minimisation then proceeds by line search along $\{\mathrm{Y}(\tau)\}_{\tau \geq 0}$.

We use a standard line search algorithm given in Nocedal and Wright (2006) and recommended in Wen and Yin (2013). The details of this method are omitted, as any one-dimensional optimisation method would suffice, but the interested reader is referred to both sources. We also perform a gradient-based algorithm to find optimal values of $\boldsymbol{\mu}$; as there are no constraints on $\boldsymbol{\mu}$ to deal with, any standard optimisation procedure can be used.

The required gradients are given as follows, their derivation can be found in Appendix

A3:

$$\frac{\partial M}{\partial \mathrm{U}_{rs}} = \sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\left(b_j'\left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\right)\right]_j\right) - x_{ij}\right) \times \left(\delta_{rj}\mathbf{U}_{[s]}^{\mathrm{T}}\left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\right) + U_{js}\left(\tilde{\theta}_{ir} - \mu_r\right)\right)\Bigg\} + \frac{\left(\lambda_L + \lambda_S P_S'\left(\left|\mathrm{U}_{rs}^{(t-1)}\right| +; \lambda, a\right)\right)\mathrm{U}_{rs}}{\varepsilon + \left|\mathrm{U}_{rs}^{(t-1)}\right|} \tag{12}$$

$$\frac{\partial M}{\partial \mu_r} = \sum_{i=1}^{n}\sum_{j=1}^{p}\left(b_j'\left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\right)\right]_j\right) - x_{ij}\right)\left(\delta_{jr} - \left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\right]_{jr}\right) \tag{13}$$

4.2.3. Implementation

Code to calculate the optimal values of $\boldsymbol{\mu}$ and U was written in R (R Foundation for Statistical Computing, Vienna (2011)). We alternated between optimising over $\boldsymbol{\mu}$ using the R built-in OPTIM function and optimising over U using our own implementation of the gradient-based method of Wen and Yin (2013), stopping when the objective function changed between iterations by less than a given tolerance, which we selected on a case-by-case basis. Our majorisation-based method requires that we have an initial estimate for $\boldsymbol{\mu}$ and U. As the gradient-based optimisation method for U preserves the value of $\mathrm{U}^{\mathrm{T}}\mathrm{U}$, we also require that our initial estimate for U is left semi-orthogonal. To that end, we follow the same procedure as for GPCA in setting $\boldsymbol{\mu}^{(0)} = \mathrm{ColMeans}\left(g\left(\mathrm{X}\right)\right)$ and $\mathrm{U}^{(0)} = \mathrm{EigVec}_l\left(g\left(\mathrm{X}\right) - \mathbb{1}\left(\boldsymbol{\mu}^{(0)}\right)^{\mathrm{T}}\right)$ (i.e. the first l eigenvectors, ordered in decreasing eigenvalue magnitude).

4.2.4. Estimation procedure

To summarise, estimation proceeds as follows:

- Step 1:
  1. Calculate $\tilde{\Theta} \coloneqq g(\mathrm{X})$
  2. Set $\boldsymbol{\mu}^{(0)}$ to the column means of $\tilde{\Theta}$
  3. Set $\mathrm{U}^{(0)}$ to the first l eigenvectors of $\tilde{\Theta} - \mathbb{1}\left(\boldsymbol{\mu}^{(0)}\right)^{\mathrm{T}}$
- Step 2: to be repeated for $t = 1, 2, \ldots$
  1. Let $\boldsymbol{\mu}^{(\mathrm{t})}$ be the minimiser of (10) with respect to $\boldsymbol{\mu}$, letting U in (10) be $\mathrm{U}^{(t-1)}$ calculated using OPTIM) with gradient given by (13)
  2. (a) Calculate the gradient G of (10) with respect to U evaluated at $\mathrm{U}^{(t-1)}, \boldsymbol{\mu}^{(\mathrm{t})}$ using (12)
     (b) Set $\mathrm{A} \coloneqq \mathrm{G}\left(\mathrm{U}^{(t-1)}\right)^{\mathrm{T}} - \mathrm{U}^{(t-1)}\mathrm{G}^{\mathrm{T}}$
     (c) Define $\mathrm{Y}(\tau) \coloneqq \left(\mathrm{I} + \frac{\tau}{2}\mathrm{A}\right)^{-1}\left(\mathrm{I} - \frac{\tau}{2}\mathrm{A}\right)\mathrm{U}^{(t-1)}$
     (d) Find $\tau^* \geq 0$ which minimises (10) evaluated at $\mathrm{U} = \mathrm{Y}(\tau), \boldsymbol{\mu}^{(\mathrm{t})}$
     (e) Set $\mathrm{U}^{(\mathrm{t})} \coloneqq \mathrm{Y}(\tau^*)$

3. Repeat until (5) at $\mathrm{U}^{(\mathrm{t})}, \boldsymbol{\mu}^{(\mathrm{t})}$ has changed less than the specified tolerance from (5) at $\mathrm{U}^{(t-1)}, \boldsymbol{\mu}^{(t-1)}$

Remark 4. Some exponential family distributions (including the Poisson distribution) have canonical link functions which require taking the logarithm of 0. Numerically, we approximate this by $-\iota$, where ι has a large positive value, as Landgraf and Lee (2015) did. The appropriate value can be determined by cross validation, but the authors have found that values of 4 and larger generally suffice, differing only in numerical stability.

5. Synthetic Data Examples

Our main motivation in exploring GPCA and SGPCA is their application to text data. Cardiff and Vale University Health Board have a very large collection of letters sent from consultants at the local hospital to general practitioners regarding outpatient care. In order to scale to working with their more than 2 million records, we are interested in dimension reduction techniques which are compatible with the discrete, non-Gaussian distribution of word counts in the vector-space model of text. The word counts in this framework can be modelled using Poisson random variables. As a result, in this section we consider models based around the Poisson distribution. At this point, it is worth noting that for the Poisson distribution, $b'(\theta) = \exp(\theta)$ and $g(\theta) = \log\theta$.

Remark 5. In the following subsections we do not include SEPCA in our comparisons. Unfortunately, we could not find combinations of the two penalties for which the algorithm would converge. Our investigations suggest that this is related to the magnitude of the Poisson distribution means. Indeed, in Section 6 (where each variable has a much smaller mean) the algorithm did successfully converge and will be included in the comparisons.

5.1. Synthetic Data

In order to compare the performance of SGPCA and existing methodology (both that which is based on PCA for feature extraction and that which has been used extensively for the analysis of text data), we use a synthetic data example which is based on hidden factors as follows

$$V_1 \sim \mathrm{Po}(25), \quad V_2 \sim \mathrm{Po}(30), \quad V_3 = V_1 + 3V_2$$

For each observation, we drew V_1, V_2 and calculated V_3. Then the first four components of the observation were set equal to V_1 with independent errors, that is $X_i = V_1 + \epsilon_i$, $i = 1, \ldots, 4$, the second four components were set equal to V_2 with independent errors, that is $X_i = V_2 + \epsilon_i$, $i = 5, \ldots, 8$, and the last two components were set equal to V_3 with independent errors, $X_i = V_3 + \epsilon_i$, $i = 9, 10$. In each case, the errors were generated by drawing from a $\mathrm{Po}(2)$ distribution and multiplying by 1 or -1 with equal probability. We drew 100 observations, and performed L^1 SGPCA with $\lambda_L = 10^7$, SCAD SGPCA with $\lambda = 0.1$ and $\lambda_S = 10^6$, and the combined penalty SGPCA with $\lambda_L = 10^6$, $\lambda_S = 10^6$ and $\lambda = 0.05$. We also performed PCA, SPCA, RPCA, NMF and LDA. The first loadings/directions from all algorithms are shown in Table 5.1, and the second in Table 5.2.

Remark 6. Note that we do not include the loadings from Sparse PCA by Rotation and Truncation (Hu et al. (2016)) here or in the following sections for reasons of space. In all the examples, the performance is very similar to SPCA, though generally sparser.

L^1	SCAD	Both	GPCA	PCA	SPCA	RPCA	NMF	LDA	SPPCA
0.085	0.074	0.107	-0.283	0.045	0.000	-0.022	0.074	0.203	-0.195
0.094	0.095	0.094	-0.306	0.051	0.002	-0.024	0.074	0.178	-0.198
0.055	0.058	0.049	-0.290	0.040	0.000	-0.013	0.069	0.118	-0.191
0.045	0.052	0.032	-0.284	0.034	0.026	-0.019	0.068	0.243	-0.189
0.418	0.426	0.396	-0.336	0.193	0.132	-0.205	0.186	0.085	-0.275
0.440	0.434	0.461	-0.334	0.207	0.133	-0.212	0.194	0.180	-0.282
0.420	0.418	0.433	-0.334	0.201	0.123	-0.198	0.190	0.229	-0.279
0.445	0.447	0.441	-0.358	0.209	0.160	-0.219	0.189	0.141	-0.283
0.342	0.341	0.330	-0.311	0.641	0.668	-0.639	0.647	0.805	-0.517
0.345	0.345	0.340	-0.316	0.646	0.691	-0.645	0.645	0.307	-0.520

Table 5.1: First loading/direction for synthetic data, using L^1 penalised SGPCA, SCAD penalised SGPCA, the combined penalty SGPCA (both), GPCA, PCA, SPCA, RPCA, NMF, LDA and SPPCA

L^1	SCAD	Both	GPCA	PCA	SPCA	RPCA	NMF	LDA	SPPCA
0.470	0.452	0.493	-0.373	-0.458	-0.485	-0.443	0.217	0.059	0.408
0.481	0.475	0.490	-0.380	-0.466	-0.483	-0.457	0.218	0.082	0.413
0.511	0.517	0.505	-0.428	-0.496	-0.526	-0.519	0.227	0.137	0.428
0.505	0.522	0.484	-0.429	-0.493	-0.479	-0.486	0.228	0.028	0.427
-0.082	-0.084	-0.068	0.270	0.133	0.065	0.148	0.126	0.209	0.236
-0.106	-0.108	-0.094	0.299	0.175	0.111	0.164	0.123	0.133	0.259
-0.082	-0.070	-0.098	0.265	0.130	0.047	0.096	0.135	0.095	0.235
-0.077	-0.079	-0.062	0.277	0.131	0.076	0.152	0.126	0.163	0.235
0.018	0.015	-0.006	0.143	-0.017	0.000	-0.070	0.608	0.412	0.175
0.022	0.019	0.008	0.141	-0.036	0.000	-0.056	0.606	0.839	0.185

Table 5.2: Second loading/direction for synthetic data, using L^1 penalised SGPCA, SCAD penalised SGPCA, the combined penalty SGPCA (both), GPCA, PCA, SPCA, RPCA, NMF, LDA and SPPCA

Analysis of Table 5.1 shows that for each of the three SGPCA variants, the first loading picks a direction corresponding primarily with the second hidden factor, and secondarily with the third hidden factor. Picking out both of these hidden factors together is to be expected, as these two factors are very highly correlated. On the other hand, GPCA's first loading does not strongly identify a direction associated with any hidden factor. PCA, SPCA and RPCA and SPPCA all have quite similar performance, all finding a direction most associated with the third hidden factor, with some contribution from the second. NMF finds a direction primarily associated with the third hidden factor, with a smaller contribution from the second hidden factor. LDA primarily identifies the third hidden factor with a smaller contribution from the first and second; it is worth noting

that the contributions across the same hidden factor appear to vary more than they do in most of the other methods.

Looking at the second loadings/directions in Table 5.2, we see that all three SGPCA variants strongly identify the first hidden factor. GPCA returns a similar loading, but with smaller coefficients for the components of the first hidden factor, and fairly large coefficients (of opposite sign) for the second. It also includes contributions from the third hidden factor. PCA returns fairly similar loadings to GPCA, with slightly larger contributions from the first hidden factor and slightly smaller from the second and third. SPCA and RPCA perform very simiarly to PCA. NMF, on the other hand, produces a loading similar in characteristics to its first loading, strongly identifying the third hidden factor with some contributions from the first and second. LDA also does not identify anything more than its first direction, finding primarily the third hidden factor. SPPCA finds a direction primarily associated with the first component, then with the second, and with a contribution from the third as well.

Of all the algorithms considered, the performance of SGPCA is the closest to the authors' idealised loadings where we might consider the hidden factors to be related to topics or classifications. Its first loading identifies the second hidden factor and the strongly correlated third hidden factor, and the second identifies the first hidden factor. The second hidden factor, having higher variance than the first, is a sensible choice for the first loading. Given that the third hidden factor has the highest variance we might expect it to dominate the first loading, but we believe that it receives lower coefficients due to its correlation with both of the other hidden factors. Although GPCA, PCA, SPCA and RPCA provide similar identification in the second loading, they all include larger contributions from the second hidden factor, which somewhat reduces interpretability.

5.2. Synthetic Classes

Although SGPCA is an unsupervised dimension reduction method, we believe that its construction should be sufficient to find features sufficient to differentiate between observations drawn from different classes. To investigate this, we performed a similar exercise to that in Section 5.1, but drawing from two different distributions. The first has hidden factors

$$V_1 \sim \mathrm{Po}\,(25)\,, \quad V_2 \sim \mathrm{Po}\,(25)\,, \quad V_3 = V_1 + 3V_2$$

and the second

$$V_1 \sim \mathrm{Po}\,(25)\,, \quad V_2 \sim \mathrm{Po}\,(35)\,, \quad V_3 = 2V_1 + V_2$$

As in Section 5.1, the first four entries of each vector observation are V_1 with error, the second four V_2 with error and the last two are V_3 with error. We construct all of the observations by the same method as before. We then perform the same algorithms as before, with the addition this time of MNIR, using as a response the class identifier 0 for the first 100 observations and 1 for the second 100. The loadings/directions for all algorithms are given in Table 5.3.

The primary differentiating factors between the two classes of data are the second and third hidden factors. As such, the strong identification of the third hidden factor by all three SGPCA variants is excellent. The first GPCA loading assigns roughly similar

L^1	SCAD	Both	GPCA	PCA	SPCA	RPCA	NMF	LDA	MNIR	SPPCA
-0.227	-0.196	-0.184	-0.193	-0.123	-0.116	-0.126	0.162	0.198	0.000	0.240
0.042	0.023	0.023	-0.225	-0.110	-0.109	-0.113	0.165	0.263	0.000	0.233
-0.119	-0.101	-0.089	-0.226	-0.127	-0.125	-0.131	0.164	0.098	0.000	0.243
-0.025	-0.030	-0.025	-0.239	-0.120	-0.119	-0.129	0.163	0.183	0.000	0.237
-0.078	-0.069	-0.062	-0.431	-0.130	-0.128	-0.134	0.196	0.245	0.447	0.243
-0.036	-0.058	-0.068	-0.437	-0.130	-0.131	-0.125	0.197	0.208	0.469	0.243
-0.066	-0.042	-0.027	-0.441	-0.129	-0.129	-0.127	0.197	0.174	0.448	0.243
-0.062	-0.053	-0.045	-0.428	-0.126	-0.123	-0.127	0.197	0.097	0.446	0.240
-0.674	-0.682	-0.684	-0.156	-0.656	-0.657	-0.650	0.605	0.307	-0.301	0.515
-0.679	-0.687	-0.691	-0.159	-0.668	-0.670	-0.670	0.610	0.782	-0.300	0.524

Table 5.3: Loadings/directions for classed synthetic data, using L^1 penalised SGPCA, SCAD penalised SGPCA, the combined penalty SGPCA (both), GPCA, PCA, SPCA, RPCA, NMF, LDA, MNIR and SPPCA.

coefficients to the first and third hidden factors, and slightly higher coefficients to the second. Once again, PCA, SPCA and RPCA give similar loadings, concentrating mostly on the third hidden factor with a smaller contribution for the first and second. NMF also gives highest weighting to the third hidden factor, with smaller contributions from the first and second. SPPCA performs similarly again, though assigns slightly less weighting to the third hidden factor and slightly more to the first two. LDA assigns its highest coefficient to one of the components corresponding to the third hidden factor, but its other coefficients are far more sporadic and inconsistent across the same hidden factor. MNIR's loading finds a (weighted) difference between the second and third hidden factors.

5.3. Robustness Against Noise

In order to investigate how robust the loadings given by SGPCA are to noise, we will perform a similar synthetic data analysis to that in Section 5.1, but varying the parameter of noise. That is,

$$V_1 \sim \mathrm{Po}\,(25)\,, \quad V_2 \sim \mathrm{Po}\,(30)\,, \quad V_3 = V_1 + 3V_2$$

We then construct each observation in the usual way, except that the i.i.d. noise for each observation component is instead drawn from a Poisson distribution with mean η and multiplied by 1 or -1 with equal probability. This is the same setup as previously, except that we can control the variance of the noise. We then find SGPCA loadings for each $\eta \in \{1, 2, 3, 4\}$. We again drew 100 observations and performed L^1 SGPCA with $\lambda_L = 10^7$, SCAD SGPCA with $\lambda = 0.1$ and $\lambda_S = 10^6$, and the combined penalty SGPCA with $\lambda_L = 10^6$, $\lambda_S = 10^6$ and $\lambda = 0.05$. The loadings for the L^1 penalised SGPCA are displayed in Table 5.4a, for the SCAD penalised SGPCA in Table 5.4b, and for the combined penalty SGPCA in Table 5.4c.

An examination of Table 5.4a suggests that, for all four magnitudes of noise, the loadings of L^1 penalised SGPCA are very similar (up to sign changes), with the first loading consistently identifying a direction which primarily captures the second and third hidden factors, while the second loading captures the first hidden factor. Table 5.4c

	$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$		$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$
Loading 1	0.1267	0.0846	0.0355	-0.0182	Loading 2	-0.4553	0.4699	-0.5496	-0.4738
	0.1385	0.0941	0.0265	-0.0007		-0.4751	0.4809	-0.4895	-0.5268
	0.1303	0.0548	0.0250	0.0473		-0.4807	0.5112	-0.4823	-0.4865
	0.1582	0.0454	0.0602	-0.0237		-0.4941	0.5052	-0.4524	-0.4912
	0.4128	0.4176	0.4151	-0.4448		0.1482	-0.0824	0.0629	0.0641
	0.4150	0.4401	0.4358	-0.3682		0.1402	-0.1060	0.0512	0.0434
	0.4012	0.4203	0.5003	-0.4802		0.1534	-0.0816	0.0637	-0.0300
	0.4293	0.4452	0.3950	-0.4873		0.1601	-0.0771	0.0684	0.0329
	0.3385	0.3417	0.3362	-0.3153		0.0200	0.0181	-0.0454	-0.0777
	0.3468	0.3453	0.3352	-0.3098		0.0227	0.0221	-0.0606	-0.0773

(a) Two loadings of L^1 penalised SGPCA under varying levels of noise on synthetic data

	$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$		$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$
Loading 1	0.1266	0.0744	0.0361	-0.0347	Loading 2	-0.4556	0.4524	-0.5505	-0.0414
	0.1388	0.0953	0.0262	-0.0237		-0.4753	0.4750	-0.4890	-0.0324
	0.1304	0.0580	0.0249	0.0019		-0.4806	0.5168	-0.4826	0.0375
	0.1579	0.0517	0.0599	-0.0195		-0.4939	0.5216	-0.4516	-0.0254
	0.4126	0.4255	0.4152	-0.0392		0.1481	-0.0843	0.0627	-0.1338
	0.4154	0.4339	0.4355	-0.0510		0.1400	-0.1084	0.0514	-0.1471
	0.4015	0.4177	0.5005	-0.0430		0.1534	-0.0700	0.0637	-0.1214
	0.4287	0.4470	0.3949	-0.0336		0.1601	-0.0792	0.0684	-0.1211
	0.3386	0.3414	0.3362	0.8715		0.0202	0.0146	-0.0454	-0.4866
	0.3468	0.3452	0.3352	-0.4808		0.0229	0.0195	-0.0606	-0.8303

(b) Two loadings of SCAD penalised SGPCA under varying levels of noise on synthetic data

	$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$		$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$
Loading 1	0.1267	0.1074	0.0358	-0.0177	Loading 2	-0.4552	0.4933	-0.5501	-0.4747
	0.1384	0.0944	0.0263	0.0004		-0.4750	0.4901	-0.4892	-0.5259
	0.1303	0.0492	0.0249	0.0469		-0.4807	0.5051	-0.4824	-0.4864
	0.1584	0.0318	0.0600	-0.0242		-0.4942	0.4841	-0.4520	-0.4913
	0.4129	0.3959	0.4151	-0.4435		0.1483	-0.0684	0.0628	0.0643
	0.4149	0.4607	0.4356	-0.3677		0.1403	-0.0936	0.0513	0.0430
	0.4011	0.4333	0.5004	-0.4783		0.1533	-0.0980	0.0637	-0.0291
	0.4296	0.4412	0.3949	-0.4875		0.1601	-0.0625	0.0684	0.0330
	0.3384	0.3304	0.3362	-0.3177		0.0200	-0.0061	-0.0454	-0.0785
	0.3467	0.3398	0.3352	-0.3124		0.0226	0.0082	-0.0606	-0.0777

(c) Two loadings of the combined penalty SGPCA under varying levels of noise on synthetic data

Table 5.4: Investigations of the performance of all three SGPCA variants across varying levels of noise.

suggests that the combined penalty SGPCA has the same characteristics. However, Table 5.4b suggests that the SCAD penalised SGPCA is not quite so robust; both loadings seem to vary more as η increases, with the loadings for $\eta = 4$ being very different from those for $\eta = 1$, no longer giving near-equal weights amongst components corresponding to the same hidden factors. This is, perhaps, not surprising, given the use of the L^1 penalty in RPCA and JSPCA for its robustness to outliers (which become more probable with increasing magnitude of noise).

5.4. Dependence on Tolerance

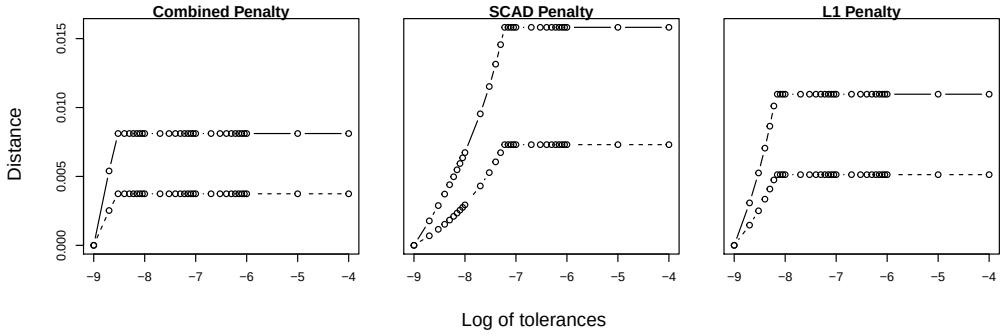
In our anecdotal experience of estimating the SGPCA approximation across a variety of datasets, we found that decreasing the numerical tolerance increased the time taken to complete the algorithm approximately exponentially. As this could quickly become untenable, we performed a further simulation study to examine the effects of the tolerance on the accuracy of the solution. The data was generated in precisely the same way as in Section 5.1 and the same three SGPCA variants were fitted with the same parameters, varying only the tolerance. We chose a range of values between 10^{-4} and 10^{-9}, taking the results of the latter as the "gold standard" by which we judged the accuracy.

In Figure 5.1a we graph the Euclidean distances of each of the two loadings from each run of the algorithm against the logarithm of the chosen tolerances. As an alternative measure, in Figure 5.1b we show the deviances from each fit, "normalised" by division by the deviance of our gold standard fit. Both figures show that tolerances too large provide fits which are nearly identical, having had only a few iterations. Beyond a critical tolerance, the fits quickly converge to the gold standard fit. However, we wish to draw attention to the axes of both plots – the largest deviance amongst all fits is less than one half of one percent larger than the gold standard, and the Euclidean distance between the gold standard and the fit further from it is of order 10^{-2}. We suggest, therefore, that a practical strategy for choosing a tolerance value would be to choose the smallest value that allows feasible optimisation times, preferably (if this study is representative) smaller than 10^{-7}.

6. Healthcare Data

The dataset we analysed is a by-product of a lexicon-based classifier which the health board developed in response to a need to systematically classify letters sent by consultants to GPs and patients. It consists of just over 40,000 exemplar sentences generated procedurally from a set of seeds. All observations are labelled either discharge or follow-up with a heavy imbalance in favour of follow-up; this imbalance is due to the nature of the problem being addressed, and approximates the expected imbalance in the collection of letters. For the analysis we drew a stratified sample of 200 observations from the dataset to calculate the projection matrices for each method. The dimension of the chosen data was 55 in this case. We then drew 50 observations from each of the two classes to use as a testing dataset, and show the results of applying the projections to this test set in Figure 6.1. For each algorithm, we plot each pair of the first three principal components. Each plot shows the 100 selected points coloured (and with different point characters) according to the class. Due to the similar performance of the three types of SGPCA, we performed only the L^1 penalised version. We also show GPCA and

(a) Euclidean distances between loadings from fits with different tolerance values for each of the SGPCA variants. The first loading is shown dashed, and the second is shown solid.



(b) Deviances from each fit across a range of tolerance values, divided by the deviance from the fit with the smallest tolerance. All three SGPCA variants are shown.

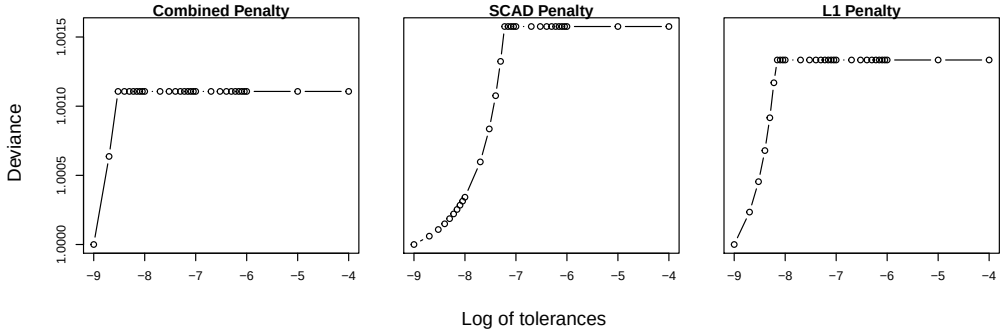
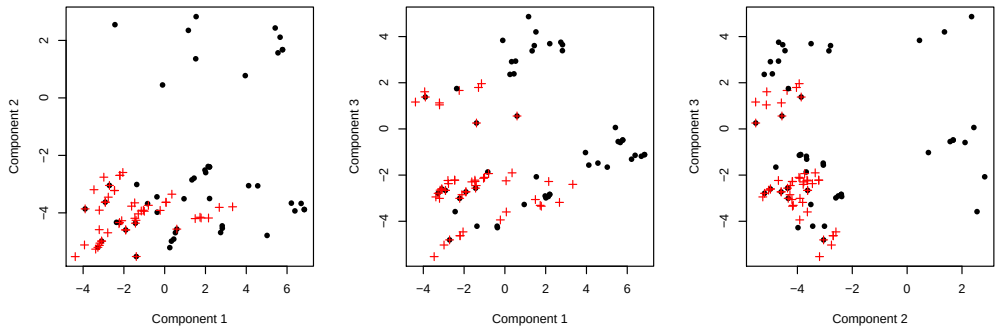
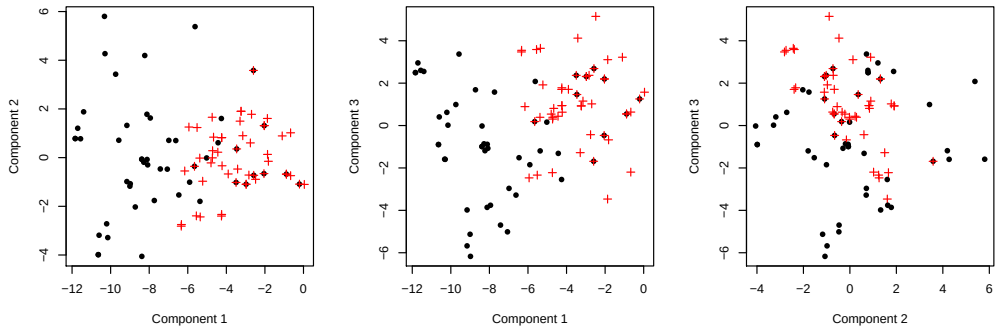


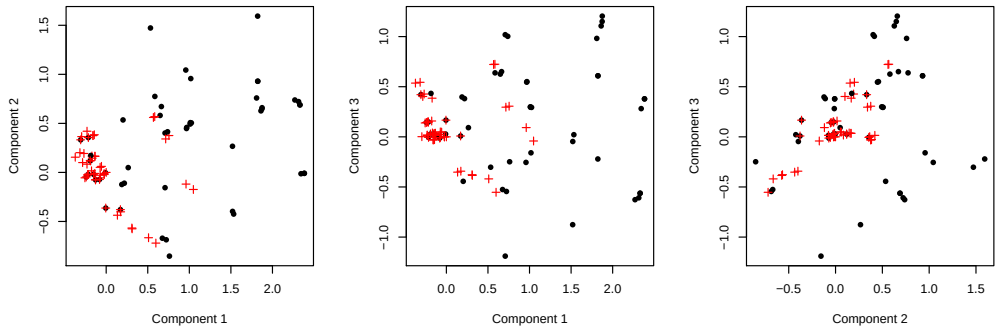
Figure 5.1: Behaviour of all SGPCA variants as tolerance is varied.



(a) L^1 penalty SGPCA



(b) GPCA



(c) PCA

Figure 6.1: Plots of pairs of the first three principal components for the healthcare dataset obtained from SGPCA, GPCA and PCA.

PCA as the most comparable and most widely used algorithms respectively; the other algorithms are omitted due to a combination of lack of space and poor performance. Qualitatively, all three algorithms have reasonably similar performance, allowing decent linear separation of the data. This suggests that our algorithm seems to inherit PCA's reputation for frequently providing useful loadings for analysing with respect to some response variable, despite not using such information itself.

7. Discussion

In this paper we have introduced a sparsifying penalty to the generalised principal components of Landgraf and Lee (2015), along with an efficient routine by which to calculate the loadings. By means of simulation and case study we have shown excellent performance in estimating appropriate (and useful) principal component loadings, giving arguably the best loadings in the synthetic cases and projections on par with the current state of the art for text in the healthcare study.

Comparison of the efficacy of the three penalties we used (L^1, SCAD and L^1 + SCAD) suggest little difference between them. As such, our preference is for the L^1 penalty due both to its simplicity and its reliance on only one tuning parameter, unlike the SCAD penalty which has two and the combination of both which has three.

It is worth noting that, as with all non-convex problems, our method will likely have multiple minima. One way this could be mitigated, should it prove problematic, is to repeatedly run the optimisation procedure from a variety of initial values and choose between the found optima by means of the deviance. However, the authors' experience is that using Gaussian PCA as the starting point is both convenient (as it provides a left semi-orthogonal matrix) and effective; the optima the optimisation procedure finds from this starting point have consistently exhibited the properties we desire.

There is significant scope for extension of this work. Firstly, one can look into a data-driven method for choosing the optimum values of λ_L, λ and λ_S. Secondly, there is scope for investigating other penalty functions, such as the L^0 penalty, to better understand how penalising the coefficients drives sparsity. Another possible extension is the development of methods for supervised dimension reduction, extending existing methods such as partial least squares. Finally, a more challenging problem is the development of kernel-based GPCA and SGPCA for non-linear feature extraction.

Acknowledgements

The authors would like to thank Cardiff and Vale University Health Board for supporting the project and allowing access to the healthcare dataset. We are also deeply grateful for the insightful comments provided by the reviewers and editor, this article is much improved for their efforts.

A1. Derivation of deviance

The deviance of our model from the saturated model is properly

$$D(\mathrm{X}|\Theta) = -2\left(\log \mathbb{P}\left(\mathrm{X}|\Theta\right) - \log \mathbb{P}\left(\mathrm{X}|\tilde{\Theta}\right)\right)$$

where $\tilde{\Theta}$ is the matrix of saturated parameters for each observation. However, as we seek only value of Θ which minimises the deviance, we can disregard the contribution of the log probability from the saturated model, as well as the factor of 2. This leaves the expression for the pseudo-deviance as

$$\begin{aligned}
D(\mathrm{X}|\Theta) &= -\log \mathbb{P}\left(\mathrm{X}|\Theta\right) \\
&= -\log \prod_{i=1}^{n} \left\{h(\mathbf{x}_i) \exp\left(\mathbf{x}_i^{\mathrm{T}} \boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)\right)\right\} \\
&= -\sum_{i=1}^{n} \left\{\log h(\mathbf{x}_i) + \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)\right\} \\
&= \sum_{i=1}^{n} \left\{b(\boldsymbol{\theta}_i) - \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\theta}_i\right\} \text{ (with an additive constant)} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \left\{b_j(\theta_{ij}) - x_{ij}\theta_{ij}\right\} \text{ (using the independence of each component)}
\end{aligned}$$

This can then be recognised as the expression in (1) by substituting in the appropriate form for θ_{ij}.

A2. Derivation of optimality conditions

In order to derive optimality conditions, we need first to construct the Lagrangian of the problem.

$$L(\mathrm{U}, \boldsymbol{\mu}) \coloneqq S(\mathrm{U}, \boldsymbol{\mu}) + \mathrm{Tr}\left(\Lambda\left(\mathrm{U}^{\mathrm{T}}\mathrm{U} - \mathrm{I}\right)\right)$$

where Λ is an $l \times l$ symmetric matrix of Lagrange multipliers which we use in order to capture the semi-orthogonality of U, and $S(\mathrm{U}, \boldsymbol{\mu})$ is given in (5). Calculating the partial derivatives with respect to U, $\boldsymbol{\mu}$ and Λ we arrive at the following first-order optimality conditions, the derivations of which can be found below.

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=1}^{p} \left(b_j'\left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\right)\right]_j\right) - x_{ij}\right)\left(\delta_{rj}\mathbf{U}_{[l]}^{\mathrm{T}}\left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\right) + U_{js}\left(\tilde{\theta}_{ir} - \mu_r\right)\right) \\
&\quad + \lambda_L \operatorname{sgn}\left(\mathrm{U}_{rs}\right) + \lambda_S P_S'\left(|\mathrm{U}_{rs}|\right)\operatorname{sgn}\left(\mathrm{U}_{rs}\right) + 2\boldsymbol{\Lambda}_l^{\mathrm{T}}\mathbf{U}_r = 0 \quad r = 1, \ldots, n,\ s = 1, \ldots, p
\end{aligned} \tag{B.1}$$

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=1}^{p} \left(b_j'\left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\left(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\right)\right]_j\right) - x_{ij}\right)\left(\delta_{jr} - \left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\right]_{jr}\right) = 0 \\
&\qquad\qquad r = 1, \ldots, n
\end{aligned} \tag{B.2}$$

$$\mathrm{U}^{\mathrm{T}}\mathrm{U} = \mathrm{I} \tag{B.3}$$

where $\mathbf{A}_i$ indicates the i^{th} row and $\mathbf{A}_{[i]}$ the i^{th} column of the matrix A (where both are taken as column vectors), and δ_{ij} is the discrete Kronecker delta. It is important to note

that, due to the non-differentiability of the absolute value function at 0, (B.1) only holds for $\mathrm{U}_{rs} \neq 0$.

We begin by deriving (B.1).

$$
\begin{aligned}
&\frac{\partial}{\partial \mathrm{U}_{rs}} \Bigg\{ \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ b_j \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \left\{ \mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right\} \\
&\qquad + \lambda_S P_S \left(|\mathrm{U}_{ij}| \right) \Bigg\} + \lambda_L |\mathrm{U}| + \mathrm{Tr} \left(\boldsymbol{\Lambda} \left\{ \mathrm{U}^{\mathrm{T}}\mathrm{U} - \mathrm{I} \right\} \right) \Bigg\} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ \left\{ b_j' \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \right\} \frac{\partial}{\partial \mathrm{U}_{rs}} \left[\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right] \Bigg\} \\
&\qquad + \lambda_S P_S' \left(|\mathrm{U}_{rs}| \right) \mathrm{sgn} \left(\mathrm{U}_{rs} \right) + \lambda_L \, \mathrm{sgn} \left(\mathrm{U}_{rs} \right) + \frac{\partial}{\partial \mathrm{U}_{rs}} \mathrm{Tr} \left(\boldsymbol{\Lambda} \left\{ \mathrm{U}^{\mathrm{T}}\mathrm{U} - \mathrm{I} \right\} \right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ \left\{ b_j' \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \right\} \Bigg\{ \underbrace{\delta_{rj} \mathrm{U}_{[s]}^{\mathrm{T}} \left[\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right]}_{(*)} + \underbrace{\mathrm{U}_{js} \left[\tilde{\theta}_{ir} - \mu_r \right]}_{(\dagger)} \Bigg\} \Bigg\} \\
&\qquad + \lambda_S P_S' \left(|\mathrm{U}_{rs}| \right) \mathrm{sgn} \left(\mathrm{U}_{rs} \right) + \lambda_L \, \mathrm{sgn} \left(\mathrm{U}_{rs} \right) + \underbrace{2 \boldsymbol{\Lambda}_l^{\mathrm{T}} \mathbf{U}_r}_{(\ddagger)}
\end{aligned}
$$

Where $(*)$ comes from the terms quadratic in U_{rs}, $(\dagger)$ is from the terms linear in U_{rs}, and $(\ddagger)$ is from the summation definition of the trace. Setting the above equation equal to 0 for each value of $r \in 1, \ldots, n$ and $s \in 1, \ldots, p$ yields the first set of first order optimality conditions. Similarly, we derive the next set by differentiating with respect to the r^{th} element of $\boldsymbol{\mu}$.

$$
\begin{aligned}
&\frac{\partial}{\partial \mu_r} \Bigg\{ \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ b_j \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \left\{ \mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right\} \\
&\qquad + \lambda_S P_S \left(|\mathrm{U}_{ij}| \right) \Bigg\} + \lambda_L |\mathrm{U}| + \mathrm{Tr} \left(\boldsymbol{\Lambda} \left\{ \mathrm{U}^{\mathrm{T}}\mathrm{U} - \mathrm{I} \right\} \right) \Bigg\} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ \left\{ b_j' \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \right\} \frac{\partial}{\partial \mu_r} \left[\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right] \Bigg\} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ \left\{ b_j' \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \right\} \left\{ \delta_{jr} - \frac{\partial}{\partial \mu_r} \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \boldsymbol{\mu} \right]_j \right\} \Bigg\} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \Bigg\{ \left\{ b_j' \left(\mu_j + \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \left\{ \tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu} \right\} \right]_j \right) - x_{ij} \right\} \left\{ \delta_{jr} - \left[\mathrm{U}\mathrm{U}^{\mathrm{T}} \right]_{jr} \right\} \Bigg\}
\end{aligned}
$$

Once again, setting the final expression equal to 0 for each value of r gives the $\boldsymbol{\mu}$ optimality conditions. Finally, the $\boldsymbol{\Lambda}$ optimality condition is easily derived, this time using matrix

differential calculus for convenience.

$$\begin{aligned}
&\frac{\partial}{\partial \Lambda}\Bigg\{\sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{b_j\left(\mu_j+\left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\left\{\tilde{\boldsymbol{\theta}}_i-\boldsymbol{\mu}\right\}\right]_j\right)-x_{ij}\left\{\mu_j+\left[\mathrm{U}\mathrm{U}^{\mathrm{T}}\left\{\tilde{\boldsymbol{\theta}}_i-\boldsymbol{\mu}\right\}\right]_j\right\}\\
&\quad+\lambda_S P_S\left(|\mathrm{U}_{ij}|\right)\Bigg\}+\lambda_L|\mathrm{U}|+\mathrm{Tr}\left(\Lambda\left\{\mathrm{U}^{\mathrm{T}}\mathrm{U}-\mathrm{I}\right\}\right)\Bigg\}\\
&=\frac{\partial}{\partial \Lambda}\mathrm{Tr}\left(\Lambda\left\{\mathrm{U}^{\mathrm{T}}\mathrm{U}-\mathrm{I}\right\}\right)\\
&=\mathrm{vec}\left(\mathrm{U}^{\mathrm{T}}\mathrm{U}-\mathrm{I}\right)
\end{aligned}$$

Setting this equal to 0 reduces to $\mathrm{U}^{\mathrm{T}}\mathrm{U}=\mathrm{I}$, the semi-orthogonality condition.

A3. Derivation of gradients

Calculating the gradients of the majorised objective function is very similar to calculating the first order optimality conditions in Appendix A2, as the terms from $D(\mathrm{X},\mathrm{U},\boldsymbol{\mu})$ will be the same. Further, the perturbed penalty has no terms in $\boldsymbol{\mu}$, so the $\boldsymbol{\mu}$-gradient is precisely $\frac{\partial}{\partial \boldsymbol{\mu}}D(\mathrm{X},\mathrm{U},\boldsymbol{\mu})$, so this gradient will not be recalculated. This simply leaves calculating $\frac{\partial}{\partial \mathrm{U}_{rs}}M_P^{\varepsilon}\left(\mathrm{U}|\mathrm{U}^{(t-1)}\right)$

$$\begin{aligned}
\frac{\partial M_P^{\varepsilon}}{\partial \mathrm{U}_{rs}}=&\frac{\partial}{\partial \mathrm{U}_{rs}}\sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\lambda_L\left|\mathrm{U}_{ij}^{(t-1)}\right|+\lambda_S P_S\left(\left|\mathrm{U}_{ij}^{(t-1)}\right|\right)\\
&+\frac{\left[\mathrm{U}_{ij}^2-\left(\mathrm{U}_{ij}^{(t-1)}\right)^2\right]\left[\lambda_L+\lambda_S P_S'\left(\left|\mathrm{U}_{ij}^{(t-1)}\right|\right)\right]}{2\left[\varepsilon+\left|\mathrm{U}_{ij}^{(t-1)}\right|\right]}\Bigg\}\\
=&\sum_{i=1}^{n}\sum_{j=1}^{p}\left\{\frac{\partial}{\partial \mathrm{U}_{rs}}\left[\mathrm{U}_{ij}^2-\left(\mathrm{U}_{ij}^{(t-1)}\right)^2\right]\frac{\lambda_L+\lambda_S P_S'\left(\left|\mathrm{U}_{ij}^{(t-1)}\right|\right)}{2\left[\varepsilon+\left|\mathrm{U}_{ij}^{(t-1)}\right|\right]}\right\}\\
=&\frac{2\mathrm{U}_{rs}\left(\lambda_L+\lambda_S P_S'\left(\left|\mathrm{U}_{rs}^{(t-1)}\right|\right)\right)}{2\left[\varepsilon+\left|\mathrm{U}_{rs}^{(t-1)}\right|\right]}
\end{aligned}$$

We omit the arguments of λ, a to P_S for notational convenience.

A4. Derivation of SCAD penalty from derivative

The SCAD penalty derivative (2) is equivalently given as

$$p'(\theta)=\begin{cases}\lambda & 0<\theta\leq\lambda\\ \frac{a\lambda-\theta}{a-1} & \lambda<\theta<a\lambda\\ 0 & a\lambda<\theta\end{cases}$$

Computing $p(\theta)$ is then a matter of integrating carefully. For $0 < \theta \leq \lambda$, $p(\theta) = \int_0^\theta \lambda dt = \lambda\theta$. For $\lambda < \theta \leq a\lambda$,

$$p(\theta) = p(\lambda) + \int_\lambda^\theta \frac{a\lambda - t}{a-1} dt = \lambda^2 + \left[\frac{a\lambda t - \frac{1}{2}t^2}{a-1}\right]_{t=\lambda}^{t=\theta} = -\frac{\theta^2 - 2a\lambda\theta + \lambda^2}{2(a-1)}$$

Finally, for $a\lambda < \theta$, $p(\theta) = p(a\lambda) + \int_{a\lambda}^\theta 0 dt = p(a\lambda) = \frac{(a+1)\lambda^2}{2}$. Together, this gives (3).

References

Blei, D. M., Ng, A. Y., Jordan, M. I., 2003. Latent dirichlet allocation. Journal of machine Learning research 3, 993–1022.

Collins, M., Dasgupta, S., Schapire, R. E., 2002. A Generalization of Principal Components Analysis to the Exponential Family. Advances in Neural Information Processing Systems 14, 617–624.

de Leeuw, J., 2006. Principal component analysis of binary data by iterated singular value decomposition. Computational Statistics & Data Analysis 50 (1), 21–39.

Diederichs, E., Juditsky, A., Nemirovski, A., Spokoiny, V., 2013. Sparse non Gaussian component analysis by semidefinite programming. Machine Learning 91, 211–238.

Ding, C., He, X., 2004. K-means clustering via principal component analysis. In: Proceedings of the twenty-first international conference on Machine learning. ACM, p. 29.

Fan, J., Li, R., 2001. Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. Journal of the American Statistical Association 96 (456), 1348–1360.

Gillis, N., 2014. The Why and How of Nonnegative Matrix Factorization. ArXiv e-prints.

Guan, Y., Dy, J., 2009. Sparse probabilistic principal component analysis. In: Artificial Intelligence and Statistics. pp. 185–192.

Hoerl, A. E., Kennard, R. W., 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics 12 (1), 55–67.

Hotelling, H., 1933. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24 (6), 417–441.

Hu, Z., Pan, G., Wang, Y., Wu, Z., 2016. Sparse principal component analysis via rotation and truncation. IEEE transactions on neural networks and learning systems 27 (4), 875–890.

Hunter, D. R., Li, R., 2005. Variable selection using MM algorithms. Annals of Statistics 33 (4), 1617–1642.

Kwak, N., 2008. Principal component analysis based on l1-norm maximization. IEEE transactions on pattern analysis and machine intelligence 30 (9), 1672–1680.

Landgraf, A. J., Lee, Y., 2015. Generalized Principal Component Analysis : Projection of Saturated Model Parameters. Ohio State University Statistics Department Technical Report 892 (892).

Lu, M., Huang, J. Z., Qian, X., 2016. Sparse exponential family Principal Component Analysis. Pattern Recognition 60, 681–691.

Nocedal, J., Wright, S., 2006. Numerical optimization. Springer Science & Business Media.

Pearson, K., 1901. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (1), 559–572.

R Foundation for Statistical Computing, Vienna, 2011. R Development Core Team. R: A Language and Environment for Statistical Computing 55, 275–286.

Schönemann, P. H., 1966. A generalized solution of the orthogonal procrustes problem. Psychometrika 31 (1), 1–10.

Taddy, M., 2013. Multinomial Inverse Regression for Text Analysis. Journal of the American Statistical Association 108 (503), 755–770.

Taddy, M., 2015. Distributed multinomial regression. The Annals of Applied Statistics 9 (3), 1394–1414.

Tibshirani, R., 1996. Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society. Series B (Methodological) 58 (1), 267–288.

Tipping, M. E., Bishop, C. M., 1999. Probabilistic Principal Component Analysis. Journal of the Royal Statistical Society. Series B (Statistical Methodology) 61 (3), 611–622.

Wen, Z., Yin, W., 2013. A feasible method for optimization with orthogonality constraints. Mathematical Programming 142 (1-2), 397–434.

Zou, H., Hastie, T., 2005. Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society. Series B (Methodological) 67 (2), 301–320.

Zou, H., Hastie, T., Tibshirani, R., 2006. Sparse Principal Component Analysis. Journal of Computational and Graphical Statistics 15 (2), 265–286.