Dimension Reduction for Exponential Family Data with Applications to Text Data

Luke Smallman

2019

Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

School of Mathematics
Ysgol Mathemateg

Summary

In this manuscript, we will address the problem of dimension reduction for data modelled by an exponential family distribution, with a particular focus on text data modelled by a Poisson-count model. We are motivated to develop new methods for such data by links between principal component analysis and the Gaussian log-likelihood, which suggests both a simple way to extend PCA to the exponential family (of which the Gaussian distribution is a member), and the unsuitability of PCA when the data is appropriately modelled by a distribution which is not well-approximated by the Gaussian distribution.

We will present three novel methods for exponential family dimension reduction. The first is "Poisson Inverse Regression", a supervised method from the family of inverse regression methods. We will demonstrate that this method provides a *sufficient dimension reduction*. That is, the transformed data is statistically sufficient with respect to the response.

The second is Sparse Generalised Principal Component Analysis, which extends the method of Generalised Principal Component Analysis put forward by Landgraf and Lee (2015b). This method is unsupervised, as is motivated by a modification of the PCA objective function to accommodate other exponential family distributions. We demonstrate that this method performs as-well or better than other state-of-the-art methods. This work has been published as Smallman, Artemiou, et al. (2018).

The third is Sparse Simple Exponential/Poisson Principal Component Analysis. This method extends Simple Exponential Principal Component Analysis, put forward by Li and Tao (2013), enforcing sparsity in the equivalent of the loadings matrix. This method is also unsupervised, and we demonstrate its state-of-the-art performance. This work was done jointly with William Underwood from Oxford University, and is published in Smallman, Underwood, et al. (2019).

Finally, we present a new framework for analysing and synthesising dimension reduction methods, which we call "Quasi-Likelihood PCA". This is based on tensor estimating equations, which we also present as a new development. We apply this method to analyse several methods in the literature.

This would not have been possible without the love and support of so many. In particular, my deepest thanks to Scott – you have been there from the beginning, have put up with so much, and have never once doubted me.

Thank you to my parents; without your help and support I would never have been in a position to begin this, let alone finish it.

Thank you to Lynn, Glenni and Mizzy, your friendship has been invaluable.

Finally, but by no means least, thank you to Andreas – you've been a wonderful mentor and friend throughout this process.

"Begin at the beginning," the King said gravely, "and go on till you come to the end: then stop."

— Lewis Carroll, *Alice in Wonderland*

Acknowledgements

I would like to acknowledge deep gratitude to Cardiff and Vale University Health Board and Cardiff University School of Mathematics for providing funding for this project.

The work of Chapter 5 was done jointly with William Underwood from Oxford University as part of a 4 week undergraduate summer project. I would like to thank William for his collaboration and enthusiasm; it was a pleasure working with him.

Contents

List of Figures

List of Tables

— Chapter 1 —

Introduction

With the explosion of "big data", methods for dimension reduction have become increasingly important; recently, there has been considerable research into different ways to perform dimension reduction for a wide variety of types of data. Text data is one of those types of data – it is plentiful, its analysis is frequently very impactful, and it has certain statistical peculiarities. In particular, text data is not Gaussian nor even symmetrically distributed which negatively impacts the performance of many of the standard methods for dimension reduction. In this work, we will develop a Poisson-model-based method for text dimension reduction in Chapter 2, sparse extensions of two exponential family methods of dimension reduction in Chapter 4 and Chapter 5, and finally a framework for constructing, comparing and deriving asymptotic results for a class of estimators known as tensor estimating equations in Chapter 6 which we use to analyse several exponential family dimension reduction methods.

In this chapter, we will begin by introducing the fundamental notions of dimension reduction in Section 1.1 and text data in Section 1.2. We then motivate the desire to include a notion of sparsity in our dimension reduction techniques in Section 1.3. For our focus on text data, we will be working extensively with the exponential family of distributions, which we introduce in Section 1.4. Finally, we examine related work in the literature in Section 1.5. The methods in this section will be divided into those with distributional assumptions in Section 1.5.1 and those without in Section 1.5.2.

1.1 Dimension Reduction

Dimension reduction methods can broadly be divided into supervised and unsupervised methods; i.e. those which take into account the values of some "response" variable, and those which do not. For the former, we will take as our canonical example the method of sliced inverse regression (SIR), which is within the family of "sufficient dimension reduction" techniques. For the latter, we will use Principal Component Analysis (PCA).

Before exploring the details of SIR, we will take a detour into the general idea of sufficient dimension reduction, as this will provide context both for SIR and for the method of Poisson Inverse Regression (PoIR) in Chapter 2.

1.1.1 Sufficient Dimension Reduction

The family of sufficient dimension reduction methods is a family of supervised dimension reduction techniques with the explicit aim of finding a projection of the original data into a lower-dimensional space such that the projected data is statistically sufficient. Formally, we give the definition:

Definition 1.1.1. Let $\gamma : \mathscr{X} \to \mathscr{Z}$, where $\mathscr{X}$ is the domain of a random vector $\mathfrak{X}$ and $\mathscr{Z}$ is the domain of the random vector $\gamma(\mathfrak{X})$ satisfying $|\mathscr{X}| < |\mathscr{Z}|$. Then γ is a sufficient dimension reduction mapping with respect to a random variable $\mathcal{Y}$ if

$$\mathcal{Y} \perp\!\!\!\perp \mathfrak{X} \mid \gamma(\mathfrak{X})$$

There is a strong advantage to this family in the application of predictive influence; by preserving all information that the observed data has about the response with our dimension reduction, we are able to perform the same quality of predictive influence with the lower-dimensional data.

Given this definition, it can be seen that such transformations are not unique; if γ is a sufficient dimension reduction, then so is $a\gamma$, for any $a \in \mathbb{R}$
$\{0\}$. In the more restricted set of linear sufficient dimension reductions (which are projection matrices), we can define the notion of a "dimension reduction subspace". This encapsulates all full-rank linear transformations of the dimension reduction, bringing us closer to identifiability.

Definition 1.1.2. A "dimension reduction subspace" is the columnspace of a sufficient linear dimension reduction matrix.

A less obvious problem still remains; a suitably "small" sufficient dimension reduction can be made larger by adding superfluous components. This leads us to question if there is a "smallest" sufficient dimension reduction. This is addressed by the concept of a "central dimension reduction subspace", defined below.

Definition 1.1.3. The "central dimension reduction subspace" is the intersection of all dimension reduction subspaces.

If the central subspace exists, then it is a dimension reduction subspace with the minimal needed dimension. To see this, note that if a lower dimensional dimension

reduction subspace exists, then it is necessarily included in the intersection forming the central dimension reduction subspace, thus the central space cannot be of larger dimension. The dimension of the central subspace is often referred to as the "structural" dimension, as the central subspace is the minimal needed structure for the data (with respect to the response). Estimating this central subspace is the goal of a number of dimension reduction methods, notably Sliced Inverse Regression (Li 1991) which popularised the idea of sufficient dimension reduction and which we will now explore.

1.1.2 Sliced Inverse Regression

Sliced inverse regression Li (1991) begins with the assumption that the relationship between the predictors $\mathcal{X}$ and the response $\mathcal{Y}$ has the form

$$\mathcal{Y} = f\left(\beta_1\mathcal{X}, \ldots, \beta_k\mathcal{X}, \epsilon\right) \tag{1.1}$$

where ε represents noise. They also the additional restriction that

$$\mathbb{E}\left(\beta\mathcal{X} \mid \beta_1\mathcal{X}, \ldots, \beta_k\mathcal{X}\right) = c_0 + c_1\beta\mathcal{X} + \ldots + c_k\beta_k\mathcal{X} \tag{1.2}$$

for all $\beta \in \mathbb{R}^p$ with $c_0, \ldots, c_k$ real constants. This assumption is often referred to in the literature as the "linear conditional expectation" condition; it was shown in Eaton (1986) that this is condition is satisfied if and only if $\mathcal{X}$ has an elliptically symmetric distribution. The central result of sliced inverse regression is then the following theorem.

Theorem 1.1.1. *Under (1.1) and (1.2), the centred inverse regression curve* $\mathbb{E}(\mathcal{X} \mid \mathcal{Y}) - \mathbb{E}(\mathcal{X})$ *lies within the central dimension reduction subspace.*

Remark. A brief history note: the notion of dimension reduction subspaces and the central dimension reduction subspace was developed after the publication of sliced inverse regression. As such, Theorem 1.1.1 is stated differently in its original published form.

Remark. The word "sliced" in sliced inverse regression refers to part of the estimation procedure, in which the response $\mathcal{Y}$ is sliced into H bands and the sample mean of the observed predictors (after standardisation) is calculated within each slice. These sample conditional means are then formed into a covariance matrix, weighted by the proportion of observations in each slice. Finally, the directions $\beta_1, \ldots, \beta_k$ are then estimated from the eigenvectors with the largest eigenvalues.

1.1.3 Principal Component Analysis

PCA was first formulated in Pearson (1901) and (perhaps more importantly) reformulated in Hotelling (1933). It is often formulated in terms of finding a successive series

of orthogonal directions which maximise variance in turn – the first direction is in the direction of greatest variance, the second direction in the direction of greatest variance which is orthogonal to the first, and so on. However, there is an equivalent definition which is more suited to our purposes. Formally, we define the PCA loadings matrix $\mathbf{U}$ (and the associated mean vector $\boldsymbol{\mu}$) by

$$\mathbf{U}, \boldsymbol{\mu} := \underset{\mathbf{U}\in\mathbb{R}^{p\times k}, \mathbf{U}^{\mathrm{T}}\mathbf{U}=\mathbf{I}, \boldsymbol{\mu}\in\mathbb{R}^{p}}{\arg\min} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \boldsymbol{\mu} - \mathbf{U}\mathbf{U}^{\mathrm{T}}(\mathbf{x}_i - \boldsymbol{\mu}) \right\|_2^2 \tag{1.3}$$

where $\mathbf{x}_i \in \mathbb{R}^p$ for $i = 1, \ldots, n$ are our observed data. This form was considered by Pearson for the optimal projection of the p-dimensional data into a k dimensional subspace under squared error loss. In this formulation, the optimal choice for $\boldsymbol{\mu}$ is $\bar{\mathbf{x}}$ (the sample mean), and for $\mathbf{U}$ the first k eigenvectors of the sample covariance matrix ordered descending by eigenvalue. Later, we will relate this form to the deviance of a particular Gaussian model for the data, and use that connection to extend PCA to all exponential family distributions. For now, we show in Figure 1.1 the result of applying PCA to some two-dimensional data (shown as black points). The line shows the first direction found by PCA, and the red points on the line are the projections of the data onto the line.

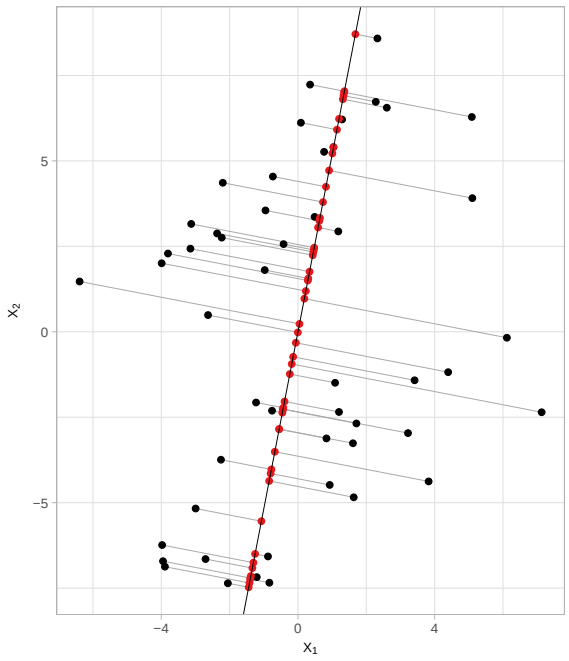


Figure 1.1: PCA direction and projections for two-dimensional data

It is worth noting that (1.3) is not the "typical" definition. The more usual definition is as the directions of maximal variance, which can be derived from the variance of the

standardised data. To be precise, we can define the first principal component by the weight vector $\mathbf{w}_1$ such that

$$\mathbf{w}_1 = \arg\max_{\|w\|=1} \sum_{i=1}^{n} (\mathbf{w} \cdot \mathbf{z}_i)^2 = \arg\max_{\|w\|=1} \mathbf{w}^{\mathrm{T}} \mathbf{Z}^{\mathrm{T}} \mathbf{Z} \mathbf{w} \tag{1.4}$$

where $\mathbf{Z}$ is the matrix with each row an observation $\mathbf{z}_i^{\mathrm{T}}$ and each $vecz_i$ is defined by $\mathbf{z}_i = \mathbf{x}_i - \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$. It can then be shown that the maximal value is the largest eigenvalue of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ which is obtained at the corresponding eigenvector.

The k^{th} component is then defined as the first principal component of the matrix

$$\hat{\mathbf{Z}}_k = \mathbf{Z} - \sum_{i=1}^{k-1} \mathbf{Z}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}$$

and it can be further shown that this is precisely the k^{th} eigenvector of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ when sorted by decreasing eigenvalue.

Remark. It is worth noting that the usual computational method for calculating the principal components is via singular value decomposition of $\mathbf{Z}$, as this avoids the need to calculate $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ which can be computationally expensive when the number of observations and/or dimension of them is high.

Remark. It is also important to note that the principal components are not generally unique. Trivially, one can note that if $\mathbf{w}$ is a principal component, then $-\mathbf{w}$ could replace it.

In general, dimension reduction methods are transformations from a high-dimensional feature space to a lower-dimensional feature space. We require an important restriction, however: these transformations must have some form of data-driven optimality. In the case of PCA, this is the minimised squared reconstruction error (or the maximised successive variances). Here, the reconstruction error means the L^2 distance between the original data and the reconstruction from the lower dimensional approximation. Our hope is that given data and a dimension reduction method with some appropriate measure of optimality, we can find a lower-dimensional representation of the data which is suitable for further statistical analysis or methodology which would have been infeasible or unsatisfactory with the original data.

1.2 Text Data

The majority of this work (specifically Chapters 2, 4 and 4) will be motivated and demonstrated using text data, under a Poisson counts model. For edification, we will now intro-

duce some key information about text data and how it can be studied from a statistical perspective.

We will use, as is standard in the literature and especially in practice, the vector space model" of text data, where documents are represented by vectors in $\mathbb{N}^d$. Each element of the vector corresponds to a stem, with the value being the number of times that stem appears in the document. A stem is effectively a word reduced down to an immutable part which does not change under conjugation, pluralisation, etc. Before vector representation, documents undergo the process of stemming, where each word is reduced to the appropriate stem. For example, the words "go", "going" and "gone" would typically all be reduced to the stem "go". This helps reduce the size of the vectors significantly, whilst preserving (by and large) meaning. Documents also typically have "stopwords" removed before stemming. Stopwords usually include "and", "the', etc. These words usually do not convey any of the sentiment or meaning of the document.

As a consequence of removing such stopwords, the remaining stems typically have quite low frequencies of occurrence. This is a key point to note for this work; with a sufficiently large λ, the $\text{Po}(\lambda)$ distribution is well-approximated by a normal distribution with mean and variance both equal to λ. This approximation performs much more poorly with small λ, which is our experience of the typical case with text data after stemming and stopword removal. Empirically, we have found that infrequent but highly informative words can have $\lambda < 1$, necessitating an alternative treatment to the normal approximation.

This poor approximation by the normal distribution is precisely why we, and other authors, have devoted time to studying extensions of PCA which are derived from an assumption of alternative exponential-family distributions. It is our expectation that when a normal approximation is inappropriate, so too is the standard PCA which, while not imposing any formal distributional assumptions, is deeply connected to the normal distribution's log-likelihood.

It is worth noting that instead of stems of individual words, it is also possible to consider stems of n words, called n-grams. However, the same ideas for modelling these apply, it is merely a matter of which one chooses to be the base unit of meaning. It is also possible to work without stemming under the premise that stemming could throw away important information which ought to be preserved. The main danger of not stemming is possible diffusing the importance that would be assigned to one stem across multiple words to the extent that they appear unimportant, whilst the stem would be important.

This representation of text data does not lend itself to modelling by continuous distributions, so we will need to develop techniques for dimension reduction which are appropriate for it. In particular, PCA (for reasons which will be discussed later) is gen-

erally unsuitable.

1.3 Sparsity

We will now address the main motivation of Chapters 4 and 5, the notion of sparse loadings. Consider the example of PCA – each new variable is a linear combination of the original variables. In general, the coefficients of those linear combinations will be almost entirely non-zero. Our assumption is that this is typically not necessary, but is an artefact of overfitting to the observed data, and that reducing the number of non-zero coefficients can be used to improve the generalisation performance of PCA on new data. In fact, this is the premise of Zou, Hastie, and Tibshirani (2006) which proposes a method for Sparse Principal Component Analysis (SPCA).

To elaborate, our assumption that sparsity will improve generalisation performance arises from the complications of working with finite sample sizes. Typically we work with estimators which are unbiased, but in the finite sample case we should expect that they will be influenced by the peculiarities of the observed data. Imposing sparsity constraints is an attempt to counteract these finite sample peculiarities by requiring a simple structure to our estimands unless there is sufficient evidence in the data to outweigh the preference for sparsity.

There is an additional benefit to sparse loadings, beyond that of improved performance; sparse loadings are generally more interpretable. In a typical setup, having thousands of non-zero components makes it difficult to interpret how the original variables are being used in the new (dimension reduced) variables. Reducing the number of non-zero components means fewer original variables are involved in the construction of each new one; this can make it possible to give a subject-specific explanation for what the new variables represent.

Generally, sparse loadings can be achieved by adding a penalty to the optimisation procedure. Popular choices are the L^1 and Smoothly Clipped Absolute Deviation (SCAD) penalties which penalise the total magnitude of the components in slightly different ways and the L^0 penalty which penalises the number of non-zero components. We will revisit the first two of these penalties in Chapter 4, and the latter in Chapter 5.

1.4 Exponential Families

Much of the following work will be centred around the exponential family of distributions. This family of distributions encompasses the Gaussian and Poisson distributions amongst others; given our interest in applying dimension reduction to text data (which

we model with the Poisson) and the link between classical PCA and the Gaussian distribution, the exponential family seems a sensible choice.

We define an exponential family (EF) distribution by the form of its probability density (or mass) function. Let $\mathfrak{X}$ have an EF distribution with parameter θ (known as the *natural parameter*). Then its density function $f(\mathbf{x} \mid \theta)$ has the form

$$h(\mathbf{x}) \exp(\theta \cdot \mathbf{t}(\mathbf{x}) - b(\theta))$$

The function $\mathbf{t}(\mathbf{x})$ is a sufficient statistic for the data; in many cases this will simply be the identity function. The function b is called the *log-partition function*, and it normalises the distribution to have integral 1 over the whole domain. The function $h(\mathbf{x})$ is usually referred to as a *base measure*. It can be shown that $\mathbb{E}(\mathfrak{X}) = \nabla b(\theta)$ if $\mathbf{t}$ is the identity. In fact, more generally it is true that $\nabla b(\theta) = \mathbb{E}(\mathbf{t}(\mathfrak{X}))$, and the higher order moments of the sufficient statistic can also be derived from the log-partition function. We define $\mathbf{g}$ as the left inverse of ∇b (that is, $\mathbf{g}(\nabla b(\theta)) = \theta$) and call it the *canonical link function*. This function is of central important when studying EF distributions, as it tells us how to map between the expectation and the natural parameters. This has an obvious application, mapping the sample mean of the sufficient statistic to an estimate of the natural parameters.

In Table 1.1 we will now give a summary of a number of common EF distributions along with their corresponding log-partition functions, base measures, sufficient statistics and canonical link functions. In the table, we use the symbol θ to denote the *natural* parameter as a function of the typical parameter(s).

Table 1.1: Usual parameter, natural parameter, sufficient statistic, log-partition, and canonical link functions for some common univariate exponential family distributions.

Distribution	Parameter[a]	θ	$h(x)$	$t(x)$	$b(\theta)$	$g(\theta)$
Poisson	λ	$\log(\lambda)$	$1/x!$	x	e^{θ}	$\log(\theta)$
Normal[b]	μ	μ/σ	$\frac{\exp\left(-x^2/2\sigma^2\right)}{\sigma\sqrt{2\pi}}$	x/σ	$\theta^2/2$	θ
Binomial[c]	p	$\log\frac{p}{1-p}$	$\binom{n}{x}$	x	$n\log\left(1+e^{\theta}\right)$	$\log\left(\frac{n}{\theta-n}\right)$
Exponential	λ	$-\lambda$	1	x	$-\log(-\theta)$	$-\frac{1}{\theta}$

[a] the "usual" parameter
[b] with known variance σ^2
[c] with known number of trials n

With the canonical link function, we can define the so-called *saturated model*; given

a matrix of observations $\mathbf{X} \in \mathbb{R}^{n \times p}$, the saturated model assumes that each component x_{ij} (for $i = 1, \dots n$ and $j = 1, \dots p$) is an observation of a random variable $\mathcal{X}_{ij}$ from some EF distribution with natural parameter $\theta_{ij} = g_{ij}(x_{ij})$. Finally, we can define the *deviance*, a measure of goodness of fit for models. Let $\ell_{\mathrm{S}}(\mathbf{X})$ denote the log-likelihood of observations $\mathbf{X}$ under the saturated model and let $\ell_{\mathrm{A}}(\mathbf{X})$ denote the log-likelihood under some alternative model within the same exponential family. Then the deviance of that alternative model is

$$-2\left(\ell_{\mathrm{A}}(\mathbf{X}) - \ell_{\mathrm{S}}(\mathbf{X})\right)$$

Remark. Although we have discussed exponential families in a fair deal of generality, for the rest of this thesis we will work under the restriction that the sufficient statistic function $\mathbf{t}$ is the identity function. Furthermore, we will usually (but not always) be referring to univariate distributions.

1.5 Related Work

In this section, we will look at related work in the literature, beginning with methods that make assumptions about the distribution of the data in Section 1.5.1, then at those which do not make distributional assumptions in Section 1.5.2. Of the two methods we have already considered, Sliced Inverse Regression (SIR) falls mostly into the non-distributional category as its primary assumption is on the relationship between the observed predictors and the response (also one could argue that the linear conditional expectation assumption is distributional, due to its equivalence to the assumption of an elliptically symmetric distribution); PCA also most naturally fits into the non-distributional category as despite its close relationship with the normal log-likelihood, it is typically motivated and derived from a non-distributional standpoint.

1.5.1 Distributional Dimension Reduction

1.5.1.1 Probabilistic Principal Component Analysis

The first distributional method we will look at is "Probabilistic Principal Component Analysis", proposed in Tipping and Bishop (1999). Here, the authors relate PCA to factor analysis. They model the observed data by

$$\mathcal{X} \mid \mathbf{t} \sim \mathrm{N}\left(\mathbf{W}\mathbf{t} + \boldsymbol{\mu}, \sigma^2 \mathbf{I}\right) \tag{1.5}$$

where $\mathbf{I}$ is the identity matrix of appropriate dimension, $\mathbf{t}$ is a k dimensional latent factor (i.e. it is unobserved) and $\mathbf{W}$ is a $p \times k$ dimensional matrix (corresponding to

the dimension p of the observed variable $\mathcal{X}$). This is often referred to as a Gaussian model with isotropic noise. They also make the assumption that $\mathbf{t} \sim \mathrm{N}(0, \mathbf{I})$, where $\dim(\mathbf{t}) < \dim(\mathcal{X})$. Marginalising, this gives that

$$\mathcal{X} \sim \mathrm{N}\left(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}\right) \tag{1.6}$$

which allows $\boldsymbol{\mu}$ to be easily estimated by maximum likelihood as the sample mean of the observed data. Using Bayes rule, the authors show that the reversed conditional distribution is

$$\mathbf{t} \mid \mathcal{X} \sim \mathrm{N}\left(\mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathcal{X} - \boldsymbol{\mu}), \sigma^2\mathbf{M}^{-1}\right) \tag{1.7}$$

where $\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + \sigma^2\mathbf{I}$. It is then shown that the maximum likelihood estimator for $\mathbf{W}$ can be given in explicit form as

$$\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_k\left(\boldsymbol{\Lambda}_k - \sigma^2\mathbf{I}\right)^{1/2}\mathbf{R} \tag{1.8}$$

where $\mathbf{U}_k$ is the matrix formed from the k principal eigenvectors of the sample covariance of $\mathcal{X}$, $\boldsymbol{\Lambda}_k$ is a diagonal matrix consisting of the corresponding eigenvalues, and $\mathbf{R}$ is an arbitrary $k \times k$ orthogonal rotation matrix. In practice, the authors recommend choosing $\mathbf{R}$ as the identity matrix.

To relate this to the classical PCA, note that given (1.7) we can summarise $\mathbf{t} \mid \mathcal{X}$ by its mean

$$\mathbb{E}(\mathbf{t} \mid \mathcal{X}) = \mathbf{M}^{-1}\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}(\mathcal{X} - \boldsymbol{\mu}) \tag{1.9}$$

which, as $\sigma^2 \to 0$ (and thus $\mathbf{M} \to \left(\mathbf{W}_{\mathrm{ML}}^{\mathrm{T}}\mathbf{W}_{\mathrm{ML}}\right)$), becomes an orthogonal projection into latent space. This, then, is the recovery of the traditional PCA from the probabilistic version. It is worth noting that the model proposed by Tipping and Bishop (1999) does become singular (and thus undefined) as $\sigma^2 \to 0$ however.

1.5.1.2 Bayesian Principal Component Analysis

Fundamentally, Bayesian Principal Component Analysis (BPCA) (proposed by Bishop (1999)) is an extension of Probabilistic Principal Component Analysis (PPCA) which specifies a full Bayesian model for the latent variable model, shown in Figure 1.2. Here, σ^2, $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ are treated as constants to be estimated for simplicity; the main difference between BPCA and PPCA then is the distribution specified over $\mathbf{W}$. This distribution is given as

$$\mathbb{P}(\mathbf{W} \mid \boldsymbol{\alpha}) = \prod_{i=1}^{k-1}\left(\frac{\alpha_i}{2\pi}\right)^{d/2}\exp\left(-\frac{1}{2}\alpha_i\|\mathbf{w}_i\|^2\right) \tag{1.10}$$

where $\mathbf{W}$ is once again a $p \times k$ dimensional matrix as in PPCA, $\mathbf{w}_i$ is the i^{th} column of $\mathbf{W}$ and α_i is the i^{th} component of $\boldsymbol{\alpha}$. The motivation for this form of prior distribution is

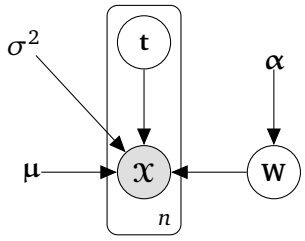


Figure 1.2: Plate diagram for Bayesian PCA

in Automatic Relevance Determination (ARD), proposed in Mackay (1995) initially for neural networks. It can be seen that α_i controls the precision (inverse variance) of the i^{th} column of $\mathbf{W}$, with an increasingly large value of α_i indicating that the probability mass is concentrated around the 0 vector. ARD proposes that sufficiently large values of α_i indicate that the i^{th} column of $\mathbf{W}$ is irrelevant, and should thus be removed from the model. This provides a significant advantage to BPCA, as it means in practice that one can set an initial value of k to $p-1$ (one less than the observed dimension of $\mathfrak{X}$) and allow ARD to remove columns from $\mathbf{W}$ to determine the appropriate value of k. This final k value is referred to by the authors as the "effective dimension".

1.5.1.3 Collins et al. 2002

The work of Collins et al. (2002) begins with the observation that PCA is equivalent to finding a set of vector parameters $\Theta_1, \ldots, \Theta_n$ which lie in a low-dimensional subspace from some observed values $x_1, \ldots, x_n$ which are corrupted by Gaussian noise. Using this, they extend the notion of PCA to the exponential family using the notion of Bregman divergences. They do this by noting that the conditional log-likelihood of an observation x given the natural parameter θ is

$$\log \mathbb{P}(x \mid \theta) = \log h(x) + x\theta - b(\theta) \tag{1.11}$$

using the notation from Section 1.4 and assuming that the sufficient statistic t is the identity. When maximising this log-likelihood with respect to θ, the term $\log h(x)$ is constant, and so can be disregarded. Therefore, the primary difference between exponential families is in the function $b(\theta)$. In order to relate this to Bregman divergences, we will first define what a Bregman divergence is.

Definition 1.5.1. Let $F : \Delta \to \mathbb{R}$ be a differentiable and strictly convex function on a closed, convex set $\Delta \subseteq \mathbb{R}$. Then for any $p, q \in real$, the Bregman divergence associated with F is

$$B_F(p||q) = F(p) - F(q) - f(q)(p-q) \tag{1.12}$$

where $f(x) = F'(x)$.

We then relate $b(\theta)$ to a "dual" Bregman divergence by the equation

$$F(b'(\theta)) + b(\theta) = b'(\theta)\theta \tag{1.13}$$

Collins et al. (2002) note that, under some fairly general conditions, $f(x) = [b']^{-1}(x)$ which, as noted in Section 1.4 is precisely the "canonical link function". Using this, we can see that the log-likelihood (1.11) can be rewritten as

$$\log \mathbb{P}(x \mid \theta) = \log h(x) + F(x) - B_F(x||\theta) \tag{1.14}$$

thus maximising the likelihood with respect to θ can be achieved simply by minimising the Bregman divergence $B_F(x||\theta)$. The authors overload the Bregman divergence notation for vectors by

$$B_F(\mathbf{v}||\mathbf{w}) = \sum_{i=1}^{p} B_F(v_i||w_i) \tag{1.15}$$

for all $\mathbf{v}, \mathbf{w} \in \mathbb{R}^p$, and for matrices by

$$B_F(\mathbf{V}||\mathbf{W}) = \sum_{i=1}^{p}\sum_{j=1}^{q} B_F\left(V_{ij}||W_{ij}\right) \tag{1.16}$$

for all $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{p\times q}$. Finally, we define $\mathbf{\Theta} = \mathbf{AV}$ as the $n \times p$ matrix of natural parameters, such that the i^{th} row is the vector of natural parameters associated with the i^{th} observation $\mathbf{x}_i$. Here, we have $\mathbf{A} \in \mathbb{R}^{n\times k}$ and $\mathbf{V} \in \mathbb{R}^{k\times p}$, with $k < p$, and p the dimension of the observed variables $\mathbf{x}_i, i = 1, \ldots, n$. Then the log-likelihood can be written as

$$\sum_{i=1}^{n} B_F(\mathbf{x}_i||g(\mathbf{\Theta}_i)) \tag{1.17}$$

where $\mathbf{\Theta}_i$ is the i^{th} row of $\mathbf{\Theta}$. One can then think of $\mathbf{V}$ defining a lower-dimensional basis for the surface $Q(\mathbf{V}) = \left\{g(\mathbf{aV})|\mathbf{a} \in \mathbb{R}^k\right\}$, such that this surface passes close to each observation $\mathbf{x}_i$, and each row of $\mathbf{A}$ giving the coefficients with respect to that basis of the closest point on $Q(\mathbf{V})$ to $\mathbf{x}_i$. Thus, $\mathbf{A}$ gives us the lower dimensional "projection" of the observed data.

1.5.1.4 Bayesian Exponential Principal Component Analysis

The method of Mohamed et al. (2009), which they call Bayesian Exponential Principal Component Analysis (BXPCA), shares a similar approach to that of Collins et al. (2002) but from a Bayesian perspective. In Figure 1.3 we show the plate diagram for the model.

To elaborate on the model, we have

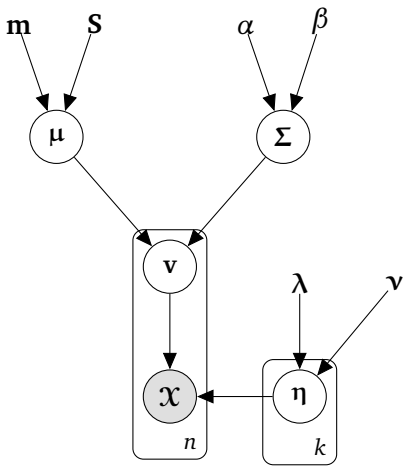


Figure 1.3: Plate diagram for Bayesian Exponential Family PCA

- $\boldsymbol{\mu}$ is drawn from a normal distribution with mean $\mathbf{m}$ and covariance matrix $\mathbf{S}$
- $\boldsymbol{\Sigma}$ is a diagonal matrix with the i^{th} diagonal element drawn from an inverse gamma distribution parametrised by α and β
- Each of the n vectors $\mathbf{v}_i$ $(i = 1, \ldots, n)$ are drawn from a normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
- Each observation $\mathbf{x}_i$ of the random variable $\mathcal{X}$ is drawn from an exponential family distribution with natural parameter $\sum_{j=1}^{k} v_{ij}\boldsymbol{\eta}_j$, where v_{ij} is the j^{th} element of $\mathbf{v}_i$
- Each $\boldsymbol{\eta}_i$ $(i = 1, \ldots, k)$ is drawn from the conjugate prior distribution corresponding to the exponential family distribution of $\mathcal{X}$

Having specified a full Bayesian model, the authors then propose a hybrid Monte Carlo for estimating the model parameters. Then the matrix $\mathbf{V}$, formed by the vectors $\mathbf{v}_i, i = 1, \ldots n$ as rows, is the analogue of the data after projection by PCA.

1.5.1.5 Simple Exponential Principal Component Analysis

Remark. Chapter 5 is an extension of this method, as such a detailed technical description will be given there. A brief description is given here for completeness.

Simple Exponential Principal Component Analysis (SePCA), put forward in Li and Tao (2013), uses a very similar model to that of Bishop (1999) in BPCA. The key difference is that, rather than the observed data having mean given by the product of the latent factors, it is the natural parameter which is given by this. As such, in the Gaussian case this method reproduces BPCA as a special case, while being able to model data

from any exponential family distribution. This method is also suitable for ARD, and the authors give details of how to apply ARD to automatically determine an appropriate dimension for the reduced dimension data.

1.5.1.6 Generalised Principal Component Analysis

Remark. Chapter 4 is an extension of this method, as such a detailed technical description will be given there. A brief description is given here for completeness.

In Landgraf and Lee (2015b) (an extension of Landgraf and Lee (2015a)), the authors propose an extension of PCA to the exponential family. Like Collins et al. (2002), this begins by recognising a connection between PCA and the estimation of the parameters of a Gaussian distribution. Generalised Principal Component Analysis (GPCA), however, uses a different form for the decomposition of the natural parameters in their extension to the exponential family, and optimises by minimising the deviance from the saturated model.

1.5.1.7 Sparse Probabilistic Principal Component Analysis

This method from Guan and Dy (2009) essentially extends PPCA (Tipping and Bishop 1999) in much the way that SPCA (detailed in Section 1.5.2.1) extended PCA. Specifically, they use the same model as in PPCA (Section 1.5.1.1), with a choice of additional prior distributions over $\mathbf{W}$.

The first proposed prior is a two-level hierarchical prior, equivalent to a Laplacian prior; the first level is a normal prior such that $\mathbf{W}_{ij}|z_{ij} \sim \mathrm{N}\left(0, z_{ij}\right)$, the second level is an exponential distribution on each z_{ij} such that $z_{ij} \sim \mathrm{Exp}(\lambda)$. When marginalising out the z_{ij}, this is equivalent to a Laplacian prior directly on $\mathbf{W}$. This prior is similar to the ARD prior, such as is used in BPCA, although this was not explored in the original paper.

The authors also propose an inverse-Gaussian prior and a Jeffrey's prior (that is, an uninformative prior proportional to the square root of the determinant of the Fisher information matrix). They show that these three priors are equivalent to sparsity-inducing penalty functions; the precise equivalences are shown in Table 1.2.

1.5.1.8 Sparse Exponential Family Principal Component Analysis

In Lu et al. (2016), the authors propose the method Sparse Exponential Family Principal Component Analysis (SEFPCA) as an ideological extension of SPCA to EF distributions. Denoting observed data by $\mathbf{X}$ composed of n observations $\mathbf{x}_i, i = 1, \ldots, n$ as the rows, they specify the form of the natural parameters for the i^{th} observation as $\mathbf{W}^{\mathrm{T}}\mathbf{z}_i + \boldsymbol{\mu}$.

Prior	Equivalent Penalty
Hierarchical Laplace	$\sum_i \sum_j \lambda \lvert \mathbf{W}_{ij} \rvert$
Inverse Gaussian	$-\frac{1}{2} \sum_i \sum_j \left(\log\left(\mathbf{W}_{ij}^2 + \lambda \right) + \log\left(K_1 \left(\frac{\sqrt{\lambda}\sqrt{\mathbf{W}_{ij}^2 + \lambda}}{\mu} \right) \right) \right)$
Jeffrey's	$\sum_i \sum_j \log\left(\mathbf{z}_{ij} \right)$

Table 1.2: Equivalences between priors on $\mathbf{W}$ and penalty functions for Sparse Probabilistic PCA. Here, K_1 is the modified Bessel function of the second kind with order 1 and λ is a penalty controlling the strength of the penalty function.

Here, $\boldsymbol{\mu}$ specifies the "average" natural parameters which are then diverged from by the lower-dimensional latent variable $\mathbf{z}_i$ through $\mathbf{W}$.

Under this specification, SEFPCA seeks to minimise the function

$$\sum_n b\left(\mathbf{W}^{\mathrm{T}} \mathbf{z}_n + \boldsymbol{\mu} \right) - \mathrm{Trace}\left(\left(\mathbf{Z}\mathbf{W} + \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}} \right) \mathbf{X}^{\mathrm{T}} \right) + P\left(\mathbf{W}, \boldsymbol{\mu} \right) \tag{1.18}$$

where b is the log-partition function as described in Section 1.4 which is specified by the choice of exponential family distribution, and $P\left(\mathbf{W}, \boldsymbol{\mu} \right)$ is a penalty function to induce sparsity. The form of this penalty function is

$$P\left(\mathbf{W}, \mathbf{b} \right) := \lambda_0 \left\| \mathbf{Z}\mathbf{W} + \mathbf{1}\mathbf{b}^{\mathrm{T}} \right\|_2^2 + \sum_{i=1}^{k} \lambda_i \left| \mathbf{W}_i \right| \tag{1.19}$$

which, in addition to the sparsifying effect, also helps with the stability of the estimation process when the dimension of the data is higher than the number of observations.

1.5.1.9 Multinomial Inverse Regression

Moving away from methods derived from PCA, we will now look at a method specifically designed for dimension reduction of text data. In Taddy (2013) and Taddy (2015), Taddy introduces Multinomial Inverse Regression (MNIR), a method for supervised dimension reduction of text data, based around a multinomial topic model. Specifically, the model is

- $\mathbf{x}_i \sim \mathrm{MN}\left(\mathbf{q}_i, m_i \right)$ for $i = 1, \ldots, n$, where $\mathbf{x}_i$ is the i^{th} observation
- $q_{ij} = \frac{\exp\left(\eta_{ij} \right)}{\sum_{l=1}^{p} \exp\left(\eta_{ij} \right)}$ is the probability of the j^{th} term appearing in the i^{th} observation, $j = 1, \ldots, p$
- $\eta_{ij} = \alpha_j + u_{ij} + \mathbf{v}_i^{\mathrm{T}} \boldsymbol{\phi}_j$

Here the $\mathbf{v}_i$ are k-dimensional "response factors" which capture the dependency of the term counts on the response $\mathcal{Y}$. That is, the $\mathbf{v}_i$ are (possibly random) functions dependent on $\mathcal{Y}$. The terms u_{ij} can be assembled into vectors $\mathbf{u}_i = \left[u_{i1}, \ldots, u_{ip}\right]^{\mathrm{T}}$ of "subject effects". In this way, the individual term probabilities are decomposed into a "mean", a subject specific effect, and the response specific effect introduced via the inverse regression coefficients $\boldsymbol{\phi}_j$.

In order to estimate these parameters, Taddy specifies a full Bayesian model of priors for $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p]^{\mathrm{T}}$, $\boldsymbol{\Phi} = \left[\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_p\right]^{\mathrm{T}}$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]^{\mathrm{T}}$. In particular, the model specified is

- $\alpha_j \sim \mathrm{N}(0, 1)$, which identifies the logistic multinomial model, removing the need to specify a null category
- $\phi_{jk} \sim \mathrm{Lap}\left(0, 1/\lambda_{jk}\right)$, where each Laplace distribution is independent of the others and the λ_{jk} are coefficient-specific precision parameters
- $\lambda_{jk} \sim \mathrm{Gamma}(s, r)$, where the s and r are shared hyperparameters for the shape and rate of the gamma distribution respectively
- $\exp\left(u_{ij}\right) \sim \mathrm{Gamma}(1, 1)$

Importantly, Taddy shows that this model produces a sufficient dimension reduction, where

$$y_i \perp\!\!\!\perp \mathbf{x}_i \mid \mathbf{v}_i \Rightarrow y_i \perp\!\!\!\perp \mathbf{x}_i | \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{x}_i \tag{1.20}$$

They also develop a bespoke optimisation algorithm for solving this problem, stating that the typical approaches to Bayesian inference are generally too slow for application to text data which tends to be very high dimensional.

1.5.1.10 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA), put forward by Blei et al. (2003), is a generative probabilistic model for text, displayed graphically as a plate diagram in Figure 1.4

To elaborate, for the d^{th} "document" (that is, for each observation), we assume a vector $\boldsymbol{\theta}_d$ of topic probabilities is drawn from a Dirichlet distribution with parameter $\boldsymbol{\alpha}$. Each of the n_d words within that document are generated by drawing a "topic" from a multinomial distribution with topic probabilities $\boldsymbol{\theta}_d$, then drawing a word from a multinomial distribution with probabilities $\boldsymbol{\beta}_z$. Here, $\boldsymbol{\beta}_z$ is the z^{th} column of matrix $\boldsymbol{\beta}$ of word probabilities.

From a dimension reduction point of view, it is the posterior distribution of $\boldsymbol{\theta}_d$ for $d = 1, \ldots, D$ which is of interest. In particular, we can use the posterior mean for each

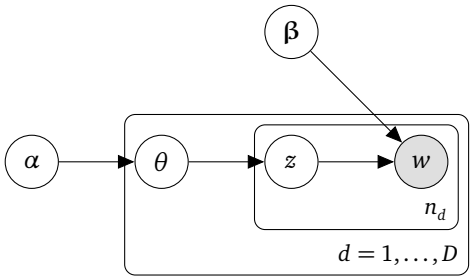


Figure 1.4: Plate diagram for Latent Dirichlet Allocation

θ_d as a lower-dimensional representation of the document. As the number of topics should be significantly lower than the number of words, this should achieve a significant reduction in the dimension of the data.

1.5.2 Non-Distributional Dimension Reduction

In contrast to the methods of the previous subsection, which have distributional assumptions at their heart, these methods are motivated primarily from a non-distributional perspective.

1.5.2.1 Sparse Principal Component Analysis

The idea behind SPCA, formulated by Zou, Hastie, and Tibshirani (2006), is to create sparse PCA loadings by enforcing an elastic net (Zou and Hastie 2005) penalty on the components of the loading matrix. To state this more precisely, the "SPCA Criterion" they gave was to find an approximation of some data $\mathbf{X} \in \mathbb{R}^{n \times p}$ of the form $\mathbf{XBA}^{\mathrm{T}}$ with $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{p \times k}$ which minimises

$$\|\mathbf{X} - \mathbf{XBA}^{\mathrm{T}}\|_2^2 + \lambda \sum_{i=1}^{k} \|\boldsymbol{\beta}_i\|_2^2 + \sum_{j=1}^{k} \lambda_{1,j} \|\boldsymbol{\beta}_j\|_1 \tag{1.21}$$

subject to the constraint $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbb{I}$, and where the vectors $\boldsymbol{\beta}_i$ are the columns of $\mathbf{B}$. The loadings matrix is $\mathbf{B}$, and the two summation terms are the elastic net penalty, a combination of the L^1 and L^2 penalties which usually gives superior performance to either of them in isolation. This penalty has been widely applied in regression analysis with success; given the regression-like formulation of PCA in minimising squared error the authors had good reason to suspect that it would be effective in this context.

The critical difficulty in applying SPCA is in the number of tuning parameters, with one parameter for the L^2 penalty, and k for the L^1 penalty. In practice, the authors suggest choosing λ a small, positive number for data with $n > p$ where it mainly fulfils

a role in helping to reduce potential collinearity problems with the data. In order to choose the values $\{\lambda_{1,j}, j = 1, \dots k\}$, the authors suggest trying a range, determining the best choice by balancing explained variance against sparsity.

1.5.2.2 Joint Sparse Principal Component Analysis

The idea of Joint Sparse Principal Component Analysis (JSPCA) (Yi et al. 2017) is similar in essence to SPCA in using penalisation to induce sparsity, but rather than use the squared L^2 norm, the authors use the unsquared L^2 norm directly for both the penalty and the reconstruction error. In contrast to SPCA, they also do not use L^1 penalisation on the components. To be precise, they solve the optimisation problem

$$\underset{\mathbf{A},\mathbf{B}\in\mathbb{R}^{p\times k}}{\arg\min} \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}^{\mathrm{T}}\|_2 + \lambda \sum_{i=1}^{k} \|\boldsymbol{\beta}_i\|_2 \tag{1.22}$$

where again the vectors $\boldsymbol{\beta}_i$ $(i = 1, \dots, k)$ are the columns of $\mathbf{B}$. The first term is the reconstruction error. The authors' rationale for using the non-squared L^2 norm is to provide better robustness to outliers in the data. They use a classification problem to demonstrate this, generating a set of noise-corrupted images (with different types of noise) and applying (amongst others) classical PCA and their JSPCA. Each observation is then classified using a nearest-neighbours classifier. Using the classification accuracy as the evaluation metric, the authors show the relative performances across multiple datasets. In some of the authors' experiments, there is good evidence that JSPCA performs better than any other PCA variant on noise-corrupted data, while some of the experiments show more mixed results.

1.5.2.3 Robust Principal Component Analysis

Though not a sparse method, we include the work of Kwak (2008) here for its similar aim. The focus is also on a PCA variant which is robust to outliers, something which sparse methods often achieve. In contrast to JSPCA, Robust Principal Component Analysis (RPCA) achieves this from the framework of the projection variance maximisation formulation of PCA. In this formulation, the aim is to find the projection matrix $\mathbf{U} \in \mathbb{R}^{p\times k}$ which maximises

$$\max \|\mathbf{U}^{\mathrm{T}}\mathbf{X}\|_1 \tag{1.23}$$

where $\mathbf{X}$ is the matrix of observed data, and where we impose the additional constraint that $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$. The authors contrast this to another possible choice of L^1 PCA-like optimisation problems

$$\min_{\mathbf{U}\in\mathbb{R}^{p\times k}} \|\mathbf{X} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}\|_1 \tag{1.24}$$

which, while more robust to outliers, is not invariant to rotations. The criteria they use, (1.23), is invariant to rotations and is robust to outliers. It can be seen as maximising the (L^1) dispersion in the feature space as opposed to the input space.

It is important to note that, while maximising the L^2 dispersion in feature space and minimising the squared L^2 reconstruction error are duals to one another, and thus both result in classical PCA, the two problems (1.23) and (1.24) are not dual problems. This means that the calculated projections will not, in general, be equal.

1.5.2.4 Sparse Principal Component Analysis by Rotation and Truncation

Proposed by Hu et al. (2016), Sparse Principal Component Analysis by Rotation and Truncation (SPCArt) is an alternative sparse PCA method to SPCA. Unlike many of the methods discussed so far, the authors do not propose either a probabilistic model to estimate, or introduce a penalisation of the PCA optimality criterion.

Instead, their method derives from a matrix approximation problem. The authors make use of the fact that $\text{Span}(\mathbf{B}) = \text{Span}(\mathbf{RB})$ for all full-rank rotation matrices $\mathbf{R}$. In particular, rotating the PCA loadings by a full-rank rotation produces an orthogonal basis which spans the same subspace. The authors propose to solve

$$\min_{\mathbf{R}\in\mathbb{R}^{k\times k}, \mathbf{R}^{\mathrm{T}}\mathbf{R}=\mathbf{I}} \|\mathbf{V}_{1:k}\mathbf{R}\|_0 \tag{1.25}$$

where $\mathbf{V}_{1:k}$ is the matrix of the k principal components, and the $\|\cdot\|_0$ norm is applied to the matrix by summing the column norms. Due to the difficulty of solving this problem, the authors solve the approximation

$$\min_{\mathbf{R}\in\mathbb{R}^{k\times k}, \mathbf{P}\in\mathbb{R}^{n\times k}} \frac{1}{2}\|\mathbf{V}_{1:k} - \mathbf{PR}\|_F^2 + \lambda \sum_{i=1}^{n} \|\mathbf{P}_i\|_1 \tag{1.26}$$

subject to $\|\mathbf{P}_i\|_2 = 1$ for $i = 1, \ldots, n$ and $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$, where $\|\cdot\|_F$ is the Frobenius norm. Here, the approximation to $\mathbf{V}_{1:k}$ is given by $\mathbf{PR}$, and in general it may not be orthogonal (though for a sufficiently good approximation, it should be close to orthogonal).

As part of the solution process to this approximation of the original problem, thresholding takes place. Although the solution properly calls for a type of soft thresholding, the authors propose four different types (including the soft thresholding), each of which possesses different properties. Of note is one type of thresholding, which the authors call “truncation by sparsity” which allows direct control over the sparsification of the result.

1.5.2.5 Non-negative Matrix Factorisation

Like SPCArt, Nonnegative Matrix Factorisation (NMF) is a method based around matrix approximation, rather than a PCA-like criteria to optimise. There are a number of distinct NMF algorithms (see Gillis (2014) for an overview), but they all focus on solving the same problem. That problem is the decomposition of a non-negative matrix $\mathbf{X}$ into the product of two non-negative matrices $\mathbf{W}$ and $\mathbf{H}$, such that $\mathbf{X} \approx \mathbf{WH}$ with $\mathbf{W}$ a $n \times k$ matrix and $\mathbf{H}$ a $k \times p$ matrix. Here we expect $k < p$. Then $\mathbf{H}$ can be interpreted as a basis matrix, with the rows of $\mathbf{W}$ giving the co-ordinates of each observation $\mathbf{x}$ forming the rows of $\mathbf{X}$ with respect to the basis $\mathbf{H}$.

In the context of text data, this factorisation has a pleasing interpretation; $\mathbf{H}$ corresponds to topics, with each column being counts (or frequencies) of terms appearing within each topic. We can then interpret $\mathbf{W}$ as decomposing each observation into its topics. Note that this interpretation is not strict, no conditions are enforced to make this interpretation exact, but it is often appropriate in practice.

In the context of dimension reduction, we can use the matrix $\mathbf{W}$ as a lower-dimensional representation of our observed data $\mathbf{X}$.

1.6 Conclusion

In this chapter we have introduced the concepts of text data, dimension reduction, and some of the popular related methods. In the following chapter, we will introduce a Poisson based model for text data, and from there specify a Bayesian model which will allow us to find a dimension reduction transformation for the data.

— Chapter 2 —

Poisson Inverse Regression

2.1 Introduction

As each column in a document-term matrix consists of observed counts, a natural idea is to model these variables as observations from a Poisson random variable. Conveniently, in Cook (2007) we find that such a model fits into a framework of exponential family inverse regression, which leads to a simple dimension reduction framework. In this vein, we will propose a simple Poisson model for text data, and from it show how it leads to a linear dimension reduction transformation which is statistically sufficient. The details of this will be given in Section 2.2; in Section 2.3 we will give two methods of estimating the linear transformation. Recalling the similar method of Multinomial Inverse Regression (MNIR) from Section 1.5, we will compare the two methods in Section 2.4, before concluding in Section 2.5

2.2 Poisson Inverse Regression

In this section, we will define the Poisson Inverse Regression (PoIR) model. We begin by discussing a very simple multi-variate extension to a Poisson model, then add dependence on the response through the natural parameter.

Recall that if λ is the mean of a Poisson distributed random variable (i.e. the usual parameter), then its natural parameter is $\kappa := \log(\lambda)$. We then have the form

$$\frac{1}{x!}\exp(x\kappa - \exp(\kappa))$$

for the probability mass function. We can extend this definition to vector-valued random variables $\mathcal{X}$ where each component is an independent Poisson distribution with natural parameter κ_i:

$$\frac{1}{\prod_{i=1}^{p} x_i!}\exp\left(\mathbf{x}\cdot\boldsymbol{\kappa} - \sum_{i=1}^{p}\exp(\kappa_i)\right) \tag{2.1}$$

where κ_i is the natural parameter for the i^{th} component of $\mathcal{X}$.

The inverse regression model we will use assumes that is it $\mathcal{X} \mid \mathcal{Y}$ which has a Poisson distribution. In order to capture this relationship, we assume that the natural parameter is a function of $\mathcal{Y}$. In particular, we assume the following form

$$\kappa_i(y) = \mu_i + \boldsymbol{\eta}_i^{\mathrm{T}} \boldsymbol{\nu}(y) \tag{2.2}$$

where y is the observed value of $\mathcal{Y}$. Here, $\boldsymbol{\eta}_i$ denotes the ith column of a matrix $\mathbf{H}$ and $\boldsymbol{\nu}(y)$ is some vector-valued function of y which, for the moment, we will place no restrictions on in terms of form. This form assumes that there is a central mean tendency to the natural parameter, captured by $\boldsymbol{\mu}$, which is shared across all values of $\mathcal{Y}$. Then the term $\boldsymbol{\eta}_i^{\mathrm{T}} \boldsymbol{\nu}(y)$ captures the divergence from that central tendency. If we briefly relate this back to the mean parameter of the Poisson distribution, then we can see that the "divergence" from the central tendency has a multiplicative effect on the expected count. In practice, this assumed form is difficult to check without making more restrictive assumptions.

We now present the following theorem, which validates the use of this method of dimension reduction.

Theorem 2.2.1. *If $\mathcal{X} \mid \mathcal{Y}$ has a Poisson distribution with natural parameter of the form (2.2), then $\mathbb{S}_{\mathbf{H}} \coloneqq \mathrm{ColSpace}(\mathbf{H})$ is a dimension reduction subspace under the definition in Def. 1.1.2.*

Proof. Substituting 2.2 into the model 2.1, we obtain

$$\begin{aligned}
&\frac{1}{\prod_{i=1}^{k} x_i!} \exp\left(\mathbf{x} \cdot \boldsymbol{\mu} + \boldsymbol{\nu}(y)^{\mathrm{T}} \mathbf{H}^{\mathrm{T}} \mathbf{x} - \sum_{i=1}^{k} \exp\left(\mu_i + \boldsymbol{\eta}_i^{\mathrm{T}} \boldsymbol{\nu}(y)\right) \right) \\
&= \left[\frac{1}{\prod_{i=1}^{k} x_i!} \exp\left(\mathbf{x} \cdot \boldsymbol{\mu} - \sum_{i=1}^{k} \exp(\mu_i) \right) \right] \times \left[\exp\left(\boldsymbol{\nu}(y)^{\mathrm{T}} \mathbf{H}^{\mathrm{T}} \mathbf{x} - \sum_{i=1}^{k} \exp\left(\boldsymbol{\nu}(y)^{\mathrm{T}} \boldsymbol{\eta}_i\right) \right) \right]
\end{aligned}$$

Notice that the left hand term is a function only of $\mathbf{x}$ and the right hand term is a function only of $\mathbf{H}^{\mathrm{T}}\mathbf{x}$ and y. Then through the Fisher-Neyman factorisation theorem for sufficiency we have that the distribution of $\mathcal{X} \mid \left(\mathbf{H}^{\mathrm{T}}\mathcal{X}, \mathcal{Y} = y\right)$ is the same as the distribution of $\mathcal{X} \mid \mathbf{H}^{\mathrm{T}}\mathcal{X}$ for all values of y. Consequently, $\mathcal{Y} \perp\!\!\!\perp \mathcal{X} \mid \mathbf{H}^{\mathrm{T}}\mathcal{X}$. □

Now that we have established that we can, indeed, use this model to perform dimension reduction on text data, it seems prudent to name it.

Definition 2.2.1. Let $\mathcal{X} \mid \mathcal{Y} = y \sim \text{Po}\left(\exp\left(\boldsymbol{\mu} + \boldsymbol{\nu}(y)^{\text{T}}\mathbf{H}\right)\right)$, where we overload the notation $\text{Po}(\cdot)$ for a vector parameter to mean independent Poisson distributions for each component of the random vector with mean given by the corresponding component of the vector parameter (and where we apply the exponential component-wise). This model is the "Poisson inverse regression" model.

In order to actually estimate this model, we will first establish a full Bayesian model which will be used to perform the estimation. To do this, we assign prior distributions to each of the parameters in the model. In order to encourage sparsity, we place a Laplace prior with mean 0 and on each component of $\mathbf{H}$. We use a shared rate parameter $1/h$ for each of these priors. For flexibility, we place an exponential hyperprior on h with a fixed rate parameter $1/\rho_h$ to be determined by experimentation. Our experience suggests that estimation is not too sensitive to the choice of ρ_h, so we may search across a coarse grid for the best value in respect to convergence of MCMC methods.

Similarly, we place a Laplace prior on each component of $\boldsymbol{\mu}$ with mean 0 and rate parameter $1/m$. This prior is flexible enough to accommodate a wide range of values for $\boldsymbol{\mu}$, while requiring increasingly more evidence from the observed data to deviate to particularly high or low values. Upon m we place an exponential hyperprior with rate parameter $1/\rho_m$.

For each value of y, we place an independent standard normal prior on $\boldsymbol{\nu}(y)$. In the case of discrete-valued responses this is easy to accomplish; in the continuous-valued case, we suggest slicing the response into discrete groups. This prior is quite uninformative; for a given dataset one may have a better understanding of the form this function should take, in this case practitioners can use that function directly rather than attempting to estimate it.

Using sequential conditioning, we can split the joint probability function into a multiplication of several probabilities whose random variables are conditionally independent as follows. Summarised, also, are the priors we have imposed.

$$\begin{aligned}\mathbb{P}(\mathbf{x}, y, h, \mathbf{H}, m, \boldsymbol{\mu}, \boldsymbol{\nu}) = \mathbb{P}(\mathbf{x} \mid y, h, \mathbf{H}, m, \boldsymbol{\mu}, \boldsymbol{\nu})\,\mathbb{P}(\mathbf{H} \mid h)\,\mathbb{P}(h) \\ \times \mathbb{P}(\boldsymbol{\mu} \mid m)\,\mathbb{P}(m)\,\mathbb{P}(\boldsymbol{\nu} \mid y)\,\mathbb{P}(y) \quad (2.3)\end{aligned}$$

$$\eta_{ij} \sim \text{Lap}\left(0, \frac{1}{h}\right) \tag{2.4}$$

$$\mu_i \sim \text{Lap}\left(0, \frac{1}{m}\right) \tag{2.5}$$

$$h \sim \text{Exp}\left(\frac{1}{\rho_h}\right) \tag{2.6}$$

$$m \sim \text{Exp}\left(\frac{1}{\rho_m}\right) \tag{2.7}$$

$$\nu_i \mid y \sim \text{N}(0,1) \tag{2.8}$$

$$\mathbf{x}_i \mid y, \boldsymbol{\mu}, \mathbf{H}, \boldsymbol{\nu} \sim \text{Po}\left(\exp\left(\mu_i + \boldsymbol{\eta}_i^{\text{T}} \boldsymbol{\nu}(y)\right)\right) \tag{2.9}$$

Here we use the convention that the pdf of a Laplace distributed variable with mean parameter μ and rate parameter b is

$$f(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

and the convention that the pdf of an Exponential random variable with rate parameter b has pdf

$$f(x) = \frac{1}{b} \exp\left(-\frac{x}{b}\right)$$

Having fully specified the Bayesian model, we can then proceed to use methods from Bayesian inference to estimate $\mathbf{H}$.

2.3 Estimation

We have used two main methods to estimate the value of $\mathbf{H}$. Firstly, we used maximum a posteriori estimation, where we maximise $\mathbb{P}(\boldsymbol{\mu}, \mathbf{H}, \boldsymbol{\nu}, m, h \mid \mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is a matrix of observed data and $\mathbf{y}$ is a vector of observed responses. Using (2.3) and Bayes's theorem we can compute the a posteriori probability. For simplicity, we have used an "off the shelf" optimisation routine in R to find the maximum values. However, it would be worth investing time in deriving a bespoke routine for calculating the MAP estimate, as was done in Taddy (2013). Note that this can be extended to vector responses in a straightforward manner, in which case we have a matrix of response values $\mathbf{Y}$

We also made use of Markov Chain Monte Carlo methods; the aim of such methods is to sample from a Markov chain whose equilibrium distribution is the posterior distribution of interest. In our case, that means that we wish to construct a Markov chain whose equilibrium distribution is the PoIR posterior distribution.

To be precise, a (discrete time) Markov chain is a type of stochastic process, where the distribution of possible values at some time-point t is independent of all previously

obtained values, except the value at $t-1$. This definition can be expanded to continuous time, but will not be necessary here. The formal definition is as follows:

Definition 2.3.1. A discrete-time Markov chain is given by a sequence of random variables $\mathcal{X}_1, \mathcal{X}_2, \ldots$ such that

$$\mathbb{P}(\mathcal{X}_t = \mathbf{x} \mid \mathcal{X}_1 = \mathbf{x}_1, \ldots, \mathcal{X}_{t-1} = \mathbf{x}_{t-1}) = \mathbb{P}(\mathcal{X}_t = \mathbf{x} \mid \mathcal{X}_{t-1} = \mathbf{x}_{t-1})$$

The equilibrium distribution is then given by $\lim_{t\to\infty} \mathbb{P}(\mathcal{X}_t)$; that is, the limiting unconditional probability density.

In Metropolis et al. (1953), a method was developed for constructing a Markov chain with the desired equilibrium distribution under some restrictions; the work of Hastings (1970) extended this to the general case. This method is known as the Metropolis-Hastings method for its progenitors. It requires the ability to calculate a function $f(\mathbf{x})$ which is proportional to the desired density $\mathbb{P}(\mathbf{x})$; in the Bayesian setting this function $f(\mathbf{x})$ can be taken to be the un-normalised posterior distribution. This allows us to avoid the (often difficult) task of calculating the integral required to normalise the posterior. To give some intuition, at time step t with current value $\mathbf{x}_t$, the method works by generating a proposed new value $\mathbf{x}'$ sampled from a "proposal distribution" $g(\mathbf{x}' \mid \mathbf{x}_t)$. This proposal distribution is often taken to be a normal distribution with mean $\mathbf{x}_t$ for simplicity. We then calculate the "acceptance ratio"

$$A = \min\left(\frac{f(\mathbf{x}')}{f(\mathbf{x}_t)}\frac{g(\mathbf{x}' \mid \mathbf{x}_t)}{g(\mathbf{x}_t \mid \mathbf{x}')}, 1\right) = \min\left(\frac{\mathbb{P}(\mathbf{x}')}{\mathbb{P}(\mathbf{x}_t)}\frac{g(\mathbf{x}' \mid \mathbf{x}_t)}{g(\mathbf{x}_t \mid \mathbf{x}')}, 1\right)$$

and "accept" the proposed point with probability A. If we do not accept the proposed point, we set $\mathbf{x}_{t+1} = \mathbf{x}_t$. If the proposal distribution is symmetric, then the acceptance ratio simplifies to

$$A = \min\left(\frac{f(\mathbf{x}')}{f(\mathbf{x}_t)}, 1\right) = \min\left(\frac{\mathbb{P}(\mathbf{x}')}{\mathbb{P}(\mathbf{x}_t)}\right)$$

We can then see that, roughly, this criteria means that as t increases, we are more likely to move to (or stay at) points that are more likely under the posterior likelihood.

We used a Gibbs-like version of the Metropolis-Hastings, in that we divided the sampling step up into a sampling step for each of the variables conditional on those variables previously sampled. This method derives from the work of Geman and Geman (1984); it works by sampling each individual component from the conditional distribution given the rest of the components. However, the intractability of sampling from the actual conditional distributions required us to use the Metropolis-Hastings style proposal-acceptance routine within each sampling sub-step. To expand on this, each sampling step was as follows:

1. Draw $h^{(*)}$ from a normal distribution left-truncated at 0 with mean $h^{(t)}$. Calculate the acceptance ratio

$$A := \frac{\mathbb{P}\left(h^{(*)} \mid \mathbf{X}, \mathbf{y}, \mathbf{H}^{(t)}, m^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}\right)}{\mathbb{P}\left(h^{(t)} \mid \mathbf{X}, \mathbf{y}, \mathbf{H}^{(t)}, m^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}\right)} \frac{\mathbb{P}\left(h^{(t)} \mid h^{(*)}\right)}{\mathbb{P}\left(h^{(*)} \mid h^{(t)}\right)}$$

where the right-hand fraction is the ratio of probabilities from the truncated-normal proposal distribution. With probability $\min(1, A)$ accept $h^{(*)}$ as $h^{(t+1)}$, otherwise let $h^{(t+1)} := h^{(t)}$

2. Draw $\mathbf{H}^{(*)}$ from an (uncorrelated) multivariate normal distribution with mean $\mathbf{H}^{(t)}$. Calculate the acceptance ratio

$$A := \frac{\mathbb{P}\left(\mathbf{H}^{(*)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, m^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}\right)}{\mathbb{P}\left(\mathbf{H}^{(t)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, m^{(t)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}\right)} \frac{\mathbb{P}\left(\mathbf{H}^{(t)} \mid \mathbf{H}^{(*)}\right)}{\mathbb{P}\left(\mathbf{H}^{(*)} \mid \mathbf{H}^{(t)}\right)}$$

and accept with probability $\min(1, A)$

3. Draw $m^{(*)}$ from a normal distribution left-truncated at 0 with mean $m^{(t)}$. Calculate acceptance ratio

$$A := \frac{\mathbb{P}\left(m^{(*)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, \mathbf{H}^{(t+1)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}\right)}{\mathbb{P}\left(m^{(t)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, \mathbf{H}^{(t+1)}, \boldsymbol{\mu}^{(t)}, \boldsymbol{\nu}^{(t)}\right)} \frac{\mathbb{P}\left(m^{(t)} \mid m^{(*)}\right)}{\mathbb{P}\left(m^{(*)} \mid m^{(t)}\right)}$$

and accept $m^{(*)}$ with probability $\min(1, A)$.

4. Draw $\boldsymbol{\mu}^{(*)}$ from an (uncorrelated) multivariate normal distribution with mean $\boldsymbol{\mu}^{(t)}$. Calculate

$$A := \frac{\mathbb{P}\left(\boldsymbol{\mu}^{(*)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, \mathbf{H}^{(t+1)}, m^{(t+1)}, \boldsymbol{\nu}^{(t)}\right)}{\mathbb{P}\left(\boldsymbol{\mu}^{(t)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, \mathbf{H}^{(t+1)}, m^{(t+1)}, \boldsymbol{\nu}^{(t)}\right)} \frac{\mathbb{P}\left(\boldsymbol{\mu}^{(t)} \mid \boldsymbol{\mu}^{(*)}\right)}{\mathbb{P}\left(\boldsymbol{\mu}^{(*)} \mid \boldsymbol{\mu}^{(t)}\right)}$$

accept $\boldsymbol{\mu}^{(*)}$ with probability $\min(1, A)$

5. For each value $y \in \mathscr{Y}$, draw an $\boldsymbol{\nu}_y^{(*)}$ from an uncorrelated multivariate normal distribution with mean $\boldsymbol{\nu}_y^{(t)}$. Calculate

$$A := \frac{\mathbb{P}\left(\boldsymbol{\nu}_y^{(*)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, \mathbf{H}^{(t+1)}, m^{(t+1)}, \boldsymbol{\mu}^{(t+1)}\right)}{\mathbb{P}\left(\boldsymbol{\nu} y^{(t)} \mid \mathbf{X}, \mathbf{y}, h^{(t+1)}, \mathbf{H}^{(t+1)}, m^{(t+1)}, \boldsymbol{\mu}^{(t+1)}\right)} \frac{\mathbb{P}\left(\boldsymbol{\nu}_y^{(t)} \mid \boldsymbol{\nu}_y^{(*)}\right)}{\mathbb{P}\left(\boldsymbol{\nu}_y^{(*)} \mid \boldsymbol{\nu}_y^{(t)}\right)}$$

and accept with probability $\min(1, A)$.

In order to evaluate convergence, we use the Gelman-Rubin $\hat{R}$ statistic Gelman and Rubin (1992). This statistic requires us to carry out C Markov Chain Monte Carlo processes all started from different, over-dispersed initial values. By overdispersed, we mean selected from a distribution with significantly higher variance than their prior distribution. This helps to ensure exploring a larger range of the parameter space, leading to better estimation. After discarding a set number of iterations for burn-in, we calculate the following quantities from the n samples each chain gives us.

$$W = \frac{1}{C}\sum_{j=1}^{C} s_j^2$$

$$B = \frac{n}{C-1}\sum_{j=1}^{C}\left(\bar{\theta}_j - \bar{\bar{\theta}}\right)^2$$

Where s_j is the variance in the jth chain, $\bar{\theta}_j$ is the sample mean from chain j and $\bar{\bar{\theta}}$ is the overall sample mean. For simplicity, we speak of a single parameter θ, but this is extended to multiple parameters component-wise. These quantities give us estimates of the within-chain and between-chain variances, respectively. We then calculate an estimate of the variance of the so-called stationary distribution.

$$\widehat{\mathbb{V}\mathrm{ar}(\theta)} = \left(1 - \frac{1}{n}\right)W + \frac{1}{n}B$$

Finally, we have

$$\hat{R} = \sqrt{\frac{\widehat{\mathbb{V}\mathrm{ar}(\theta)}}{W}}$$

The usual convention is to keep drawing samples (and possibly discarding more as burn-in samples) until the value of $\hat{R}$ is approximately 1 (usually < 1.1) for every parameter. Once we have that, we can pool our samples from each chain together and calculate the mean of each parameter. Asymptotically, this parameter mean is convergent to the expected value of that parameter over its true distribution.

Both the maximum a-posteriori (MAP) method and the Markov Chain Monte Carlo (MCMC) method were effective in estimating the parameters of the PoIR model. However, given that we are not as interested in the posterior distributions as we are in the posterior means, we will proceed to use the MAP as it is less computationally expensive.

2.4 Evaluation

The closest method to our formulated Poisson inverse regression is MNIR; in this section we will evaluate the performance of our method using MNIR as the benchmark. To

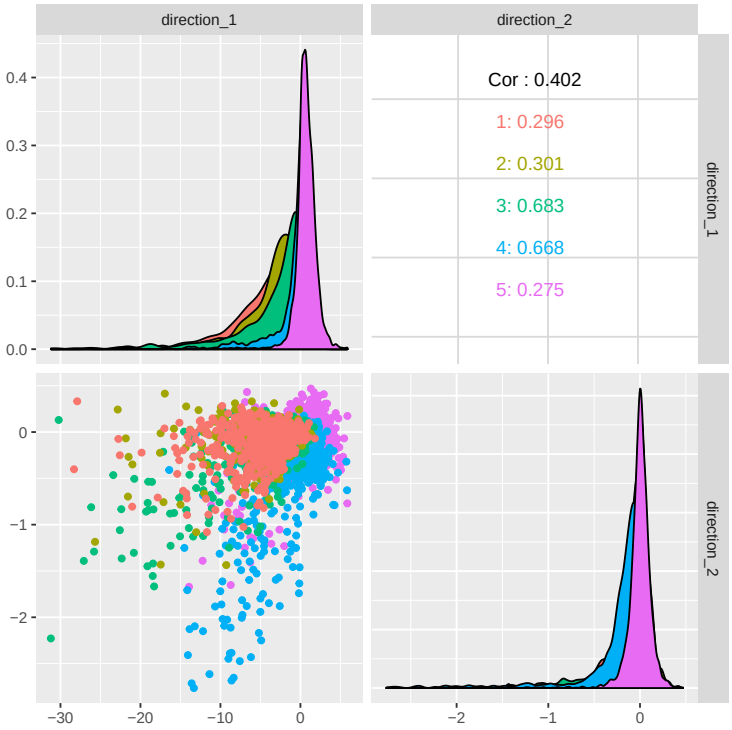


Figure 2.1: Plot of two directions recovered by POIR from the we8there data.

do this, we will use a test dataset used in Taddy (2013) and made available in the accompanying R package, namely the "we8there" dataset of restaurant reviews. The data, as made available in the "textir" package, is already preprocessed which makes it very convenient to work with. Figure 2.2 shows three directions recovered from the data using POIR, and Figure 2.3 shows the one direction MNIR is able to recover. The limitation of MNIR to one direction is due to us using only a one-dimensional response (the overall rating). MNIR is, as implemented in "textir" and as focused on in Taddy (2013), limited to recovering only as many directions as the dimension of the response variable. The way we have specified the POIR model leaves us able to estimate however many directions are appropriate.

We estimate the PoIR model using MAP, as we only need a point estimate. MNIR is estimated using the "textir" package method, detailed in Taddy (2013). In Figure 2.1 we show the results of estimating a two-dimensional PoIR model; a three-dimensional PoIR model is shown in Figure 2.2. These two figures are in the form of "pairs" plots, showing for each combination of two directions a scatterplot of the projected data (coloured according to the value of the response), the correlations between the two directions on a per-response-value level, as well as a one-dimensional density plot for each direction,

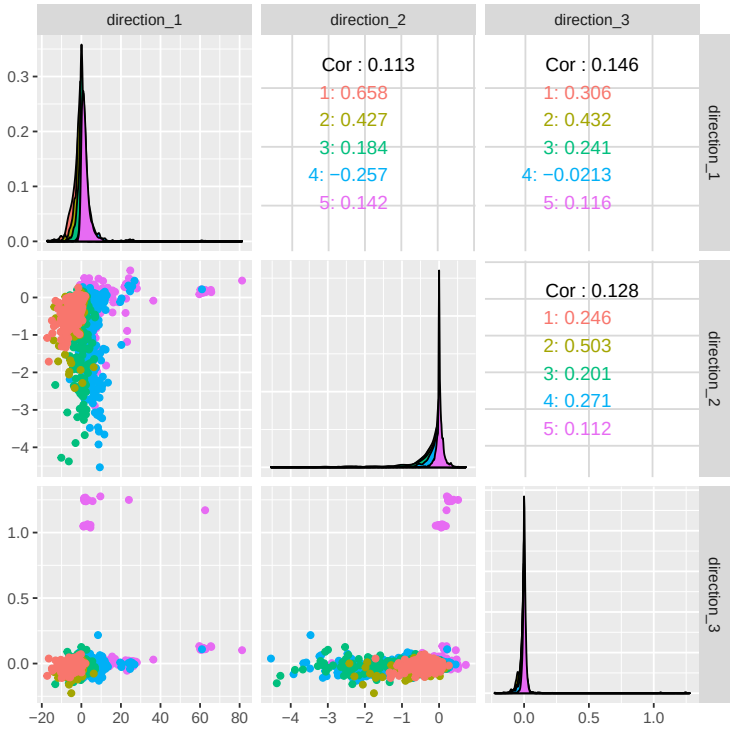


Figure 2.2: Plot of three directions recovered by POIR from the we8there data.

separated by response value. We also show the (one-dimensional) results from MNIR in Figure 2.3 in both a scatterplot of the direction against the rating, and as a density plot separated by the rating.

From Figure 2.1, we can see that two directions is not sufficient to distinguish the five response classes; in particular the three lowest ratings (1, 2, and 3 stars) are particularly difficult to distinguish from one another. It is no surprise, then, that the one-dimensional MNIR shown in Figure 2.3 also struggles significantly to distinguish the five response classes. On the other hand, Figure 2.2 shows the three-dimensional PoIR giving a more reasonable separation of the data.

As can be seen from the two figures, the extra dimensions we are able to extract using POIR give us significantly better separation of the five different response levels which are indicated using colours on both plots. One direction only, in the case of MNIR, leaves the different response levels largely inseparable; it would be difficult to identify correctly any more than some of the 1-rated and some of the 5-rated responses.

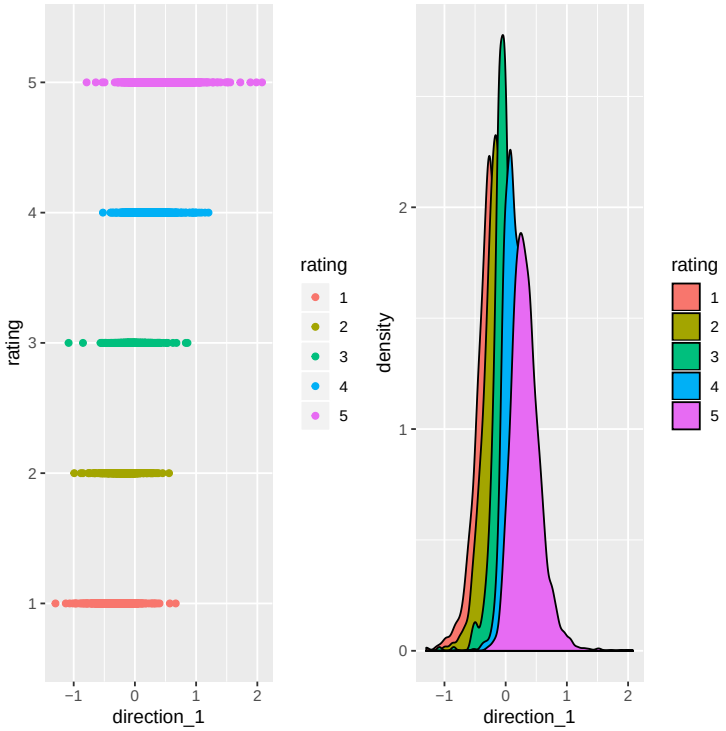


Figure 2.3: Scatterplot of the one MNIR direction recovered from the we8there data against the rating, and density of that direction separated by rating.

2.5 Conclusion

We have shown in Section 2.4 that our PoIR is an effective method for text data dimension reduction. In fact, from 2.2.1, we know that it is a sufficient dimension reduction. We have given two methods for estimation in Section 2.3, a method based on MAP, and one based on MCMC.

There are some interesting directions for future work. Firstly, the model for text data upon which PoIR is based is a very simple one which does not take into account potential correlation between terms. Secondly, the estimation algorithms given are not tuned for computational efficiency. As such, they are currently slower than the method provided in Taddy (2013) and then further improved in Taddy (2015) for MNIR. It should be possible to significantly decrease the computational time for the MAP estimation method by using a customised optimisation algorithm, rather than the "out of the box" method we used in our tests. It may, in fact, be possible to extend Taddy's work on distributed multinomial regression to the Poisson case, and adapt it for PoIR. For the MCMC method, there has been considerable research into improved algorithms for MCMC which we have not

taken advantage of in this work.

— Chapter 3 —

Healthcare Data

In this chapter, we will introduce a dataset which will be used as a "real data" example in Chapter 4 and Chapter 5. This dataset was kindly provided by Cardiff and Vale University Health Board.

This dataset was a by-product of a lexicon-based classifier built by the health board in order to classify letters sent by consultants at a large hospital to general practitioners and patients. Amongst other things, these letters either recommend further treatment, or a discharge from the out-patient service, but almost all of these letters are unlabelled. In order to better model patient pathways, the health board informatics team wanted to have labels assigned to each; given that they have over one million such letters, it was economically infeasible to label each of these by hand.

In order to deal with this problem, the health board developed a lexicon of just over 40,000 procedurally generated sentences, built from exemplar phrases and words related to either a follow-up or discharge decision. This lexicon could then be used with a nearest-neighbour classifier to predict the label for the letters. In order to validate this process, they had a number of letters manually labelled. This process also helped suggest further "seeds" for the lexicon generation process.

Some example sentences with their classification are included below.

Discharge "I DID NOT FEEL THERE WAS ANY NEED FOR HIM TO BE FOLLOWED UP"

Discharge "I DID NOT FEEL THERE WAS ANY NEED FOR HIM TO BE REVIEWED"

Discharge "I DID NOT FEEL THERE WAS ANY NEED FOR HIM TO BE SEEN AGAIN"

Follow-up "I WILL SEE HER AGAIN IN CLINIC [˜CLINIC˜]"

Follow-up "I WILL SEE HER AGAIN IN [˜CLINIC˜]"

Follow-up "I WILL SEE HER IN CLINIC [˜CLINIC˜]"

Due to the procedural generation process, which produces many very similar "documents", we encountered difficulties applying any algorithms to the entire dataset. To combat this (with the added benefit of making experiments quicker to run), we worked with a subset of 300 examples from the lexicon.

The lexicon which was generated had a heavy imbalance in favour of follow-up examples. Because the methods in this work are not designed to cope with such a heavy imbalance as is present in the original dataset, this subset was randomly sampled to consist of 150 "discharge" sentences and 150 "follow-up" sentences.

In order to prepare the lexicon data for analysis with the methods developed in Chapters 4 and 5, we first stemmed the words using the R package "RTextTools". The stemmed documents were then converted into a document-term matrix using the same package.

— Chapter 4 —

Sparse Generalised Principal Component Analysis

In this chapter we introduce Sparse Generalised Principal Component Analysis (SGPCA), a dimension reduction technique developed from the Generalised Principal Component Analysis (GPCA) of Landgraf and Lee (2015b). We begin by a recollection of the vector-space model of text data and an explanation of how that motivates sparsity in Section 1.2.

We will introduce GPCA as an extension of Principal Component Analysis (PCA) which is suitable for data from any exponential family of distribution in Section 4.2, before defining our extension SGPCA in Section 4.3. This will involve an exploration of methods of penalisation (Section 4.3.1) and an efficient algorithm for estimation of the loadings matrix (Section 4.3.2).

In Sections 4.4 and 4.6 we will examine the efficacy of SGPCA in several ways. Firstly, we will conduct synthetic data experiments using a hidden factor model in Section 4.4.1, a two-class hidden factor model in Section 4.4.2 and a varying-noise hidden factor model to explore robustness against noise in Section 4.4.3. Section 4.6 will comprise of the application and analysis of SGPCA to a healthcare text dataset.

Finally, we will summarise the results of this chapter in Section 4.7.

4.1 Introduction

4.1.1 Text Data

Recalling from Section 1.2, in the vector-space model for text data we represent each document in a corpus as a row in a data matrix $\mathbf{X} = \left[x_{ij}\right]_{i=1,j=1}^{i=n,j=p}$. Each column of $\mathbf{X}$ represents a term (either a word or n-gram) which appears in the corpus. Then the element x_{ij} is the number of times the i^{th} document contains term j. Often, a suitable

model for the data is that each term has a Poisson distribution with some mean λ_j for $j = 1, \ldots, p$.

It is important for our future discussion to note two points. Firstly, that such data is not Gaussian; in fact, whilst the Gaussian distribution is a reasonable approximation to the Poisson distribution for large enough λ, it is rare in a text mining scenario for such values of λ to occur. Secondly, across a collection of documents, we can often expect many terms to be common to most of the documents. We are often interested in either classifying or clustering documents, but these common terms are uninformative in these contexts. As such, we might expect that a dimension reduction across the corpus would not include contributions from these terms, thus motivating our belief in sparsity.

4.2 Generalised Principal Component Analysis

4.2.1 Generalised Principal Component Analysis Definition

Let us now return to the earlier definition of PCA in (1.3). In Landgraf and Lee (2015a) and Landgraf and Lee (2015b), the authors recognised that this squared reconstruction error is equivalent (up to a constant) to the deviance of a model where the observations are from a Gaussian distribution with known variance and a natural parameter matrix of the form

$$\widehat{\mathbf{\Theta}} = \mathbb{1}_n \boldsymbol{\mu}^{\mathrm{T}} + \left(\widetilde{\mathbf{\Theta}} - \mathbb{1}_n \boldsymbol{\mu}^{\mathrm{T}}\right) \mathbf{U}\mathbf{U}^{\mathrm{T}} \tag{4.1}$$

where $\widetilde{\mathbf{\Theta}}$ is the matrix of saturated natural parameters, $\mathbf{U} \in \mathbb{R}^{p \times k}$, $\boldsymbol{\mu} \in \mathbb{R}^p$ and we constrain $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbb{I}$. This connection can be seen by recognising that the canonical link function for the Gaussian distribution is merely the identity function, so the saturated natural parameters are simply the observed values. Note that the matrix $\widehat{\mathbf{\Theta}}$ has rank k, compared to $\widetilde{\mathbf{\Theta}}$ which has rank p. It is precisely this low-rank approximation which can be used as a lower-dimensional analogue to the observed data. Once optimal $\boldsymbol{\mu}$ and $\mathbf{U}$ have been identified, $\widehat{\mathbf{\Theta}}$ can be calculated.

In order to generalise this to exponential family data (binary in Landgraf and Lee (2015a), most exponential families in Landgraf and Lee (2015b)), we seek the deviance-optimal parameters $\mathbf{U}$ and $\boldsymbol{\mu}$. Here, the deviance refers to a measure of model quality, comparing one model (within the same family) to the saturated model (the model with natural parameters equal to the saturated natural parameters). Formally, the deviance is defined by the log of the likelihood ratio; this is equivalent to the difference of the log-likelihood. The deviance is non-negative, with minimum 0 achieved at the saturated model. We now prove a lemma regarding the form of the deviance under the GPCA model.

Lemma 4.2.1. *Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be n observations of a p-dimensional random vector $\mathcal{X} = [\mathcal{X}_1, \ldots, \mathcal{X}_p]^{\mathrm{T}}$ where $\mathcal{X}_j$ has an exponential family distribution with log-partition function b_j for $j = 1, \ldots, p$. Denote by $\widetilde{\mathbf{\Theta}}$ the matrix of saturated natural parameters and let $\widehat{\mathbf{\Theta}}$ be the low-rank approximation with ij^{th} component $\hat{\theta}_{ij}$. Then the deviance up to an additive constant and multiplication by a positive factor is given by*

$$D\left(\mathbf{X} \mid \widehat{\mathbf{\Theta}}\right) \overset{+\times}{\propto} \sum_{i=1}^{n}\sum_{j=1}^{p}\left[b_j\left(\hat{\theta}_{ij}\right) - x_{ij}\hat{\theta}_{ij}\right] \tag{4.2}$$

Proof. Properly, we have

$$\begin{aligned}
D\left(\mathbf{X} \mid \widehat{\mathbf{\Theta}}\right) &= -2\left(\log \mathbb{P}\left(\mathbf{X} \mid \widehat{\mathbf{\Theta}}\right) - \log \mathbb{P}\left(\mathbf{X} \mid \widetilde{\mathbf{\Theta}}\right)\right) && (4.3)\\
&\overset{+\times}{\propto} -\log \mathbb{P}\left(\mathbf{X} \mid \widehat{\mathbf{\Theta}}\right) && (4.4)\\
&= \log \prod_{i=1}^{n}\prod_{j=1}^{p}\left[h_j(x_{ij}) \exp\left(x_{ij}\hat{\theta}_{ij} - b_j\left(\hat{\theta}_{ij}\right)\right)\right] && (4.5)\\
&= -\sum_{i=1}^{n}\sum_{j=1}^{p}\left[\log h_j\left(x_{ij}\right) + x_{ij}\hat{\theta}_{ij} - b_j\left(\hat{\theta}_{ij}\right)\right] && (4.6)\\
&\overset{+}{\propto} \sum_{i=1}^{n}\sum_{j=1}^{p}\left[b_j\left(\hat{\theta}_{ij}\right) - x_{ij}\hat{\theta}_{ij}\right] && (4.7)
\end{aligned}$$

□

Remark. From this point on, when we refer to $D(\mathbf{X} \mid \widehat{\mathbf{\Theta}})$ we will actually refer to the right hand side of (4.2) unless otherwise stated. From the point of view of optimisation, the two are entirely equivalent.

Definition 4.2.1. For some observed data $\mathbf{X} \in \mathbb{R}^{n\times p}$, the rank k GPCA approximation is the pair $\mathbf{U}$ and $\boldsymbol{\mu}$ satisfying

$$\mathbf{U}, \boldsymbol{\mu} = \underset{\mathbf{U}\in\mathbb{R}^{p\times k}, \boldsymbol{\mu}\in\mathbb{R}^{p}}{\arg\min}\; D\left(\mathbf{X} \mid \mathbb{1}_n\boldsymbol{\mu}^{\mathrm{T}} + \left(\widetilde{\mathbf{\Theta}} - \mathbb{1}_n\boldsymbol{\mu}^{\mathrm{T}}\right)\mathbf{U}\mathbf{U}^{\mathrm{T}}\right) \quad \text{subject to } \mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbb{I}$$

4.3 Sparse Generalised Principal Component Analysis

4.3.1 Penalisation

As the goal of SGPCA is to provide a method comparable to GPCA but with sparse loadings, the first step to defining it is to discuss penalisation in the context of penalised optimisation. We will be placing a penalty function $P(\mathbf{U})$ on the components of $\mathbf{U}$, and rather than optimising the deviance directly, we shall optimise the deviance plus the

penalty function, with the aim to jointly minimise both, finding a good compromise between sparsity and deviance-optimality. We will consider two well-used penalty functions in order to construct our penalty function, the Smoothly Clipped Absolute Deviation (SCAD) penalty and the L^1 penalty. Both of these have well-understood properties, and are often used in the sparsity context.

Remark. We also briefly investigated the L^2 penalty due to its frequent use in sparsifying methods (especially in conjuction with the L^1 penalty in the so-called elastic net (Zou and Hastie 2005)). Unfortunately, due to the semi-orthogonality constraint on $\mathbf{U}$, the L^2 penalty on matrices (often known as the Frobenius norm) would be always equal to k on $\mathbf{U}$.

4.3.1.1 Smoothly Clipped Absolute Deviation Penalty

The SCAD penalty, due to Fan and Li (2001), is usually defined by its derivative

$$P'_S(\theta;\lambda,a) := \lambda I(\theta \le \lambda) + \frac{(a\lambda-\theta)_+}{a-1} I(\theta > \lambda) \tag{4.8}$$

where $I(A)$ is the indicator function on the proposition A and $(x)_+ = \max\{x, 0\}$. An application of integration shows that the penalty function itself is

$$P_S(\theta;\lambda,a) = \begin{cases} \lambda\theta & 0 < \theta \le \lambda \\ -\frac{\theta^2 - 2a\lambda\theta + \lambda^2}{2(a-1)} & \lambda < \theta \le a\lambda \\ \frac{(a+1)\lambda^2}{2} & a\lambda < \theta \end{cases} \tag{4.9}$$

A graph of $P_S(|\theta|)$ is shown in Figure 4.1. Examination either of (4.9) or Figure 4.1 will reveal that within λ of the origin the penalty decreases towards 0 linearly, more than $a\lambda$ from the origin the penalty is a (positive) constant, and between the two regions the penalty is a quadratic joining the two regions. With this in mind, one can understand the motivation behind the penalty – to drive coefficients close to 0 down to 0, to penalise coefficients larger than a certain threshold only a constant amount, under the recognition that large enough coefficients are probably needed by the model, and to transition smoothly between the two behaviours. To illustrate the way in which the penalty changes with λ, in Figure 4.2 we show the penalty for three different values of λ.

We will be applying the SCAD penalty to $\mathbf{U}$ as follows:

$$P_S(\mathbf{U};\lambda,a) = \sum_{i=1}^{n}\sum_{j=1}^{p} P_S\left(|U_{ij}|;\lambda,a\right) \tag{4.10}$$

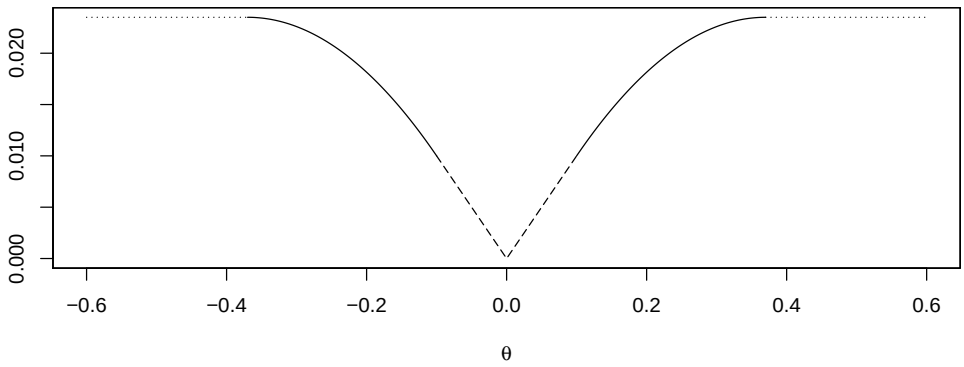


Figure 4.1: The (symmetrised) SCAD penalty, with $\lambda = 0.1$ and $a = 3.7$. In the region $[-\lambda, \lambda]$, the function is shown dashed; in the region $(-\infty, -a\lambda) \cup (a\lambda, \infty)$ it is shown dotted; the quadratic joining sections are shown solid.

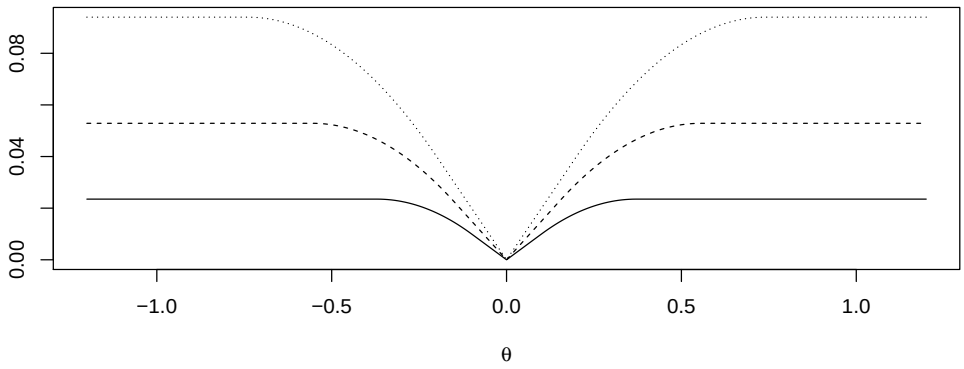


Figure 4.2: The (symmetrised) SCAD penalty shown for $\lambda = 0.1$ (solid line), $\lambda = 0.15$ (dashed line), and $\lambda = 0.2$ (dotted line).

Note that the SCAD penalty takes two parameters, λ and a. The latter is usually taken to be 3.7 due to an argument in the original paper, but the former must be chosen on a case-by-case basis. We will usually take it to be a small value on the order of 10^{-1} or smaller.

4.3.1.2 L^1 Penalty

The L^1 penalty on $\mathbf{U}$ is simply $\|\mathbf{U}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{p}|U_{ij}|$. Its simplicity in comparison with the SCAD penalty is obvious, but it does come with the disadvantage of reduced flexibility and power. We will investigate whether this is truly a disadvantage in our application, or whether its performance is sufficient.

4.3.1.3 Total Penalty Function

We define the total penalty function

$$P(\mathbf{U}; a, \lambda, \lambda_S, \lambda_L) \coloneqq \lambda_S P_S(\mathbf{U}; a, \lambda) + \lambda_L \|\mathbf{U}\|_1 \tag{4.11}$$

Note that this penalty function has two additional parameters, λ_S and λ_L; these are used both to control the relative importance of each penalty function to each other, and the relative importance to the deviance. A linear combination of the two penalties was chosen in order to make their comparison easier; in practice we would not advise using both penalties, although we will later compare the performance of this "doubly penalised" model against the singly penalised versions for thoroughness. SGPCA is fairly sensitive to appropriate specification of these parameters to manage the desire for sparsity against the need for a deviance-optimal fit. We will discuss some heuristic strategies for choosing appropriate values in Sec 4.7, along with the need to develop a general data-driven method for their selection.

4.3.1.4 Definition of SGPCA

We can now define the SGPCA approximation.

Definition 4.3.1. Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be a matrix of n observations of p variables and let the j^{th} variable have an exponential family distribution with log-partition function b_j. Let $\widetilde{\mathbf{\Theta}} \in \mathbb{R}^{n\times p}$ denote the matrix of saturated natural parameters. Then, letting D and P have the definitions given in (4.2) (4.11). Then the SGPCA approximation for $\mathbf{X}$ is the pair $\mathbf{U} \in \mathbb{R}^{p\times k}$ and $\boldsymbol{\mu} \in \mathbb{R}^p$ satisfying

$$\mathbf{U}, \boldsymbol{\mu} = \underset{\mathbf{U}\in\mathbb{R}^{p\times k}, \boldsymbol{\mu}\in\mathbb{R}^p}{\arg\min} \left[D\left(\mathbf{X} \mid \widehat{\mathbf{\Theta}}\right) + P\left(\mathbf{U}; a, \lambda, \lambda_S, \lambda_L\right)\right]$$

4.3.2 Estimation

Now that we have defined the SGPCA approximation, it remains to discuss how to estimate the optimum $\mathbf{U}$ and $\boldsymbol{\mu}$. Unfortunately, this is not an easy task due to the singularities of the penalty function at 0 components and the semi-orthogonality condition on $\mathbf{U}$.

This is further complicated by the non-convexity of the objective function. In general, this means that finding the global minimum is not guaranteed. In practice, we have not found this to be particularly problematic, as local minima have been sufficiently good for our purposes.

In order to optimise the objective function, we would like to be able to use gradient-based methods, but the singularities of the penalty function preclude doing so directly. To avoid this problem, we will use the method of Hunter and Li (2005) to derive an appropriate, differentiable and non-singular majoriser of the penalties. To summarise, they showed that, for penalties of the form $P(\boldsymbol{\theta}) = \lambda p(|\theta_i|)$ where p is non-decreasing, convex and have $\lim_{\theta \downarrow 0} p'(\theta) < \infty$ the function

$$\Phi_{\theta_0}(\theta) := \lambda p(|\theta|) + \frac{\left(\theta^2 - \theta_0^2\right) \lambda p'(|\theta_0|+)}{2|\theta_0|}$$

is a majoriser. That is, $\Phi_{\theta_0} \geq \lambda p(|\theta|)$ for all θ with equality when $\theta = \theta_0$. Note that here $p'(|\theta_0|+)$ means the right hand limit of the function at $|\theta_0|$.

As the SCAD component function is convex, and our total SCAD penalty is merely a weighted sum (with non-negative weights) of the component functions evaluated at the absolute value of the components, we can apply this result to majorise the SCAD term of our penalty function. Under the notation of Hunter and Li, the L^1 penalty is also of this form, as the corresponding p function is merely the identity function which is convex. Thus, we can majorise our total penalty function $P(\mathbf{U}; a, \lambda, \lambda_S, \lambda_L)$ by an appropriately calculated combination of Φ_{θ_0} functions.

Lemma 4.3.1. *The function $\Phi^{(t-1)}$ defined below is a majoriser of the total penalty function.*

$$\begin{aligned}\Phi^{(t-1)} := \sum_{i=1}^{n}\sum_{j=1}^{p} \Bigg\{ & \lambda_L \left[\left|u_{ij}^{(t-1)}\right| + \frac{u_{ij}^2 - \left(u_{ij}^{(t-1)}\right)^2}{2\left|u_{ij}^{(t-1)}\right|} \right] + \\ & \lambda_S \left[P_S\left(\left|u_{ij}^{(t-1)}\right|; \lambda, a\right) + \frac{\left(u_{ij}^2 - \left(u_{ij}^{(t-1)}\right)^2\right) P_S'\left(\left|u_{ij}^{(t-1)}\right|+; \lambda, a\right)}{2\left|u_{ij}^{(t-1)}\right|} \right] \Bigg\} \end{aligned} \tag{4.12}$$

Proof. An application of the result of Hunter and Li (2005). □

However, our majoriser is not defined everywhere; it contains a singularity at all $\mathbf{U}_0$ with at least one component equal to 0. Fortunately, we can remove this singularity with a perturbed version of the majoriser:

$$\Phi^{(t-1)} := \sum_{i=1}^{n}\sum_{j=1}^{p}\left\{\lambda_L\left[\left|u_{ij}^{(t-1)}\right| + \frac{u_{ij}^2 - \left(u_{ij}^{(t-1)}\right)^2}{2\left(\left|u_{ij}^{(t-1)}\right| + \varepsilon\right)}\right] + \lambda_S\left[P_S\left(\left|u_{ij}^{(t-1)}\right|;\lambda,a\right) + \frac{\left(u_{ij}^2 - \left(u_{ij}^{(t-1)}\right)^2\right)P_S'\left(\left|u_{ij}^{(t-1)}\right|+;\lambda,a\right)}{2\left(\left|u_{ij}^{(t-1)}\right| + \varepsilon\right)}\right]\right\} \quad (4.13)$$

This new perturbed majoriser is not, in fact, a majoriser for our penalty function except in the limit $\varepsilon \downarrow 0$, but we will refer to it as such under this understanding. Hunter and Li showed that such perturbed majorisers do majorise a perturbed version of the problem, differing by a term with a multiplicative factor of ε. As such, we can hope that with small enough ε, solving the perturbed problem will provide a good approximation to solving the original problem.

Now we have an expression for a (perturbed) majoriser which is defined everywhere and is differentiable, we can proceed to use gradient methods to find the optimum $\mathbf{U}$ and $\boldsymbol{\mu}$. We will treat the two quantities separately, in a batched coordinate-descent style. As optimising over $\boldsymbol{\mu}$ is fairly simple, we will concentrate on how to perform optimisation over $\mathbf{U}$. We need to preserve the left semi-orthogonality condition; we could use a Langragian method, but we prefer to use a more direct method from Wen and Yin (2013). Given a feasible point $\mathbf{U}$ and the gradient at that point $\mathbf{G}$, they first define the matrix $\mathbf{A} := \mathbf{G}\mathbf{U}^{\mathrm{T}} - \mathbf{U}\mathbf{G}^{\mathrm{T}}$ (in notation consistent with our use) which is skew-symmetric, that is, $\mathbf{A}^{\mathrm{T}} = -\mathbf{A}$. Next, they define $\mathbf{Y}(\tau)$ by the Crank-Nicholson-like scheme $\mathbf{Y}(\tau) = \mathbf{U} - \frac{\tau}{2}\mathbf{A}(\mathbf{U} + \mathbf{Y}(\tau))$. The solution to this can be given explicitly as

$$\mathbf{Y}(\tau) = \left(\mathbb{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbb{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{U} \quad (4.14)$$

Wen and Yin show that $\mathbf{Y}(\tau)$ possesses several very important features: for all $\tau \geq 0$ we have that $(\mathbf{Y}(\tau))^{\mathrm{T}}\mathbf{Y}(\tau) = \mathbf{U}^{\mathrm{T}}\mathbf{U}$, it is smooth in τ, $\mathbf{Y}(0) = \mathbf{U}$, and finally $\mathbf{Y}(\tau)$ defines a descent path for $\tau \geq 0$. It is important to note that, while this function does involve two matrix inversions, an efficient algorithm for calculating them is given in Wen and Yin (2013) which we recommend.

Now all that is needed is the appropriate gradients for the objective function which we will denote by S.

Lemma 4.3.2. *The gradient of the objective function*

1. *with respect to U_{kl} is*

$$\frac{\partial S}{\partial \mathbf{U}_{kl}} = \sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\Big(b_j'\Big(\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big)\big]_j\Big) - x_{ij}\Big) \times \Big(\delta_{kj}\mathbf{U}_{[l]}^{\mathrm{T}}\big(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big) + \mathbf{U}_{jl}\big(\tilde{\theta}_{ik} - \mu_k\big)\Big)\Bigg\} + \frac{\Big(\lambda_L + \lambda_{\mathrm{S}} P_{\mathrm{S}}'\Big(\Big|\mathbf{U}_{kl}^{(t-1)}\Big| +; \lambda, a\Big)\Big)\mathbf{U}_{kl}}{\varepsilon + \Big|\mathbf{U}_{kl}^{(t-1)}\Big|} \tag{4.15}$$

2. *with respect to μ_k is*

$$\frac{\partial S}{\partial \mu_k} = \sum_{i=1}^{n}\sum_{j=1}^{p}\Big(b_j'\Big(\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big(\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big)\big]_j\Big) - x_{ij}\Big)\Big(\delta_{jk} - \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big]_{jk}\Big) \tag{4.16}$$

Proof. First, we calculate the derivative of the majorised penalty with respect to the kl element of $\mathbf{U}$, i.e. $\frac{\partial}{\partial \mathbf{U}_{rs}} M_P^{\varepsilon}\big(\mathbf{U}|\mathbf{U}^{(t-1)}\big)$

$$\begin{aligned}
\frac{\partial M_P^{\varepsilon}}{\partial \mathbf{U}_{kl}} &= \frac{\partial}{\partial \mathbf{U}_{kl}} \sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\lambda_L\Big|\mathbf{U}_{ij}^{(t-1)}\Big| + \lambda_{\mathrm{S}} P_{\mathrm{S}}\Big(\Big|\mathbf{U}_{ij}^{(t-1)}\Big|\Big) + \frac{\Big[\mathbf{U}_{ij}^2 - \Big(\mathbf{U}_{ij}^{(t-1)}\Big)^2\Big]\Big[\lambda_L + \lambda_{\mathrm{S}} P_{\mathrm{S}}'\Big(\Big|\mathbf{U}_{ij}^{(t-1)}\Big|\Big)\Big]}{2\Big[\varepsilon + \Big|\mathbf{U}_{ij}^{(t-1)}\Big|\Big]}\Bigg\} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\frac{\partial}{\partial \mathbf{U}_{kl}}\Big[\mathbf{U}_{ij}^2 - \Big(\mathbf{U}_{ij}^{(t-1)}\Big)^2\Big]\frac{\lambda_L + \lambda_{\mathrm{S}} P_{\mathrm{S}}'\Big(\Big|\mathbf{U}_{ij}^{(t-1)}\Big|\Big)}{2\Big[\varepsilon + \Big|\mathbf{U}_{ij}^{(t-1)}\Big|\Big]}\Bigg\} \\
&= \frac{2\mathbf{U}_{kl}\Big(\lambda_L + \lambda_{\mathrm{S}} P_{\mathrm{S}}'\Big(\Big|\mathbf{U}_{kl}^{(t-1)}\Big|\Big)\Big)}{2\Big[\varepsilon + \Big|\mathbf{U}_{kl}^{(t-1)}\Big|\Big]} = \frac{\mathbf{U}_{kl}\Big(\lambda_L + \lambda_{\mathrm{S}} P_{\mathrm{S}}'\Big(\Big|\mathbf{U}_{kl}^{(t-1)}\Big|\Big)\Big)}{\Big[\varepsilon + \Big|\mathbf{U}_{kl}^{(t-1)}\Big|\Big]}
\end{aligned}$$

where we suppress the arguments P_{S} for notational clarity.

Next, we calculate the gradient of the deviance function with respect to $\mathbf{U}_{kl}$

$$\begin{aligned}
\frac{\partial D}{\partial \mathbf{U}_{kl}} &= \frac{\partial}{\partial \mathbf{U}_{kl}}\Bigg\{\sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{b_j\Big(\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big\{\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big\}\big]_j\Big) - x_{ij}\Big\{\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big\{\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big\}\big]_j\Big\}\Bigg\} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\Big\{b_j'\Big(\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big\{\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big\}\big]_j\Big) - x_{ij}\Big\}\frac{\partial}{\partial \mathbf{U}_{kl}}\Big[\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big\{\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big\}\big]_j\Big]\Bigg\} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{p}\Bigg\{\Big\{b_j'\Big(\mu_j + \big[\mathbf{U}\mathbf{U}^{\mathrm{T}}\big\{\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big\}\big]_j\Big) - x_{ij}\Big\}\Big\{\underbrace{\delta_{kj}\mathbf{U}_{[l]}^{\mathrm{T}}\big[\tilde{\boldsymbol{\theta}}_i - \boldsymbol{\mu}\big]}_{(*)} + \underbrace{\mathbf{U}_{jl}\big[\tilde{\theta}_{ik} - \mu_k\big]}_{(\dagger)}\Big\}\Bigg\}
\end{aligned}$$

Where $(*)$ comes from the terms quadratic in $\mathbf{U}_{kl}$ and $(\dagger)$ is from the terms linear in $\mathbf{U}_{kl}$. Combining the two expressions, we get (4.15).

Next, we calculate the derivative of the objective function with respect to μ_k. First, note that the majorised penalty does not depend on $\boldsymbol{\mu}$, so we need only calculate the gradient of the deviance function.

$$
\begin{aligned}
&\frac{\partial}{\partial \mu_k}\left\{\sum_{i=1}^{n}\sum_{j=1}^{p}\left\{b_j\left(\mu_j+\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\left\{\tilde{\theta}_i-\boldsymbol{\mu}\right\}\right]_j\right)-x_{ij}\left\{\mu_j+\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\left\{\tilde{\theta}_i-\boldsymbol{\mu}\right\}\right]_j\right\}\right\}\right.\\
&=\sum_{i=1}^{n}\sum_{j=1}^{p}\left\{\left\{b_j'\left(\mu_j+\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\left\{\tilde{\theta}_i-\boldsymbol{\mu}\right\}\right]_j\right)-x_{ij}\right\}\frac{\partial}{\partial\mu_k}\left[\mu_j+\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\left\{\tilde{\theta}_i-\boldsymbol{\mu}\right\}\right]_j\right]\right\}\\
&=\sum_{i=1}^{n}\sum_{j=1}^{p}\left\{\left\{b_j'\left(\mu_j+\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\left\{\tilde{\theta}_i-\boldsymbol{\mu}\right\}\right]_j\right)-x_{ij}\right\}\left\{\delta_{jk}-\frac{\partial}{\partial\mu_k}\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\boldsymbol{\mu}\right]_j\right\}\right\}\\
&=\sum_{i=1}^{n}\sum_{j=1}^{p}\left\{\left\{b_j'\left(\mu_j+\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\left\{\tilde{\theta}_i-\boldsymbol{\mu}\right\}\right]_j\right)-x_{ij}\right\}\left\{\delta_{jk}-\left[\mathbf{U}\mathbf{U}^{\mathrm{T}}\right]_{jk}\right\}\right\}
\end{aligned}
$$

□

Remark. Although the gradients are given in Lemma 4.3.2, in practice we approximate the gradients by finite differences for reasons of computational efficiency.

In Algorithm 1 we summarise the procedure for calculating the estimated $\mathbf{U}$ and $\boldsymbol{\mu}$ for SGPCA. In our implementation, we find the minimising value of $\boldsymbol{\mu}$ using the OPTIM function in R (R Foundation for Statistical Computing, Vienna 2011). We find the appropriate value of τ by the line search algorithm of Nocedal and Wright (1999); we do not give explanation of this method here as the exact implementation of this aspect of the algorithm is fairly arbitrary, any line search algorithm would be sufficient. The algorithm of Nocedal and Wright is chosen upon the recommendation in Wen and Yin (2013).

A naive implementation of this algorithm tends to be computationally expensive, but with suitable care taken with evaluating the necessary functions it can be evaluated in a reasonable time-frame. In particular, we recommend using the efficient algorithm for the matrix inverses in $\mathbf{Y}(\tau)$. We have also found success using an automatic differentiation scheme to calculate both the objective function and its gradient simultaneously, though we do not yet have a concrete recommendation for this.

Remark. We remark that this algorithm for estimation does not come with any guarantees of convergence to the global minimum. Fortunately, our experience in practice

is that starting at the recommended initial values leads to convergence to a minimum which is sufficiently good.

4.4 Synthetic Data Examples

We will now investigate the performance of SGPCA in three situations using synthetic data; firstly, we will explore its performance on some synthetic data generated by a hidden factor model and evaluate the ability of SGPCA to identify directions corresponding to those hidden factors, secondly we will perform a similar analysis but using data drawn from two different hidden factor models and test the ability of SGPCA to find directions which can differentiate the two classes of data, finally we will test the robustness of the SGPCA loadings to noise by repeating the analysis from the first synthetic model but with varying levels of noise.

Throughout this analysis, we will use the following distribution for noise, which we will denote by $\mathcal{E}$ and observations from it by ϵ.

$$\mathcal{E}(\eta) \coloneqq \mathcal{P} \times (-1)^{\mathcal{B}}, \quad \mathcal{P} \sim \text{Po}(\eta), \quad \mathcal{B} \sim \text{Bernoulli}(1/2) \tag{4.17}$$

That is, it is a Poisson random variable of mean η multiplied with equal probability by 1 or -1.

We will differentiate between three types of SGPCA in this and the following section: L^1 penalised SGPCA, SCAD penalised SGPCA and the combined SGPCA. The first will denote the case where $\lambda_S = 0$, the second will denote the case where $\lambda_L = 0$ and the third will denote the case where both λ_S and λ_L are non-zero. This will allow us to analyse the performance of each penalty together and in isolation.

4.4.1 "Classless" Data Exploration

The hidden factor model we will use to test the performance of SGPCA will be constructed around three hidden factors, $\mathcal{V}_1$, $\mathcal{V}_2$ and $\mathcal{V}_3$ which will be a linear combination of the previous two. In particular,

$$\mathcal{V}_1 \sim \text{Po}(25) \qquad \mathcal{V}_2 \sim \text{Po}(30) \qquad \mathcal{V}_3 = 1\mathcal{V}_1 + 3\mathcal{V}_2$$

Each observations $\mathbf{x}_i$ will be constructed by drawing v_{1i}, v_{2i} and v_{3i} from $\mathcal{V}_1$, $\mathcal{V}_2$ and $\mathcal{V}_3$ respectively, then denoting the j^{th} component of $\mathbf{x}_i$ by x_{ij}, we have

$$x_{ij} = v_{1i} + \epsilon_{ij},\ j \in \{1,2,3,4\} \quad x_{ij} = v_{2i} + \epsilon_{ij},\ j \in \{5,6,7,8\} \quad x_{ij} = v_{3i} + \epsilon_{ij},\ j \in \{9,10\}$$

where each $\epsilon_{ij} \overset{\text{i.i.d}}{\sim} \mathcal{E}(2)$. 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, Sparse Principal Component Analysis (SPCA), Robust Principal Component Analysis (RPCA), Nonnegative Matrix Factorisation (NMF) and Latent Dirichlet Allocation (LDA). The first loadings/directions from all algorithms are shown in Table 4.1, and the second in Table 4.2.

Analysis of Table 4.1 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, SPPCA and RPCA 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 small contribution from the second hidden factor and slightly larger contribution from the first hidden factor. LDA primarily identifies the third hidden factor.

Looking at the second loadings/directions in Table 4.2, 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, SPPCA 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 contribution from the second. LDA also does not identify anything more than its first direction, finding primarily the third hidden factor.

Of all the algorithms considered, the performance of SGPCA is the best. Its first loading identifies the second hidden factor and the strongly correlated third hidden factor, and the second identifies the first hidden factor. Although GPCA, PCA, SPCA and RPCA provide similar identification in the second loading, they all include larger contributions from the second hidden factor, somewhat reducing interpretability.

4.4.2 Classed Synthetic Data Exploration

Although SGPCA is not a supervised dimension reduction method, we believe it should be capable of identifying directions suitable for differentiating between classes. To investigate this, we perform a similar study to that in Section 4.4.1, but drawing 100

Data: $\mathbf{X}$, tol, λ_S, λ, a, λ_L
Result: $\mathbf{U}$, $\boldsymbol{\mu}$
calculate saturated natural parameters matrix $\widetilde{\boldsymbol{\Theta}}$;
set $\boldsymbol{\mu}^{(0)}$ equal to the column means of $\widetilde{\boldsymbol{\Theta}}$;
set $\mathbf{U}^{(0)}$ equal to the first k eigenvalues of $\widetilde{\boldsymbol{\Theta}} - \mathbb{1}\boldsymbol{\mu}^{\mathrm{T}}$;
set $t = 1$;
while $\|\mathbf{U}^{(t)} - \mathbf{U}^{(t-1)}\|_2 >$ tol *or* $\|\boldsymbol{\mu}^{(t)} - \boldsymbol{\mu}^{(t-1)}\|_2 >$ tol *or* $t = 1$ **do**
  set $\boldsymbol{\mu}^{(t)}$ to the minimiser of the objective function w.r.t. $\boldsymbol{\mu}$;
  set $\mathbf{G}$ to the gradient of the objective function w.r.t. $\mathbf{U}$;
  set $\mathbf{A} = \mathbf{G}\left(\mathbf{U}^{(t-1)}\right)^{\mathrm{T}} - \mathbf{U}^{(t-1)}\mathbf{G}^{\mathrm{T}}$;
  define function $\mathbf{Y}(\tau) = \left(\mathbb{I} + \frac{\tau}{2}\mathbf{A}\right)^{-1}\left(\mathbb{I} - \frac{\tau}{2}\mathbf{A}\right)\mathbf{U}^{(t-1)}$;
  set τ^* to the value of τ which minimises the objective function along the path of $\mathbf{Y}(\tau)$, using $\boldsymbol{\mu}^{(t)}$ as the value of $\boldsymbol{\mu}$;
  set $\mathbf{U}^{(t)} = \mathbf{Y}(\tau^*)$;
  set $t = t + 1$;
end

Algorithm 1: SGPCA Estimation Procedure

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 4.1: First loading/direction for synthetic data, using L^1 penalised SGPCA, SCAD penalised SGPCA, the combined penalty SGPCA, GPCA, PCA, SPCA, RPCA, NMF, LDA and Sparse Probabilistic Principal Component Analysis (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 4.2: Second loading/direction for synthetic data, using L^1 penalised SGPCA, SCAD penalised SGPCA, the combined penalty SGPCA, GPCA, PCA, SPCA, RPCA, NMF and LDA

observations each from two different hidden factor models. The first 100 observations have hidden factors

$$\mathcal{V}_1 \sim \text{Po}(25) \qquad \mathcal{V}_2 \sim \text{Po}(25) \qquad \mathcal{V}_3 = 1\mathcal{V}_1 + 3\mathcal{V}_2$$

and the second 100 observations have hidden factors

$$\mathcal{V}_1 \sim \text{Po}(25) \qquad \mathcal{V}_2 \sim \text{Po}(35) \qquad \mathcal{V}_3 = 2\mathcal{V}_1 + 1\mathcal{V}_2$$

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 Multinomial Inverse Regression (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 4.3.

The primary differentiating factors between the two classes of data are the second hidden factor and the third hidden factor. As such, the strong identification of the second hidden factor by the loading of the L^1 penalised SGPCA is excellent. Interestingly, the SCAD and combined penalty SGPCA loadings are somewhat confused, suggesting that the L^1 penalty may be preferable. The first GPCA loading assigns roughly similar coefficients to the first and second hidden factors, and slightly lower coefficients to the third. Once again, PCA, SPCA, SPPCA and RPCA give similar loadings, concentrating mostly on the third hidden factor with a smaller contribution of the first. NMF also gives highest weighting to the third hidden factor, with a smaller contribution from the first and an even smaller by the second. LDA assigns mostly small coefficients, except to one of the components containing the third hidden factor. MNIR performs quite similarly to GPCA, giving very similar coefficients to both the first and second hidden factors, and smaller coefficients to the third hidden factor.

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 4.3: Loadings/directions for classed synthetic data, using L^1 penalised SGPCA, SCAD penalised SGPCA, the combined penalty SGPCA, GPCA, PCA, SPCA, RPCA, NMF, LDA and MNIR

4.4.3 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 4.4.1, but varying the parameter of noise. That is,

$$\mathcal{V}_1 \sim \text{Po}(25) \qquad \mathcal{V}_2 \sim \text{Po}(30) \qquad \mathcal{V}_3 = 1\mathcal{V}_1 + 3\mathcal{V}_2$$

We then construct each observation $\mathbf{x}_i$ by drawing v_{1i} from $\mathcal{V}_1$, v_{2i} from $\mathcal{V}_2$ and constructing $v_{3i} = 3v_{1i} + v_{2i}$, then we have

$$x_{ij} = v_{1i} + \epsilon_{ij},\ j \in \{1,2,3,4\} \quad x_{ij} = v_{2i} + \epsilon_{ij},\ j \in \{5,6,7,8\} \quad x_{ij} = v_{3i} + \epsilon_{ij},\ j \in \{9,10\}$$

where the ϵ_{ij} are independent observations from $\mathcal{E}(\eta)$. Then η is our noise parameter, we will 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 4.4a, for the SCAD penalised SGPCA in Table 4.4b, and for the combined penalty SGPCA in Table 4.4c.

An examination of Table 4.4a 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 primarily capturing the second and third hidden factors, and the second loading capturing the first hidden factor. Table 4.4c suggests that the combined penalty SGPCA has the same characteristics. However, Table 4.4b suggests that the SCAD penalised SGPCA is not quite so robust; both loadings seem to

	$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$		$\eta = 1$	$\eta = 2$	$\eta = 3$	$\eta = 4$
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 4.4: Investigations of the performance of all three SGPCA variants across varying levels of noise.

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 Joint Sparse Principal Component Analysis (JSPCA) for its robustness to outliers (which become more probable with increasing magnitude of noise).

4.5 Dependence on Tolerance

As one might expect, the time taken to estimate SGPCA is heavily dependent on the numerical tolerance; decreasing the tolerance by an order of magnitude has, in our experience, roughly the effect of increasing the time taken by an order of magnitude. Thus it is important to assess how critical a small tolerance is to accurate estimation. In order to determine this, we ran a further simulation study using the same data generation setup as Section 4.4.1. We fitted the three SGPCA variants with the same parameters, varying only the tolerance. We chose a range of tolerance values between 10^{-4} and 10^{-9}, taking the latter as sufficiently converged to use as a reference standard.

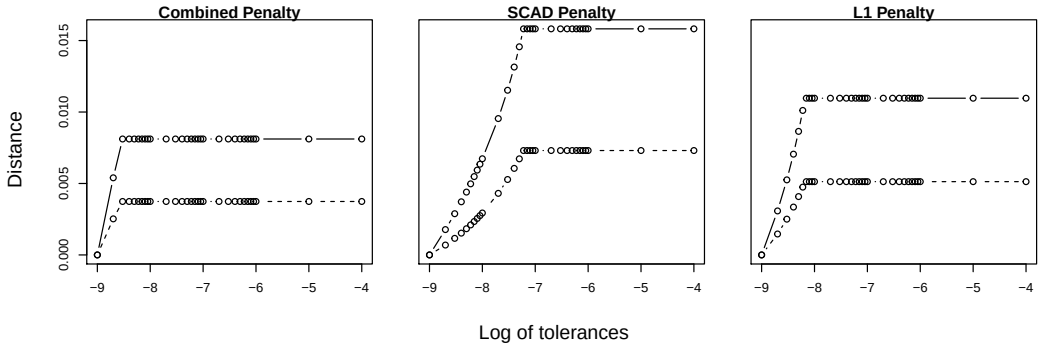
In Figure 4.3a we graph the Euclidean distances between the loadings from the reference fit and each of the lower tolerance fits. In Figure 4.3b we graph the deviances of each fit, normalised by the deviance of the reference fit. Both of these figures demonstrate the same behaviour: a period of nearly identical fits followed by rapid convergence to our reference fit. However, it is important to notice the scales on the two figures: the largest deviation amongst all fits is less than one half of one percent from the reference fit, and the greatest Euclidean distance is of order 10^{-2}. Consequently, we suggest that a practical strategy for choosing an appropriate tolerance value would be to choose the smallest value which allows for feasible optimisation times. If this study is representative of the general situation, a good starting point would be 10^{-7}.

4.6 Healthcare Data

As a "real-world" example, we apply GPCA, SGPCA and classical PCA to the healthcare data, described in Chapter 3. We chose to estimate three directions with each algorithm; experimenting with fewer directions failed to adequately separate the data with any of the methods. The results of this are shown in Figure 5.4.

Certainly it seems that both GPCA and SGPCA provide better separation of the data by pairs of components. The performances of GPCA and SGPCA, while producing rather different pictures in appearance, are comparable in quality. We should be reassured that,

(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.

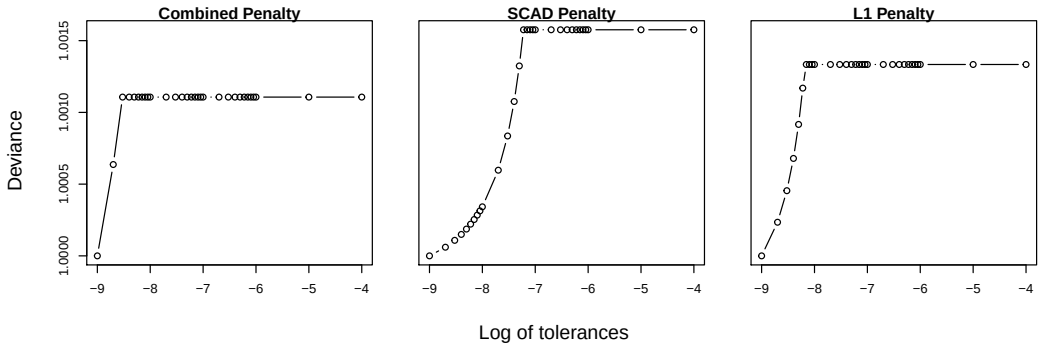


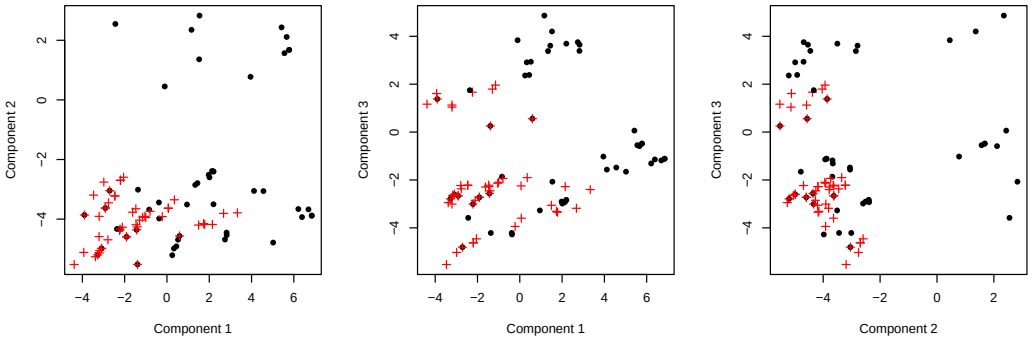
Figure 4.3: Behaviour of all SGPCA variants as tolerance is varied.

despite inducing sparse loadings, SGPCA's performance is on-par with the current state of the art on a real-world task.

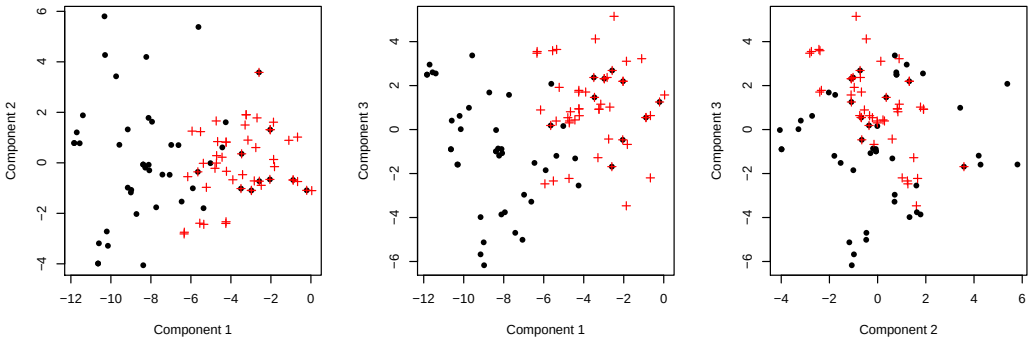
4.7 Discussion

In this chapter, we have extended GPCA by introducing a sparsifying penalty, and shown that we achieve as-good or superior performance in extracting appropriate principal component loadings which prioritise the selection of informative variables in each principal component, assigning much smaller coefficients to non- or less-informative variables.

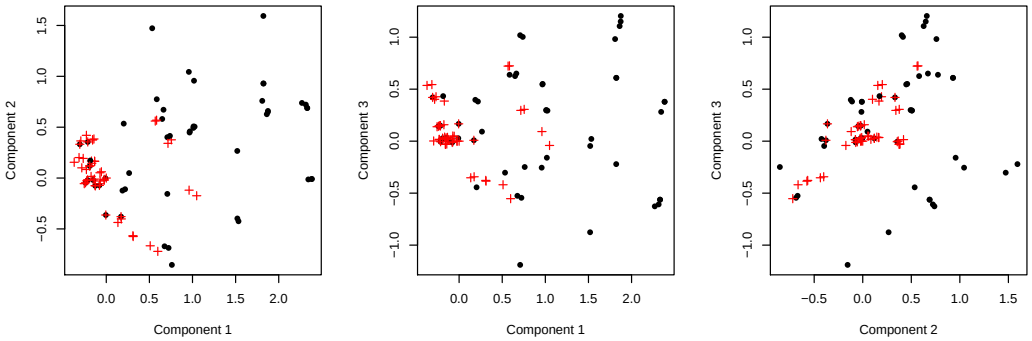
Comparison of the efficacy of the three penalties we used (L^1, SCAD and the combined penalty) suggest that, despite the more complex nature of the SCAD penalty, the L^1 penalty has the best performance. As such, our preference is for the L^1 penalty which is significantly simpler than the SCAD penalty, and whose evaluation computationally



(a) L^1 penalty SGPCA



(b) GPCA



(c) PCA

Figure 4.4: Plots of pairs of the first three principal components for the healthcare dataset obtained from SGPCA, GPCA and PCA. Red "+" symbols denote the "discharge" class; black dots represent the "follow-up" class.

is cheaper, not requiring any conditional statements and requiring fewer multiplications and no additions. It has the further advantage of only requiring one parameter (λ_L), unlike the SCAD penalty which requires λ_S, a and λ. As finding optimal choices of these parameters is most easily accomplished by repeated application of the SGPCA algorithm until one finds a suitable compromise between interpretability and predictive/discriminative power, having only one parameter to find is a significant advantage. We note that this is the method by which we chose the values of these parameters in the applications shown.

We suggest several important topics to extend this work. Firstly, a data-driven method for choosing these tuning parameters which does not require manual examination of the generated coefficients, thus facilitating an automated choice. Secondly, the construction of an appropriate order determination test to choice the required value of k. Such a test would likely be applicable also to GPCA. Thirdly, we suggest that this method could likely be extended to a supervised method for dimension reduction, similar to partial least squares (PLS). Finally, we suggest a more challenging problem: to develop a kernel-based extension of GPCA and SGPCA for the extraction of non-linear features.

— Chapter 5 —

Sparse Simple Exponential Family Principal Component Analysis

In this chapter, we will introduce an extension to Simple Exponential Principal Component Analysis (SePCA), the work of Li and Tao (2013). As with Generalised Principal Component Analysis (GPCA) in Chapter 4, we will propose a method which incorporates sparsity in the (analogy to) the principal component loadings in order to improve interpretability and performance. Unlike GPCA, SePCA starts with a fully specified Bayesian model, though the two methods are ultimately similar in concept, applying to any exponential family distribution. Here, we will focus on the Poisson case for its application to text data.

5.1 Simple Exponential Principal Component Analysis

In Li and Tao (2013), the authors specify a Bayesian model for some data $\mathbf{X}$ following an exponential family (EF) distribution. A plate diagram for the model is given in Figure 5.1. Specifically, they prescribe the model:

- $\mathcal{X}_i|\mathcal{W},\mathcal{Y}_i \sim \text{ExpFam}(\mathcal{W}\mathcal{Y}_i)$ for $i = 1,\dots,n$
- $\mathcal{Y}_i \sim \text{N}(\mathbf{0},\mathbf{I})$ for $i = 1,\dots,n$
- $\mathcal{W}_j|\alpha_j \sim \text{N}\left(\mathbf{0},\alpha_j^{-1}\mathbf{I}\right)$ for $j = 1,\dots,k$

where each $\mathcal{X}_i \in \mathbb{R}^p$, $\mathcal{Y}_i \in \mathbb{R}^k$, $\mathcal{W} \in \mathbb{R}^{p\times k}$. We denote by $\mathcal{W}_j$ the j^{th} row of $\mathcal{W}$. Finally, $\boldsymbol{\alpha} \coloneqq (\alpha_1,\dots,\alpha_k)$ is a vector of precision hyperparameters.

Here we identify the observed value $\mathbf{W}$ of $\mathcal{W}$ with the classical loadings matrix and the observed value $\mathbf{Y}$ formed of the n random vectors $\mathcal{Y}_i$ $(i = 1,\dots,n)$ with the classical scores matrix. The integer k here (as previously) is the number of principal components. It is these three parameters which we are particularly interested in. From a dimension

reduction standpoint, $\mathbf{Y}$ is the most important, it is this which we suggest can be used as a lower-dimensional analogue to the observed data.

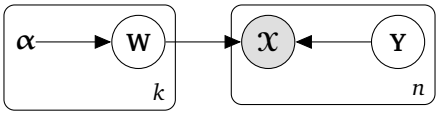


Figure 5.1: Plate diagram for SePCA

We now draw attention to one of SePCA's greatest advantages: using Automatic Relevance Determination (ARD), introduced in Mackay (1995), SePCA is able to determine the appropriate number of principal components k. Note that, according to the model we have given for SePCA, if (for some j) $\alpha_j \to \infty$, then $\mathbf{W}_j \xrightarrow{p} \mathbf{0}$. Thus, if we have a sufficiently large α_j, then we can reasonably determine that the corresponding principal component is not useful to the model. We will refer to the critical value for this as M, and remove the column of $\mathbf{W}$ and row of $\mathbf{Y}$ corresponding to any α_j which exceeds this threshold. We note that it is precisely because of the precision hyperparameters for $\mathcal{W}$ that we can apply ARD.

5.2 Sparse Simple Exponential Principal Component Analysis

In the same spirit as Sparse Generalised Principal Component Analysis (SGPCA) (Chapter 4), we will impose a sparsifying penalty on the loadings in the interest of increasing performance and interpretability. In the case of SePCA, the prior distribution for each $\mathcal{W}_j|\alpha_j$ roughly corresponds to an L^2 penalty on its entries, so we will not consider such a penalty. Instead, we approach the problem using the method of Frommlet and Nuel (2016) to place an (approximate) L^0 penalty on $\mathbf{W}$ through an iterative method. We could consider the same penalties as in SGPCA (namely, the SCAD and L^1 penalties), and we suspect that they would be reasonable choices. Our motivation for choosing the L^0 penalty here is primarily out of curiosity; it is known to be a strongly sparsifying penalty, and the new development of Frommlet and Nuel's method for approximating it allowed us to use it. Note that

$$\|\mathbf{W}\|_0 = \sum_{i=1}^{p}\sum_{j=1}^{k} I\left(\mathbf{W}_{ij} \neq 0\right) \tag{5.1}$$

where $I(\cdot)$ is the indicator function. Frommlet and Nuel use the following approximation (written in the specific case of our problem):

$$\|\mathbf{W}\|_0 \approx \sum_{i=1}^{p}\sum_{j=1}^{k} \frac{\mathbf{W}_{ij}^2}{\left(\mathbf{W}_{ij}^{(t-1)}\right)^2 + \delta} \tag{5.2}$$

where $\mathbf{W}^{(t-1)}$ is the value of $\mathbf{W}$ from the previous interation and $\delta > 0$ is a very small value. The advantage to this method is the differentiability of (5.2), allowing us to use gradient-based optimisation methods in the estimation procedure. The L^0 penalty, on the other hand, is not differentiable at 0.

Remark. In Frommlet and Nuel (2016), the authors' method requires that each L^0 penalty be approximate by multiple interations of the approximation (5.2). However, in our method we will use only a single interation. This is due to the iterative nature of the estimation algorithm we will develop in Section 5.3.

We will precisely define Sparse Simple Exponential Principal Component Analysis (SSePCA) in the following section by its objective function.

Remark. Note that, like SePCA, SSePCA is also amenable to using ARD to determine the number of principal components needed.

5.3 Estimation

In this section, we will describe jointly the process of estimating SePCA and SSePCA. We fit both by maximum a-posteriori (MAP) estimation, with the caveat that SSePCA is technically penalised MAP. Both methods require the specification of the exponential family in order to write down explicitly the objective functions and gradients, so we give them for the specific case of the Poisson distribution and will note which parts must be adapted for other EF distributions.

We begin by deriving the objective function for SePCA, from which we can calculate the objective function for SSePCA. We begin with the conditional probability $\mathbb{P}(\mathbf{X} \mid \mathbf{W}, \mathbf{Y})$.

As this is in the exponential family, we know the general form:

$$\begin{aligned}\mathbb{P}(\mathbf{X}|\mathbf{W},\mathbf{Y}) &\propto \exp\sum_{i=1}^{n}\left[\mathbf{x}_i^{\mathrm{T}}\mathbf{W}\mathbf{y}_i + g(\mathbf{W}\mathbf{y}_i)\right] \\ &\propto \exp\sum_{i=1}^{n}\left[\mathbf{x}_i^{\mathrm{T}}\mathbf{W}\mathbf{y}_i - \sum_{j=1}^{k}\exp\left(\left[\mathbf{W}\mathbf{y}_i\right]_j\right)\right] \\ &\propto \exp\left[\mathrm{Trace}(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{Y}) - \sum_{i=1}^{n}\sum_{j=1}^{p}\exp\left([\mathbf{W}\mathbf{Y}]_{ij}\right)\right]\end{aligned}$$

where $\propto$ indicates proportionality up to a constant multiple (with respect to $\mathbf{X}$, which is fixed during an optimisation over $\mathbf{W}$ and $\mathbf{Y}$).

Similarly, the joint log-posterior of $\mathbf{W},\mathbf{Y}|\mathbf{X},\boldsymbol{\alpha}$ is, up to addition of a constant:

$$\log\mathbb{P}(\mathbf{X},\mathbf{Y},\mathbf{W},\boldsymbol{\alpha}) = \log\mathbb{P}(\mathbf{X}|\mathbf{W},\mathbf{Y}) + \log\mathbb{P}(\mathbf{Y}) + \log\mathbb{P}(\mathbf{W}|\boldsymbol{\alpha}) + \text{const}$$

Then the objective function for SePCA is

$$P = \mathrm{Trace}\left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{Y}\right) - \sum_{i=1}^{n}\sum_{j=1}^{p}\exp\left(\mathbf{W}\mathbf{Y}\right) - \frac{1}{2}\mathrm{Trace}\left(\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) - \frac{1}{2}\mathrm{Trace}\left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\,\mathrm{Diag}\left(\boldsymbol{\alpha}\right)\right) \quad (5.3)$$

where we can ignore the constant.

It is now very simple to define the objective function for SSePCA at the t^{th} iteration as

$$P^{\mathrm{s}} = P - \lambda\sum_{i=1}^{p}\sum_{j=1}^{k}\frac{\mathbf{W}_{ij}^2}{\left(\mathbf{W}_{ij}^{(t-1)}\right)^2 + \delta} \quad (5.4)$$

where the second term comes from (5.2).

We see now that SSePCA is the weighted sum of the SePCA objective function and the approximate L^0 penalty. The value of λ controls to what extent we prioritise sparsity over explanatory power for the observed data; determining the appropriate value is a heavily data- and context-dependent task. Our practical advice is to try a logarithmic sequence of λ values and select the value which best balances sparsity against predictive/discriminative power in your application.

In order to optimise these two objective functions, we make use of the differentiability with respect to both $\mathbf{W}$ and $\mathbf{Y}$. The required derivatives are

$$\frac{\partial P}{\partial \mathbf{W}} = \mathbf{X}\mathbf{Y}^{\mathrm{T}} - \exp(\mathbf{W}\mathbf{Y})\mathbf{Y}^{\mathrm{T}} - \mathbf{W}\,\mathrm{Diag}(\boldsymbol{\alpha}) \tag{5.5}$$

$$\frac{\partial P^{s}}{\partial \mathbf{W}} = \frac{\partial P}{\partial \mathbf{W}} - 2\lambda \frac{\mathbf{W}}{\left(\mathbf{W}^{(t-1)}\right)^{2} + \delta} \tag{5.6}$$

$$\frac{\partial P}{\partial \mathbf{W}} = \frac{\partial P^{s}}{\partial \mathbf{W}} = \mathbf{W}^{\mathrm{T}}\mathbf{X} - \mathbf{W}^{\mathrm{T}}\exp(\mathbf{W}\mathbf{Y}) - \mathbf{Y} \tag{5.7}$$

We can now give an algorithm for the estimation of SePCA and SSePCA in Algorithm 2. This algorithm is novel, being unlike the algorithm originally given for SePCA in Li and Tao 2013. The original algorithm used an expectation-maximisation scheme, where we directly optimise the posterior likelihood using gradient information for efficiency. We note that, like the algorithm for SGPCA, this algorithm does not have a guarantee of finding the global optimum. One strategy may be to initialise the algorithm at randomly distributed points and choose the best performing solution. In practice, we have not found this to be necessary; the (most likely local) optimum found from the proposed initial values has been sufficiently performant.

Remark. It is worth noting that we have generally found it useful to temporarily increase the value of M for the first ten iterations of the optimisation as this time is when we are most likely to spuriously remove components due to a poor starting point.

5.4 Synthetic Data Studies

All investigations in this section will use the same basic model, with some small adaptations. We will use the two hidden factors

$$V_1 \sim \text{Poisson}(20) \qquad V_2 \sim \text{Poisson}(30) \tag{5.8}$$

We will also use an error distribution E, constructed by drawing an observation from a Poisson(2) distribution and multiplying by 1 or -1 with equal probability.

The first analysis will use two datasets, with "true" dimensions 1 and 2 respectively, which we will refer to as **X1D** and **X2D**. Each consists of 100 observations of a random vector of length 10, but the construction of that vector differs. For the component selection procedure we set $M = 100$ except the first 10 iterations where $M = 500$ (as was mentioned in Section 5.3 we do this to avoid removing components too early). Also for the Sparse Simple Poisson Principal Component Analysis (SSPPCA) algorithm $\delta = 10^{-8}$.

To construct **X1D**, let v_{1i}, $i = 1,\ldots,100$ be independently observed values of V_1 and let ε_{ij}, $i = 1,\ldots,100$, $j = 1,\ldots,10$ be independently observed values of E. Then

the i^{th} observation in **X1D** has its first two components equal to v_{1i} plus error, and the remaining eight components are equal to $2 * v_{1i}$ plus error. Formally each observation has the form

$$[(v_{1i} + \varepsilon_{i1}), (v_{1i} + \varepsilon_{i2}), (2v_{1i} + \varepsilon_{i3}), \ldots, (2v_{1i} + \varepsilon_{i10})]^{\mathrm{T}}$$

To give a bit more insight here, one should expect that a good dimension reduction in this case will identify that we need exactly one component, which has larger coefficients to variables 3 to p and it has smaller coefficients for variables 1 and 2.

Similarly, the i^{th} observation in **X2D** has its first two components equal to an observed value v_{1i} of V_1 plus independent errors, its second two components equal to an observed value v_{2i} of V_2 plus independent errors, and its final six components equation to $v_{1i} + 3v_{2i}$ plus independent errors:

$$[(v_{1i} + \varepsilon_{i1}), (v_{1i} + \varepsilon_{i2}), (v_{2i} + \varepsilon_{i3}), (v_{2i} + \varepsilon_{i4}), (v_{1i} + 3v_{2i} + \varepsilon_{i5}) \ldots, (v_{1i} + 3v_{2i} + \varepsilon_{i10})]^{\mathrm{T}}$$

To both of these datasets we applied each of Sparse Probabilistic Principal Component Analysis (SPPCA), SSPPCA, PCA, Sparse Principal Component Analysis (SPCA), GPCA and SGPCA. For the latter three we needed to specify the dimension; for SPPCA and SSPPCA the automatic relevance determination criterion successfully identified the true dimension. The loadings for the one-dimensional data are given in Table 5.1; SPPCA, SSPPCA, PCA and SPCA all give very similar results qualitatively, giving equal weighting to components three through ten (corresponding to the $2v_1$ term) and slightly smaller values to the first two components corresponding to the v_1 term. GPCA gives approximately equal weighting to all the terms. SGPCA, on the other hand, gives considerably more sporadic loadings. This is perhaps due to the lack of sparsity of the underlying data.

In Table 5.2 we give the two loadings for the two-dimension data. Here, the first SPPCA loading gives roughly equal weight to the first two and last six components, corresponding to the v_1 and $v_1 + 3v_2$ terms respectively, and a slightly lower loading to the second two components (corresponding to the v_2 terms). The second SPPCA loading gives most weight to the last six components, with small weights for the second pair of components and the lowest weights to the first pair of components. The performance of SSPPCA is more easily interpretable; the first loading gives highest weighting to the last six components, with smaller weight for the first four; the second loading strongly identifies the first two components with near-zero weighting given to all other terms. PCA's first loading primarily identifies the $v_1 + 3v_2$ term, with its second primarily identifying the v_1 term; SPCA does similarly with sparser loadings. GPCA's first loading gives

approximately equal weighting to all terms (except for the very first component), with its second primarily emphasising the v_1 components. Finally, SGPCA's first loading identifies a combination of the v_1 and $v_1 + 3v_2$ terms, while its second fairly strong identifies the v_1 components. Of all the loadings, the most successful at identifying the hidden factors are the second loadings of SSPPCA, PCA, SPCA, GPCA and SGPCA, with SSPPCA, SPCA and SGPCA slightly better as the other components are driven closer to 0.

5.4.1 Order Determination

In order to investigate the accuracy of the order determination provided by ARD, we conducted similar experiments to those in Section 5.4, varying several parameters. We looked at $p \in \{10, 20\}$, $p \in \{25, 50, 100, 200\}$, $k \in \{1, 2, 3\}$. For each combination of parameters, we constructed data by the following method and used both SPPCA and SSPPCA to estimate k, repeating this 50 times in order to understand the average behaviour. When $k = 1$, the i^{th} observation ($i = 1, \ldots, n$) was given by

$$[(v_{1i} + \varepsilon_{i1}), (v_{1i} + \varepsilon_{i2}), (2v_{1i} + \varepsilon_{i3}), \ldots, (2v_{1i} + \varepsilon_{ip})]^{\mathrm{T}}$$

When $k = 2$, it was given by

$$[(v_{1i} + \varepsilon_{i1}), (v_{1i} + \varepsilon_{i2}), (v_{2i} + \varepsilon_{i3}), (v_{2i} + \varepsilon_{i4}), (v_{1i} + 3v_{2i} + \varepsilon_{i5}) \ldots, (v_{1i} + 3v_{2i} + \varepsilon_{ip})]^{\mathrm{T}}$$

Finally, when $k = 3$ it was given by

$$[(v_{1i} + \varepsilon_{i1}), (v_{1i} + \varepsilon_{i2}), (v_{2i} + \varepsilon_{i3}), (v_{2i} + \varepsilon_{i4}), (v_{3i} + \varepsilon_{i5}), (v_{3i} + \varepsilon_{i6}), \\ (3v_{1i} + 2v_{2i} + 2v_{3i} + \varepsilon_{i7}) \ldots, (3v_{1i} + 2v_{2i} + 2v_{3i} + \varepsilon_{ik})]^{\mathrm{T}}$$

For each of these, v_{1i}, v_{2i} and v_{3i} denote observations from V_1, V_2 and V_3 defined in (5.8) respectively.

Tables 5.3a and 5.3b give the percentage of times each algorithm correctly identified k for a given choice of n and k with $p = 20$. Generally, it appears that SPPCA performs better for small n, but its performance degrades as n increases. However, our proposed SSPPCA's performance improves as n increases and in fact performs significantly better by $n = 200$.

5.4.2 Synthetic Data with Classes

Although SPPCA and SSPPCA are not supervised methods, it is instructive to see whether, given data arising from two or more classes, they are able to find principal components which are able to distinguish between these classes. This gives some indication of their

Data: $\mathbf{X}$, λ, M, tol
Result: $\mathbf{W}$, $\mathbf{Y}$, k
initialise $k = p - 1$;
initialise $\mathbf{W}^{(0)}$ and $\mathbf{Y}^{(0)}$ as the loadings and score vectors of standard Principal Component Analysis (PCA);
initialise $\boldsymbol{\alpha}^{(0)} = (1, \ldots, 1)$;
set $t = 1$;
while *P changed more than tol or components removed on previous interation* **do**
  set $\mathbf{W}^{(t)}, \mathbf{Y}^{(t)} = \arg\max_{\mathbf{W},\mathbf{Y}} P$;
  for $j = 1$ **to** k **do**
    set $\alpha_j = p / \left\| \mathbf{W}_j^{(t)} \right\|_2^2$;
  end
  if *$\alpha_j > M$ for any j* **then** /* this block is the application of ARD */
    remove j^{th} component from $\boldsymbol{\alpha}$;
    remove j^{th} column from $\mathbf{W}$;
    remove j^{th} row from $\mathbf{Y}$;
    reorder components of $\boldsymbol{\alpha}$ from smallest to largest;
    reorder columns of $\mathbf{W}$ and rows of $\mathbf{Y}$ in the same order;
    decrement k;
  end
  increment t;
end

Algorithm 2: SePCA and SSePCA Estimation Procedure.

SPPCA	SSPPCA	PCA	SPCA	GPCA	SGPCA
-0.26	-0.26	0.17	0.00	0.34	0.06
-0.27	-0.27	0.17	0.00	0.33	0.42
-0.33	-0.33	0.35	0.47	0.32	-0.00
-0.33	-0.33	0.33	0.27	0.29	-0.50
-0.33	-0.33	0.34	0.35	0.30	-0.43
-0.33	-0.33	0.36	0.31	0.33	-0.06
-0.33	-0.33	0.34	0.37	0.31	-0.44
-0.33	-0.33	0.33	0.32	0.31	0.19
-0.33	-0.33	0.36	0.39	0.33	-0.01
-0.33	-0.33	0.35	0.32	0.31	-0.40

Table 5.1: Loadings for **X1D**

SPPCA	SSPPCA	PCA	SPCA	GPCA	SGPCA
-0.33	-0.23	0.01	0.00	-0.18	0.37
-0.31	-0.23	0.03	0.00	-0.31	0.04
-0.23	-0.26	0.12	0.00	-0.35	0.10
-0.23	-0.26	0.12	0.00	-0.35	0.06
-0.34	-0.36	0.41	0.40	-0.33	0.40
-0.34	-0.36	0.40	0.36	-0.32	0.40
-0.34	-0.36	0.40	0.42	-0.32	0.38
-0.34	-0.36	0.41	0.40	-0.33	0.37
-0.34	-0.36	0.40	0.45	-0.33	0.31
-0.34	-0.36	0.39	0.42	-0.31	0.38

(a) First loading

SPPCA	SSPPCA	PCA	SPCA	GPCA	SGPCA
0.12	-0.76	0.73	-0.82	0.71	0.67
0.14	-0.64	0.64	-0.57	0.61	0.68
0.29	0.09	-0.18	0.00	-0.20	-0.10
0.28	0.08	-0.16	0.00	-0.19	-0.05
0.37	0.00	0.01	0.00	-0.09	-0.11
0.36	0.00	0.07	0.00	-0.08	-0.09
0.37	-0.00	-0.00	0.00	-0.09	-0.12
0.37	0.00	-0.02	0.00	-0.10	-0.12
0.37	0.00	-0.00	0.00	-0.09	-0.08
0.37	0.00	-0.02	0.00	-0.09	-0.15

(b) Second Loading

Table 5.2: Two loadings from **X2D**

	d		
N	1	2	3
25	94	24	18
50	82	62	26
100	62	24	26
200	24	16	14

(a) SPPCA

	d		
N	1	2	3
25	2	8	4
50	42	10	8
100	82	60	18
200	78	70	50

(b) SSPPCA

Table 5.3: Percentage of correct identification of d for SPPCA and SSPPCA

suitability for use as a step before applying a clustering or classification algorithm (depending on whether labels are available or not). To this end, we construct two sets of classed data; the first having observations from two classes with equal sample sizes from both, the second having three classes with imbalanced sample sizes.

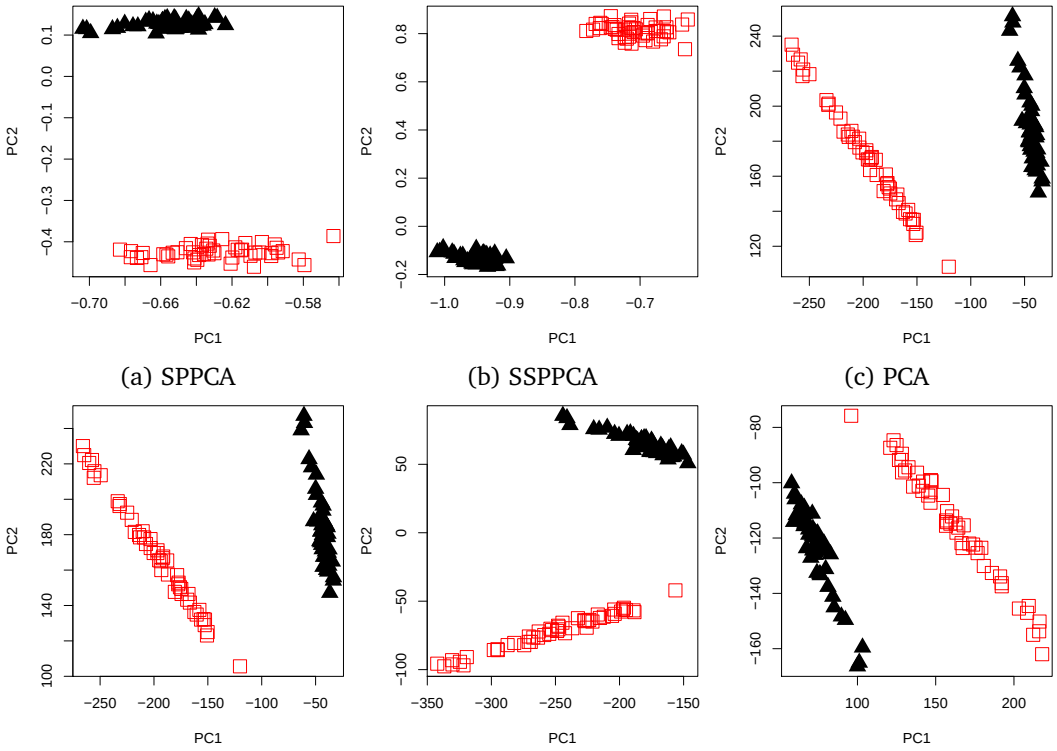
We will use again the hidden factors from (5.8) and both datasets have dimension $p = 10$ and total sample size $n = 100$. We will denote the two-class data by **X2C** and the three-class data by **X3C**. The first class for both datasets will have its first two components equal to observations v_2 of V_2 with independent error E and the remaining eight components equal to $3v_2$ with independent error. The second class for both will have first two components equal to $2v_3$ with independent error and the remaining eight components equal to v_3, where the v_3 are observations of V_3. The third class will have all components equal to observations from V_1 with independent error. The two-class data **X2C** has 50 observations from the first class and 50 from the second. The three-class data **X3C** is divided between 25 observations of the first class, 25 observations of the second class, and 50 observations of the third class.

The loadings from applying SPPCA, SSPPCA, GPCA, SGPCA, PCA and SPCA to **X2C** are given in Figure 5.2. For GPCA, SGPCA, PCA and SPCA we must specify a dimension: as both SPPCA and SSPPCA choose $k = 2$ we use that value. All six algorithms achieve good separation of the two classes. Visually, it appears that SPPCA and SSPPCA (in Figures 5.2a and 5.2b respectively) give the best clustering of the two classes. We use the method of silhouettes put forward by Rousseeuw (1987) to confirm this, using the Euclidean distance metric and clusters found using k-medioid clustering. The silhouette of the i^{th} observation is given by

$$\frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average dissimilarity of the i^{th} observation to the other members of its cluster and $b(i)$ is the lowest average dissimilarity of the i^{th} observation to any other cluster. We can thus interpret the silhouette as a measure of how well a data point is assigned to its cluster; the average silhouette over a dataset gives a measure for how well clustered the data is. Average silhouette values range between -1 and 1; the closer to 1 the better the clustering. In Table 5.4 we give average silhouettes for **X2C** for each of the six algorithms. Our visual intuition that SPPCA and SSPPCA give the best clustering is confirmed. For **X3C** we can see that SSPPCA is actually the only algorithm that is able to achieve separation of the 3 classes using only the first principal component. All other methods need the second direction to achieve this separation. When we compare the average silhouette measure in Table 5.4 we see that SSPPCA is actually behind the other

methods which have similar values but this may as well be due to the fact that we do not apply the penalty for multiple iterations as was explained in Section 5.3.



(a) SPPCA (b) SSPPCA (c) PCA

(d) SPCA (e) GPCA (f) SGPCA

Figure 5.2: Two directions from each algorithm for **X2C**, with one class shown with red squares, the other with black triangles.

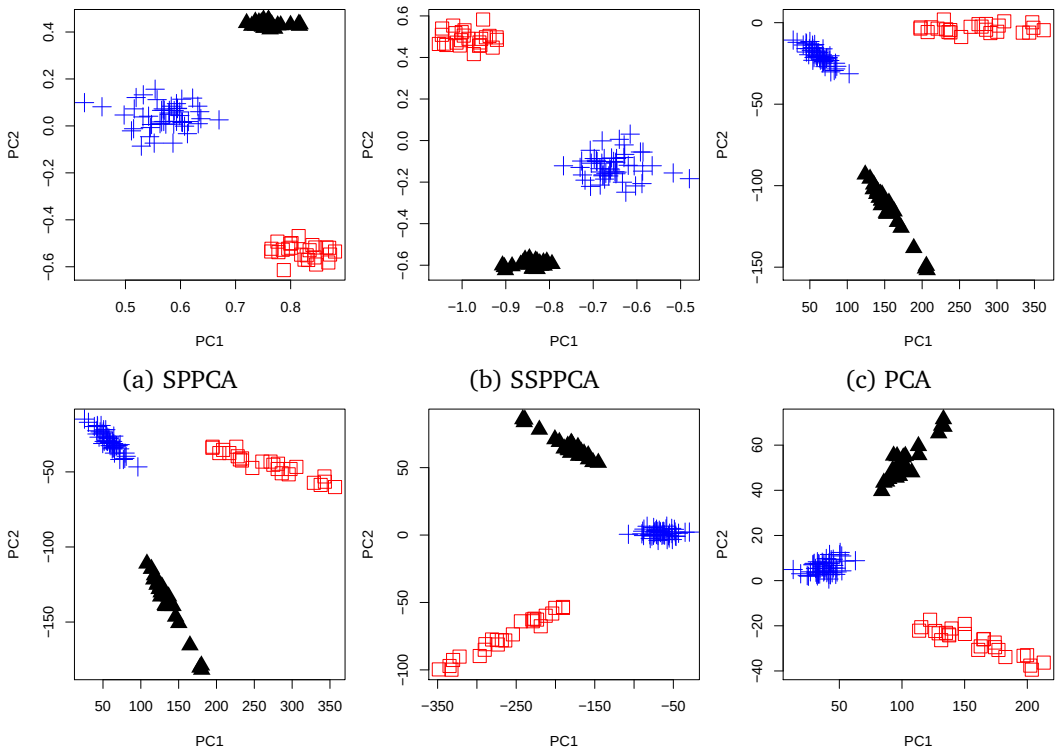
	SPPCA	SSPPCA	PCA	SPCA	GPCA	SGPCA
X2C	0.94	0.95	0.75	0.75	0.75	0.67
X3C	0.86	0.86	0.78	0.79	0.78	0.78

Table 5.4: Average (Euclidean) silhouettes

5.5 Healthcare Data

We will now examine the efficacy of SPPCA and SSPPCA in reducing the dimension of the healthcare dataset as detailed in Chapter 3.

In Figure 5.4 we show the results of applying SPPCA, SSPPCA, GPCA and SGPCA to this dataset. Discharge data points are shown with crosses and follow up points are



(a) SPPCA (b) SSPPCA (c) PCA

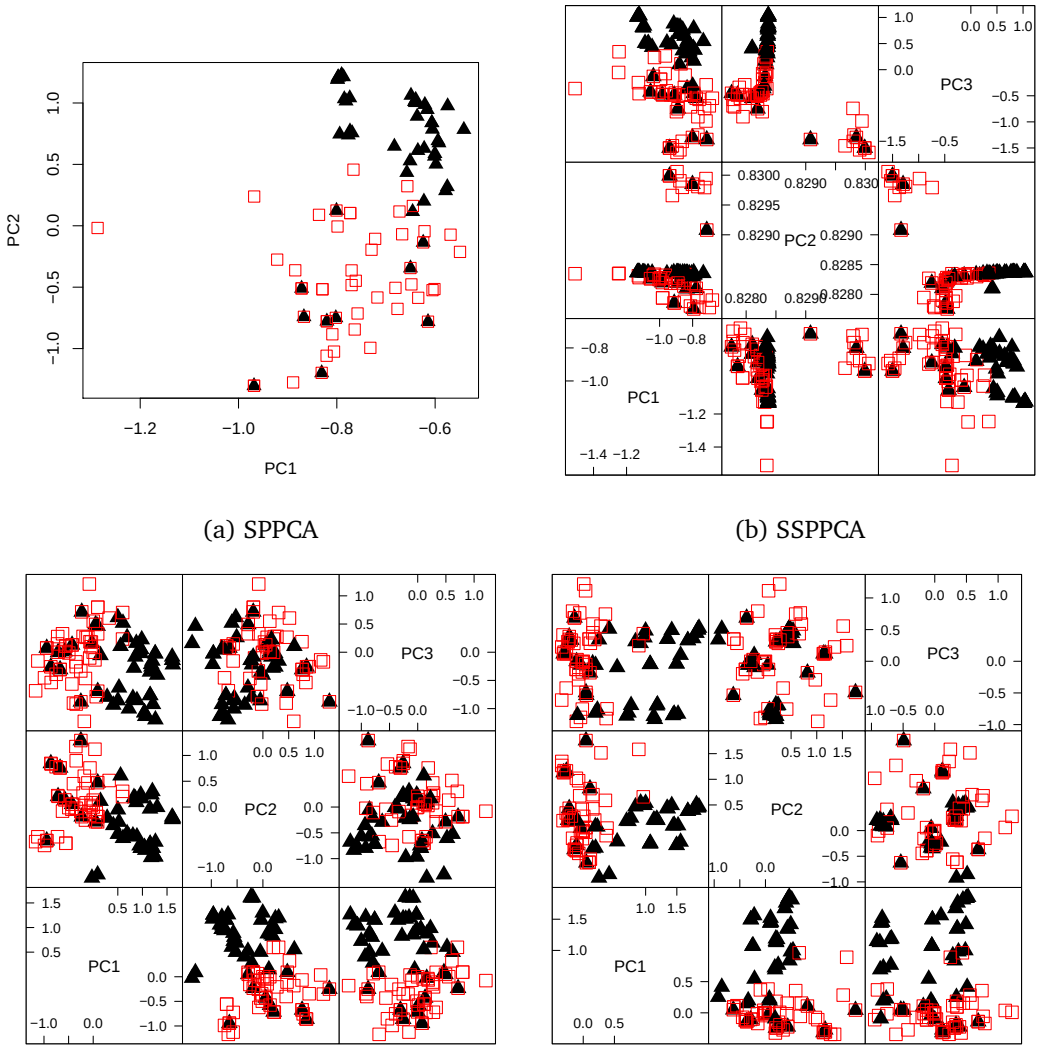
(d) SPCA (e) GPCA (f) SGPCA

Figure 5.3: Two directions from each algorithm for **X3C**

shown with circles. For both SPPCA and SSPPCA we used $M = 100$, for the latter we chose $\lambda = 0.06$. From Figure 5.4a we can see that SPPCA estimated k as 1; on the other hand, from Figure 5.4b we see that SSPPCA chose $k = 2$. Based on this, we chose $k = 2$ for GPCA and SGPCA, which require a fixed value. The difficulty inherent in achieving good class separation as a result of the strong class imbalance is evident. Of the four, GPCA (Figure 5.4c) is the worst, with the discharge points not particularly tightly clustered and difficult to separate from the follow up points. SGPCA (Figure 5.4d, on the other hand, tightly clusters the discharge points. SPPCA (Figure 5.4a, also manages to tightly cluster the discharge points. SSPPCA (Figure 5.4b) does not cluster the discharge points particularly closely, but does achieve better visual separation than GPCA.

In order to better quantify the clustering, we give the average (Euclidean) silhouettes in Table 5.5. Based on this performance metric, SPPCA and SGPCA are the best performers. Average silhouette is designed to measure and inform about the performance of clustering algorithms and not the accuracy of feature extraction. The fact that SPPCA seems to be worst may either be due to this or to the fact that we are not doing

multiple iterations of the penalty.



(a) SPPCA

(b) SSPPCA

(c) GPCA

(d) SGPCA

Figure 5.4: The resulting principal components from applying SPPCA, SSPPCA, GPCA and SGPCA to the healthcare data. The "discharge" class is plotted as red squares, the "follow-up" class is shown as black triangles.

5.6 Discussion

In this paper we have developed a Poisson based PCA algorithm which we called SPPCA and which was based on the SePCA (Li and Tao 2013). We use a different algorithm

SPPCA	SSPPCA	GPCA	SGPCA	PCA
0.38	0.34	0.23	0.17	0.21

Table 5.5: Average silhouettes first the healthcare data.

for inference on **W** and **Y** than SePCA. We have illustrated that by example in the specific case where the distribution is Poisson, to give SPPCA. We have also introduced an approximate L^0 sparsity penalty in this context to allow for Sparse SPPCA. In a more general framework this can be seen as a unified way of achieving sparse or non-sparse feature extraction from a Poisson-based PCA algorithm. At the same time this algorithm should be straightforward to extend to other distributions in the exponential family by modifying appropriately the formulas.

The sparse algorithm performs particularly well, both in latent dimension discovery and in class separation for multi class Poisson data. Computation times are acceptable for small samples ($n \leq 500$), but become a slightly more burdensome for larger samples. It is worth noting that there exist multiple solutions or local maxima. This is also dealt with simply, by evaluating multiple optima using fully specified probability model. In practice, we have found that this has not been necessary, the maxima obtained starting from the Gaussian PCA have performed perfectly well.

There is scope for extension of this work. First of all it is interesting to introduce different more complex sparsity penalties, such as the L^1 or SCAD penalties and compare their performance. Another possible extension is the development of nonlinear feature extraction methods as well as sparse nonlinear feature extraction method in the generalised PCA setting for non-Gaussian data.

— Chapter 6 —

Quasi-Likelihood Principal Component Analysis

6.1 Introduction

The two Principal Component Analysis (PCA)-like methods we have concentrated on so far share a number of similarities – both aim to optimise some criterion. In the case of (S)GPCA this is the (penalised) deviance, and in the case of (S)SePCA it is the (penalised) posterior likelihood. As an optimisation problem of fairly simple functions, we can express these methods as finding the zeros of the derivatives. In fact, there is a rich theory for estimating statistical parameters from such a premise, the theory of estimating equations. However, this theory has, to date, focused on solving problems with scalar or vector parameters, whilst dimension reduction methods like we are interested in require higher dimensional analogues. In this chapter, we will develop the theory of estimating equations for tensor parameters. We shall do so with the additional restriction of requiring that our estimating equations are themselves tensors in order to have a pleasing invariance with respect to changes of parametrisations.

The purpose of this theory is to provide a framework for the development and understanding of PCA-like methods. As such, having developed a theory for tensor-parameter estimating equations, we will then apply that theory to a range of dimension reduction methods in the literature in order to better understand the differences between each. To illustrate this technique, we shall apply it to three important methods: Generalised Principal Component Analysis (Landgraf and Lee 2015b), the method of Collins et al. (2002), and Simple Exponential Family Principal Component Analysis (Li and Tao 2013).

It is worth noting that, by expressing these methods as estimating equations, one can very easily write code to find the dimension reduction parameters using a standard root-finding algorithm. In many programming languages, it is even possible to input the

appropriate equations symbolically. While this will rarely be the most computationally efficient method, it illustrates a very compelling feature of the estimating equations viewpoint – newly devised methods can be tried experimentally with very little time investment.

6.2 Tensor Estimating Equations

Before defining tensor estimating equations, we will recap the vector case, in order to illustrate the concepts and provide motivation to the tensor extension. We will then introduce tensors and some key properties, before showing the extension of vector estimating equations to the tensor case.

6.2.1 Vector Parameter Estimating Equations

Suppose we are given a sample of data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ from a random variable $\mathbf{X}$ which takes values in $\mathscr{X} \subseteq \mathbb{R}^p$ with a family of probability distributions $\mathscr{P} := \{\mathbb{P}_\theta : \theta \in \Theta\}$ and set of possible parameters Θ being an open subset of $\mathbb{R}^q$. As a method for estimating the "true" parameter value θ, we are interested in determining an optimal estimating function $\mathbf{G} : \mathscr{X}^n \times \Theta \to \mathbb{R}^q$ whose roots give good estimates of the "true" θ.

Let $\mathscr{G}$ be the class of estimation functions. We require that each $\mathbf{G}$ has zero mean, is square-integrable, and for which $\mathbb{E}\left(\dot{\mathbf{G}}\right) := \left(\mathbb{E}\left(\frac{\partial \mathbf{G}_i}{\partial \theta_j}\right)\right)$ and $\mathbb{E}\left(\mathbf{G}\mathbf{G}^{\mathrm{T}}\right)$ are non-singular. To compare estimating functions, we will need a notion of standardisation, given in Definition 6.2.1. This will allow us to find an optimal estimating function for parameter estimation.

Definition 6.2.1 (Standardised Estimating Function)**.** Given an estimating function $\mathbf{G} \in \mathscr{G}$, the corresponding *standardised* estimating function $\underset{\sim}{\mathbf{G}}$ is

$$-\mathbb{E}\left(\dot{\mathbf{G}}^{\mathrm{T}}\right)\left(\mathbb{E}\left(\mathbf{G}\mathbf{G}^{\mathrm{T}}\right)\right)^{-1}\mathbf{G}$$

We now seek a notion of finite-sample optimality, which we will denote O_F-optimality. This will use an information criterion $\mathscr{E}(\cdot)$, whose definition is given in Definition 6.2.2, which is an analogue to the Fisher Information.

Definition 6.2.2 (Information Criterion)**.** The *information criterion* $\mathscr{E}(\mathbf{G})$ is given by

$$\mathscr{E}(\mathbf{G}) := \mathbb{E}\left(\underset{\sim}{\mathbf{G}}\underset{\sim}{\mathbf{G}}^{\mathrm{T}}\right) = \left(\mathbb{E}\left(\dot{\mathbf{G}}\right)\right)^{\mathrm{T}}\left(\mathbb{E}\left(\mathbf{G}\mathbf{G}^{\mathrm{T}}\right)\right)^{-1}\left(\mathbb{E}\left(\dot{\mathbf{G}}\right)\right)$$

We are now able to give the definition of O_F-optimality. It will be given in terms of a subclass $\mathscr{H}$ of estimating functions, as this will usually be the case in practice.

Definition 6.2.3 (O_F-Optimality)**.** Let $\mathscr{H} \subseteq \mathscr{G}$. Then we call $\mathbf{G}_* \in \mathscr{H}$ the O_F-optimal estimating function if $\mathscr{E}(\mathbf{G}_*) - \mathscr{E}(\mathbf{G})$ is positive semi-definite for all $\mathbf{G} \in \mathscr{H}$, $\theta \in \Theta$ and $\mathbb{P}_\theta \in \mathscr{P}$.

This concept is known as Loewner optimality in experimental design, its name originating from the Loewner partial ordering on positive semi-definite matrices. Should the score function $\mathbf{U}$ exist, there is an alternative and equivalent definition:

Definition 6.2.4. The estimating function $\mathbf{G}_* \in \mathscr{H}$ is O_F-optimal in $\mathscr{H}$ if for all $\mathbf{G} \in \mathscr{H}$, $\theta \in \Theta$ and $\mathbb{P}_\theta \in \mathscr{P}$

$$\mathbb{E}\left(\left(\underset{\sim}{\mathbf{U}} - \underset{\sim}{\mathbf{G}}\right)\left(\underset{\sim}{\mathbf{U}} - \underset{\sim}{\mathbf{G}}\right)^{\mathrm{T}}\right) - \mathbb{E}\left(\left(\underset{\sim}{\mathbf{U}} - \underset{\sim}{\mathbf{G}}_*\right)\left(\underset{\sim}{\mathbf{U}} - \underset{\sim}{\mathbf{G}}_*\right)^{\mathrm{T}}\right)$$

is positive semi-definite.

Definition 6.2.4 can be seen as minimising the dispersion distance of $\underset{\sim}{\mathbf{G}}_*$ from the score function. For this reason, such optimal estimating functions are also known as "quasi-score" functions, as they behave like score functions in many respects and they are the optimal approximation to this score function (if it exists) within a given class of functions.

Finally, we give one more equivalent criterion for O_F-optimality which will be useful in practice.

Theorem 6.2.1. *The estimating function $\mathbf{G}_* \in \mathscr{H} \subseteq \mathscr{G}$ is O_F-optimal if*

$$\mathbb{E}\left(\underset{\sim}{\mathbf{G}}_* \underset{\sim}{\mathbf{G}}^{\mathrm{T}}\right) = \mathbb{E}\left(\underset{\sim}{\mathbf{G}} \underset{\sim}{\mathbf{G}}_*^{\mathrm{T}}\right) = \mathbb{E}\left(\underset{\sim}{\mathbf{G}} \underset{\sim}{\mathbf{G}}^{\mathrm{T}}\right) \tag{6.1}$$

for all $\mathbf{G} \in \mathscr{H}$. Equivalently, if $\left(\mathbb{E}\left(\dot{\mathbf{G}}\right)\right)^{-1}\mathbb{E}(\mathbf{G})\,\mathbb{E}(\mathbf{G}_*)$ is a constant matrix. In the converse direction, if $\mathbf{G}_* \in \mathscr{H}$ is an O_F-optimal estimating function and $\mathscr{H}$ is convex then (6.1) holds.

Theorem 6.2.2. *If the score function $\mathbf{u}$ exists and $\mathbf{u} \in \mathscr{H}$, then $\mathbf{u}$ is the optimal estimating function.*

6.2.2 Tensor Preliminaries

Perhaps the most familiar way to view tensors is as multi-dimensional arrays. Much as vectors are "larger" scalars, and matrices are "larger" vectors, tensors form a natural way to think about such objects in arbitrary dimensions. To be more precise, any tensor can be represented as a multi-dimensional array with respect to a basis, just as any vector can be represented by a one-dimensional array with respect to a basis in the appropriate vector space.

The final distinctive element of tensors is the concept of contravariance and covariance. These describe the way that a given vector transforms under a change of basis. Briefly, these are defined below.

Definition 6.2.5. A vector $\mathbf{v}$ with components $v^1, \ldots, v^n$ with respect to a basis $\mathbf{e}_1, \ldots, \mathbf{e}_n$ is *contravariant* if the new components $\hat{v}^1, \ldots, \hat{v}^n$ with respect to a new basis $\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_n$ are given by

$$\hat{v}^i = \sum_{j=1}^{n} \left(\mathbf{R}^{-1}\right)_j^i v^j$$

where $\mathbf{R}$ is the mapping such that $\hat{\mathbf{e}}_i = \sum_{j=1}^n \mathbf{e}_j \mathbf{R}_i^j$.

Definition 6.2.6. A vector $\mathbf{w}$ with components $w_1, \ldots, w_n$ with respect to a basis $\mathbf{e}_1, \ldots, \mathbf{e}_n$ is *covariant* if the new components $\hat{w}_1, \ldots, \hat{w}_n$ with respect to the new basis $\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_n$ are given by

$$\hat{w}_i = \sum_{j=1}^{n} w_j \mathbf{R}_i^j$$

with $\mathbf{R}$ again the mapping such that $\hat{\mathbf{e}}_i = \sum_{j=1}^n \mathbf{e}_j \mathbf{R}_i^j$.

The above definitions also introduce a piece of common notation when working with tensors, where subscript indices are used for covariant components and superscript indices are used for contravariant components. We will make use of this notation going forward. A second piece of notation which is very common is that of "Einstein notation" or the "Einstein summation convention". Many tensor definitions and operations involve summations, in order to reduce the notation needed, it will be taken as understood that a repeated algebraic index such as i or j is to be summed over. For example, we can write the following:

$$\hat{w}_i = \sum_{j=1}^{n} w_j \mathbf{R}_i^j = w_j \mathbf{R}_i^j$$

We will use this convention for the remainder of this chapter.

We are now in a position to give a formal definition for a tensor.

Definition 6.2.7. A tensor of type (p, q) has a multidimensional array associated with respect to the basis $\mathbf{f}$, denoted by $T^{i_1,\ldots,i_p}_{j_1,\ldots,j_q}[\mathbf{f}]$ which, under the basis transformation $\mathbf{f} \mapsto \mathbf{f} \cdot \mathbf{R} = \left(\mathbf{e}_i \mathbf{R}_1^i, \ldots, \mathbf{e}_i \mathbf{R}_n^i\right)$ transforms as

$$T^{i'_1,\ldots,i'_p}_{j'_1,\ldots,j'_q}[\mathbf{f} \cdot \mathbf{R}] = \left(R^{-1}\right)^{i'_1}_{i_1} \cdot \left(R^{-1}\right)^{i'_p}_{i_p} T^{i_1,\ldots,i_p}_{j_1,\ldots,j_q} R^{j_1}_{j'_1} \cdot R^{j_q}_{j'_q}$$

In this definition, type (p,q) means that the tensor has p contravariant indices and q covariant indices. We use the prime indices (e.g. i_1') to denote indices under the new basis. From here, we will usually drop the explicit reference to the basis in the notation, so we write (for example) $T^{i_1,\ldots,i_p}_{j_1,\ldots,j_q}$, and will use an asterisk $(*)$ to denote the tensor under a change of basis.

6.2.3 Vector Estimating Equations as Tensors

In our extension to tensor estimating equations, we will begin by considering the simple case of a vector estimating equation, and give the following theorem showing when such objects are tensors.

Theorem 6.2.3. *If, for a given estimating equation* $\mathbf{G}$*, there exists a scalar potential function* g *such that* $\mathbf{G} = \nabla_\theta g$*, then* $\mathbf{G}$ *obeys the tensor transformation laws over all differentiable transformations of the parameters.*

Proof. Denote the original parameters by $\theta = \theta^1, \ldots, \theta^p$, and the new parameters by $\boldsymbol{\phi} = \phi^1, \ldots, \phi^p$. Let $\mathbf{a} : \Phi \to \Theta$ map each parameter $\boldsymbol{\phi}$ to the corresponding θ. Then our potential function under change of variables is g^*, defined as $g^* := g \circ \mathbf{a}$. The estimating equation generated by g^* is $\mathbf{G}^* := \nabla_\phi$. Then we have

$$\mathbf{G^*}_r = \mathbf{G}_i a^i_r \tag{6.2}$$

where $a^i_j = \partial a^i / \partial \phi^j$. This is precisely the tensor transformation rule for a covector. □

We must now check that standardisation of estimating equations preserves the tensor transformation rule. First, let us write the standardisation in tensor notation. We will denote by σ_{ij} the covariance matrix $\mathbb{E}\left(\mathbf{G}\mathbf{G}^{\mathrm{T}}\right)$, and by σ^{ij} its inverse. The expectation of the first derivative, $\mathbb{E}\left(\dot{\mathbf{G}}\right)$ will be denoted by γ_{ij}. Then we have:

$$\underset{\sim}{\mathbf{G}}_r = -\gamma_{ir}\sigma^{ij}\mathbf{G}_j \tag{6.3}$$

and

$$\mathbf{G}_r = -\gamma^{ij}\sigma_{rj}\underset{\sim}{\mathbf{G}}_i \tag{6.4}$$

Then, with τ_{ij} being the covariance matrix of $\mathbb{E}\left(\mathbf{G}^*\mathbf{G}^{*\mathrm{T}}\right)$ and τ^{ij} its inverse and ς_{ij} the expectation of the first derivative, we also have

$$\underset{\sim}{\mathbf{G}^*}_r = -\varsigma_{ir}\tau^{ij}\mathbf{G^*}_j \tag{6.5}$$

For later simplification, we first give a lemma regarding τ and ς:

Lemma 6.2.1. *τ, its inverse and ς are given by the following expressions*

$$\tau_{ij} = \gamma_{ik}\gamma_{jl}\sigma^{kl} \tag{6.6}$$

$$\tau^{ij} = \gamma^{ik}\gamma^{jl}\sigma_{kl} \tag{6.7}$$

$$\varsigma_{ij} = \gamma_{ik}\gamma_{jl}\sigma^{kl} \tag{6.8}$$

We now give a theorem for the transformation law of standardised estimating equations.

Theorem 6.2.4. *Standardised estimating equations follow a tensor transformation law.*

Proof. Combining (6.2) and (6.5), we obtain:

$$\underset{\sim}{\mathbf{G}}^*_r = -\varsigma_{ir}\tau^{ij}\left(\mathbf{G}_k a^k_j\right) = -a^k_j\tau^{ij}\varsigma_{ir}\mathbf{G}_k = c^i_r\mathbf{G}_i \tag{6.9}$$

where $c^i_r := -a^i_j\tau^{kj}\varsigma_{kr}$. Then, using (6.4), we obtain:

$$\underset{\sim}{\mathbf{G}}^*_r = a^i_j\tau^{kj}\varsigma_{kr}\gamma^{lm}\sigma_{im}\underset{\sim}{\mathbf{G}}_l \tag{6.10}$$

Finally, using (6.8) and (6.7), we obtain

$$\underset{\sim}{\mathbf{G}}^*_r = a^i_j\gamma^{lk}\gamma^{mj}\sigma_{lm}\gamma_{kn}\gamma_{ro}\sigma^{no}\gamma^{pq}\sigma_{iq}\underset{\sim}{\mathbf{G}}_p = a^i_r\sigma_{ik}\gamma^{kj}\underset{\sim}{\mathbf{G}}_j \tag{6.11}$$

□

Finally, we give the information criterion and optimality criterion in tensor notation:

Definition 6.2.8. Denoting, as in Theorem 6.2.4, the the covariance matrix $\mathbb{E}\left(\mathbf{G}\mathbf{G}^{\mathrm{T}}\right)$ by σ_{ij}, and by σ^{ij} its inverse and $\mathbb{E}\left(\dot{\mathbf{G}}\right)$ by γ_{ij} then the information criterion $\mathscr{E}(\mathbf{G})$, denoted by $\mathscr{E}_{ij}$ is:

$$\mathscr{E}_{ij} := \gamma_{ik}\gamma_{jl}\sigma^{kl}$$

Theorem 6.2.5. *The information criterion is invariant under change of coordinates.*

Proof. Using the previous notation, we have:

$$\mathscr{E}(\mathbf{G}^*)_{ij} = \varsigma_{ki}\tau^{kl}\varsigma_{lj} \tag{6.12}$$

Substituting using (6.7) and (6.8), we obtain:

$$\begin{aligned}\mathscr{E}(\mathbf{G}^*)_{ij} &= \gamma_{km}\gamma_{in}\sigma^{mn}\gamma^{ko}\gamma^{lp}\sigma_{op}\gamma_{lq}\gamma_{jr}\sigma^{qr}\\ &= \gamma_{km}\gamma^{ko}\gamma_{in}\gamma^{ko}\gamma_{lq}\gamma^{lp}\sigma_{op}\sigma^{qr}\end{aligned}$$

$$\mathscr{E}(\mathbf{G}^*)_{ij} = \gamma_{ik}\gamma_{jl}\sigma^{kl} \tag{6.13}$$

□

6.2.4 Tensor Estimating Equations

We are now ready to formulate the theory of tensor estimating equations. We will make the following necessary definitions, analogues of those given for vector estimating equations in the previous section.

Definition 6.2.9. A **tensor estimating equation of order** $(p_1, \ldots, p_D)$ is a multi-index array of functions $\mathbf{G}_{i_1,\ldots,i_p} : \mathscr{X} \times \Theta \to \mathbb{R}$, where the indices $i_1, \ldots, i_D$ run from 1 through to the corresponding element of $(p_1, \ldots, p_D)$. When there is no confusion, we will write $\mathscr{I}$ for $i_1, \ldots, i_D$, and similarly with other indices. Using this notation, we will write an estimating equation of order $(p_1, \ldots, p_D)$ as $\mathbf{G}_{\mathscr{I}}$. We require the following properties:

- There exists a potential g such that $\mathbf{G}_{\mathscr{I}} = \frac{\partial g}{\partial \theta_{\mathscr{I}}}$
- $\mathbb{E}(\mathbf{G}_{\mathscr{I}}) = 0$
- $\mathbb{E}\left(\dot{\mathbf{G}}_{\mathscr{I}\mathscr{J}}\right) \coloneqq \mathbb{E}\left(\frac{\partial G_{\mathscr{I}}}{\partial \theta_{\mathscr{J}}}\right)$ is non-singular, i.e. invertible.

Remark. We remark that the first condition, the existence of a potential function g is not standard in the consideration of general estimating equations. However, general multi-index arrays satisfying the conditions of estimating equations will fail the tensor transformation laws. The restriction to the class of multi-index arrays generated as the gradient of a scalar function guarantees the desired tensor properties, as we will prove. For its importance, we will refer to this condition as the assumption of potential (AOP).

Definition 6.2.10. The variance-covariance tensor for a tensor of order $(p_1, \ldots, p_D)$ is

$$\sigma_{\mathscr{I}\mathscr{J}} = \mathbb{E}\left(\mathbf{G}_{\mathscr{I}}\mathbf{G}_{\mathscr{J}}\right) \tag{6.14}$$

Definition 6.2.11. Given an estimating equation $\mathbf{G}$, the standardised estimating equation $\underset{\sim}{\mathbf{G}}$ is defined by:

$$\underset{\sim}{\mathbf{G}}_{\mathscr{R}} \coloneqq -\gamma_{\mathscr{I}\mathscr{R}}\sigma^{\mathscr{I}\mathscr{J}}\mathbf{G}_{\mathscr{J}} \tag{6.15}$$

where $\gamma_{\mathscr{I}\mathscr{J}}$ is the expected first derivative of $\mathbf{G}_{\mathscr{I}}$ with respect to $\theta_{\mathscr{J}}$.

Definition 6.2.12. The information criterion $\mathscr{E}_{\mathscr{I}\mathscr{J}}$ associated with an estimating equation $\mathbf{G}_{\mathscr{I}}$ is

$$\mathscr{E}_{\mathscr{I}\mathscr{J}} = \gamma_{\mathscr{I}\mathscr{K}}\gamma_{\mathscr{J}\mathscr{L}}\sigma^{\mathscr{K}\mathscr{L}} \tag{6.16}$$

We are now ready to prove the tensor versions of the three theorems of Section 6.2.3. The proofs are almost identical, save for the additional complication that where we before largely dealt with pairs of indices, we are now dealing with pairs of index

sets. However, the choice of notation largely protects us from this additional complexity. Firstly, we will prove that our referring to these objects as tensor estimating equations is, in fact, justified.

Theorem 6.2.6. *Tensor estimating equations are tensors.*

Proof. Denote the original parameters by $\theta_{\mathscr{I}}$, and the new parameters by $\phi_{\mathscr{I}}$. Let $a : \Phi \to \Theta$ map each parameter $\phi_{\mathscr{I}}$ to the corresponding $\theta_{\mathscr{I}}$. Here a is a multi-dimensional array of the same dimensions as θ. Then our potential function under change of variables is g^*, defined as $g^* \coloneqq g \circ a$. The estimating equation generated by g^* is $\mathbf{G}^* \coloneqq \nabla_\phi g^*$. Then we have

$$\mathbf{G}^*{}_{\mathscr{R}} = \mathbf{G}_{\mathscr{I}} a^{\mathscr{I}}_{\mathscr{R}} \tag{6.17}$$

where $a^{\mathscr{I}}_{\mathscr{J}} \coloneqq \partial a^{\mathscr{I}} / \partial \phi^{\mathscr{J}} = \left(\nabla_\phi a\right)^{\mathscr{I}}_{\mathscr{J}}$. Then we have that estimating equations transform according to the law for a covector; that is, they are tensors. □

As in Section 6.2.3, we will now prove a short lemma giving some useful identities which will be necessary to prove that standardised estimating equations also followed a tensor transformation law.

Lemma 6.2.2. *τ, its inverse and ς are given by the following expressions*

$$\tau_{\mathscr{I}\mathscr{J}} = \gamma_{\mathscr{I}\mathscr{K}} \gamma_{\mathscr{J}\mathscr{L}} \sigma^{\mathscr{K}\mathscr{L}} \tag{6.18}$$

$$\tau^{\mathscr{I}\mathscr{J}} = \gamma^{\mathscr{I}\mathscr{K}} \gamma^{\mathscr{J}\mathscr{L}} \sigma_{\mathscr{K}\mathscr{L}} \tag{6.19}$$

$$\varsigma_{\mathscr{I}\mathscr{J}} = \gamma_{\mathscr{I}\mathscr{K}} \gamma_{\mathscr{J}\mathscr{L}} \sigma^{\mathscr{K}\mathscr{L}} \tag{6.20}$$

Theorem 6.2.7. *Standardised tensor estimating equations are tensors.*

Proof. Denoting, as before, the variance tensor of the estimating equation under a change of coordinates ($\mathbf{G}_{*\mathscr{I}}$) by $\varsigma_{\mathscr{I}\mathscr{J}}$ and the expected first derivative by $\tau_{\mathscr{I}\mathscr{J}}$, we have:

$$\underset{\sim}{\mathbf{G}}^*_{\mathscr{R}} = -\varsigma_{\mathscr{I}\mathscr{R}} \tau^{\mathscr{I}\mathscr{J}} \mathbf{G}^*{}_{\mathscr{J}} \tag{6.21}$$

We will also need the following expression for the inverse of standardisation.

$$\mathbf{G}_{\mathscr{R}} = -\gamma^{\mathscr{I}\mathscr{J}} \sigma_{\mathscr{R}\mathscr{J}} \underset{\sim}{\mathbf{G}}_{\mathscr{I}} \tag{6.22}$$

Combining (6.17) and (6.21), we obtain:

$$\underset{\sim}{\mathbf{G}}^*_{\mathscr{R}} = -\varsigma_{\mathscr{I}\mathscr{R}} \tau^{\mathscr{I}\mathscr{J}} \left(\mathbf{G}_{\mathscr{K}} a^{\mathscr{K}}_{\mathscr{J}}\right) = -a^{\mathscr{K}}_{\mathscr{J}} \tau^{\mathscr{I}\mathscr{J}} \varsigma_{\mathscr{I}\mathscr{R}} \mathbf{G}_{\mathscr{K}} = c^{\mathscr{I}}_{\mathscr{R}} \mathbf{G}_{\mathscr{I}} \tag{6.23}$$

where $c^{\mathscr{I}}_{\mathscr{R}} := -a^{\mathscr{I}}_{\mathscr{J}} \tau^{\mathscr{K}\mathscr{J}} \varsigma_{\mathscr{K}\mathscr{R}}$. Then, using (6.22), we obtain:

$$\underset{\sim}{\mathbf{G}}^*_{\mathscr{R}} = a^{\mathscr{I}}_{\mathscr{J}} \tau^{\mathscr{K}\mathscr{J}} \varsigma_{\mathscr{K}\mathscr{R}} \gamma^{\mathscr{L}\mathscr{M}} \sigma_{\mathscr{I}\mathscr{M}} \underset{\sim}{\mathbf{G}}_{\mathscr{L}} \tag{6.24}$$

Finally, using (6.20) and (6.19), we obtain

$$\underset{\sim}{\mathbf{G}}^*_{\mathscr{R}} = a^{\mathscr{I}}_{\mathscr{J}} \gamma^{\mathscr{L}\mathscr{K}} \gamma^{\mathscr{M}\mathscr{J}} \sigma_{\mathscr{L}\mathscr{M}} \gamma_{\mathscr{K}\mathscr{N}} \gamma_{\mathscr{R}\mathscr{O}} \sigma^{\mathscr{N}\mathscr{O}} \gamma^{\mathscr{P}\mathscr{Q}} \sigma_{\mathscr{I}\mathscr{Q}} \underset{\sim}{\mathbf{G}}_{\mathscr{P}} = a^{\mathscr{I}}_{\mathscr{R}} \sigma_{\mathscr{I}\mathscr{K}} \gamma^{\mathscr{K}\mathscr{J}} \underset{\sim}{\mathbf{G}}_{\mathscr{J}} \tag{6.25}$$

which is a tensor transformation law. □

Finally, we will show that, as in the vector case, the information criterion remains an invariant under coordinate transformations. This result is fundamental to the use of estimating equations, reassuring us that our choice of coordinate system for parameter space does not influence the optimality of our estimating equations. As such, we are free to choose convenient systems of coordinates, whether they simplify the formulation of our models or of our estimating equations.

Theorem 6.2.8. *The information criterion is an invariant.*

Proof. As before, we have:

$$\mathscr{E}(\mathbf{G}^*)_{\mathscr{I}\mathscr{J}} = \varsigma_{\mathscr{K}\mathscr{I}} \tau^{\mathscr{K}\mathscr{L}} \varsigma_{\mathscr{L}\mathscr{J}} \tag{6.26}$$

Using the expressions given in (6.19) and (6.20), we obtain:

$$\begin{aligned}\mathscr{E}(\mathbf{G}^*)_{\mathscr{I}\mathscr{J}} &= \gamma_{\mathscr{K}\mathscr{M}} \gamma_{\mathscr{I}\mathscr{N}} \sigma^{\mathscr{M}\mathscr{N}} \gamma^{\mathscr{K}\mathscr{O}} \gamma^{\mathscr{L}\mathscr{P}} \sigma_{\mathscr{O}\mathscr{P}} \gamma_{\mathscr{L}\mathscr{Q}} \gamma_{\mathscr{J}\mathscr{R}} \sigma^{\mathscr{Q}\mathscr{R}} \\ &= \gamma_{\mathscr{K}\mathscr{M}} \gamma^{\mathscr{K}\mathscr{O}} \gamma_{\mathscr{I}\mathscr{N}} \gamma^{\mathscr{K}\mathscr{O}} \gamma_{\mathscr{L}\mathscr{Q}} \gamma^{\mathscr{L}\mathscr{P}} \sigma_{\mathscr{O}\mathscr{P}} \sigma^{\mathscr{Q}\mathscr{R}}\end{aligned}$$

$$\mathscr{E}(\mathbf{G}^*)_{\mathscr{I}\mathscr{J}} = \gamma_{\mathscr{I}\mathscr{K}} \gamma_{\mathscr{J}\mathscr{L}} \sigma^{\mathscr{K}\mathscr{L}} \tag{6.27}$$

which is precisely the expression for $\mathscr{E}(\mathbf{G})_{\mathscr{I}\mathscr{J}}$, so the information criterion remains invariant under change of coordinates. □

6.2.5 Asymptotic Consistency

In order to establish consistency of tensor estimating equations, we will begin by using the uniform law of large numbers to establish that there is a continuous function $\mathbb{E}(g(X,\theta))$ to which the sample mean of the estimating function $g(x,\theta)$ converges almost surely. First, we state the uniform law of large numbers for convenience.

Theorem 6.2.9 (Uniform Law of Large Numbers)**.** *Given*

1. *Θ is compact*
2. *The potential function $g(x,\theta)$ is continuous for each $\theta \in \Theta$ for almost all x*
3. *$g(x,\theta)$ is a measurable function of x for each θ*
4. *There is a dominating function $d(x)$ satisfying*
   a) $\mathbb{E}(d(x)) < \infty$
   b) $|g(x,\theta)| \leq d(x)\ \forall \theta \in \Theta$

then $\mathbb{E}(g(X,\theta))$ is continuous in θ and

$$\sup_{\theta\in\Theta}\left|\frac{1}{T}\sum_{i=1}^{T} g(X_i,\theta) - \mathbb{E}(g(X,\theta))\right| \xrightarrow{a.s.} 0$$

We now give a theorem using the uniform law of large numbers which establishes, under some conditions, that tensor estimating equations are consistent. At a high level, we establish that they are well-behaved extremum estimators, which Wald (1949) proved are consistent.

Theorem 6.2.10. *Under the following assumptions, the tensor estimating equation estimate is consistent. We shall use θ_0 to denote the true parameter value.*

A.1 $\mathbb{E}_\theta(|g(x,\theta_0)|) < \infty$

A.2 $\mathbb{E}_\theta(g(x,\theta)) < \infty$ *for all* $\theta \in \Theta$

A.3 *There exists a score function U for the underlying distribution of the data, generated by log-likelihood function ℓ*

A.4 *The argument maxima of ℓ and g coincide (i.e. both are maximised at θ_0)*

A.5 *The parameter space Θ is a compact subset of $\mathbb{R}^{p_1\times\ldots\times p_D}$*

A.6 *$g(x,\theta)$ is continuous in θ over the parameter space for almost all values of x*

A.7 *$g(x,\theta)$ is a measurable function of x for all $\theta \in \Theta$*

Proof. We now need to establish a dominating function $d(x)$, as required for the Uniform Law of Large Numbers. Our candidate will be the function in **A.1**, i.e. we define

$$d(x) \coloneqq \mathbb{E}_\theta(g(x,\theta_0))$$

We seek to establish that

$$\mathbb{E}_\theta(g(X,\theta)) < \mathbb{E}_\theta(g(X,\theta_0)) \quad \forall \theta \neq \theta_0$$

Now, if $\mathbb{E}_\theta(g(X,\theta)) = -\infty$ then this holds trivially, so we will consider only the case when $\mathbb{E}_\theta(g(X,\theta)) > -\infty$. Then by **A.2** we have that

$$\mathbb{E}_\theta(|g(X,\theta)|) < \infty$$

Now consider the random variable $m := g(X,\theta) - g(X,\theta_0)$. We can write $g = \ell + f$, thus we have

$$m(X) = \ell(X,\theta) - \ell(X,\theta_0) - (f(X,\theta_0) - f(X,\theta))$$

We now use the gradient theorem to write

$$f(X,\theta_0) - f(X,\theta) = \int_{\gamma[\theta,\theta_0]} \nabla_\theta f_{\mathscr{I}}(X,r) dr_{\mathscr{I}}$$

where $\gamma[\theta,\theta_0]$ is any path through the parameter space Θ from θ to θ_0. Now write $f = g - \ell$, so we have

$$f(X,\theta_0) - f(X,\theta) = \int_{\gamma[\theta,\theta_0]} \nabla_\theta (g-\ell)_{\mathscr{I}}(X,r) dr_{\mathscr{I}} = \int_{\gamma[\theta,\theta_0]} (G-U)_{\mathscr{I}}(X,r) dr_{\mathscr{I}}$$

Then we consider the expectation

$$\begin{aligned}
\mathbb{E}(f(X,\theta_0) - f(X,\theta)) &= \mathbb{E}\left(\int_{\gamma[\theta,\theta_0]} (G-U)_{\mathscr{I}}(X,r) dr_{\mathscr{I}} \right) \\
&= \int_{\gamma[\theta,\theta_0]} \mathbb{E}((G-U)(X,r))_{\mathscr{I}}\, dr_{\mathscr{I}} \\
&= \int_{\gamma[\theta,\theta_0]} 0_{\mathscr{I}} dr_{\mathscr{I}} \\
&= 0
\end{aligned}$$

Thus we have that

$$\mathbb{E}(m(X)) = \mathbb{E}(\ell(X,\theta) - \ell(X,\theta_0))$$

From Wald (1949), then, we have that $\mathbb{E}(m(X)) < 0$, i.e. that $d(X)$ is the dominating function we require. Thus, by the uniform law of large numbers, we have the result

$$\sup_{\theta\in\Theta} \left| \frac{1}{T} \sum_{i=1}^{T} g(X_i,\theta) - \mathbb{E}(g(X,\theta)) \right| \xrightarrow{a.s.} 0$$

Then, as an extremum estimator, the estimating function consistently estimates its maximum θ_0. □

6.3 Categorising Generalisations of PCA

Having developed the theory of tensor estimating equations, we will now apply it to the methods of Landgraf and Lee (2015b), Collins et al. (2002) and Li and Tao (2013).

6.3.1 Generalised Principal Component Analysis

We now return to the Generalised Principal Component Analysis of Landgraf and Lee (2015b), as detailed in Section 4.2.

As we have a full expression for the likelihood of $\mathbf{U}$ given data $\mathbf{X}$ which is differentiable, we can construct the score function. For convenience, we write it as two separate estimating functions.

$$\mathbf{G}_{\mathbf{U}} = \left(\left(\mathbf{X} - b'\left(\hat{\mathbf{\Theta}}\right)\right)^{\mathrm{T}}\left(\tilde{\mathbf{\Theta}} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right) + \left(\tilde{\mathbf{\Theta}} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X} - b'\left(\hat{\mathbf{\Theta}}\right)\right)\right)\mathbf{U} \tag{6.28}$$

$$\mathbf{G}_{\mu} = \left(\mathbf{I} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\right)\left(\mathbf{X} - b'\left(\hat{\mathbf{\Theta}}\right)\right)^{\mathrm{T}}\mathbf{1} \tag{6.29}$$

where $\tilde{\mathbf{\Theta}}$ is the matrix of saturated natural parameters, $\hat{\mathbf{\Theta}}$ is given by $\mathbf{1}\boldsymbol{\mu}^{\mathrm{T}} + \mathbf{U}\mathbf{U}^{\mathrm{T}}\left(\tilde{\mathbf{\Theta}} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right)$ and b' is the inverse of the canonical link function.

As a score function, by Theorem 6.2.2 we know that this is an optimal estimating function if we choose $\mathscr{H}$ to include it. However, Generalised Principal Component Analysis (GPCA) requires that $\mathbf{U}$ satisfy the left semi-orthonormality constraint; as such we must add a non-linear constraint to be enforced. We denote the constraint by $\phi(\mathbf{U}) \coloneqq \mathbf{U}^{\mathrm{T}}\mathbf{U}$ and require that our solution to (6.28) and (6.29) satisfies $\phi(\mathbf{U}) = \mathbf{I}$. For theoretical simplicity, we can now write this as a third estimating function

$$\mathbf{G}_{\text{orth.}} = \mathbf{U}^{\mathrm{T}}\mathbf{U} - \mathbf{I} \tag{6.30}$$

We can also think of this constraint as a change of space $\mathscr{H}$ on which we seek the optimal estimating function. In this case, the estimating functions (6.28) and (6.29) will again be optimal.

6.3.2 Collins et al.

This method, though typically formulated in terms of Bregmann distances, is ultimately based on an exponential family distribution, so we can be confident that it will satisfy the requirements of a tensor estimating equation. We now give the estimating functions.

$$\mathbf{G}_{\mathbf{A}} = \left(G'(\mathbf{A}\mathbf{V}) - \mathbf{X}\right)\mathbf{V}^{\mathrm{T}} \tag{6.31}$$

$$\mathbf{G}_{\mathbf{V}} = \mathbf{A}^{\mathrm{T}}\left(G'(\mathbf{A}\mathbf{V}) - \mathbf{X}\right) \tag{6.32}$$

From the estimating equations, we can see that, heuristically, we are attempting to make the difference between the observed data and the lower-dimensional approximation of it equal to zero, with some appropriate waiting (the $\mathbf{V}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}$ terms respectively) in each of $\mathbf{G}_{\mathbf{A}}$ and $\mathbf{G}_{\mathbf{V}}$.

6.3.3 Simple Poisson Principal Component Analysis

Due to the complexity introduced by the unspecified functions involved in the general Simple Exponential PCA of Li and Tao (2013), we will give as an example the estimating functions of the Poisson case:

$$\mathbf{G}_{\mathbf{W}} = \mathbf{X}\mathbf{Y}^{\mathrm{T}} - \exp(\mathbf{W}\mathbf{Y})\,\mathbf{Y}^{\mathrm{T}} - \mathbf{W}\mathrm{Diag}(\boldsymbol{\alpha}) \tag{6.33}$$

$$\mathbf{G}_{\mathbf{Y}} = \mathbf{W}^{\mathrm{T}}\mathbf{X} - \mathbf{W}^{\mathrm{T}}\exp(\mathbf{W}\mathbf{Y}) - \mathbf{Y} \tag{6.34}$$

$$\mathbf{G}_{\alpha j} = p - \|\mathbf{w}_j\|^2 \boldsymbol{\alpha}_j \tag{6.35}$$

Here, the exponential function is applied component-wise, and we use the convention of Li and Tao (2013) and Smallman, Underwood, et al. (2019) for the arrangement of the data matrix $\mathbf{X}$ as n columns of observations. Note that (6.35) is derived from the approximation of $\boldsymbol{\alpha}$ proposed by Li and Tao, rather than the conditional posterior likelihood.

6.4 Comparisons

In estimating equation form, it is now easy to compare how these three methods different. Let us start by comparing GPCA and the method of Collins et al. (2002). If we assume $\boldsymbol{\mu} = 0$ in (6.28), then we have (with a change of sign)

$$\mathbf{G}_{\mathbf{U}} = \left(b'\left(\mathbf{U}\mathbf{U}^{\mathrm{T}}\tilde{\boldsymbol{\Theta}}\right) - \mathbf{X}\right)^{\mathrm{T}}\tilde{\boldsymbol{\Theta}}\mathbf{U} + \tilde{\boldsymbol{\Theta}}^{\mathrm{T}}\left(b'\left(\mathbf{U}\mathbf{U}^{\mathrm{T}}\tilde{\boldsymbol{\Theta}}\right) - \mathbf{X}\right)\mathbf{U}$$

Comparing this to (6.31) and (6.32), we then see that this estimating equation also considers some (differently weighted) difference between the lower-dimensional representation and the observations. The difference, then, is in the weighting and the use of $\boldsymbol{\mu}$ as a centring parameter (which requires the inclusion of another estimating equation).

To compare Simple Exponential Principal Component Analysis (SePCA) with these methods, let us make some changes of notation. We rewrite $\exp(\mathbf{W}\mathbf{Y})$ as $\tilde{\mathbf{X}}$, recalling that exp maps the natural parameters $\mathbf{W}\mathbf{Y}$ to the expected value of the distribution, i.e.

our best guess for $\mathbf{X}$ without further information. Then we have

$$\begin{aligned}
\mathbf{G}_{\mathbf{W}} &= \mathbf{X}\mathbf{Y}^{\mathrm{T}} - \tilde{\mathbf{X}}\mathbf{Y}^{\mathrm{T}} - \mathbf{W}\mathrm{Diag}(\boldsymbol{\alpha}) &&= \left(\mathbf{X} - \tilde{\mathbf{X}}\right)\mathbf{Y}^{\mathrm{T}} - \mathbf{W}\,\mathrm{Diag}(\boldsymbol{\alpha}) \\
\mathbf{G}_{\mathbf{Y}} &= \mathbf{W}^{\mathrm{T}}\mathbf{X} - \mathbf{W}^{\mathrm{T}}\tilde{\mathbf{X}} - \mathbf{Y} &&= \mathbf{W}^{\mathrm{T}}\left(\mathbf{X} - \tilde{\mathbf{X}}\right) - \mathbf{Y} \\
\mathbf{G}_{\alpha j} &= p - \|\mathbf{w}_j\|^2 \boldsymbol{\alpha}_j
\end{aligned}$$

In this form, we can see how SePCA relates to GPCA and Collins et al. (2002); once again we are looking at weighted differences between the estimated and observed data. In fact, if it were not for the $-\mathbf{W}\,\mathrm{Diag}(\boldsymbol{\alpha})$) and $-\mathbf{Y}$ terms in the first two equations, and the existence of the third equation, this would be precisely the method of Collins et al. (2002).

6.5 Asymptotic Consistency

None of the methods considered in this chapter have previously had any asymptotic results published in the literature, which is a significant disadvantage to the justification of their use.

Recall Theorem 6.2.10. Given that assumptions A.1 through A.7 hold, we can establish the asymptotic consistency of each of the three methods. It is easy to see that all but A.4 hold for each of GPCA, SePCA and Collins et al. (2002). In general, A.4 will likely have to be assumed, as it cannot be easily verified. However, in these cases, all three methods can be readily seen to be equivalent to maximum likelihood, so A.4 will also hold. Precise details can be found in the respective papers, but briefly: GPCA is estimated by minimum deviance which is equivalent to maximum likelihood, SePCA is estimated by maximum a-posteriori (MAP) which is equivalent to maximum likelihood for the posterior likelihood, and Collins et al. (2002) is derived from maximum likelihood via Bregman divergences.

For illustration, we will briefly investigate each of these assumptions for the Poisson case of GPCA. Recall that here, the potential function is the deviance of a Poisson model.

A.1 Write the deviance as

$$\log\left(\frac{1}{x!}\right) + x\theta_s - \exp\theta_s - \log(1x!) - x\theta_0 - \exp\theta_0$$

where θ_s is the saturated natural parameters. This simplifies to

$$x(\theta_s - \theta_0) + \exp\theta_s - \exp\theta_0$$

which is almost-everywhere finite for finite values of θ_s and θ_0. Thus its absolute value is also almost everywhere finite, and it follows that so is the expectation

thereof. Of course, this is for the one-dimensional case, but the true vector case follows along the same lines with more complicated notation

A.2 The first term of the deviance is the log-likelihood under the saturated model, which is finite and has finite expectation, so this assumption is equivalent to verifying that

$$\mathbb{E}\left(-\log\left(\mathbb{P}\left(x,\theta\right)\right)\right)$$

is finite. In the Poisson case, this is

$$-\mathbb{E}\left(\log\left(\frac{1}{x!}\right)+x\theta-\exp\theta\right)$$

thus, as the expectation of an almost-everywhere finite quantity, it is also finite. Again, this is the one-dimensional case, but the true case follows simply from this

A.3 Trivially true by assumption

A.4 As we discussed, because the minimum deviance estimator is identical to the maximum likelihood estimator, we can be assured that this holds

A.5 As the parameter space is simply $\mathbb{R}^{p_1\times\ldots\times p_D}$, this is trivially true

A.6 The deviance function is everywhere continuous in θ for all values of x (which are restricted to vectors of non-negative integers)

A.7 Satisfied

6.6 Conclusions

In this chapter, we have seen how the framework of (tensor) estimating equations enables specifications of dimension reduction methods in a way which allows both for universal computational algorithms (via standard root-finding algorithms) and direct comparison between methods.

In particular, we have stated three important methods of exponential family dimension reduction in estimating equation form, and used that form to see how fundamentally similar these methods are. At their core, all of them are concerned by the difference between the observed and expected data, differing primarily in how they are weighted. This does, of course, provide a framework for exploring new dimension reduction techniques by modifying the estimating equations.

We have also used the asymptotic results for tensor estimating equations to show consistency of each of the three studied methods.

Any important future direction for this work is the ability to incorporate sparsifying penalties. This will allow almost all of the exponential family dimension reduction methods to be analysed under the same framework. Currently, the obstacle for this is the use of penalties which are not everywhere differentiable, such as the L^1 and SCAD penalties. This could, perhaps, be done by using differentiable approximations to these penalties, but it is not yet clear how to treat these in the asymptotic framework.

— Chapter 7 —

Conclusions

In this work, we have given three novel methods for exponential family dimension reduction. Firstly, we derived Poisson Inverse Regression (PoIR) in Chapter 2, which is specialised to the Poisson distribution for its use with text data. We gave two methods of estimation, using either maximum a-posteriori (MAP) or Markov Chain Monte Carlo (MCMC), and demonstrated its efficacy using real text data against its closest competitor, Multinomial Inverse Regression (MNIR).

We then extended the work of Landgraf and Lee (2015b) (known as Generalised Principal Component Analysis (GPCA)) in Chapter 4, incorporating a sparsifying penalty which improves performance and intepretability of the resulting loadings. Using both simulation studies and a real text data example, we showed that our proposed method performs on par or better than the current state of the art.

In Chapter 5, we extended Simple Exponential Principal Component Analysis (SePCA) (Li and Tao 2013) in a similar fashion, applying a sparsifying penalty to the loadings matrix. Again, we demonstrated the advantages of our new method with simulation and real data studies. In particular, we showed that our method performed better at estimating the underlying structural dimension of the data than the original method.

Taking a new approach, in Chapter 6 we presented a framework for tensor-valued estimating equations. We showed several important results, including that tensor estimating equations are asymptotically consistent. The development of this framework then allowed us to re-examine three important methods for exponential family dimension reduction in a new light in. We used the framework to show how essentially similar these methods are, and precisely how they differ. We also applied the asymptotic theory from tensor estimating equations to show that each of these methods is consistent.

Bibliography

Bishop, Christopher M (1999). “Bayesian pca”. In: *Advances in neural information processing systems*, pp. 382–388.

Blei, David M, Andrew Y Ng, and Michael I Jordan (2003). “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3, pp. 993–1022. arXiv: 1111.6189v1.

Collins, Michael, S Dasgupta, and Robert E Schapire (2002). “A Generalization of Principal Components Analysis to the Exponential Family”. In: *Advances in Neural Information Processing Systems 14*, pp. 617–624.

Cook, R. Dennis (2007). “Fisher Lecture: Dimension Reduction in Regression”. In: *Statistical Science* 22.1, pp. 1–26. arXiv: 0708.3774.

Eaton, Morris L (1986). “A characterization of spherical distributions”. In: *Journal of Multivariate Analysis* 20.2, pp. 272–276.

Fan, Jianqing and Runze Li (2001). “Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties”. In: *Journal of the American Statistical Association* 96.456, pp. 1348–1360.

Frommlet, Florian and Grégory Nuel (2016). “An adaptive ridge procedure for L0 regularization”. In: *PLoS ONE* 11.2, pp. 1–23.

Gelman, A and D B Rubin (1992). “A Single Series from the Gibbs Sampler Provides a False Sense of Security”. In: *Bayesian Statistics* 4.July, pp. 625–631.

Geman, Stuart and Donald Geman (1984). “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6, pp. 721–741.

Gillis, Nicolas (2014). “The Why and How of Nonnegative Matrix Factorization”. In: pp. 1–25. arXiv: 1401.5226.

Guan, Y and Jg Dy (2009). “Sparse probabilistic principal component analysis”. In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 185–192.

Hastings, W Keith (1970). “Monte Carlo sampling methods using Markov chains and their applications”. In:

Hotelling, Harold (1933). “Analysis of a complex of statistical variables into principal components.” In: *Journal of Educational Psychology* 24.6, pp. 417–441.

Hu, Zhenfang et al. (2016). “Sparse Principal Component Analysis via Rotation and Truncation”. In: *IEEE Transactions on Neural Networks and Learning Systems* 27.4, pp. 875–890. arXiv: arXiv:1403.1430v2.

Hunter, David R and Runze Li (2005). “Variable selection using MM algorithms”. In: *Annals of Statistics* 33.4, pp. 1617–1642. arXiv: 0508278 [math].

Kwak, Nojun (2008). “Principal component analysis based on L1-norm maximization”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.9, pp. 1672–1680.

Landgraf, Andrew J and Yoonkyung Lee (Oct. 2015a). “Dimensionality Reduction for Binary Data through the Projection of Natural Parameters”. In: *arXiv:1510.06112 [stat.ML]*. arXiv: 1510.06112.

— (2015b). “Generalized Principal Component Analysis: Projection of Saturated Model Parameters”. In: *Ohio State University Statistics Department Technical Report* 892.892.

Li, Jun and Dacheng Tao (2013). “Simple exponential family PCA”. In: *IEEE Transactions on Neural Networks and Learning Systems* 24.3, pp. 485–497.

Li, Ker-Chau (1991). “Sliced inverse regression for dimension reduction”. In: *Journal of the American Statistical Association* 86.414, pp. 316–327.

Lu, Meng, Jianhua Z. Huang, and Xiaoning Qian (Dec. 2016). “Sparse exponential family Principal Component Analysis”. In: *Pattern Recognition* 60, pp. 681–691.

Mackay, David J C (1995). “Probable networks and plausible predictionsa review of practical Bayesian methods for supervised neural networks”. In: *Network: Computation in Neural Systems* 6.3, pp. 469–505.

Metropolis, Nicholas et al. (1953). “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6, pp. 1087–1092.

Mohamed, Shakir, Katherine Heller, and Zoubin Ghahramani (2009). “Bayesian exponential family PCA”. In: *Advances in Neural Information Processing Systems*, pp. 1089–1096.

Nocedal, J and S J Wright (1999). *Numerical Optimization*. Vol. 43. 2, pp. 164–75. arXiv: NIHMS150003.

Pearson, Karl (1901). “On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.1, pp. 559–572.

R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0 (2011). “R Development Core Team”. In: *R: A Language and Environment for Statistical Computing* 55, pp. 275–286.

Rousseeuw, Peter J. (Nov. 1987). “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. arXiv: z0024.

Smallman, Luke, Andreas Artemiou, and Jennifer Morgan (2018). “Sparse generalised principal component analysis”. In: *Pattern Recognition* 83, pp. 443–455.

Smallman, Luke, William Underwood, and Andreas Artemiou (2019). “Simple Poisson PCA: an algorithm for (sparse) feature extraction with simultaneous dimension determination”. In: *Computational Statistics*, pp. 1–19.

Taddy, Matt (2013). “Multinomial Inverse Regression for Text Analysis”. In: *Journal of the American Statistical Association* 108.503, pp. 755–770. arXiv: 1012.2098.

— (Sept. 2015). “Distributed multinomial regression”. In: *The Annals of Applied Statistics* 9.3, pp. 1394–1414.

Tipping, Michael E. and Christopher M. Bishop (1999). “Probabilistic Principal Component Analysis”. In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 61.3, pp. 611–622.

Wald, Abraham (1949). “Note on the consistency of the maximum likelihood estimate”. In: *The Annals of Mathematical Statistics* 20.4, pp. 595–601.

Wen, Zaiwen and Wotao Yin (Dec. 2013). “A feasible method for optimization with orthogonality constraints”. In: *Mathematical Programming* 142.1-2, pp. 397–434.

Yi, Shuangyan et al. (Jan. 2017). “Joint sparse principal component analysis”. In: *Pattern Recognition* 61, pp. 524–536.

Zou, Hui and Trevor Hastie (2005). “Regularization and Variable Selection via the Elastic Net”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 67.2, pp. 301–320.

Zou, Hui, Trevor Hastie, and Robert Tibshirani (June 2006). “Sparse Principal Component Analysis”. In: *Journal of Computational and Graphical Statistics* 15.2, pp. 265–286. eprint: 1205.0121v2.