Computational Statistics
https://doi.org/10.1007/s00180-019-00903-0

ORIGINAL PAPER

Simple Poisson PCA: an algorithm for (sparse) feature extraction with simultaneous dimension determination

Luke Smallman[1] · **William Underwood**[2] · **Andreas Artemiou**[1]

Received: 11 July 2018 / Accepted: 6 June 2019

Abstract
Dimension reduction tools offer a popular approach to analysis of high-dimensional big data. In this paper, we propose an algorithm for sparse Principal Component Analysis for non-Gaussian data. Since our interest for the algorithm stems from applications in text data analysis we focus on the Poisson distribution which has been used extensively in analysing text data. In addition to sparsity our algorithm is able to effectively determine the desired number of principal components in the model (order determination). The good performance of our proposal is demonstrated with both synthetic and real data examples.

Keywords L0 penalty · Exponential family · Text data analysis · Dimension reduction

1 Introduction

Principal Component Analysis (PCA), and its variants, are popular and well-established methods for unsupervised dimension reduction through feature extraction. These methods attempt to construct low-dimensional representations of a dataset which minimise the reconstruction error for the data, or for its associated parameters. An attractive property of PCA is that it produces a linear transformation of the data which allows reconstructions to be calculated using only matrix multiplication. It was developed initially for Gaussian-distributed data, and has since been extended to include other distributions, as well as extended for use in a Bayesian framework. Examples of such extensions include Sparse PCA (SPCA) (Zou et al. 2006), Probabilistic PCA (PPCA) (Tipping and Bishop 1999), Bayesian PCA (BPCA) (Bishop 1999), Multinomial PCA (Buntine 2002), Bayesian Exponential Family PCA (Mohamed

✉ Luke Smallman
smallmanl@cardiff.ac.uk

[1] School of Mathematics, Cardiff University, Senghennydd Road, Cardiff CF24 4AG, UK

[2] Mathematical Institute, University of Oxford, Woodstock Road, Oxford OX2 6CG, UK

Published online: 11 June 2019

et al. 2009), Simple Exponential Family PCA (SePCA) (Li and Tao 2013), Generalised PCA (GPCA) (Landgraf and Lee 2015) and Sparse Generalised PCA (SGPCA) (Smallman et al. 2018). The wide variety of PCA extensions can be attributed to a combination of the very widespread use of PCA in high-dimensional applications and to its insuitability for problems not well-approximated by the Gaussian distribution.

In this paper, we present a method for Simple Poisson PCA (SPPCA) based on the SePCA (Li and Tao 2013), an extension of PCA which (through the prescription of a simple probabilistic model) can be applied to data from any exponential family distribution. Firstly, by focusing on the Poisson distribution we provide an alternative inferential algorithm to the original paper which is simpler and makes use of gradient-based methods of optimisation. This simple algorithm then facilitates our second extension: the application of a sparsity-inducing adaptive L^0 penalty to the loadings matrix which improves the ability of the algorithm when dealing with high-dimensional data with uninformative components. We call our sparse extension Sparse Simple Poisson PCA (SSPPCA). We note that similarly one can focus on any other exponential family distribution and create the respective gradient-based optimisation for that distribution. We discuss only the Poisson distribution in this work due to the fact that we focus on text data which are usually modelled using a Poisson distribution. We also note that in the simulation studies and real-world example which we will investigate in this work, we do not use particularly high-dimensional data. While our method should work the same with such data, the necessary computational adaptations to work with very high-dimensional data are beyond the scope of this paper.

A common feature of high-dimensional data is the presence of uninformative dimensions. In text data, observed data is often represented by a document-term matrix $\boldsymbol{X}$ whose ijth entry is the number of times the ith document contains the jth term. In practice, a considerable number of these terms are irrelevant to classification, clustering or other analysis. Such examples have lead to the development of many methods for sparse dimension reduction, such as Sparse Principal Component Analysis (Zou et al. 2006), Joint Sparse Principal Component Analysis (Yi et al. 2017), Sparse Generalised Principal Component Analysis (Smallman et al. 2018) and more. In this paper, our sparsifying procedure for SPPCA uses the adaptive L^0 penalty to induce sparsity. We will present the method in general terms for any exponential family distribution and illustrate with a case study using the Poisson distribution and text data. We stress that although this method does not give a classification algorithm, dimension reduction methods are often used as a preprocessing step before classification or clustering techniques; as such these tasks provide both important applications and methods of validation for this work. As such, we will spend some time investigating the ability of our proposed methods to provide dimension reduction transformations which leave the data amenable to such applications.

One difficulty associated with PCA algorithms is the need to choose the desired number of principal components, a process known generally in dimension reduction frameworks as *order determination*. This is in practice a difficult task, particularly with data involving large numbers of features, and often necessitates multiple experiments to determine the best number. For suitable models, Automatic Relevance Determination (ARD) (Mackay 1995), provides an automatic method for order determination; like SePCA, SPPCA is able to make use of ARD, as is our sparse adaptation. We will investigate the behaviour of this order determination for both the original SPPCA and for Sprse SPPCA.

In Sect. 2 we discuss the exponential family of distributions and the previous work on SePCA, as well as outline the process of Automatic Relevance Determination. Section 2.3 defines Simple Poisson Principal Component Analysis (SPPCA), with details of the techniques involved for numerical computation, and reconstruction of the data. Section 3 introduces sparsity, giving Sparse Simple Poisson Principal Component Analysis (SSPPCA). We give a detailed estimation procedure for both the SPPCA and SSPPCA in Sect. 4.1. We investigate numerical performance in Sect. 5, focusing on artificial data sets in Sects. 5.1 and 5.3, the performance of the order determination via ARD in Sect. 5.2, and a healthcare dataset in Sect. 5.4. Finally, we will discuss implications and plans for future work in Sect. 6. To improve readability, certain results and derivations are included as appendices to the text.

2 SePCA

Since our work is based on Simple Exponential PCA by Li and Tao (2013), we will introduce in this section the general method for the exponential family of distributions. We first introduce the most frequently used notation from Li and Tao (2013). We let N be the sample size, D the number of features and d the number of principal components. Moreover, $\mathbf{X}$ is the $D \times N$ data matrix, $\mathbf{x}_n \in \mathbb{R}^D$ is the nth observation, $\mathbf{Y}$ is the $d \times N$ scores matrix and $\mathbf{y}_n \in \mathbb{R}^d$ is the score of the nth observation. Furthermore, $\mathbf{W}$ is a $D \times d$ loadings matrix, $\mathbf{w}_j \in \mathbb{R}^D$ is the loadings of the jth principal component, $\boldsymbol{\Theta}$ is the $D \times N$ parameter matrix and $\boldsymbol{\theta}_n \in \mathbb{R}^N$ is the parameter vector of the nth observation. Finally, we denote the joint posterior likelihood of $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\alpha}$ with $P(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\alpha})$.

Then we state the definition of the exponential family of distributions in terms of its conditional distribution to motivate our work.

Definition 1 The exponential family distribution has a probability function which is conditional on a single vector parameter $\boldsymbol{\theta}$, and takes the canonical form:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \exp\left[\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + g(\boldsymbol{\theta}) + h(\mathbf{x})\right]$$

where $g : \mathbb{R}^D \to \mathbb{R}$ and $h : \mathbb{R}^D \to \mathbb{R}$.

2.1 Model specification of SePCA

In SePCA (see Li and Tao 2013) proposed modelling the sampling process of $\boldsymbol{x}_n$ by the distribution $p(\boldsymbol{x}_n|\boldsymbol{W}, \boldsymbol{y}_n) = \mathrm{Exp}(\boldsymbol{x}_n|\boldsymbol{\theta}_n)$ where $\mathrm{Exp}(\boldsymbol{x}_n|\boldsymbol{\theta}_n)$ is the conditional distribution of the exponential family as defined above, and $\boldsymbol{\theta}_n = \boldsymbol{W}\boldsymbol{y}_n$ is the natural parameter vector for the exponential family distribution that generates $\boldsymbol{x}_n$. On $\boldsymbol{y}_n$ they place a Gaussian prior: $p(\boldsymbol{y}_n) = \mathcal{N}(\boldsymbol{y}_n|\mathbf{0}_d, \boldsymbol{I}_d)$ where $\mathbf{0}_d$ is the d-dimensional vector with all entries zero and $\boldsymbol{I}_d$ the $d \times d$ identity matrix. Finally, on each principal component $\boldsymbol{w}_i$, $i = 1, \ldots, d$ they place another Gaussian prior: $p(\boldsymbol{W}|\boldsymbol{\alpha}) = \prod_{j=1}^{d} \mathcal{N}(\boldsymbol{w}_j|\mathbf{0}_D, \alpha_j^{-1}\boldsymbol{I}_D)$ where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_d\}$ is a set of precision hyperparameters. This is what differentiates SePCA from previous methodology of exponential family distribution PCA. Bishop (1999) proposed a similar framework but assumed that $p(\boldsymbol{x}_n|\boldsymbol{W}, \boldsymbol{y}_n)$ is from a Gaussian distribution. Mohamed et al. (2009) used a fully probabilistic treatment where the mean and variance of the prior distribution of $\boldsymbol{W}$ are themselves probabilistic and not fixed, whereas Collins et al. (2002) gave a deterministic model for $\boldsymbol{W}$. In contrast, Landgraf and Lee (2015) [and thus Smallman et al. (2018)] proceed from a different starting point, instead likening the squared reconstruction error definition of PCA to the deviance function of a model with constrained natural parameters.

Li and Tao (2013) suggested to estimate $\boldsymbol{\alpha}$ by maximising the marginal likelihood $p(\boldsymbol{X}|\boldsymbol{\alpha})$ which results in:

$$\alpha_j^{\mathrm{M}} \approx \frac{D}{||\boldsymbol{w}_j^{\mathrm{MP}}||_2^2} \tag{1}$$

where $\boldsymbol{w}_j^{\mathrm{MP}}$ is the maximum a posteriori (MAP) estimate of $\boldsymbol{w}_j$. In essence this implies an iterative procedure as the posterior estimate of $\boldsymbol{W}$ depends on $\boldsymbol{\alpha}$ and vice versa. Here, the value of α_j, $j = 1, \ldots, d$, indicates whether a principal component $\boldsymbol{w}_j$ should be kept or ignored. This is done using Automatic Relevance Determination (ARD) which was introduced in Mackay (1995). If $\alpha_j > M$ where M is sufficiently large then we may infer that all components of $\boldsymbol{w}_j$ are within a small neighbourhood of 0 with high probability; thus we may safely discard them. In practice, we have usually found $M \approx 100$ to be sufficient.

2.2 Inference procedure on W and Y in SePCA

To make inference on $\boldsymbol{W}$ and $\boldsymbol{Y}$ we use the MAP estimation of the log posterior which has the form:

$$\begin{aligned}\log P(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\alpha}) &= \log p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{Y}, \boldsymbol{\alpha}) + \log p(\boldsymbol{Y}) + \log p(\boldsymbol{W}|\boldsymbol{\alpha}) + \text{constant}\\ &= \sum_{n=1}^{N} \left[\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{W} \boldsymbol{y}_n + g(\boldsymbol{W} \boldsymbol{y}_n)\right] - \frac{1}{2}\mathrm{tr}(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}) - \frac{1}{2}\mathrm{tr}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\mathrm{Diag}(\boldsymbol{\alpha}))\end{aligned}$$

where $\mathrm{Diag}(\boldsymbol{\alpha})$ is the $d \times d$ diagonal matrix with entries $\boldsymbol{\alpha}$. In Li and Tao (2013), the authors based their estimation procedure on the fact that the conditional distribution $p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{Y})$ is some general exponential family distribution. Therefore, they suggested to approximate the log-likelihood with a lower bound and adopt an expectation-maximisation (EM) approach for optimisation. This led to a rather complicated inference procedure on $\boldsymbol{W}$ and $\boldsymbol{Y}$.

In this paper we propose a different inference procedure on $\boldsymbol{W}$ and $\boldsymbol{Y}$. Although in Li and Tao (2013) the conditional distribution $p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{Y})$ is some general exponential family distribution, in specific problems the distribution is well defined, for example in their simulated data they used a binomial distribution and in the text data framework we are interested in this paper we use Poisson distribution. Therefore, we suggest that the estimation of $\boldsymbol{W}$ and $\boldsymbol{Y}$ is done using simple gradient based methods for optimisation to find the MAP estimates of $P(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\alpha})$. We will illustrate the details of our method by example in the next section, where we discuss Simple Poisson PCA (SPPCA).

2.3 Simple Poisson PCA (SPPCA)

As we said earlier in this paper, we are interested in a PCA algorithm which is appropriate for text data. Text data is usually transformed into numeric vectors where we measure the number of times a word appears in a document (or a sentence or a paragraph). The usual distribution to measure counts like this is a Poisson distribution and therefore here we present the special version of SePCA where the Poisson distribution is used instead of the general exponential distribution.

The Poisson distribution is a discrete probability distribution, and is a member of the exponential family distribution. It has probability mass function, conditional on λ,

$$p(x|\lambda) = \frac{\lambda^x \, \mathrm{e}^{-\lambda}}{x!}$$

The joint distribution of D independent Poisson variates, with means $\boldsymbol{\lambda} = (\lambda_1 \dots \lambda_D)$ is also a member of the exponential family distribution, with mass function

$$p(\boldsymbol{x}|\boldsymbol{\lambda}) = \prod_{i=1}^{D} p(x_i|\lambda_i) = \exp\left[\sum_{i=1}^{D} x_i \log(\lambda_i) - \sum_{i=1}^{D} \lambda_i - \sum_{i=1}^{D} \log(x_i!)\right]$$

This is in canonical form with $\theta_i = \log(\lambda_i)$, $g(\boldsymbol{\theta}) = -\sum_{i=1}^{D} \mathrm{e}^{\theta_i}$ and $h(\boldsymbol{x}) = -\sum_{i=1}^{D} \log(x_i!)$ where g and h are defined in Definition 1.

To run SPPCA, we need to define a number of distributions as was the case with SePCA. The prior distributions defined for SePCA are the same. Since the forms of g, h are determined though, the likelihood of $\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{Y}, \boldsymbol{\alpha}$ can be explicitly stated:

$$\begin{aligned} p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{Y}) &\propto \exp \sum_{n=1}^{N} \left[\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{W} \boldsymbol{y}_n + g(\boldsymbol{W} \boldsymbol{y}_n)\right] \\ &\propto \exp \sum_{n=1}^{N} \left[\boldsymbol{x}_n^{\mathrm{T}} \boldsymbol{W} \boldsymbol{y}_n - \sum_{i=1}^{D} \mathrm{e}^{(\boldsymbol{W} \boldsymbol{y}_n)_i}\right] \\ &\propto \exp\left[\mathrm{tr}(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{Y}) - \sum \sum \mathrm{e}^{\boldsymbol{W} \boldsymbol{Y}}\right] \end{aligned}$$

where $\sum\sum$ indicates the sum over all entries and $\mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}}$ is the *component-wise* exponential (and not the matrix exponential).

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

$$\begin{aligned} \log P(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}, \boldsymbol{\alpha}) &= \log p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{Y}) + \log p(\boldsymbol{Y}) + \log p(\boldsymbol{W}|\boldsymbol{\alpha}) + \mathrm{const} \\ &= \mathrm{tr}(\boldsymbol{X}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{Y}) - \sum\sum \mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}} - \frac{1}{2}\mathrm{tr}(\boldsymbol{Y}^{\mathrm{T}} \boldsymbol{Y}) - \frac{1}{2}\mathrm{tr}(\boldsymbol{W}^{\mathrm{T}} \boldsymbol{W} \mathrm{Diag}(\boldsymbol{\alpha})) \end{aligned}$$

Now the above can be used to directly make inference on $\boldsymbol{W}$ and $\boldsymbol{Y}$ using MAP estimates of the log-posterior. As was mentioned earlier, the fact that the exponential distribution is specified makes the estimation procedure much easier and there is no need to rely on an EM approach to make inference. For completeness we mention that for estimation of $\boldsymbol{\alpha}$ we use the equation used in SePCA as shown in Eq. (1).

The algorithm which we developed alternates between parameter estimation for $\boldsymbol{\alpha}$ and inference on $\boldsymbol{W}$, $\boldsymbol{Y}$. We will give more details on the estimation after we introduce the Sparse SPPCA as the estimation algorithms for the two are similar.

3 Sparse simple exponential PCA (SSePCA)

When we do feature extraction for dimension reduction the features are often a function of all the original variables in our model. In most cases though a lot of the coefficients for the original variables are close to zero and you expect that these variables are not significant in the feature construction and you would like to remove them by setting their coefficients to zero. Sparse PCA algorithms have been proposed over the years [such as Zou et al. (2006) and Yi et al. (2017)] to address sparsity in the classic PCA setting for Gaussian data. In the generalised setting where the data is not Gaussian there has been limited effort. To the best of our knowledge, only recently has there been interest in developing sparse algorithms for non-Gaussian PCA settings (Smallman et al. 2018). In Smallman et al. (2018) the authors propose the use of SCAD (Fan and Li 2001) and LASSO (Tibshirani 1996) penalties (or a combination of the two) to be applied to a generalised PCA algorithm proposed in Landgraf and Lee (2015). In this work, we propose an algorithm which has advantages over the work in Smallman et al. (2018). First, we use a penalty proposed in Frommlet and Nuel (2016) which allows for a simpler computational algorithm than the one proposed before. More importantly, this algorithm can automatically detect the working dimension d of the problem at the same time as estimating the principal components (as was the case with SePCA). To the best of our knowledge, this is the first sparse and non-Gaussian based PCA algorithm that simultaneously achieves this. In this section, we discuss how one can introduce sparsity to SePCA and then focus on introducing sparsity in the SPPCA framework.

3.1 The adaptive L^0 penalty

It is known in the literature that LASSO and SCAD penalties computationally complex problems and are computationally expensive in extracting sparse features. Therefore, to maintain the simplicity of our estimation algorithm and not add unnecessary complexity we propose the use of an iterative approximation to the L^0 norm penalty (see Frommlet and Nuel 2016) on $\boldsymbol{W}$:

$$||\boldsymbol{W}||_0 = \sum_{i=1}^{D}\sum_{j=1}^{d} \mathbb{1}(\boldsymbol{W}_{ij} \neq 0) \approx \sum_{i=1}^{D}\sum_{j=1}^{d} \frac{(\boldsymbol{W}_{ij})^2}{(\boldsymbol{W}^0_{ij})^2 + \delta}$$

where $\boldsymbol{W}^0$ is the previous value of $\boldsymbol{W}$, and $\delta > 0$ a very small value. The sparsity penalty is weighted by a constant k:

$$S = k \sum_{i=1}^{D} \sum_{j=1}^{d} \frac{(\boldsymbol{W}_{ij})^2}{(\boldsymbol{W}_{ij}^0)^2 + \delta}$$

Here we note that the optimal value of k is data specific and should be estimated using cross-validation while the value of δ does not affect the performance of the algorithm as was demonstrated by Frommlet and Nuel (2016) (as long as it is small compared to the entries of matrix $\boldsymbol{W}^0$).

Although Frommlet and Nuel (2016) suggested an iterative procedure which approximates the L^0 norm penalty, successively minimising a penalised objection function then recalculating the weights for that penalty, we will instead recalculate the weights within each penalised objective function minimisation. We will discuss the precise differences in Sect. 4.2.

3.2 Sparse simple Poisson PCA (SSPPCA)

As was mentioned before in this work we focus on using specifically the Poisson distribution as a result of its use in modelling text data. Therefore, we move one step further and model the Sparse Simple Poisson PCA to achieve sparsity under the assumption that the general exponential distribution is replaced with a Poisson distribution. The objective function which will be used for the inference on $\boldsymbol{W}$ and $\boldsymbol{Y}$ is the following:

$$P_p^s = \mathrm{tr}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{Y}) - \sum\sum \mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}} - \frac{1}{2}\mathrm{tr}(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}) - \frac{1}{2}\mathrm{tr}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W}\mathrm{Diag}(\boldsymbol{\alpha}))$$
$$-k \sum_{i=1}^{D} \sum_{j=1}^{d} \frac{(\boldsymbol{W}_{ij})^2}{\left(\boldsymbol{W}_{ij}^0\right)^2 + \delta}.$$

4 Estimation algorithm for SPPCA and SSPPCA

In this section we present the necessary steps we need to take so that we are ready to run our estimation algorithm for SPPCA. Then we discuss what changes in the SSPPCA algorithm. It is important to make clear here that this estimation algorithm will work if instead of Poisson distribution we use any other exponential family distribution.

4.1 Estimation of SPPCA

To run our estimation algorithms we need the derivatives of the objective function with respect to $\boldsymbol{W}$ and $\boldsymbol{Y}$ to aid with optimisation. Using matrix algebra gives the following:

$$\frac{\partial P}{\partial \boldsymbol{W}} = \boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}} - \mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}}\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{W}\mathrm{Diag}(\boldsymbol{\alpha})$$
$$\frac{\partial P}{\partial \boldsymbol{Y}} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{X} - \boldsymbol{W}^{\mathrm{T}}\mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}} - \boldsymbol{Y} \qquad (2)$$

where $\mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}}$ denotes component-wise operations. The element-wise derivatives from which these were derived are given in “Appendix A”.

Then we suggest using the following algorithm for SPPCA:

1. **Initialisation** $\boldsymbol{W}_{\text{Init}}$ and $\boldsymbol{Y}_{\text{Init}}$ are found by running the standard PCA algorithm (Pearson 1901) on the data, using `prcomp` in R, to find loadings and scores matrices respectively.
   $\boldsymbol{\alpha}_{\text{Init}}$ is initialised as a vector with each entry equal to 1.
   Finally we set $d = D - 1$.
2. **Optimisation** Optimisation of the objective function was performed by R’s `optim` function, using the quasi-Newton gradient-based Broyden–Fletcher–Goldfarb–Shanno (BFGS) method. The derivatives used are the ones in (2).
3. **Removal** Principal component j was removed if $\alpha_j \geq M$. To avoid removing components too early, this threshold was raised to a larger value for the first 10 iterations. To further stabilise this process, only one component was removed at a time. The PCs were reordered to have increasing α_j values. Removal of the PCs allowed for quicker optimisation times as the complexity of the problem was reduced, and also avoided issues with large α_j values dominating the objective function and causing difficulties with the optimiser.
4. **Convergence criteria**: The steps of **Optimisation** and **Removal** were repeated until the following two rules were both satisfied:

   (a) The test for convergence was satisfied, that is

   $$\left|\frac{P - P_0}{P_0}\right| < \epsilon$$

   where P_0 is the previous value of P and $\epsilon > 0$ is small.

   (b) There were no components removed during the current iteration. If a component was removed during an iteration we run at least one more iteration to ensure the latent dimension d had converged.

4.2 Estimation of SSPPCA

The algorithm is similar for the sparse version of the algorithm, namely the SSPPCA. The only things which change are the objective function P_p^s and its derivatives which take the form:

$$\frac{\partial P_p^s}{\partial \boldsymbol{W}} = \boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}} - \mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}}\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{W}\mathrm{Diag}(\boldsymbol{\alpha}) - 2k\frac{\boldsymbol{W}}{(\boldsymbol{W}^0)^2 + \delta}$$
$$\frac{\partial P_p^s}{\partial \boldsymbol{Y}} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{X} - \boldsymbol{W}^{\mathrm{T}}\mathrm{e}^{\boldsymbol{W}\boldsymbol{Y}} - \boldsymbol{Y}$$

where $(\mathbf{W}^0)^2$ and $\mathrm{e}^{\mathbf{WY}}$ are component-wise operations. As in the previous section the elementwise derivatives from which these were derived are relegated in Appendix B. The rest of the steps are similar to the algorithm for SPPCA with the only difference being the need to define δ which is a tuning parameter for the adaptive L^0 norm penalty we are using to induce sparsity.

It is very important to clarify here that the gradient $\frac{\partial P_p^s}{\partial \boldsymbol{W}}$ only exists because we are approximating the L^0 norm on $\boldsymbol{W}$ by a differentiable function. This means that if other penalties, e.g. LASSO or SCAD, were to be used the estimation algorithm would have been computationally more complex.

Finally, we make a note that we pass this penalty into R's `optim` function on each iteration so that we provide a unified framework of sparse and non-sparse feature extraction. One can achieve sparsity in a different way which resembles the idea of Frommlet and Nuel (2016) more accurately. Although this is closer to the idea presented by Frommlet and Nuel (2016) it does not allow us to use the simple computational algorithm for sparse feature extraction. In simulation studies not presented here, we found that implementing the L^0 penalty as Frommlet and Nuel (2016) suggest provides a statistically insignificant gain of approximately 1% in average Euclidean silhouette on classed data over our combined method. We deemed that this did not merit the more computationally intensive implementation or the de-unification of the sparse and non-sparse algorithms.

5 Numerical studies

In this section, we will investigate the performance of SPPCA and SSPPCA and compare their performances against those of PCA, SPCA (Zou et al. 2006), GPCA (Landgraf and Lee 2015) and SGPCA (Smallman et al. 2018). We compare with PCA to demonstrate that an algorithm based on Gaussian data is not going to work as well in this setting, and with SPCA to show that even adding sparsity will not counteract this problem. We also compare with GPCA which is another exponential family PCA algorithm and with SGPCA which (to the best of our knowledge) is the only other sparse PCA algorithm for exponential family distributions. In Sect. 5.1 we will work with synthetic data drawn from a Poisson hidden-factor model. This model will then be extended to a two-class hidden-factor model in Sect. 5.3. Finally, we will investigate a real-world healthcare dataset in Sect. 5.4.

5.1 Synthetic data

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

$$V_1 \sim \text{Poisson}(20) \qquad V_2 \sim \text{Poisson}(30) \qquad V_3 \sim \text{Poisson}(50) \tag{3}$$

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. We specify the following three models, which will be used extensively.

$$[(v_{1i} + \varepsilon_{i1}), (v_{1i} + \varepsilon_{i2}), (2v_{1i} + \varepsilon_{i3}), \ldots, (2v_{1i} + \varepsilon_{i10})]^{\mathrm{T}} \tag{4}$$

$$[(v_{1i}+\varepsilon_{i1}),(v_{1i}+\varepsilon_{i2}),(v_{2i}+\varepsilon_{i3}),(v_{2i}+\varepsilon_{i4}),(v_{1i}+3v_{2i}+\varepsilon_{i5})\ldots,\\ (v_{1i}+3v_{2i}+\varepsilon_{i10})]^{\mathrm{T}} \tag{5}$$

$$[(v_{1i}+\varepsilon_{i1}),(v_{1i}+\varepsilon_{i2}),(v_{2i}+\varepsilon_{i3}),(v_{2i}+\varepsilon_{i4}),(v_{3i}+\varepsilon_{i5}),\\ (v_{3i}+\varepsilon_{i6}),(3v_{1i}+2v_{2i}+2v_{3i}+\varepsilon_{i7}),\ldots,\\ (3v_{1i}+2v_{2i}+2v_{3i}\varepsilon_{iD})]^{\mathrm{T}} \tag{6}$$

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 Sect. 4.1 we do this to avoid removing components too early). Also for the 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 ith 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 given in (4). 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 for variables 3–10 and it has smaller coefficients for variables 1 and 2.

Similarly, the ith 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, as given in (5).

To both of these datasets we applied each of SPPCA, SSPPCA, PCA, 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 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. Out of these four, PCA has arguably the best performance, with the loadings accurately capturing the data generation model. 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 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

Table 1 Loadings for **X1D**

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

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 arguably slightly better as the other components are driven closer to 0.

5.2 Order determination

In order to investigate the accuracy of the order determination provided by ARD, we conducted similar experiments to those in Sect. 5.1, varying several parameters. We looked at $D \in \{10, 20\}$, $N \in \{25, 50, 100\}$, $d \in \{1, 2, 3\}$. For each combination of parameters, we constructed data by the following method and used both SPPCA and SSPPCA to estimate d, repeating this 50 times in order to understand the average behaviour. When $d = 1$, the ith observation ($i = 1, \ldots, N$) was given by (4), when $d = 2$, it was given by (5), and when $d = 3$ it was given by (6).

Table 3a and b give the percentage of times each algorithm correctly identified d for a given choice of N and d with $D = 20$. 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 $N = 200$. From our experiments, we find that SPPCA increasingly struggles with noise as the value of n increases, mistaking it for signal with greater confidence. As such, it generally over-estimates the dimension of the data as n increases.

5.3 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

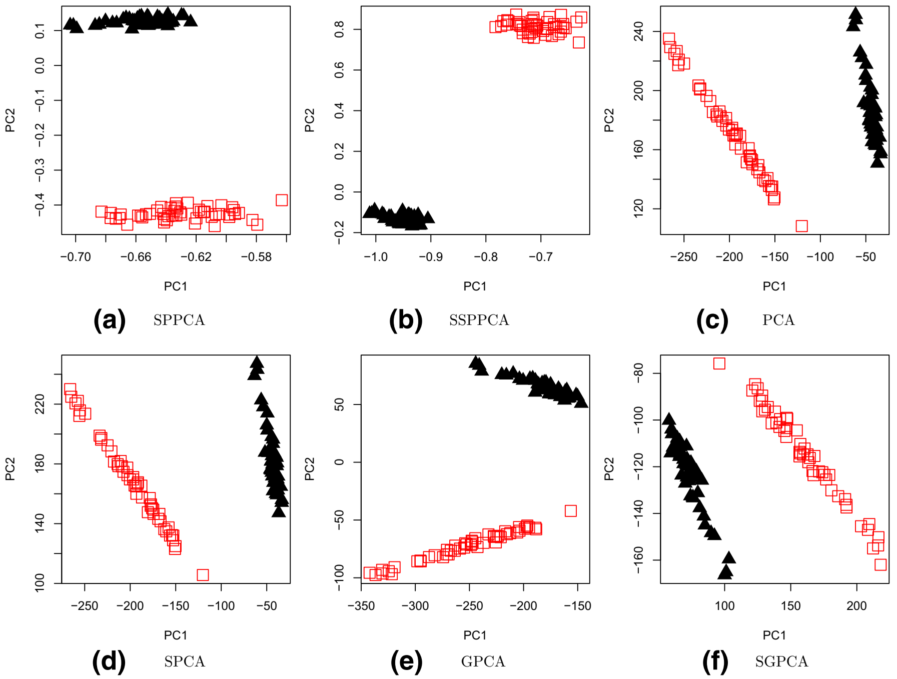
Table 2 Two loadings from **X2D**

SPPCA	SSPPCA	PCA	SPCA	GPCA	SGPCA
(a) First loading					
−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
(b) Second loading					
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

Table 3 Percentage of correct identification of d for SPPCA and SSPPCA

N	d		
	1	2	3
(a) SPPCA			
25	94	24	18
50	82	62	26
100	62	24	26
200	24	16	14
(b) SSPPCA			
25	2	8	4
50	42	10	8
100	82	60	18
200	78	70	50

indication of their 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



(a) SPPCA

(b) SSPPCA

(c) PCA

(d) SPCA

(e) GPCA

(f) SGPCA

Fig. 1 Scores from **X2C**. The (red) outline-only squares represent data drawn from the first class, while the (black) filled triangles represent data drawn from the second class (colour figure online)

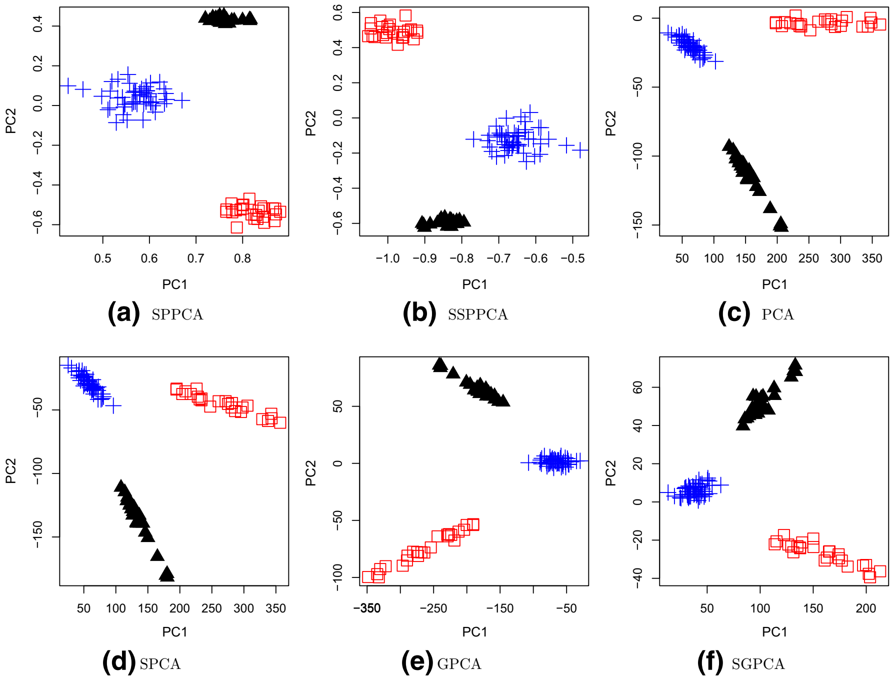
with equal sample sizes from both, the second having three classes with imbalanced sample sizes.

We will use again the hidden factors from (3) and both datasets have dimension $D = 10$ 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 Fig. 1. For GPCA, SGPCA, PCA and SPCA we must specify a dimension: as both SPPCA and SSPPCA choose $d = 2$ we use that value. All six algorithms achieve good separation of the two classes, although it is worth noting that GPCA achieves much worse separation using only the first principal component than the other methods. Visually, it appears that SPPCA and SSPPCA (in Fig. 1a, b respectively) give the best clustering of the two classes. Note, though, that all of the algorithms except GPCA separate the data (except for a single outlying point in

Table 4 Average (Euclidean) silhouettes

	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

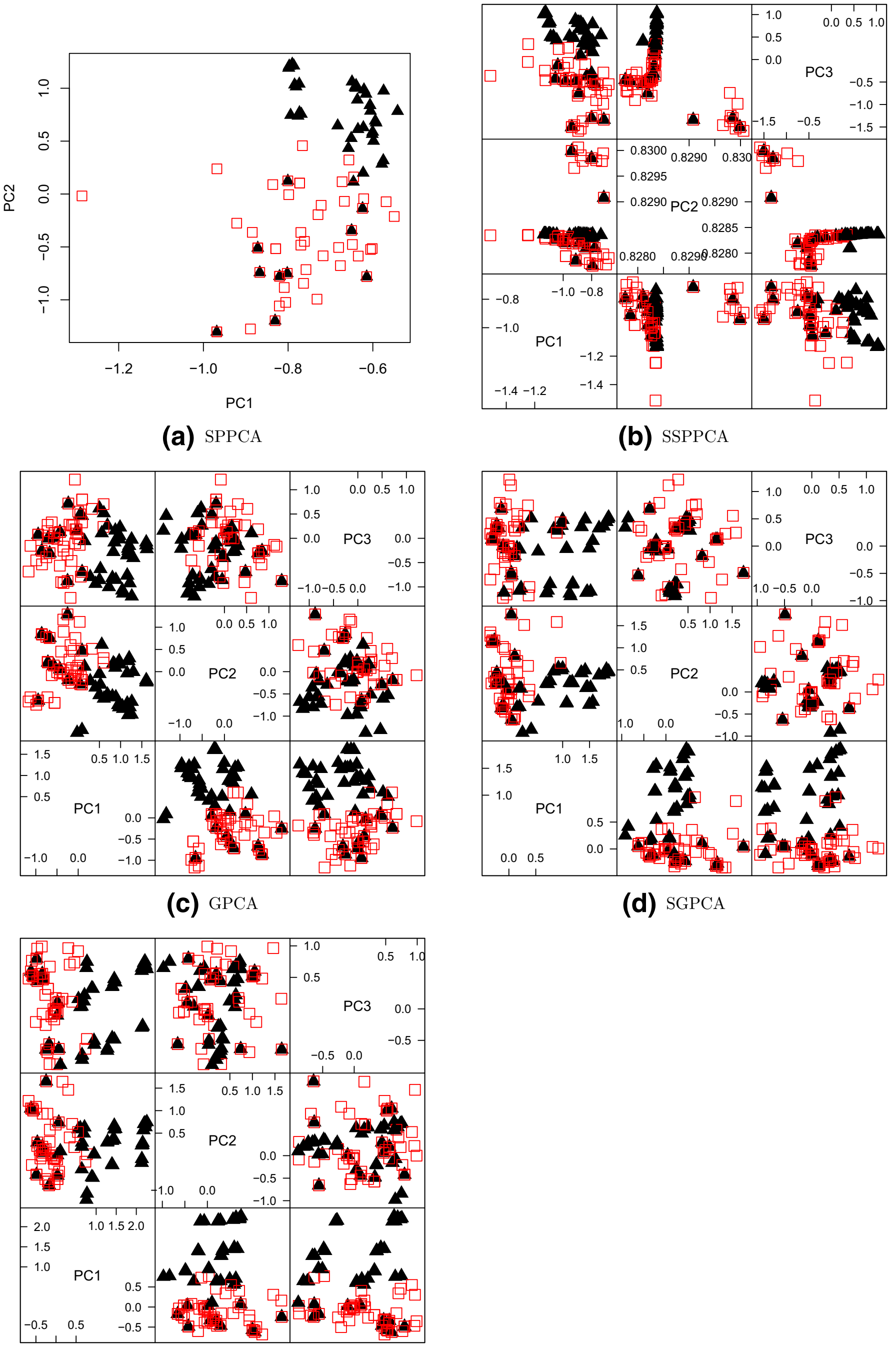


(a) SPPCA **(b)** SSPPCA **(c)** PCA

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

Fig. 2 Scores from **X3C**. The (red) outline-only squares represent data drawn from the first class, the (black) filled triangles represent data drawn from the second class, and the (blue) + symbols represent data drawn from the third (majority) class (colour figure online)

SSPPCA) with only the first direction, which is encouraging, especially for PCA and SPCA which are not specialised to this situation. We use the method of silhouettes put forward by Rousseeuw (1987) to analyse the performance further, using the Euclidean distance metric and clusters found using k-medioid clustering. The silhouette of the ith observation is given by $\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$, where $a(i)$ is the average dissimilarity of the ith observation to the other members of its cluster and $b(i)$ is the lowest average dissimilarity of the ith 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 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, differing from PCA by a little over 25%. The superior performance of SPPCA and SSPPCA is



(a) SPPCA

(b) SSPPCA

(c) GPCA

(d) SGPCA

(e) PCA

Fig. 3 The resulting principal components from applying SPPCA, SSPPCA, GPCA, SGPCA and PCA to the healthcare data

Table 5 Average silhouettes first the healthcare data

SPPCA	SSPPCA	GPCA	SGPCA	PCA
0.38	0.34	0.23	0.17	0.21

continued in the three-class study (Fig. 2), though the gap does narrow, as we can see from the silhouettes. We note that none of the tested methods perform poorly, were one to achieve separation on a real-world dataset like, for example, PCA does with this synthetic example, it would be a significant success. However, real-world data is rarely so amiable as a synthetic example like this, and we suggest that the performance gain from the SPPCA and SSPPCA methods on a real-world dataset may well be crucial to providing a workable dimension reduction.

5.4 Healthcare data

We will now examine the efficacy of SPPCA and SSPPCA in reducing the dimension of a real-world dataset. The data is a sample of 100 observations from a lexicon classifier dataset used by Cardiff and Vale University Health Board in the analysis of letters sent from consultants at a hospital to general practitioners about outpatients. Broadly, the data falls into two classes: discharge letters and follow-up appointment letters. Due to the nature of these letters, there is a heavy imbalance between the two classes. However, in order to better illustrate the performance of the methods in this manuscript, we have randomly selected an equal sample size from each class. This leaves us with 100 observations of dimension $D = 55$.

In Fig. 3 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 shown with circles. For both SPPCA and SSPPCA we used $M = 40$. For SSPPCA we also used $k = 0.07$. From Fig. 3a we can see that SPPCA estimated d as 2; on the other hand, from Fig. 3b we see that SSPPCA chose $d = 3$. Based on this, we chose $d = 3$ for GPCA, SGPCA and PCA, which require a fixed value.

There is evidence of class separation in all the principal component diagrams, even in just the pairs of dimensions for the 3-dimensional methods. However, it is unclear just from these visualisations which of the methods has the best performance. In order to better quantify the clustering, we give the average (Euclidean) silhouettes in Table 5. Based on this performance metric, SPPCA and SSPPCA are the best performers, performing significantly better than previous methods, including PCA which is the default method in practice.

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 for inference on $\boldsymbol{W}$ and $\boldsymbol{Y}$ than SePCA. We have illustrated this in the specific case where the distribution is Poisson, by developing the SPPCA algorithm. 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 easily extendable 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 the fully specified probability model upon which SePCA is based; for more details on this model we direct the reader to Li and Tao (2013). 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.

Acknowledgements The second author would like to thank St John's College in Oxford for part-funding this project and Cardiff University's School of Mathematics for hosting him for 8 weeks. Support from both schools was vital to the successful completion of this project.
The authors would like to thank Cardiff and Vale University Health Board for graciously providing the healthcare dataset.
The authors express their gratitude for the constructive comments received from the editor and two reviewers which were instrumental in improving the quality of this manuscript.

A Component wise gradients for SPPCA

Following on from Sect. 4.1, we will now derive the element wise derivatives of P:

$$
\begin{aligned}
P &= \mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{Y}) - \sum_{i=1}^{D}\sum_{j=1}^{N} \mathrm{e}^{\mathbf{W}\mathbf{Y}} - \frac{1}{2}\mathrm{tr}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y}) - \frac{1}{2}\mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathrm{Diag}(\boldsymbol{\alpha}))\\
&= \sum_{i=1}^{N}\sum_{j=1}^{d}\sum_{k=1}^{D} \mathbf{X}_{ki}\mathbf{W}_{kj}\mathbf{Y}_{ji} - \sum_{i=1}^{D}\sum_{j=1}^{N}\exp\left(\sum_{k=1}^{d}\mathbf{W}_{ik}\mathbf{Y}_{kj}\right)\\
&\quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{d}\mathbf{Y}_{ji}^2 - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{D}\mathbf{W}_{ji}^2\alpha_i
\end{aligned}
$$

Hence the gradient with respect to $\mathbf{W}$ is

$$\begin{aligned}\frac{\partial P}{\partial \mathbf{W}_{ab}} &= \sum_{i=1}^{N} \mathbf{X}_{ai}\mathbf{Y}_{bi} - \sum_{j=1}^{N} \mathbf{Y}_{bj} \exp\left(\sum_{k=1}^{d} \mathbf{W}_{ak}\mathbf{Y}_{kj}\right) - \mathbf{W}_{ba}\alpha_a \\ &= (\mathbf{X}\mathbf{Y}^{\mathrm{T}})_{ab} - (\mathrm{e}^{\mathbf{W}\mathbf{Y}}\mathbf{Y}^{\mathrm{T}})_{ab} - (\mathbf{W}\mathrm{Diag}\boldsymbol{\alpha})_{ab}\end{aligned}$$

and the gradient with respect to $\mathbf{Y}$ is

$$\begin{aligned}\frac{\partial P}{\partial \mathbf{Y}_{ab}} &= \sum_{k=1}^{D} \mathbf{X}_{kb}\mathbf{W}_{ka} - \sum_{i=1}^{D} \mathbf{W}_{ia} \exp\left(\sum_{k=1}^{d} \mathbf{W}_{ik}\mathbf{Y}_{kb}\right) - \mathbf{Y}_{ab} \\ &= (\mathbf{W}^{\mathrm{T}}\mathbf{X})_{ab} - (\mathbf{W}^{\mathrm{T}}\mathrm{e}^{\mathbf{W}\mathbf{Y}})_{ab} - (\mathbf{Y})_{ab}.\end{aligned}$$

B Component wise gradients for SSPPCA

As needed for the estimation algorithm for SSPPCA, described in Sect. 4.2, we will now derive the gradients of P_p^s element wise:

$$\begin{aligned}P_p^s &= \mathrm{tr}(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{Y}) - \sum_{i=1}^{D}\sum_{j=1}^{N} \mathrm{e}^{\mathbf{W}\mathbf{Y}} - \frac{1}{2}\mathrm{tr}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y}) \\ &\quad -\frac{1}{2}\mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathrm{Diag}(\boldsymbol{\alpha})) - k\ \sum_{i=1}^{D}\sum_{j=1}^{d}\ \frac{(\mathbf{W}_{ij})^2}{(\mathbf{W}_{ij}^0)^2 + \delta} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{d}\sum_{k=1}^{D} \mathbf{X}_{ki}\mathbf{W}_{kj}\mathbf{Y}_{ji} - \sum_{i=1}^{D}\sum_{j=1}^{N} \exp\left(\sum_{k=1}^{d} \mathbf{W}_{ik}\mathbf{Y}_{kj}\right) - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{d} \mathbf{Y}_{ji}^2 \\ &\quad -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{D} \mathbf{W}_{ji}^2\alpha_i - k\ \sum_{i=1}^{D}\sum_{j=1}^{d}\ \frac{(\mathbf{W}_{ij})^2}{(\mathbf{W}_{ij}^0)^2 + \delta}\end{aligned}$$

Hence the gradient with respect to $\mathbf{W}$ is

$$\begin{aligned}\frac{\partial P_p^s}{\partial \mathbf{W}_{ab}} &= \sum_{i=1}^{N} \mathbf{X}_{ai}\mathbf{Y}_{bi} - \sum_{j=1}^{N} \mathbf{Y}_{bj} \exp\left(\sum_{k=1}^{d} \mathbf{W}_{ak}\mathbf{Y}_{kj}\right) - \mathbf{W}_{ba}\alpha_a - 2k\frac{\mathbf{W}_{ab}}{(\mathbf{W}_{ab}^0)^2 + \delta} \\ &= (\mathbf{X}\mathbf{Y}^{\mathrm{T}})_{ab} - (\mathrm{e}^{\mathbf{W}\mathbf{Y}}\mathbf{Y}^{\mathrm{T}})_{ab} - (\mathbf{W}\mathrm{Diag}\boldsymbol{\alpha})_{ab} - 2k\left(\frac{\mathbf{W}}{(\mathbf{W}^0)^2 + \delta}\right)_{ab}\end{aligned}$$

and the gradient with respect to $\mathbf{Y}$ (though we note it is identical to the above) is

$$\frac{\partial P_p^s}{\partial \mathbf{Y}_{ab}} = \sum_{k=1}^{D} \mathbf{X}_{kb}\mathbf{W}_{ka} - \sum_{i=1}^{D} \mathbf{W}_{ia} \exp\left(\sum_{k=1}^{d} \mathbf{W}_{ik}\mathbf{Y}_{kb}\right) - \mathbf{Y}_{ab}$$

$$= (\mathbf{W}^{\mathrm{T}}\mathbf{X})_{ab} - (\mathbf{W}^{\mathrm{T}}\mathrm{e}^{\mathbf{WY}})_{ab} - (\mathbf{Y})_{ab}.$$

□

References

Bishop CM (1999) Bayesian PCA. In: Advances in neural information processing systems. vol 11, pp 382–388

Buntine W (2002) Variational extensions to EM and multinomial PCA. Mach Learn ECML 2002:23–34

Collins M, Dasgupta S, Schapire RE (2002) A generalization of principal components analysis to the exponential family. Adv Neural Inf Process Syst 14:617–624

Fan J, Li R (2001) Variable selection via nonconcave penalized likelihood and its oracle properties. J Am Stat Assoc 96(456):1348–1360

Frommlet F, Nuel G (2016) An adaptive ridge procedure for L0 regularization. PLoS ONE 11(2):1–23

Landgraf AJ, Lee Y (2015) Generalized principal component analysis: projection of saturated model parameters. Ohio State University Statistics Department Technical Report, (890). Available from: http://www.stat.osu.edu/~yklee/mss/tr890.pdf

Li J, Tao D (2013) Simple exponential family PCA. IEEE Trans. Neural Netw. Learn. Syst. 24(3):485–497

Mackay DJC (1995) Probable networks and plausible predictions–a review of practical Bayesian methods for supervised neural networks. Netw Comput Neural Syst 6(3):469–505

Mohamed S, Heller K, Ghahramani Z (2009) Bayesian exponential family PCA. Adv Neural Inf Process Syst 21:1089–1096

Pearson K (1901) On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 2(1):559–572

Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65

Smallman L, Artemiou A, Morgan J (2018) Sparse generalised principal component analysis. Pattern Recognit 83:443–455

Tibshirani R (1996) Regression shrinkage and selection via the lasso. J R Stat Soc Ser B (Methodol) 58(1):267–288

Tipping ME, Bishop CM (1999) Probabilistic principal component analysis. J R Stat Soc Ser B (Stat Methodol) 61(3):611–622

Yi S, Lai Z, He Z, Cheung Y, Liu Y (2017) Joint sparse principal component analysis. Pattern Recognit 61:524–536

Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. J Comput Graph Stat 2:265–286

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.