Erasmus University Rotterdam

Erasmus School of Economics

Bachelor Thesis Econometrics & Operations Research

A comparison of deflation techniques for sparse principal component analysis applied to the penalized matrix decomposition

Arthur Ning (558688)

Supervisor:	dr. Pieter C. Schoonees
Second assessor:	dr. Flavius Frasincar
Date final version:	July 2, 2023

Abstract

Witten et al. (2009) introduce a framework known as the penalized matrix decomposition (PMD), which uses low-rank approximations of the data matrix to derive sparse principal components. To obtain multiple sparse principal components, PMD performs matrix deflation using Hotelling's deflation scheme (Hotelling, 1933). However, Mackey (2008) states that utilizing Hotelling's deflation is inappropriate in the context of sparse principal component analysis (PCA) and thus introduces two new deflation schemes: Schur complement deflation and the generalized deflation method. In this paper, we extend the research of Witten et al. (2009) by incorporating the deflation schemes of Mackey (2008) within the PMD framework and comparing their performance in comparison to Hotelling's deflation method. To perform the comparative study, we replicate part of the research done by Witten et al. (2009). Namely, we apply the corresponding three variants of PMD to the gene expression data set used by Witten et al. (2009). When we apply the variations of PMD to the gene expression data set, we compare the deflation schemes in terms of how well PMD is able the capture the variance in the data and to what extent the interpretation of the sparse principal components is hindered by artifacts, new directions introduced in the deflated data matrix that are not present in the original data. As part of our extension to the research of Witten et al. (2009), we also generate simulated data. We use the cosine similarity to evaluate how well each variant of PMD is able to recover the sparse loading vectors used to simulate the data. We find that the variant of PMD that is able to recover the sparse loading vectors most optimally is dependent on the number of observations in the simulated data matrix. Additionally, when we apply the three variants of PMD to the gene expression data set, we observe that PMD with Mackey's generalized deflation method is the most preferred method as it captures the most variance in the data and its components have a fairly low percentage of artifacts.

Contents

1 Introduction

In today's digital age, the amount of data generated has grown exponentially, resulting in an increase in the size and complexity of the data sets used in fields such as business analytics, healthcare and finance. It can be quite a difficult task to extract meaningful insights from data sets with a large number of variables. *Principal component analysis* (PCA), invented by Pearson (1901) and further developed by Hotelling (1933), is a powerful statistical technique that can help address this challenge by reducing the number of variables in the data set while minimizing the amount of information that is lost. PCA does this by constructing *principal components*. These are linear combinations of the variables from the data set, which aim to capture as much of the variance in the data as possible. The coefficients in these linear combinations are called *loadings*.

Nevertheless, PCA has a drawback. Namely, the principal components are typically a linear combination of all the variables from the data set. This results in the principal components often being difficult to interpret when the data set contains a large number of variables (Zou et al., 2006). For this reason, sparse PCA has been a topic of extensive research in the past couple of decades (e.g. Jolliffe et al., 2003; Zou et al., 2006; Witten et al., 2009; Lai et al., 2014; Guerra-Urzola et al., 2021). This variant of PCA aims to improve the interpretability of the principal components by simply setting some of the loadings to zero. However, using less variables sacrifices some of the variance in the data captured by the principal components. Thus, sparse PCA tries to strike a balance between two conflicting goals (Journée et al., 2010). On one hand, it seeks to capture the maximum amount of variation in the data. On the other hand, it aims to achieve this by constructing principal components that use the fewest possible variables.

Many approaches to derive sparse principal components have been put forward by researchers. In this paper, we focus on the *penalized matrix decomposition* (PMD) proposed by Witten et al. (2009). To find a single sparse principal component using PMD, we perform the following two steps. First, we find a rank-1 approximation of the data matrix, which among other things consists of a vector containing the sparse loadings of the principal component, known as the sparse loading vector. Subsequently, we deflate the data matrix by subtracting the rank-1 approximation from it; we can use the deflated data matrix to find the next sparse principal component. The deflation method used is known as *Hotelling's deflation* (Hotelling, 1933; Weylandt, 2019). Mackey (2008) states that it is often not appropriate to utilize Hotelling's deflation in the context of sparse PCA. Namely, since the sparse loading vectors are typically non-orthogonal in contrast to ordinary PCA, it is possible that two different sparse principal components found by PMD might capture some of the same variance in the data. This makes it more difficult to interpret the components, as it is harder to attribute specific patterns or variations in the data to a single component (Mackey, 2008; Camacho et al., 2020). Hence, Mackey (2008) introduces the deflation schemes *Schur complement deflation* and the *generalized deflation method* to rectify this issue. The goal of this paper is to extend the research done by Witten et al. (2009) by assessing the performance of Hotelling's deflation scheme in comparison to Schur complement deflation and the generalized deflation method when we incorporate these deflation methods within the PMD framework.

For our comparative study, we replicate part of the research done by Witten et al. (2009). Specifically, we apply the corresponding three variants of PMD to the gene expression data set used by Witten et al. (2009). When we apply the variations of PMD to this gene expression data set, we compare the deflation schemes in terms of how well PMD is able the capture the variance in the data and to what extent the interpretation of the sparse principal components is hindered by artifacts, new directions introduced in the deflated data matrix that are not present in the original data. To measure these two factors, we compute the cumulative proportion of variance explained and the percentage of artifacts in each principal component, respectively.

As part of our extension to the research of Witten et al. (2009), we create simulated data. To generate the data matrices for the simulation study, we need to manually specify the sparse loading vectors. After we have created a data matrix, we apply the three variations of PMD to it which try to recover the sparse loading vectors we specified previously. In order to assess how well the variants of PMD are able to recover the true sparse loading vectors, we compute the cosine similarities between the true sparse loading vectors and the loading vectors found by the variants of PMD. We perform this simulation study multiple times by varying the number of observations in the data matrix.

We obtain the following results when we perform our comparative analysis. To begin, we find that none of the deflation schemes have a dominant performance when we apply PMD to our simulated data. Specifically, the variant of PMD that best recovers the true sparse loading vectors is dependent on the number of observations present in this particular simulated data set. Moreover, when we apply the three variants of PMD to the gene expression data set, we observe that PMD with Mackey's generalized deflation method is the most preferred method as it captures the most variance in the data and its components have a fairly low percentage of artifacts.

Our contribution to the broader literature is two-fold. Firstly, we obtain results which contradict the outcomes of the previous works by Mackey (2008), Weylandt (2019) and Camacho et al. (2021), stimulating researchers to further investigate the comparative performance of these deflation techniques. Secondly, although Camacho et al. (2021) incorporate the generalized deflation method of Mackey (2008) within the PMD framework, to our knowledge there is currently no literature available on the application of Schur complement deflation to PMD. Thus, we fill this gap in the literature by exploring the application of Schur complement deflation to PMD, providing novel insights into the performance of this deflation scheme.

We structure the remainder of the paper in the following way. In Section 2, we summarize and discuss some of the research that has been done on matrix deflation in the context of sparse PCA. Then, in Section 3 we describe all the methods that are relevant to retrieve the results for our research. Subsequently, we describe the data used in the comparative study in Section 4. In Section 5 we apply our methods to the data and report our results. Finally, in Section 6 we conclude this paper by summarizing our key findings, discussing the limitations of our research, and offering suggestions for future investigations.

2 Literature review

There is a plethora of research on the comparison of various deflation techniques in the context of sparse PCA. One of the pioneers in this topic is Mackey (2008), who highlights that deflation methods do not receive attention in the literature of sparse PCA, as many authors typically borrow Hotelling's deflation scheme from ordinary PCA without proper reasoning. Mackey (2008) demonstrates that Hotelling's deflation is often not appropriate when deriving sparse principal components. Therefore, they propose several deflation schemes which are more suitable for the sparse PCA setting. These methods are projection deflation, Schur complement deflation, orthogonalized Hotelling's deflation and the generalized deflation method. Mackey (2008) applies these deflation schemes to the methods generalized spectral bounds for sparse linear discriminant analysis (GSLDA) and sparse PCA using difference of convex functions (D.C.) programming (DC-PCA), and performs some analyses on real data sets. Mackey (2008) finds that all deflation schemes except for the orthogonalized Hotelling's deflation scheme allow for DC-PCA and GSLDA to explain more of the variance in the data than Hotelling's deflation.

In a more recent study, Weylandt (2019) extends the projection deflation and Schur complement deflation approaches proposed by Mackey (2008). Most sparse PCA methods depend on the covariance matrix of the data matrix and its eigenvectors. Thus, Mackey (2008) uses these elements to formulate the deflation schemes. However, Weylandt (2019) reformulates the deflation methods such that they consist of the (deflated) data matrix and its corresponding singular vectors. This reformulation makes these deflation schemes appropriate for sparse PCA methods which utilize low-rank approximations and the singular value decomposition (SVD) like PMD or the sparse and functional principal components analysis (SPFCA) method introduced by Allen and Weylandt (2019). When applying the reformulated projection deflation and Schur complement deflation to SPFCA, Weylandt (2019) finds that SPCFA is able to capture more variability in the data when projection deflation or Schur complement deflation is used instead of Hotelling's deflation. This result is similar to that found by Mackey (2008).

Camacho et al. (2021) incorporate the generalized deflation method of Mackey (2008) within the PMD framework and compare this deflation scheme to Hotelling's deflation. Instead of measuring the amount variance that PMD is able to capture with each of the deflation schemes, Camacho et al. (2020) utilize other metrics to evaluate the deflation methods. Namely, they manually specify sparse loading vectors and generate data using these loading vectors. Subsequently, they apply PMD with the various deflation methods to these simulated data and try to recover the loading vectors used to generate the data. To assess how well PMD is able to recover the true loading vectors with each of the deflation methods, Camacho et al. (2021) compute the cosine similarity between the true loading vector and the loading vector obtained by PMD. They find that PMD with the generalized deflation method of Mackey (2008) is able to better recover the true sparse loading vectors compared to PMD with Hotelling's deflation. Similarly to Mackey (2008) and Weylandt (2019), Camacho et al. (2021) provides evidence in favor against the usage of Hotelling's deflation in the sparse PCA setting.

3 Methodology

Section 3 gives an overview of the methods we use in this paper. Specifically, in Section 3.1 we describe PCA and its connection to the SVD and low-rank approximations of matrices. Then, we introduce the PMD framework proposed by Witten et al. (2009) in Section 3.2. In Section 3.3, we present the deflation schemes which we apply to PMD and compare to one another. Lastly, in Section 3.4 we discuss the metrics used to evaluate the deflation methods.

3.1 Principal component analysis

3.1.1 The method

Let $\mathbf{X} \in \mathbb{R}^{N \times P}$ be the data matrix containing N observations and P variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{P \times P}$ the covariance matrix of $\mathbf{X}$. Without loss of generality, we assume for the remainder of the paper that the variables of $\mathbf{X}$ are centered. PCA aims to reduce the dimensionality of $\mathbf{X}$ by constructing $K \leq \min\{N, P\}$ linear combinations of the variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P$. These linear combinations are known as principal components. We can write the K principal components $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K$ as follows:

$$\begin{aligned}
\boldsymbol{z}_1 &= \mathbf{X}\boldsymbol{v}_1 = v_{11}\boldsymbol{x}_1 + v_{12}\boldsymbol{x}_2 + \cdots + v_{1P}\boldsymbol{x}_P \\
\boldsymbol{z}_2 &= \mathbf{X}\boldsymbol{v}_2 = v_{21}\boldsymbol{x}_1 + v_{22}\boldsymbol{x}_2 + \cdots + v_{2P}\boldsymbol{x}_P \\
&\vdots \\
\boldsymbol{z}_K &= \mathbf{X}\boldsymbol{v}_K = v_{K1}\boldsymbol{x}_1 + v_{K2}\boldsymbol{x}_2 + \cdots + v_{KP}\boldsymbol{x}_P,
\end{aligned} \tag{1}$$

where $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ are the loading vectors and v_{ki} is element i of $\boldsymbol{v}_k, k = 1, \ldots, K$.

Since we are reducing the dimensionality of $\mathbf{X}$, we are losing some information contained in the matrix. Hence, we want to minimize this loss of information. We do this by sequentially constructing principal components that are uncorrelated and maximize their own variance. We determine the first principal component by finding the linear combination $\boldsymbol{z}_1 = \mathbf{X}\boldsymbol{v}_1$ such that $\widehat{Var}(\boldsymbol{z}_1) = \boldsymbol{v}_1^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{v}_1$ is maximized, where $\hat{\boldsymbol{\Sigma}} = \frac{\mathbf{X}^\top \mathbf{X}}{N}$ is the sample covariance matrix of $\mathbf{X}$ (Zou and Xue, 2018). As we can increase $\widehat{Var}(\boldsymbol{z}_1)$ by multiplying $\boldsymbol{v}_1$ by some constant, we restrict $\boldsymbol{v}_1$ to have unit length i.e. $\boldsymbol{v}_1^\top \boldsymbol{v}_1 = 1$. Subsequently, we look for the second principal component $\boldsymbol{z}_2 = \mathbf{X}\boldsymbol{v}_2$ that maximizes $\widehat{Var}(\boldsymbol{z}_2)$ subject to $\boldsymbol{v}_2^\top \boldsymbol{v}_2 = 1$ and $\widehat{Cov}(\boldsymbol{z}_2, \boldsymbol{z}_1) = \boldsymbol{v}_2^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{v}_1 = 0$. We can generalize this procedure for the k'th principal component. Namely, we determine the k'th principal component by finding the linear combination $\mathbf{X}\boldsymbol{v}_k$ that maximizes $\widehat{Var}(\boldsymbol{z}_k)$ subject to $\boldsymbol{v}_k^\top \boldsymbol{v}_k = 1$ and $\widehat{Cov}(\boldsymbol{z}_k, \boldsymbol{z}_l) = 0,\ l < k$.

From the iterative process described above, we can establish the following properties of PCA:

1. **Ordered maximized variances**: $\widehat{Var}(\boldsymbol{z}_1) \geq \cdots \geq \widehat{Var}(\boldsymbol{z}_K) \geq 0$.

2. **Orthogonal loading vectors**: $\boldsymbol{v}_k^\top \boldsymbol{v}_l = 0$ for $k, l = 1, \ldots, K,\ k \neq l$.

3. **Uncorrelated principal components**: $\widehat{Cov}(\boldsymbol{z}_k, \boldsymbol{z}_l) = 0$ for $k, l = 1, \ldots, K,\ k \neq l$.

It is easy to obtain linear combinations that satisfy all three conditions. Johnson and Wichern (2014) provide a detailed proof in Section 8.2 that the eigenvectors of $\boldsymbol{\Sigma}$ satisfy all the conditions.

3.1.2 Connection to the singular value decomposition & low-rank approximations

Suppose that rank($\mathbf{X}$) $= R$ and $K \leq R$. Then, it is possible to approximate $\mathbf{X}$ using a rank-K matrix. Such a matrix is known as a low-rank approximation, which we denote as $\hat{\mathbf{X}}$. We can find $\hat{\mathbf{X}}$ by optimizing the criterion

$$\underset{\hat{\mathbf{X}} \in M(K)}{\operatorname{argmin}} \, ||\mathbf{X} - \hat{\mathbf{X}}||_F^2, \tag{2}$$

where $||.||_F$ is the Frobenius norm and $M(K)$ is the set of rank-K $N \times P$ matrices. To solve this optimization problem, we use the SVD of $\mathbf{X}$ which is given by

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top. \tag{3}$$

Here, $\mathbf{U} \in \mathbb{R}^{N\times N}$ and $\mathbf{V} \in \mathbb{R}^{P\times P}$ are orthogonal matrices with the columns $\boldsymbol{u}_1, \dots, \boldsymbol{u}_N$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_P$, respectively. $\mathbf{D} \in \mathbb{R}^{N\times P}$ is a rectangular diagonal matrix containing the non-negative diagonal elements $d_{11}, \dots, d_{MM}$ such that $M = \min\{N, P\}$ and $d_{11} \geq \dots \geq d_{MM} \geq 0$.

Denote $\mathbf{U}_K \in \mathbb{R}^{N\times K}$ and $\mathbf{V}_K \in \mathbb{R}^{P\times K}$ as orthogonal matrices which contain the first K columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. Additionally, let $\mathbf{D}_K \in \mathbb{R}^{K\times K}$ be a diagonal matrix containing the K largest diagonal elements from $\mathbf{D}$, $d_{11}, \dots, d_{KK}$. Then, we can compute the rank-K approximation $\hat{\mathbf{X}}$ using the solution proposed by Eckart and Young (1936):

$$\hat{\mathbf{X}} = \mathbf{U}_K\mathbf{D}_K\mathbf{V}_K^\top. \tag{4}$$

This low-rank approximation of $\mathbf{X}$ is connected to PCA. Namely, it can be shown that the columns of $\mathbf{V}_K$ (the rows of $\mathbf{V}_K^\top$) are the eigenvectors of $\boldsymbol{\Sigma}$. In other words, $\mathbf{V}_K$ holds the desired K loading vectors that we need to construct the principal components (Zou et al., 2006; Shen and Huang, 2008). This fact motivates the use of low-rank approximations in the PMD framework to determine sparse principal components. We introduce PMD in Section 3.2.

3.2 Penalized matrix decomposition

Witten et al. (2009) introduce a new framework for computing a rank-K approximation of a matrix known as PMD. This framework imposes ℓ_1- or fused LASSO penalties on the columns of $\mathbf{U}_K$ and $\mathbf{V}_K$ to ensure that they are sparse i.e. the column vectors contain many zero elements. To derive the rank-K approximation of the data matrix $\mathbf{X}$, we start with the rank-1 approximation. Let $d = \mathbf{D}_1$, $\boldsymbol{u} = \mathbf{U}_1$ and $\boldsymbol{v} = \mathbf{V}_1$. Then, we can determine the rank-1 approximation by optimizing the PMD(ℓ_1, ℓ_1) criterion, which is given by the following expression:

$$\max_{\boldsymbol{u},\boldsymbol{v}} \boldsymbol{u}^\top\mathbf{X}\boldsymbol{v} \text{ s.t. } ||\boldsymbol{u}||_1 \leq c_1, ||\boldsymbol{v}||_1 \leq c_2,\ ||\boldsymbol{u}||_2^2 \leq 1,\ ||\boldsymbol{v}||_2^2 \leq 1, \tag{5}$$

where $||.||_1$ is the ℓ_1-norm, $||.||_2$ is the ℓ_2-norm and $c_1 \in [1, \sqrt{N}]$ and $c_2 \in [1, \sqrt{P}]$ are parameters that control the amount of sparsity imposed on $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively. The larger the value for c_1, the less sparsity we induce on $\boldsymbol{u}$. The same holds for c_2. Optimizing the PMD(ℓ_1, ℓ_1) criterion gives us $\boldsymbol{u}$ and $\boldsymbol{v}$, which we use to compute $d = \boldsymbol{u}^\top\mathbf{X}\boldsymbol{v}$.

Since the optimization problem given in Equation (5) is biconvex, Witten et al. (2009) propose an iterative procedure for solving it. We present this procedure in Algorithm 1.

Algorithm 1: Rank-1 approximation using PMD(ℓ_1, ℓ_1)

```
  Input: $\mathbf{X}$, $c_1$, $c_2$
  Output: $d$, $\boldsymbol{u}$, $\boldsymbol{v}$
1 Initialize $\boldsymbol{v}$ to be a random vector with an $\ell_2$-norm equal to 1.
2 repeat
3     $\boldsymbol{u} = \frac{S(\mathbf{X}\boldsymbol{v},\lambda_1)}{||S(\mathbf{X}\boldsymbol{v},\lambda_1)||_2}$, where $\lambda_1 = 0$ if this ensures that $||\boldsymbol{u}||_1 \leq c_1$; otherwise, use binary
        search to find the value of $\lambda_1$ such that $||\boldsymbol{u}||_1 = c_1$.
4     $\boldsymbol{v}_{old} = \boldsymbol{v}$.
5     $\boldsymbol{v} = \frac{S(\mathbf{X}^\top\boldsymbol{u},\lambda_2)}{||S(\mathbf{X}^\top\boldsymbol{u},\lambda_2)||_2}$, where $\lambda_2 = 0$ if this ensures that $||\boldsymbol{v}||_1 \leq c_2$; otherwise, use binary
        search to find the value of $\lambda_2$ such that $||\boldsymbol{v}||_1 = c_2$.
6 until $|\boldsymbol{v} - \boldsymbol{v}_{old}| \leq 10^{-7}$
7 $d = \boldsymbol{u}^\top\mathbf{X}\boldsymbol{v}$.
```

In lines 3 and 4 of Algorithm 1, we use the soft thresholding operator S to determine $\boldsymbol{u}$ and $\boldsymbol{v}$. This operator ensures that all elements in $\boldsymbol{u}$ and $\boldsymbol{v}$ are shrunk towards 0 such that the conditions $||\boldsymbol{u}||_1 \leq c_1$ and $||\boldsymbol{v}||_1 \leq c_2$ are satisfied (Liu and Foygel Barber, 2020). We define the soft thresholding operator as $S(a, b) = \text{sgn}(a)(|a| - b)_+$, where $b > 0$ and x_+ is equal to x if $x > 0$ and 0 otherwise.

In order to obtain the rank-K approximation of $\mathbf{X}$, we utilize Algorithm 2.

Algorithm 2: PMD(ℓ_1, ℓ_1) rank-K approximation using Hotelling's deflation

```
  Input: $\mathbf{X}$, $c_1$, $c_2$
  Output: $d_k$, $\boldsymbol{u}_k$, $\boldsymbol{v}_k$ for $k = 1, \ldots, K$
1 Let $\mathbf{X}_1 = \mathbf{X}$.
2 for $k = 1, \ldots, K$ do
3     Find $d_k, \boldsymbol{u}_k$ and $\boldsymbol{v}_k$ by applying Algorithm 1 to $\mathbf{X}_k$.
4     $\mathbf{X}_{k+1} = \mathbf{X}_k - d_k\boldsymbol{u}_k\boldsymbol{v}_k^\top$.
```

This procedure finds the matrices $\mathbf{U}_K$, $\mathbf{V}_K$ and $\mathbf{D}_K$ by repeatedly performing Algorithm 1. In the first iteration, we apply Algorithm 1 to the matrix $\mathbf{X}_1 = \mathbf{X}$ to obtain the rank-1 approximation $d_1\boldsymbol{u}_1\boldsymbol{v}_1^\top$. We cannot apply Algorithm 1 to $\mathbf{X}_1$ to get d_2, $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$. Thus, we deflate $\mathbf{X}_1$ by subtracting $d_1\boldsymbol{u}_1\boldsymbol{v}_1^\top$ from it, giving us the new matrix $\mathbf{X}_2$. In Section 3.3, we explain in more detail the potential problems of utilizing this particular deflation method. If we now apply Algorithm 1 to $\mathbf{X}_2$, we retrieve d_2, $\boldsymbol{u}_2$ and $\boldsymbol{v}_2$. We subtract $d_2\boldsymbol{u}_2\boldsymbol{v}_2^\top$ from $\mathbf{X}_2$ to obtain $\mathbf{X}_3$. Repeating this procedure $K - 2$ more times gives us $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ and $d_{11}, \ldots, d_{KK}$.

To derive sparse principal components, Witten et al. (2009) modify the PMD(ℓ_1, ℓ_1) criterion such that an ℓ_1-penalty is imposed on $\boldsymbol{v}$ but not $\boldsymbol{u}$. This new criterion is known as PMD(., ℓ_1) and is given by the expression

$$\max_{\boldsymbol{u},\boldsymbol{v}} \boldsymbol{u}^\top \mathbf{X}\boldsymbol{v} \text{ s.t. } ||\boldsymbol{v}||_1 \leq c_2,\ ||\boldsymbol{u}||_2^2 \leq 1,\ ||\boldsymbol{v}||_2^2 \leq 1. \tag{6}$$

We can still use Algorithms 1 and 2 to optimize the PMD(., ℓ_1) criterion. However, since we do not impose sparsity on the elements of $\boldsymbol{u}$, we update $\boldsymbol{u}$ using $\boldsymbol{u} = \frac{\mathbf{X}\boldsymbol{v}}{||\mathbf{X}\boldsymbol{v}||_2}$ in line 3 of Algorithm 1.

3.3 Deflation techniques

As mentioned in Section 3.2, deflating the matrix $\mathbf{X}_k$ is a crucial step when iteratively determining the rank-K approximation of $\mathbf{X}$ using PMD. In Section 3.3.1, we discuss the deflation technique known as Hotelling's deflation, which is used in Algorithm 2, and the issues that may arise when using it in the context of sparse PCA. In Sections 3.3.2 and 3.3.3, we present the Schur complement deflation and the generalized deflation method, respectively. These deflation methods aim to solve the problem presented in Section 3.3.1.

3.3.1 Hotelling's deflation

Let $\mathbf{X}_1 = \mathbf{X}$. Then, we use Hotelling's deflation to sequentially obtain the deflated matrices $\mathbf{X}_2, \ldots, \mathbf{X}_K$ as follows:

$$\mathbf{X}_{k+1} = \mathbf{X}_k - d_k \boldsymbol{u}_k \boldsymbol{v}_k^\top. \tag{7}$$

Recall from Section 3.1 that the loading vectors should be orthogonal in the context of ordinary PCA. We can easily satisfy this condition by setting the loading vectors equal to the eigenvectors of the covariance matrix of $\mathbf{X}$. Nevertheless, the sparse loading vectors found by PMD are generally not eigenvectors and non-orthogonal (Mackey, 2008). This has negative implications for the usage of Hotelling's deflation scheme. Namely, when we are sequentially deflating the data matrix with respect to a series of non-orthogonal loading vectors, this deflation technique does not guarantee that $\mathbf{X}_{k+s}\boldsymbol{v}_k = \mathbf{0}_N, k = 1, \ldots, K, s > 0$ (Weylandt, 2019). In other words, principal component $k + s$ could explain some of the same variability in the data as principal component k. This makes it more difficult to interpret the components since it is harder to attribute specific patterns or variations in the data to a single component (Camacho et al., 2020; Mackey, 2008).

For the remainder of the paper, we refer to the variant of PMD using Hotelling's deflation scheme as PMD-HD.

3.3.2 Schur complement deflation

To rectify the issue discussed in Section 3.3.1, Mackey (2008) introduces a new deflation scheme known as Schur complement deflation, which is given by the following expression:

$$\mathbf{X}_{k+1} = \Big(\mathbf{I}_N - -\frac{\mathbf{X}_k \boldsymbol{v}_k \boldsymbol{v}_k^\top \mathbf{X}_k^\top}{\boldsymbol{v}_k^\top \mathbf{X}_k \boldsymbol{v}_k}\Big)\mathbf{X}_k, \tag{8}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix. We can interpret this expression as the projection of the columns of $\mathbf{X}_k$ onto the orthogonal complement of the space spanned by the principal component k. Although it may not seem intuitive from this interpretation that $\mathbf{X}_{k+s}\boldsymbol{v}_k = \mathbf{0}_N, k = 1, \dots, K, s > 0$, in Section 2.3.3 Mackey (2008) gives a detailed mathematical proof that the deflated matrices found using Schur complement deflation satisfy this condition.

In order to incorporate this deflation scheme within PMD framework, we simply replace line 4 of Algorithm 2 with the expression from (8). We refer to the variant of PMD using Schur complement deflation as PMD-SC.

3.3.3 Mackey's generalized deflation method

Mackey (2008) introduces another deflation technique known as the generalized deflation method for sparse PCA. This method is based on the projection deflation scheme, which deflates the matrix $\mathbf{X}_k$ using

$$\mathbf{X}_{k+1} = \mathbf{X}_k(\mathbf{I}_P - \boldsymbol{v}_k \boldsymbol{v}_k^\top). \tag{9}$$

By projecting $\mathbf{X}_k$ onto the orthogonal complement of the space spanned by $\boldsymbol{v}_k$ and thus removing the variance in the data captured by principal component k, we ensure that we satisfy the condition $\mathbf{X}_{k+1}\boldsymbol{v}_k = \mathbf{0}_N, k = 1, \dots, K$. However, using projection deflation repeatedly does not guarantee that $\mathbf{X}_{k+s}\boldsymbol{v}_k = \mathbf{0}_N,\ k = 1, \dots, K,\ s > 1$, leading to the same problem discussed in Section 3.3.1 (Mackey, 2008; Weylandt, 2019; Camacho et al., 2020).

Mackey (2008) addresses the shortcoming of projection deflation as follows. Let $\boldsymbol{q}_1 = \boldsymbol{v}_1$ and $\boldsymbol{q}_k = (\mathbf{I}_P - \mathcal{P}_k)\boldsymbol{v}_k$, where $\mathcal{P}_k$ is the projection onto the space spanned by $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$. Mackey (2008) suggests that we rewrite (9) to

$$\mathbf{X}_{k+1} = \mathbf{X}_k(\mathbf{I}_P - \boldsymbol{q}_k \boldsymbol{q}_k^\top), \tag{10}$$

as this expression eliminates the variance in the data captured by the first k principal components.

In contrast to Hotelling's deflation and Schur complement deflation, it is slightly more difficult to implement Mackey's generalized deflation method in the PMD framework. That is, we can not simply only replace line 4 of Algorithm 2 with (10). Namely, we also need to update $\boldsymbol{q}_k$ in each step. To do this, we use the fact that $\boldsymbol{q}_1, \dots, \boldsymbol{q}_k$ form an orthonormal basis for the space spanned by the loading vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$, and thus $\mathbf{I}_P - \mathcal{P}_k = \mathbf{I}_P - \sum_{l=1}^{k} \boldsymbol{q}_l \boldsymbol{q}_l^\top = \prod_{l=1}^{k} (\mathbf{I}_P - \boldsymbol{q}_l \boldsymbol{q}_l^\top)$. This allows us to summarize the generalized deflation method as follows in Algorithm 3.

Algorithm 3: PMD(ℓ_1, ℓ_1) rank-K approximation using Mackey's generalized deflation method

```
  Input: $\mathbf{X}$, $c_1$, $c_2$
  Output: $d_k$, $\boldsymbol{u}_k$, $\boldsymbol{v}_k$ for $k = 1, \dots, K$
1 Let $\mathbf{X}_1 = \mathbf{X}$.
2 Let $\mathbf{B}_1 = \mathbf{I}_P$.
3 for $k = 1, \dots, K$ do
4     Find $d_k, \boldsymbol{u}_k$ and $\boldsymbol{v}_k$ by applying Algorithm 1 to $\mathbf{X}_k$.
5     $\boldsymbol{q}_k = \mathbf{B}_k \boldsymbol{v}_k$.
6     $\mathbf{X}_{k+1} = \mathbf{X}_k(\mathbf{I}_P - \boldsymbol{q}_k \boldsymbol{q}_k^\top)$.
7     $\mathbf{B}_{k+1} = \mathbf{B}_k(\mathbf{I}_P - \boldsymbol{q}_k \boldsymbol{q}_k^\top)$.
```

We refer to the variant of PMD using Mackey's generalized deflation method as PMD-GD.

3.4 Evaluation metrics

We compare the performance of the PMD-HD, PMD-SC and PMD-GD by applying these methods to simulated data and real data, which we describe in Section 4. In this section, we introduce the metrics that we use to compare the three methods. First, in Section 3.4.1 we present the formulation used to compute the cumulative proportion of variance explained. Next, in Section 3.4.2 we describe how we calculate the percentage of artifacts that is present in the principal components found by PMD. Finally, in Section 3.4.3 we introduce the cosine similarity.

3.4.1 Cumulative proportion of variance explained

We use the same metric as Witten et al. (2009) to evaluate the performance of PMD when applied to the gene expression data set. This metric is the cumulative proportion of variance explained, which measures how much of the variance in the data the first k principal components are able to explain. The formulation of this metric is introduced by Shen and Huang (2008) and is given by the expression

$$CPVE_k = \frac{\operatorname{tr}(\mathbf{X}_k^\top \mathbf{X}_k)}{\operatorname{tr}(\mathbf{X}^\top \mathbf{X})}, \tag{11}$$

where tr(.) is the trace of a matrix and $\mathbf{X}_k = \mathbf{X}\mathbf{V}_k(\mathbf{V}_k^\top \mathbf{V}_k)^{-1}\mathbf{V}_k^\top$ is the projection of $\mathbf{X}$ onto the subspace spanned by the first k sparse loading vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$.

3.4.2 Percentage of artifacts in the principal components

Let $\mathbf{X}_1 = \mathbf{X}$ and $R(\mathbf{X}_1)$ be the row space of $\mathbf{X}_1$. If we obtain $\mathbf{X}_2$ by deflating $\mathbf{X}_1$ using any of the methods mentioned in Section 3.3, it is possible that $R(\mathbf{X}_2) \nsubseteq R(\mathbf{X}_1)$. Thus, we incorporate new directions in $\mathbf{X}_2$ that are not initially present in the original data matrix $\mathbf{X}_1$ (Camacho et al., 2021). We refer to these new directions as artifacts. These artifacts are potentially harmful, because the principal component obtained from $\mathbf{X}_2$ may capture some of the variance pertaining to the artifacts, leading to interpretations of the data that are incorrect.

Although artifacts may be present in a deflated matrix, the derived principal components are not necessarily affected by them. Camacho et al. (2021) propose a metric which measures the percentage of artifacts in principal component $k+1$. We can compute this metric using

$$VarA_{k+1} = 100 \times \frac{\mathrm{tr}(\mathbf{O}_{k+1}^\top \mathbf{O}_{k+1})}{\mathrm{tr}(d_{k+1}^2 \boldsymbol{v}_{k+1}\boldsymbol{u}_{k+1}^\top \boldsymbol{u}_{k+1}\boldsymbol{v}_{k+1}^\top)}, \tag{12}$$

where $\mathbf{O}_{k+1} = (\mathbf{I}_P - \mathbf{X}_1^\top(\mathbf{X}_1^\top)^+)(\mathbf{X}_k^\top(\mathbf{X}_k^\top)^+)d_{k+1}\boldsymbol{v}_{k+1}\boldsymbol{u}_{k+1}^\top$ and $\mathbf{A}^+$ is the Moore-Penrose inverse of the matrix $\mathbf{A}$. $\mathbf{O}_{k+1}$ measures how much of the variance captured by principal component $k{+}1$, which Camacho et al. (2021) represent as $d_{k+1}\boldsymbol{v}_{k+1}\boldsymbol{u}_{k+1}^\top$, lies in $R(\mathbf{X}_k)$, denoted as $(\mathbf{X}_k^\top(\mathbf{X}_k^\top)^+)$, and not in $R(\mathbf{X}_1)$, represented by $(\mathbf{I}_P - \mathbf{X}_1^\top(\mathbf{X}_1^\top)^+)$. In other words, we can interpret $\mathbf{O}_{k+1}$ as the amount of variance captured by principal component $k+1$ that can not be attributed to the original data in $\mathbf{X}_1$. We use (12) to assess to what extent the deflation techniques from Section 3.3 introduce artifacts, as interpretation of the data and the principal components is an important aspect of sparse PCA. We apply this metric to the gene expression data only.

3.4.3 Cosine similarity

Since we specify the structure of the simulated data, denote $\boldsymbol{v}_k$ as the true k'th sparse loading vector and $\hat{\boldsymbol{v}}_k$ as the k'th sparse loading vector estimated by PMD. Similarly to Camacho et al. (2021), we use the absolute value of the cosine similarity between $\boldsymbol{v}_k$ and $\hat{\boldsymbol{v}}_k$ to measure how well the variants of PMD are able to recover the true sparse loading vectors. We can write this metric as

$$Similarity(\boldsymbol{v}_k, \hat{\boldsymbol{v}}_k) = \left| \frac{\boldsymbol{v}_k \cdot \hat{\boldsymbol{v}}_k}{||\boldsymbol{v}_k||_2 ||\hat{\boldsymbol{v}}_k||_2} \right|. \tag{13}$$

The absolute value of the cosine similarity takes a value between 0 and 1. If $Similarity(\boldsymbol{v}_k, \hat{\boldsymbol{v}}_k) = 0$, then PMD completely fails to recover the true sparse loading vector, while $Similarity(\boldsymbol{v}_k, \hat{\boldsymbol{v}}_k) =$

1 indicates that PMD is able to fully recover the sparse loading vector.

As we discuss in Section 4.1.2, using the same procedure we generate multiple data matrices for which we try to recover the true sparse loading vectors. Suppose that we create n data matrices $\mathbf{X}_1, \ldots, \mathbf{X}_n$. For $i = 1, \ldots, n$, let $\boldsymbol{v}_{k,i}$ be the true k'th sparse loading vector corresponding to $\mathbf{X}_i$ and $\hat{\boldsymbol{v}}_{k,i}$ be the k'th sparse loading vector corresponding to $\mathbf{X}_i$ estimated by PMD. Then, to assess how well PMD is able to recover the structure of $\boldsymbol{v}_k$, we use the following expression to compute the average of the cosine similarities over the n data matrices:

$$AvgSim(\boldsymbol{v}_k, \hat{\boldsymbol{v}}_k) = \frac{1}{n} \sum_{i=1}^{n} Similarity(\boldsymbol{v}_{k,i}, \hat{\boldsymbol{v}}_{k,i}). \tag{14}$$

4 Data

We use Section 4 to describe the data we use for our research. In Section 4.1, we explain how we obtain the simulated data for an example and the comparative study. Then, in Section 4.2 we briefly describe the gene expression data set.

4.1 Simulated data

4.1.1 Simulated data for the example

Before we perform the comparative study, we demonstrate PMD on a simulated example. We replicate the simulation performed by Witten et al. (2009). This example involves simulated comparative genomic hybridization (CGH) data. These data measure genomic DNA copy number changes, which are key genetic events in the growth of many cancers (Lingjærde et al., 2005). According to Witten et al. (2009), we can characterize some cancers in the data by regions of chromosomal gain or loss.

Suppose that the data matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$ contains $N = 12$ CGH samples and $P = 1000$ copy number measurements on CGH spots on a single chromosome. Then, we generate the CGH data for $\mathbf{X}$ utilizing the procedure described by Witten et al. (2009):

1. $X_{ij} \sim N(1, 1)$ for $i = 1, \ldots, 5$ and $j = 100, \ldots, 500$.

2. $X_{ij} \sim N(0, 1)$ for $i = 6, \ldots, 12$ and $j = 1, \ldots, 99, 501, \ldots, 1000$.

Here, X_{ij} is the element present on the i'th row and j'th column of $\mathbf{X}$. We generate the data such that five of the twelve samples contain a region of gain on the spots 100 to 500. We can clearly see this in Figure 1a and 1b, where we show the true vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ from the rank-1 approximation of $\mathbf{X}$ which correspond to the samples and the chromosomal regions, respectively.

By optimizing the PMD(ℓ_1, ℓ_1) criterion, we try to uncover these regions of gain in the simulated data.

4.1.2 Simulated data for the comparative study

As mentioned previously, we extend the research of Witten et al. (2009) by incorporating the deflation schemes from Section 3.3 within the PMD framework and assessing the performance of these deflation methods. To perform this comparative study, we generate random data following the procedure proposed by Shen and Huang (2008) and Journée et al. (2010). Let $\mathbf{\Sigma} = \mathbf{V}\mathbf{S}\mathbf{V}^\top$ be the eigendecomposition of the covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{P \times P}$ of the data matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$, where the columns $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_P$ of $\mathbf{V} \in \mathbb{R}^{P \times P}$ are the eigenvectors of $\mathbf{\Sigma}$ and the elements $s_{11}, \ldots, s_{PP}$ of the diagonal matrix $\mathbf{S} \in \mathbb{R}^{P \times P}$ are the eigenvalues corresponding to the eigenvectors. The elements of $\mathbf{S}$ are ordered in decreasing order i.e. $s_{11} \geq \cdots \geq s_{PP} \geq 0$. Furthermore, we assume that the first $q < P$ columns of $\mathbf{V}$ are manually specified to sparse and orthonormal. For this simulation study, we consider a setup with $q = 5$ to be able to make a good assessment on the comparative performance of the deflation methods. We vary N across the values 50, 100, 200 and 500 and we set $P = 100$.

To begin, we form a full-rank matrix $\mathbf{V}^* = [\boldsymbol{v}_1 | \ldots | \boldsymbol{v}_q | \boldsymbol{v}^*_{q+1} | \ldots | \boldsymbol{v}^*_P]$. As mentioned previously, we manually specify the first $q = 5$ eigenvectors to be sparse and orthonormal. Denote v_{ij} as element i of the eigenvector $\boldsymbol{v}_j$. We define the elements of the first q eigenvectors as follows:

$$v_{ij} = \begin{cases} \frac{1}{\sqrt{10}} & \text{for } i = 10(j-1), \ldots, 10(j-1)+10 \\ 0 & \text{otherwise} \end{cases}$$

Subsequently, we determine the vectors $\boldsymbol{v}^*_{q+1}, \ldots, \boldsymbol{v}^*_P$ in $\mathbf{V}^*_2$. We randomly draw the elements for these vectors from $U(0, 1)$ until $\mathbf{V}^*$ is a full-rank matrix. Then, we apply the Gram-Schmidt process to $\mathbf{V}^*$ in order to retrieve its QR decomposition. From this QR decomposition, we need the matrix $\mathbf{Q}$ as this is the desired orthogonal matrix $\mathbf{V} = [\boldsymbol{v}_1 | \ldots | \boldsymbol{v}_q | \boldsymbol{v}_{q+1} | \ldots | \boldsymbol{v}_P]$. To construct the matrix $\mathbf{S}$ we specify the eigenvalues of the first $q = 5$ sparse eigenvectors to be 600, 500, 400, 300 and 200, respectively. Similarly to Journée et al. (2010), we set the eigenvalues of the remaining $P - q$ eigenvectors to 1. Now, we can use the expression $\mathbf{V}\mathbf{S}\mathbf{V}^\top$ to construct $\mathbf{\Sigma}$. We use the fact that $\mathbf{X} \sim N_P(\mathbf{0}, \mathbf{\Sigma})$ to generate the data. We repeat the data generating procedure 100 times for each value of N.

4.2 Real data

We apply the variants of PMD to same real gene expression data set used by Witten et al. (2009).[1] This data set contains a total of $P = 19672$ gene expressions and $N = 89$ samples. Similarly to Witten et al. (2009), we only consider 5% of the genes with the highest variance due to computational reasons.

5 Results

In Section 3, we report our findings. Specifically, in Section 5.1 we present the results after applying the three variants of PMD to the simulated data sets described in the previous section. Subsequently, in Section 5.2 we describe and discuss the findings when we apply the variations of PMD to the gene expression data set.

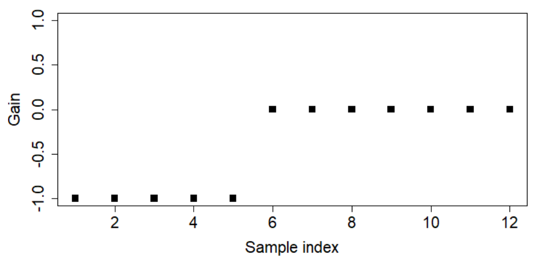
5.1 Simulated data

5.1.1 Example

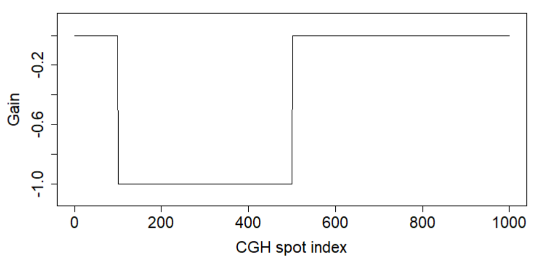
We optimize the PMD(ℓ_1, ℓ_1) criterion to find the rank-1 approximation of the matrix $\mathbf{X}$ containing the CGH data described in Section 4.1.1. In order to obtain results that closely resemble the ones found in Figure 2 of the paper by Witten et al. (2009), we set $c_1 = 2.80$ and $c_2 = 6.00$. These parameter values ensure that PMD only chooses five samples to exhibit a region of gain while also identifying the region of gain within CGH spot indices 100 to 500.

In Figure 1, we show the true $\boldsymbol{u}$ and $\boldsymbol{v}$ along with the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ found by optimizing the PMD(ℓ_1, ℓ_1) criterion.

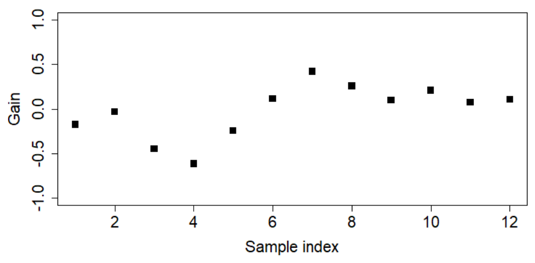
[1]Downloaded from: https://tibshirani.su.domains/PMA on May 10th 2023.



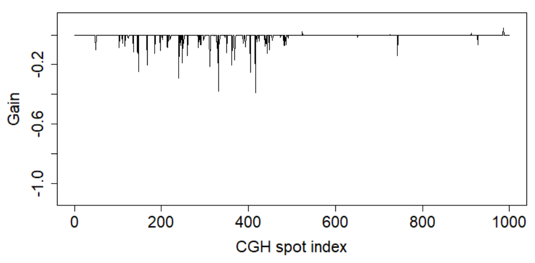
(a) True $\boldsymbol{u}$



(b) True $\boldsymbol{v}$



(c) $\boldsymbol{u}$ found by optimizing PMD(ℓ_1, ℓ_1)



(d) $\boldsymbol{v}$ found by optimizing PMD(ℓ_1, ℓ_1)

Figure 1: $\boldsymbol{u}$ and $\boldsymbol{v}$ from the generative model and PMD

Similarly to Figure 2 in the paper written by Witten et al. (2009), we find that optimizing the PMD(ℓ_1, ℓ_1) criterion does not precisely uncover the samples that have a region of gain and the region of gain itself. Namely, in Figure 1c we see that PMD incorrectly specifies sample 7 to be one of the samples with a region of gain instead of sample 2. Moreover, in Figure 1d we observe that PMD does not smoothly identify the region of gain within spot indices 100 to 500. Specifically, some spots have a much lower gain than others or no gain at all, while we expect all spots to have equal gain like we see in Figure 1b. Witten et al. (2009) state that PMD yields better results if we impose the fused LASSO penalty on $\boldsymbol{v}$ instead of an ℓ_1 penalty. However, we do not discuss this new penalty as it falls outside the scope of this paper.

5.1.2 Comparative study

For all three variants of PMD, we use $c_2 = 3.00$ as this value achieves the true level of sparsity for these simulated data. Specifically, this value of c_2 allows for all variants of PMD to generate sparse loading vectors which each have exactly 10 nonzero elements similarly to the true sparse loading vectors. In Table 1, we report the average cosine similarities between the actual loading vector and the loading vector obtained by one of the three variants of PMD. As mentioned previously, these average cosine similarities are computed over 100 data matrices. Furthermore, we highlight the variants of PMD with the highest cosine similarity for a given loading vector. Lastly, we denote the standard errors in parenthesis.

Table 1: Average cosine similarities between the actual loading vector and the loading vector obtained by one of the three variants of PMD over 100 data matrices

$N = 25$					
Method	$AvgSim(\boldsymbol{v}_1, \hat{\boldsymbol{v}}_1)$	$AvgSim(\boldsymbol{v}_2, \hat{\boldsymbol{v}}_2)$	$AvgSim(\boldsymbol{v}_3, \hat{\boldsymbol{v}}_3)$	$AvgSim(\boldsymbol{v}_4, \hat{\boldsymbol{v}}_4)$	$AvgSim(\boldsymbol{v}_5, \hat{\boldsymbol{v}}_5)$
PMD-HD	**0.5313** (0.4733)	0.3795 (0.4671)	0.4079 (0.4720)	0.4079 (0.4720)	**0.7149** (0.4063)
PMD-SC	**0.5313** (0.4733)	**0.3984** (0.4706)	**0.4364** (0.4752)	**0.4459** (0.4759)	0.7020 (0.4182)
PMD-GD	**0.5313** (0.4733)	0.3795 (0.4671)	0.4079 (0.4720)	0.4079 (0.4720)	**0.7149** (0.4063)
$N = 50$					
PMD-HD	**0.7400** (0.3950)	0.5597 (0.4689)	**0.6167** (0.4548)	0.7400 (0.3950)	0.8823 (0.2433)
PMD-SC	**0.7400** (0.3950)	**0.5787** (0.4650)	**0.6167** (0.4548)	**0.7495** (0.3883)	**0.9012** (0.2078)
PMD-GD	**0.7400** (0.3950)	0.5597 (0.4689)	**0.6167** (0.4548)	0.7400 (0.3950)	0.8823 (0.2433)
$N = 100$					
PMD-HD	**0.7495** (0.3884)	**0.6451** (0.4448)	0.7495 (0.3884)	0.8633 (0.2729)	**0.9392** (0.095)
PMD-SC	**0.7495** (0.3884)	**0.6451** (0.4448)	**0.7684** (0.3740)	**0.8728** (0.2587)	**0.9392** (0.095)
PMD-GD	**0.7495** (0.3884)	**0.6451** (0.4448)	0.7495 (0.3884)	0.8633 (0.2729)	**0.9392** (0.095)
$N = 200$					
PMD-HD	**0.8254** (0.3206)	**0.7684** (0.3740)	**0.8633** (0.2786)	**0.9297** (0.1335)	**0.9487** (1.4970×10^{-16})
PMD-SC	**0.8254** (0.3206)	0.7589 (0.3814)	0.8443 (0.2983)	0.9202 (0.1626)	**0.9487** (1.3666×10^{-16}
PMD-GD	**0.8254** (0.3206)	**0.7684** (0.3740)	**0.8633** (0.2786)	**0.9297** (0.1335)	**0.9487** (1.5701×10^{-16})
$N = 500$					
PMD-HD	**0.9012** (0.2078)	**0.8918** (0.2643)	**0.9392** (0.0949)	**0.9487** (2.7671×10^{-16})	**0.9487** (2.2840×10^{-16})
PMD-SC	**0.9012** (0.2078)	**0.8918** (0.2264)	**0.9392** (0.0949)	**0.9487** (2.6001×10^{-16})	**0.9487** (2.0994×10^{-16})
PMD-GD	**0.9012** (0.2078)	**0.8918** (0.2643)	**0.9392** (0.0949)	**0.9487** (2.4035×10^{-16})	**0.9487** (2.2428×10^{-16})

When $N = 25$, we observe that PMD-SC has a higher average cosine similarity for the components 2 to 4. Only for the fifth component are PMD-GD and PMD-HD able to outperform PMD-SC. This implies that overall PMD-SC is able to recover the sparse loading vectors the best for this particular value of N.

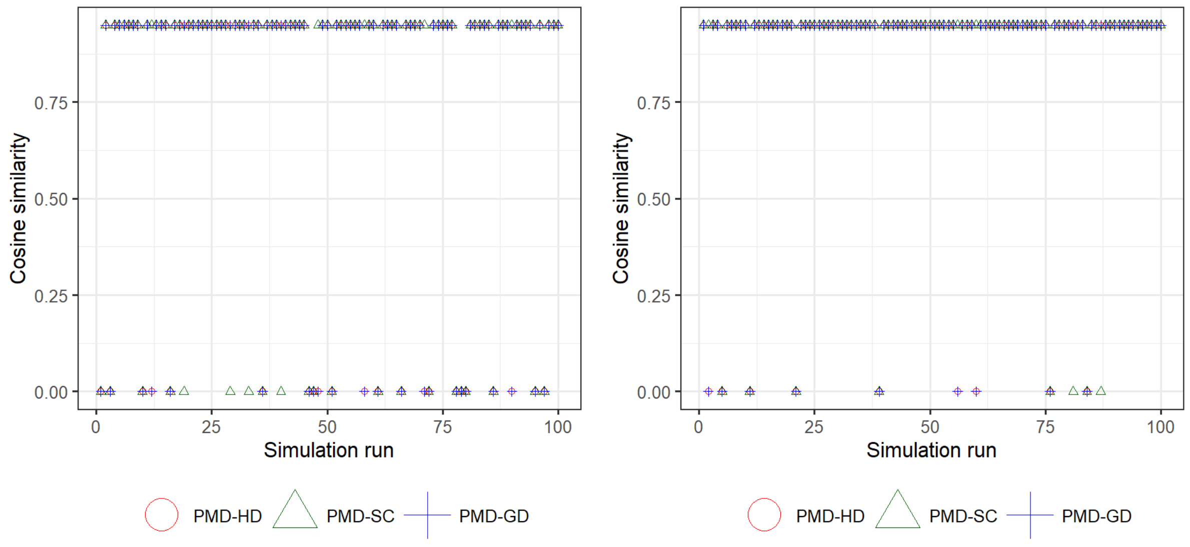
We arrive at the same conclusion for $N = 50$ and $N = 100$. Namely, if $N = 50$ we see that PMD-SC has a higher average cosine similarity for the second, fourth and fifth component. For the remaining components, all three variants of PMD have exactly the same value for the average cosine similarity. When $N = 100$, PMD-SC is slightly less dominant as it only has a higher average cosine similarity for the third and fourth component. However, none of the other variants are able to outperform PMD-SC.

Nevertheless, we are not able to draw the same conclusion when $N = 200$. Namely, we observe that both PMD-HD and PMD-GD manage to outperform PMD-SC in recovering the sparse loading vectors for components 2 to 4. For the other components, all three variants of PMD have exactly the same average cosine similarity.

All the variants of PMD on average perform equally when $N = 500$. Moreover, we see a significant increase in the average cosine similarities for all components compared to $N = 50$. Hence, we can infer from this observation that PMD is better able to recover the sparse loading vectors on average as N increases.

Interestingly, we notice that PMD-HD and PMD-GD share the same value for the average cosine similarity for each component across all values of N. This could highly suggest that

PMD-HD and PMD-GD are equivalent when trying to recover the sparse loading vectors for these particular data. To further investigate this notion, in Figure 2 we plot the individual cosine similarities across all runs of the simulations for the fourth principal component when $N = 50$ and $N = 100$.



(a) $N = 50$ (b) $N = 100$

Figure 2: Individual cosine similarities between $\boldsymbol{v}_4$ and $\hat{\boldsymbol{v}}_4$ for all 100 data matrices

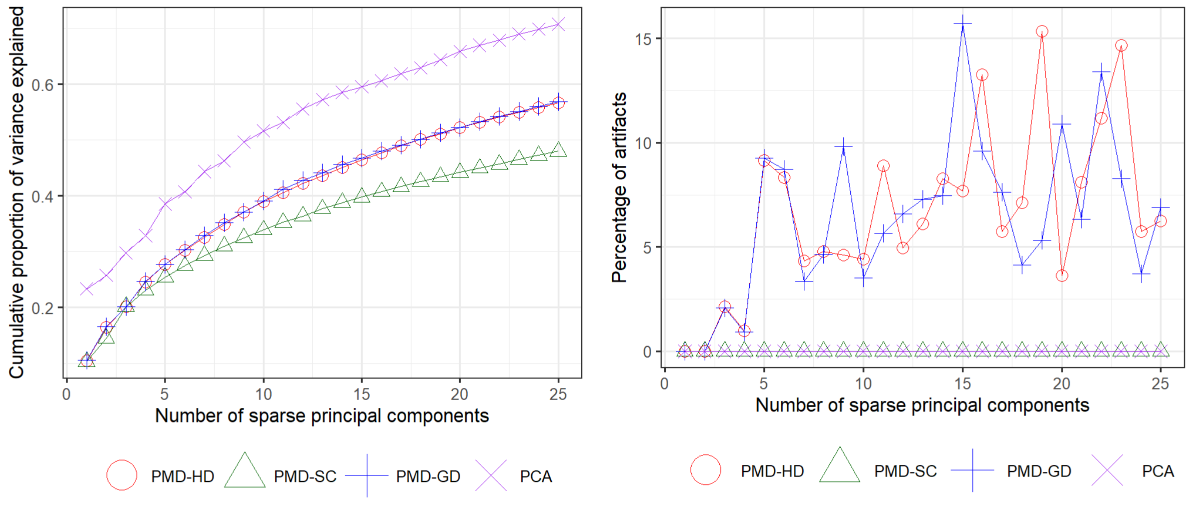
From this figure, we see that, indeed, PMD-HD and PMD-GD have equal values for the cosine similarity. One possible cause for this could be the structure of the sparse loading vectors. Namely, the true sparse loading vectors are specified in a way that the non-zero elements do not overlap. Thus, PMD-HD and PMD-GD may have differing performances when the structure of the loading vectors changes.

All in all, we conclude that no single deflation scheme is the best at recovering the true sparse loading vectors. For these particular data, we observe that PMD-SC is able to recover the sparse loading vectors the best when $N \leq P$. However, when $P > N$, both PMD-HD and PMD-GD outperform PMD-SC. For a value of N that is large enough, all deflation methods perform equally.

5.2 Real data

When applying PMD to the gene expression data set, Witten et al. (2009) choose a value for c_2 such that the average number of nonzero elements in the sparse loading vectors is approximately 195. For the methods PMD-HD, PMD-SC and PMD-GD, we find the values 9.55, 9.42, and 9.60, respectively.

In Figure 3, we show the cumulative proportion of variance explained and the percentage of artifact corresponding to the first 25 principal components found by PCA and the three variants of PMD.



(a) Proportion of variance explained

(b) Percentage of artifacts

Figure 3: Cumulative proportion of variance explained and the percentage of artifacts in the loading vectors corresponding to the three variants of PMD

We find that the first 25 principal components found by PMD-HD and PMD-GD manage to explain 56.61% and 56.90% of the variance in the data, respectively. In contrast, the first 25 principal components obtained by PMD-SC only capture 47.99% of the variance. Therefore, both PMD-HD and PMD-GD outperform PMD-SC in terms of capturing the variability in the data. Nevertheless, none of the variants of PMD manage to explain more variance in the data than ordinary PCA, since the first 25 principal components found by the latter method explain 70.7% of the variance in the data. We expect this since the sparsity constraints imposed on the loading vectors result in some of the information captured by the principal components to be lost, as discussed in Section 1.

Furthermore, in Figure 3a we see that overall PMD-HD and PMD-GD have a very comparable performance, though PMD-GD does explain slightly more of the variance in the data for some of the components compared to PMD-HD. This observation suggests that PMD-GD is the most preferred method in terms of capturing the variability in these particular data. However, PMD-HD is also a good option if one wishes to capture as much variance in these data as possible using PMD, especially when we consider that it is easier to implement Hotelling's deflation scheme in the PMD framework than Mackey's generalized deflation method.

When we look at Figure 3b, we see that the principal components found by PMD-SC are not affected by artifacts, whereas the components obtained by PMD-HD and PMD-GD do contain artifacts. Nonetheless, the percentage of artifacts in the components found by PMD-HD and

PMD-GD is fairly low. Namely, the average percentage of artifacts contained in the principal components of PMD-HD and PMD-GD is only 6.63% and 6.45%, respectively. Moreover, none of the percentages ever go above 15.71%. According to Camacho et al. (2021), these percentages are considered very low and should not hamper the interpretation of the sparse principal components too much.

Thus, if we take into account the low average percentage of artifacts for PMD-GD along with its good comparative performance in terms of explaining the variance in these data, then we can conclude that for these gene expression data Mackey's generalized deflation method seems to be the most preferred method. Though, one could make a case for Hotelling's deflation scheme, as it is easier to implement than Mackey's generalized deflation method and the performance of PMD-HD is very comparable to PMD-GD.

6 Conclusion & discussion

In this paper, we extend the research done by Witten et al. (2009) by assessing the comparative performance of three different deflation schemes when applied to the PMD framework —a framework used for deriving sparse principal components. Specifically, we compare the performance of Hotelling's deflation, Schur complement deflation and Mackey's generalized deflation method. We perform this study by first replicating part of the research of Witten et al. (2009). Namely, we apply the corresponding three variants of PMD to the gene expression data set used by Witten et al. (2009). Subsequently, we compare the different deflation schemes in terms of how well PMD is able the capture the variability in the data and to what extent the interpretation of the sparse principal components is hindered by artifacts. To measure these two factors, we compute the cumulative proportion of variance explained and the percentage of artifacts in each principal component, respectively. In our extension to the research done by Witten et al. (2009), we generate simulated data. Utilizing the simulation study, we assess how well PMD is able to recover the structure of the true sparse loading vectors with each of the different deflation methods. We measure the recovery by computing the cosine similarity between the true sparse loading vector and the loading vector found by PMD. We generate the data matrices for different number of observations.

From our comparative analysis, we obtain the following results. To begin, we find that the variant of PMD that is able to recover the sparse loading vectors most optimally is dependent on the number of observations in the simulated data matrix. Namely, when the number of observations is smaller or equal to the number of variables in the data set, we observe that PMD with Schur complement deflation recovers the true sparse loading vectors the best. Though, if there are more observations than variables in the data set, both PMD with Hotelling's deflation and Mackey's generalized deflation method perform equally, and recover the loading vectors much better than PMD with Schur complement deflation. When the number of observations is large enough, all deflation schemes have equal performance.

Additionally, we find that PMD with Mackey's generalized method is able to capture most of the variance in the gene expression data, though PMD with Hotelling's deflation has a very comparable performance. PMD with Schur complement deflation explains the least variability in the data. Even so, PMD with Schur complement deflation introduces no artifacts at all in all of its components, whereas the two other variants of PMD do. However, the average percentage of artifacts introduced by PMD with Hotelling's deflation and Mackey's generalized deflation method is low enough that it does not hinder the interpretation of the principal components too much (Camacho et al., 2021). Hence, Mackey's deflation scheme is the most preferred method if we take into account its high cumulative proportion of variance explained in comparison the other variants of PMD and fairly low average percentage of artifacts in the principal components.

Nevertheless, there are limitations to our research which we address along with suggestions for future research. Firstly, due to time constraints the number of data sets we use is very limited. This does not allow us to draw definitive conclusions or make strong generalizations about the performance of the deflation schemes. Thus, for future research we suggest to apply the three variants of PMD to other data sets in order to obtain a more conclusive result.

Secondly, we believe that our simulation study could be extended to better assess the behaviour and the comparative performance of the deflation methods. In our simulation study, we vary the number of observations. Though, one could consider varying the number of variables in the data set or adjusting the level of sparsity in the loading vectors. Additionally, we specify the sparse loading vectors such that none of the vectors have overlapping non-zero elements. In general, sparse loading vectors may have non-zero elements that are overlapping. Hence, we suggest for a future analysis to consider a simulation study in which the loading vectors have overlapping non-zero elements. One way to achieve this is to consider utilizing Given's rotation.

Finally, when we apply the three variants of PMD to the gene expression data, we keep the value of each variant's respective sparsity controlling parameter constant for all principal components. We fix the values to ensure that the average number of nonzero elements in the loading vectors obtained by each variant of PMD is equal. However, the comparison may be more fair if we specify the sparsity controlling parameter for each principal component separately. In other words, we ensure that the number of nonzero elements in the loading vectors is the same for a specific principal component across all variants of PMD.

References

Allen, G. I. and Weylandt, M. (2019). Sparse and functional principal components analysis. In *2019 IEEE Data Science Workshop (DSW)*, pages 11–16.

Camacho, J., Smilde, A. K., Saccenti, E., Westerhuis, J., and Bro, R. (2021). All sparse PCA models are wrong, but some are useful. Part ii: Limitations and problems of deflation. *Chemometrics and Intelligent Laboratory Systems*, 208:104212.

Camacho, J., Smilde, A. K., Saccenti, E., and Westerhuis, J. A. (2020). All sparse PCA models are wrong, but some are useful. Part i: computation of scores, residuals and explained variance. *Chemometrics and Intelligent Laboratory Systems*, 196:103907.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Guerra-Urzola, R., Van Deun, K., Vera, J. C., and Sijtsma, K. (2021). A guide for sparse PCA: Model comparison and applications. *Psychometrika*, 86(4):893–919.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Edcuational Psychology*, 24(6):417.

Johnson, R. and Wichern, D. (2014). *Applied Multivariate Statistical Analysis*. Pearson Education Limited.

Jolliffe, I. T., Trendafilov, N. T., and Uddin, M. (2003). A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12(3):531–547.

Journée, M., Nesterov, Y., Richtárik, P., and Sepulchre, R. (2010). Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(2):517–553.

Lai, Z., Xu, Y., Chen, Q., Yang, J., and Zhang, D. (2014). Multilinear sparse principal component analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 25(10):1942–1950.

Lingjærde, O. C., Baumbusch, L. O., Liestøl, K., Glad, I. K., and Børresen-Dale, A.-L. (2005). CGH-explorer: a program for analysis of array-CGH data. *Bioinformatics*, 21(6):821–822.

Liu, H. and Foygel Barber, R. (2020). Between hard and soft thresholding: optimal iterative thresholding algorithms. *Information and Inference: A Journal of the IMA*, 9(4):899–933.

Mackey, L. (2008). Deflation methods for sparse PCA. *Advances in Neural Information Processing Systems*, 21:1017–1024.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Shen, H. and Huang, J. Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034.

Weylandt, M. (2019). Multi-rank sparse and functional PCA manifold optimization and iterative deflation techniques. In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 500–504. IEEE.

Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.

Zou, H. and Xue, L. (2018). A selective overview of sparse principal component analysis. *Proceedings of the IEEE*, 106(8):1311–1320.

A Programming code

We attach a .zip file to this paper which contains the R code used to obtain the results for this study. We provide a short description of all the files contained in the .zip file.

1. **PMD.R** - This script runs the penalized matrix decomposition by Witten et al. (2009) to find a sparse rank-K approximation of a matrix $\mathbf{X}$.

2. **utils.R** - This script contains some of the methods that we need to run **PMD.R**.

3. **PCA.R** - This script runs ordinary PCA.

4. **geneexpression.R** - This script contains the R code for the analysis we perform on the gene expression data.

5. **simulation1.R** - This script generates the data using the procedure described in Section 4.1.1 and applies PMD to these data. Furthermore, the script generates the images for Figure 1.

6. **simulation2.R** - This script generates the data using the procedure described in Section 4.1.2 and applies the three variants of PMD to these data. Moreover, the script obtains the cosine similarities that we report in Table 1.