The Bias Method for Robust Centered Principal Component Analysis

BAOKUN HE*, The University of Texas at Dallas
GUIHONG WAN*, The University of Texas at Dallas
RONG JIN, The University of Texas at Dallas
HAIM SCHWEITZER, The University of Texas at Dallas

Many robust implementations of principal component analysis (PCA) remove outliers from the data and compute the principal components of the remaining data. The robust centered variant requires knowledge of the center of the non-outliers. Unfortunately, the non-outliers center is unknown until after the outliers are determined, and using an inaccurate center may lead to the detection of wrong outliers. We demonstrate this problem in several known robust PCA algorithms. We describe a method that implicitly centers the non-outliers, implemented by appending a constant (bias) value to each data point. This bias method can be used with "black box" robust PCA algorithms by augmenting their input with minimal change to the algorithm itself.

CCS Concepts: • **Computing methodologies** → **Principal component analysis**; • **Mathematics of computing** → **Dimensionality reduction**.

Additional Key Words and Phrases: Robust Principal Component Analysis, RPCA, Outlier Detection, Centering.

ACM Reference Format:
Baokun He, Guihong Wan, Rong Jin, and Haim Schweitzer. 2020. The Bias Method for Robust Centered Principal Component Analysis. 1, 1 (April 2020), 17 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Principal Component Analysis (PCA) is arguably the most widely used dimensionality reduction technique. See, e.g. [2, 8, 13, 22]. We are interested in the following variants of PCA: the centered, contrasted with the uncentered variant, and the robust, contrasted with the non-robust variant. In the centered variant there is a preliminary centering step where the data mean (center) is subtracted from each data point. It is much more popular than the uncentered variant [3, 13]. The robust variant that we consider here, sometimes called Robust Subspace Recovery, identifies some data points as outliers to be removed, and computes the PCA of the remaining data. See, e.g. [11, 18, 19, 23–25], and the recent survey [14].

There are other variants of robust PCA that do not discard outliers entirely. One approach is to use error criteria that are not highly sensitive to outliers. For example one can use the sum of absolute differences as an error criterion instead

*Both authors contributed equally to the research.

Authors' addresses: Baokun He, The University of Texas at Dallas, 800 W. Campbell Road, Richardson, Texas, 75080, Baokun.He@utdallas.edu; Guihong Wan, The University of Texas at Dallas, 800 W. Campbell Road, Richardson, Texas, 75080, Guihong.Wan@utdallas.edu; Rong Jin, The University of Texas at Dallas, 800 W. Campbell Road, Richardson, Texas, 75080, Rong.Jin@utdallas.edu; Haim Schweitzer, The University of Texas at Dallas, 800 W. Campbell Road, Richardson, Texas, 75080, haim@utdallas.edu.

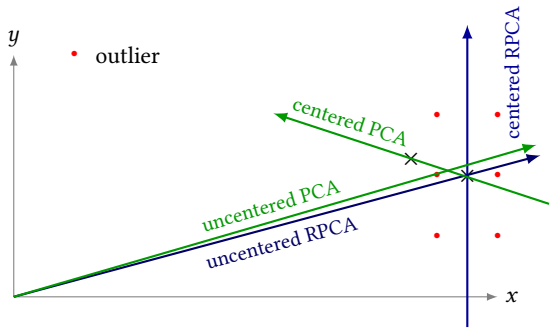


Fig. 1. The direction of the dominant principal component for several PCA variants on a simple dataset of 7 points with one outlier. The PCA directions are computed from the entire data. The RPCA directions are computed from the 6 non-outliers.

of the sum of squared differences as in the classical PCA. See, e.g. [6, 14, 17]. Another approach allows outlier values to be wrong in some coordinates but right in other coordinates. Computing the robust PCA under this assumption can be formulated as partitioning of the data matrix into the sum of two matrices, one low rank and the other sparse. See [4] and the recent surveys [1, 21].

The variants that we consider in this paper are illustrated in Figure 1. The data consists of 7 points, and the direction of the dominant principal component of each variant is shown as an arrow. We use the notation PCA to indicate the non-robust PCA, and RPCA for the corresponding robust variant. Observe that the single outlier has a huge effect on the centered variants, and a much smaller effect on the uncentered variants.

From a computational point of view there is little difference between the centered and the uncentered variants (see Section 2), and some algorithms for computing PCA ignore the centering of the data (e.g., [9, 15]). The situation is different for robust PCA. The problem is that the center should be computed only from the non-outliers, but these outliers are unknown until the algorithm terminates, which leads to a "chicken and egg" problem. Some robust PCA algorithms perform an initial centering of the data, but do not update that center. These include [19, 24, 25]. Such approach is justified under the assumption that removing outliers should not result in a significant change to the center location. Another approach is not to rely on a center, taken by algorithms such as [18, 23] that do not explicitly center the data; other algorithms such as RobPCA of [11] handle the centering as part of the algorithm.

Our main contribution is the surprising observation that appending a large bias value to each data point unifies the centered and the uncentered PCA variants. Specifically, we show that computing uncentered robust PCA from the modified data gives (almost) the same result as computing the centered robust PCA of the unmodified data. This leads to simplified algorithms for centered robust PCA. For example, using the optimal uncentered algorithm of [19] on the modified data gives the best possible centered robust PCA according to natural error criteria discussed in Section 2. To the best of our knowledge this is the first (and currently the only) nontrivial centered algorithm with optimal guarantees for the column outlier model.

A simple example illustrating the bias method is shown in Figure 2, where the data is given by the matrix X, and we assume no outliers. To compute the centered PCA of X one centers the data by subtracting the mean μ from each column, producing the matrix X_c. The centered PCA is obtained by computing the spectral decomposition of $X_c X_c^T$ (the scaled covariance matrix). The principal vectors computed in this way are the columns of the matrix V_c, and the corresponding eigenvalues are λ_c. (See Section 2 for the exact computation.) Using the bias method we append the

$$\text{Data:}\quad X=\begin{pmatrix}1&2&3\\3&0&0\end{pmatrix},\quad \mu=\begin{pmatrix}2\\1\end{pmatrix},\quad X_c=\begin{pmatrix}-1&0&1\\2&-1&-1\end{pmatrix},\quad V_c=\begin{pmatrix}-0.4718&0.8816\\0.8816&0.4718\end{pmatrix},\quad \lambda_c=\begin{pmatrix}7.6055\\0.3944\end{pmatrix}$$

$$\text{Data with added bias:}\quad X_b=\begin{pmatrix}1&2&3\\3&0&0\\100&100&100\end{pmatrix},\quad V_b=\begin{pmatrix}0.01999&\mathbf{-0.4718}&\mathbf{0.8814}\\0.0099&\mathbf{0.8816}&\mathbf{0.4717}\\0.9997&0.0006&-0.0223\end{pmatrix},\quad \lambda_b=\begin{pmatrix}30015\\\mathbf{7.6055}\\\mathbf{0.3942}\end{pmatrix}$$

Fig. 2. A numeric example of computing the centered PCA of X with the bias method. The standard approach computes the mean μ, and then the centered data matrix X_c. The principal vectors V_c are computed as the eigenvectors of $X_c X_c^T$ with the corresponding eigenvalues λ_c. To use the bias method we form X_b by adding a large bias to each column of X. The eigenvectors of $X_b X_b^T$ are V_b, with the corresponding eigenvalues λ_b. Observe that an approximation of V_c is embedded within V_b, and an approximation of λ_c is embedded within λ_b.

value of 100 to each column of X, producing X_b. Its uncentered PCA is V_b[1] , with the corresponding eigenvalues λ_b. It is clear that an accurate approximation of the centered PCA is embedded within V_b and λ_b.

Similar data changes are used to unify algorithms in other fields. In computer graphics one manipulates 2D points such as $p=(x,y)^T$. The rotation of p can be expressed as matrix multiplication but it is not possible to do the same for a translation of p. Moving to homogeneous coordinates by mapping each point p into the 3D vector $p_h=(x,y,1)^T$ unifies translation and rotation, since it can be shown that both 2D translation and 2D rotation can be expressed as 3×3 matrix multiplications by p_h. See, e.g. [12]. Another example is from the study of neural networks. The standard "perceptron" can be viewed as a linear threshold unit with zero threshold, or, alternatively, with an unknown threshold value that must be learned. These two cases lead to two different learning algorithms, but they can be unified by appending a constant bias of 1 to each data point. See, e.g. [16].

The bias method that we describe is similar but not identical to these examples. In our case the appended bias must be large, in the sense that the approximation of the centered PCA is improved when the bias value is increased. We show that this gives a practical algorithm, since "not too large" bias values give "sufficiently accurate" approximations. This statement is quantified in Section 3. To the best of our knowledge the observation that appending bias to the data unifies centered and uncentered PCA is new. (The key idea was previously described by the authors in [10].) While applying it is straightforward, our correctness proofs are quite elaborate.

Converting a robust uncentered PCA algorithm to a robust centered PCA algorithm with the bias method is straightforward and does not require any code change to the algorithm itself. Once the bias is appended to the algorithm input, the outliers computed by the uncentered algorithm are those corresponding to robust centered PCA. Therefore, this conversion retains favorable properties of the uncentered algorithm such as accuracy, sparsity, and efficiency.

Paper organization

The paper is organized as follows: The centered and the robust PCA variants are formally define in Section 2. An error is associated with each variant, enabling a quantitative comparison of different algorithms. The proposed bias method is described in Section 3. Its correctness is proved in the appendix. Experimental results are described in Section 5.

[1] There is a sign ambiguity in calculating eigenvectors, since if v is an eigenvector with λ eigenvalue then so is $-v$. All eigenvectors in Figure 2 were normalized so that their coordinate sum is positive.

2 THE CENTERED AND THE ROBUST VARIANTS

Let $X = (x_1, \ldots, x_n)$ be the data matrix of size $m \times n$, where n is the number of data points and m is the dimension of each point. The PCA computes a linear mapping from the m dimensional x_i to the r dimensional y_i, where $r \leq m$. The inverse of this mapping gives an approximate reconstruction of x_i from y_i. The reconstruction error of x_i is denoted by e_i. As in the classical development of PCA (e.g., [13]) the quality of the mapping can be measured by the average reconstruction error. We proceed to derive explicit formulas of the reconstruction error for the robust / non-robust and the centered / uncentered variants.

The non-robust uncentered variant

Here the dimension reduction mapping is $y_i = V^T x_i$, where V is $m \times r$ with orthogonal columns. The reconstruction error to be minimized by V is given by:

$$e_i(V) = \|VV^T x_i - x_i\|^2, \quad E_u(V) = \frac{1}{n}\sum_{i=1}^{n} e_i(V) \tag{1}$$

The uncentered PCA is obtained by computing V that minimizes $E_u(V)$. It is known (e.g., [13]) that the minimizer V can be taken as the r eigenvectors corresponding to the r largest eigenvalues of $B = XX^T$.

The non-robust centered variant

Let $\mu = \frac{1}{n}\sum_i x_i$ be the data mean. For each i define $x_i^c = x_i - \mu$, and the centered data matrix: $X_c = (x_1^c, \ldots, x_n^c)$. The centered PCA of X is defined as the uncentered PCA of X_c. The dimension reduction mapping is $y_i = V_c^T x_i^c = V_c^T (x_i - \mu)$, and V_c is computed from the eigendecomposition of the matrix $C = X_c X_c^T$. It is possible to compute C directly without centering all the points of X using the following formula: $C = B - n\mu\mu^T$ (e.g., [3]). The reconstruction error is given by:

$$\mu = \frac{1}{n}\sum_i x_i, \quad e_i(V_c) = \|V_c V_c^T (x_i - \mu) - (x_i - \mu)\|^2, \quad E_c(V_c) = \frac{1}{n}\sum_i e_i(V_c) \tag{2}$$

The centered PCA is obtained by computing V_c that minimizes $E_c(V_c)$. It is known (e.g., [13]) that V_c can be taken as the r eigenvectors corresponding to the r largest eigenvalues of the scaled covariance matrix $C = X_c X_c^T$.

The robust uncentered variant

Let $k \leq n$ be the number of outliers that we assume to be known. Let O be a subset of k indices in the range $[1, n]$. In this case the dimension reduction mapping from x_i to y_i is: $y_i = V^T x_i$, and the reconstruction error is given by:

$$e_i(V) = \|VV^T x_i - x_i\|^2, \quad E_{ru}(V, O) = \frac{1}{n-k}\sum_{i \notin O} e_i(V), \quad E_{ru}(V) = \min_{|O|=k} E_{ru}(V, O) \tag{3}$$

For future reference we also define:

$$E_{ru}(r, O) = \min_{V} E_{ru}(V, O), \quad \text{where } V \text{ is } m \times r \text{ with orthogonal columns} \tag{4}$$

Computing V and the corresponding O to minimize $E_{ru}(V, O)$ in (3) is the challenge addressed by robust uncentered PCA algorithms. Observe that it is easy to find the minimizer V when O is given, since it is the uncentered PCA of the non-outliers. It is also easy to compute O when V is given, as the index set of the k points with largest $e_i(V)$. However, alternating these two steps in a "k-means" style converges quickly to a local minimum that can be much worse than the global minimum (see Section 5).

The robust centered variant

The dimension reduction mapping here is the same as the dimension reduction mapping in the robust uncentered variant: $y_i = V_c^T(x_i - \mu)$. However, computing μ is no longer straightforward since it depends on the outliers. The error is given by:

$$\begin{aligned} \mu(O) &= \frac{1}{n-k}\sum_{i\notin O} x_i, \quad e_i(V_c, O) = \|V_c V_c^T(x_i - \mu(O)) - (x_i - \mu(O)\|^2 \\ E_{rc}(V_c, O) &= \frac{1}{n-k}\sum_{i\notin O} e_i(V_c, O), \quad E_{rc}(V_c) = \min_{|O|=k} E_{rc}(V_c, O) \end{aligned} \tag{5}$$

For future reference we also define:

$$E_{rc}(r, O) = \min_{V} E_{rc}(V, O), \quad \text{where } V \text{ is } m \times r \text{ with orthogonal columns} \tag{6}$$

Computing V_c and the corresponding O to minimize $E_{rc}(V_c, O)$ in (5) is the challenge addressed by robust centered PCA algorithms. It appears more difficult than the minimization in (3). In particular, observe that it is not even clear how to compute O when V_c is given. A "k-means" style algorithm would need to compute μ, V_c from O, and then O from μ, V_c (see Section 5). We point out that **our bias method reduces solving (5) to solving (3).**

3 THE BIAS METHOD

In this section we describe the bias method. It can be used to convert any algorithm that computes uncentered PCA into an algorithm that computes centered PCA. It can also be used to convert any algorithm that computes outliers for uncentered PCA into an algorithm that computes outliers for centered PCA. We treat these two cases separately. For the first case let ALG be an algorithm that computes uncentered PCA. Let X be the $m\times n$ data matrix, and let r be the desired number of principal components. The bias method converts ALG into an algorithm that computes the centered PCA of X.

Algorithm 1: The bias method for computing centered PCA.

Goal: compute rank-r centered PCA of X.

Input: X, r, and an algorithm ALG for computing uncentered PCA.

Output: the principal vectors $v_1 \dots v_r$, and the corresponding eigenvalues $\lambda_1 \dots \lambda_r$.

Step 1. Select a large value b. See Section 4 for the explicit formula: $b = 10\|X\|_F$.

Step 2. Append b to each column of X, creating X_b of size $(m+1)\times n$.

Step 3. Run ALG on X_b to compute rank $r+1$ uncentered PCA.

Step 4. Let $\lambda_1^b \dots \lambda_{r+1}^b$ be the eigenvalues computed in Step 3. Then the r eigenvalues of the centered PCA are $\lambda_2^b \dots \lambda_{r+1}^b$.

Step 5. Let $v_1^b \dots v_{r+1}^b$ be the eigenvectors computed in Step 3. Then v_j, the j'th eigenvector of the centered PCA is given by the top m values of v_{j+1}^b.

Clearly, the bias method is not an improvement over the standard algorithm for computing non-robust centered PCA, as it is more costly and less accurate. However, it also works for the robust variants, where the standard approach cannot be applied. For the case of robust PCA let ALG be a robust uncentered PCA algorithm. As discussed in Section 2 the key challenge is to identify the outliers in the data. Thus, we consider the output of ALG to be the k outliers. The robust PCA can be easily calculated once the outliers are determined, as discussed in Section 2. The input of ALG is the data X, the desired rank r, and the number k of outliers to be detected.

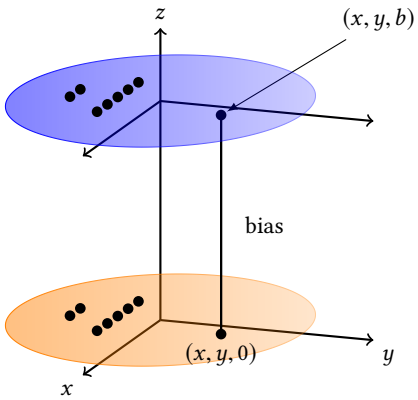


Fig. 3. The bias method applied to compute the centered PCA of the 2D points in the lower plane computes instead the uncentered PCA of the 3D points in the higher plane.

Algorithm 2: The bias method for identifying outliers.

Goal: identify k outliers for computing rank-r robust centered PCA of X.

Input: X, r, k, and an algorithm ALG for identifying outliers for robust uncentered PCA.

Output: A set O of k outliers.

Step 1. Select a large value b. See Section 4 for the explicit formula: $b = 10\|X\|_F$.

Step 2. Append b to each column of X, creating X_b of size $(m+1) \times n$.

Step 3. Run ALG on X_b to compute k outliers for rank $r+1$ uncentered PCA. Return these outliers as the output.

To explain the idea behind the bias method consider the 3-dimensional plot in Figure 3. The 2D rank-1 uncenterd PCA of the points in the lower plane is a line that must pass through the origin, the point $(0, 0)$. This imposes a constraint that leads to a suboptimal answer when there is a better solution that does not pass through $(0, 0)$. By contrast, the 3D rank-2 uncentered PCA of the points in the higher plane is a plane that must pass through the origin, the point $(0, 0, 0)$. The bias method computes the intersection between this PCA plane and the upper plane, and this intersection can be any line in the upper plane. The derivation in the appendix shows that when b is sufficiently large this intersection gives the desired 2D rank-1 centered PCA.

3.1 Correctness

In this section we prove the correctness of the two bias method algorithms of Section 3. Both proofs are corollaries of Theorem A that is proved in the appendix. These proofs require b to be "sufficiently large". In Section 4 we quantify what is meant by "sufficiently large".

Theorem 1: For any desired accuracy of approximating the eigenpairs of the centered PCA with Algorithm 1 there exists $\gamma \geq 0$ such that setting $b \geq \gamma\|X\|_F$ gives the desired accuracy. ($\|X\|_F$ is the Frobenius norm of X.)

The proof follows trivially from Theorem A in the appendix. Observe the following:

1. For correctness it is enough to use γ as the bound instead of $\gamma\|X\|_F$. The reason we use the additional factor $\|X\|_F$ is that according to the derivation in the appendix and the experiments described in this section the term $\|X\|_F$ leads to uniform behavior for different datasets: similar values of γ give similar errors regardless of the dataset.

2. The approximation promised by the theorem can be measured in any norm, since all norms are equivalent in the sense that they are withing a constant of each other (e.g., [7]).
3. In the case of robust PCA let X_O be the (unknown) data matrix of the non-outliers. The expression in the theorem needs not be changed since $\|X_O\|_F \leq \|X\|_F$. Therefore, $b \geq \gamma\|X\|_F$ implies $b \geq \gamma\|X_O\|_F$.
4. The theorem does not take into account round off errors.

The correctness of Algorithm 2 follows from Theorem 2 which shows that for any set of outliers the error of rank r robust centered PCA of X is almost the same as the error of rank r+1 robust uncentered PCA of X_b. We need the following result from linear algebra:

Lemma 1: Let $\sigma_1 \ldots \sigma_m$ be the singular values of the matrix X arranged so that $\sigma_i \geq \sigma_{i+1}$. The reconstruction error of rank r uncentered PCA measured in Frobenius norm is $\sum_{i=r+1}^{m} \sigma_i^2$.

Proof: This well known result gives the approximation error of truncated SVD in Frobenius norm. See for example Theorem 4.18 in [20], where the proof of the more general case of unitarily invariant norms is attributed to Schmidt and Mirsky. An equivalent statement of the theorem is in terms of the eigenvalues of XX^T. If $\lambda_1 \ldots \lambda_m$ are the eigenvalues of XX^T then the reconstruction error is $\sum_{i=r+1}^{m} \lambda_i$, since $\lambda_i = \sigma_i^2$.

Theorem 2: Consider a dataset X, a bias value b, the augmented dataset X_b, and the rank parameter r, as used by Algorithm 2. Let O be an outlier set (not necessarily optimal). Let $E_{rc}(r, O)$ be the reconstruction error of rank r robust centered PCA of X, as defined in (6). Let $E_{ru}(r{+}1, O)$ be the reconstruction error of rank r+1 robust uncentered PCA of X_b, as defined in (4). Then for any ϵ>0 there exists b_ϵ such that $|E_{rc}(r, O) - E_{ru}(r{+}1, O)| \leq \epsilon$ whenever $b \geq b_\epsilon$.

Proof:

1. Let X^O be the clean dataset obtained by removing all the outliers in O from X.
2. Let X_c^O be the result of centering X^O by removing the mean of X^O from each column.
3. Let $\lambda_1 \ldots \lambda_m$ be the eigenvalues of $(X_c^O)(X_c^O)^T$ arranged in decreasing order.
4. Then from Lemma 1: $E_{rc}(r, O) = \sum_{i=r+1}^{m} \lambda_i$.
5. Let X_b^O be the result of appending the bias b to each column of X^O.
6. Let $\lambda_1^b \ldots \lambda_{m+1}^b$ be the eigenvalues of $(X_b^O)(X_b^O)^T$ arranged in decreasing order.
7. Then from Lemma 1: $E_{ru}(r+1, O) = \sum_{i=r+2}^{m+1} \lambda_i^b$.
8. From Theorem A in the appendix $\lambda_i \approx \lambda_{i+1}^b$ for $i = 1 \ldots m$, so that

$$E_{rc}(r, O) = \sum_{i=r+1}^{m} \lambda_i \approx \sum_{i=r+2}^{m+1} \lambda_i^b = E_{ru}(r+1, O)$$

9. From Definition A in the appendix the interpretation of "≈" is precisely the one stated in the theorem. ■

4 PRACTICALITY

To investigate the practicality of the bias method we study experimentally the relationship between the parameter γ in Theorem 1 that controls the value of b and the accuracy. Clearly, selecting b too big will result in round off errors that may reduce the accuracy.

Figure 4 shows the error of computing the eigenvalues as a function of γ on various datasets from the UC Irvine repository [5]. The datasets vary in size from the smallest (Iris) of size 4×150, to the largest (Madelon) of size 500×4400.

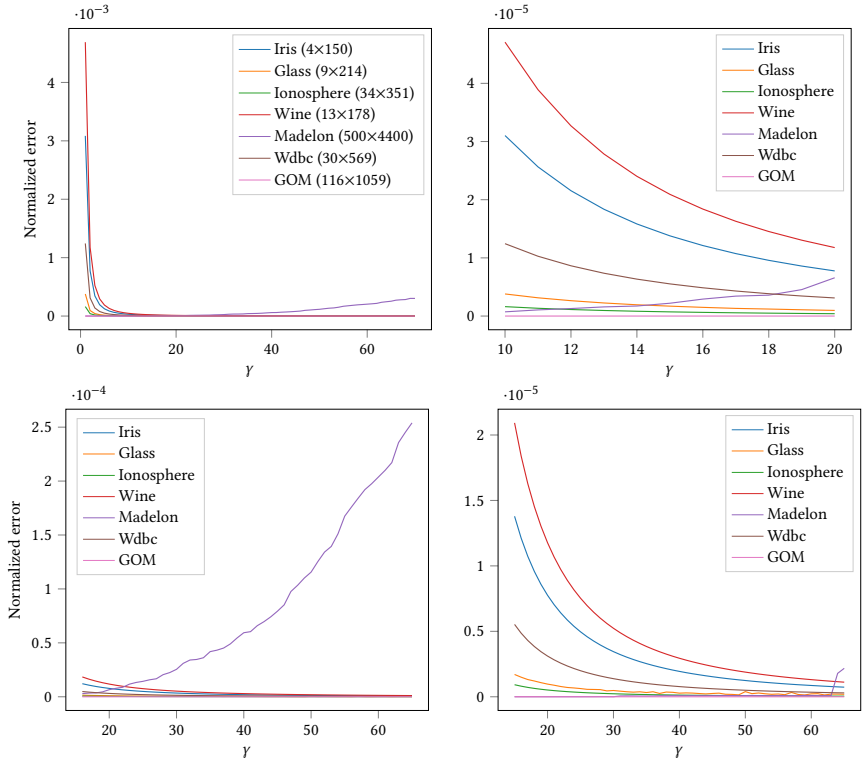


Fig. 4. Error of using the bias method to approximate eigenvalues of centered PCA as a function of γ on various datasets from UC Irvine. Top two panels and bottom left panel: error of estimating all eigenvalues. Bottom right panel: error of estimating only the largest 10 eigenvalues.

The normalized error was computed as follows: Let $\lambda_1 \ldots \lambda_m$ be the eigenvalues of the centered PCA. Let $\lambda_1^b \ldots \lambda_m^b$ be the approximate eigenvalues computed by the bias method. Then:

$$\text{Normalized error } = \frac{\sum_i |\lambda_i - \lambda_i^b|}{\sum_i \lambda_i} = \frac{\sum_i |\lambda_i - \lambda_i^b|}{\|X\|_F^2}$$

Based on similar experiments (some are described in Section 5) we recommend selecting γ in the range $[10, 20]$, which typically gives normalized errors in the range $[10^{-10}, 10^{-5}]$. Our experiments show that when using the standard Python software significant round off errors occur for $b > 10^7$. These round off errors are associated with the QR iterations inside the implementation of the eigendecomposition algorithm used by Python. The experiments described in Figure 4 show a decline in accuracy for increased γ only for the 500×4400 Madelon dataset. Still, the bias method does not break down and the only effect is a slight decrease in accuracy.

Combining the constraint $b<10^7$ with our recommendation of $b = 10\|X\|_F$ gives the bound: $\|X\|_F < 10^6$. The bias method can still be used with matrices of bigger norms although it may not be possible to obtain arbitrarily high accuracy. If one wishes to apply the bias method for larger datasets then there are several options: 1. Scale the dataset. 2. Use higher accuracy spectral decomposition routines. 3. Compute only the top few eigenpairs. The third option is what is typically required by PCA algorithms. The bottom right panel of Figure 4 shows an example of using Option 3,

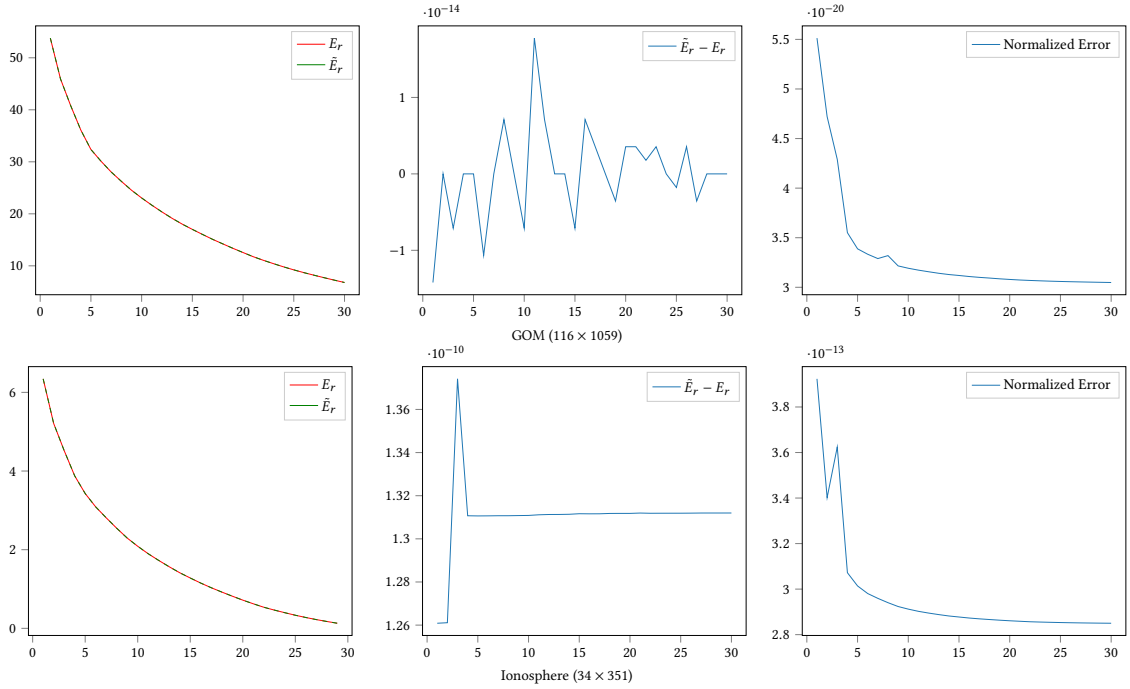


Fig. 5. Error of using the Algorithm 1 to approximate the centered PCA as a function of r.

restricting the eigen decomposition to 10 eigenpairs. Clearly, the results for the Madelon dataset are significantly improved.

5 EXPERIMENTAL RESULTS

In this section we describe experiments with the bias method. In all cases the bias value was taken as $b = 10\|X\|_F$. The experiments were performed on toy datasets and on real datasets from the UCI repository.

5.1 Experiment with Algorithm 1

In Section 4 we described experiments evaluating the accuracy of the eigenvalues computed by Algorithm 1. In this section we describe additional experiments evaluating the accuracy of the eigenvectors computed by Algorithm 1. For a given dataset X and a rank parameter r let $\tilde{V}_c$ be the r eigenvectors computed by Algorithm 1, where ALG is the standard Python routine for computing the eigenvectors of XX^T. Let V_c be the r eigenvectors of the centered PCA variant computed according to Equation (2), where the eigenvectors of $C = X_c X_c^T$ are calculated using the same standard Python routine. Compute $E_r = E_c(V_c)$, and $\tilde{E}_r = E_c(\tilde{V}_c)$, where $E_c()$ is defined in Equation (2). We investigate experimentally the relationship between E_r and $\tilde{E}_r$. Observe that inaccuracies in E_r are the result of inaccuracies in the Python routine for computing eigenvectors, but the inaccuracies in $\tilde{E}_r$ are also influenced by the accuracy of Algorithm 1.

The results of experiments on two datasets from the UCI repository are shown in Figure 5. The panels in the first column show that the two reconstruction errors are almost identical. The panels in the second column show that the

differences between the two reconstruction errors are extremely small. Observe that for GOM dataset, the error $\tilde{E}_r$ is sometimes even smaller than the error E_r. In Section 4 we showed that γ value in the range $[10, 20]$ typically gives normalized errors in the range $[10^{-10}, 10^{-5}]$. Similarly, we define the normalized reconstruction errors as: $\frac{|\tilde{E}_r - E_r|}{E_r}$. The figures in the third column show that the normalized reconstruction errors are much smaller than 10^{-10}.

5.2 Experiment with Algorithm 2

In this section the bias method is used in conjunction with a robust PCA algorithm viewed as a black box. Some algorithms were implemented by us, and for others we use code made available by the authors. The algorithms we experimented with are:

EU. The "Exhaustive Uncentered" algorithm examines all non-outlier subsets of a specified size. The uncentered PCA error is computed for each subset, and the one with the smallest error is returned. This computes optimal uncentered PCA but has exponential running time. (Our implementation.)

EC. The "Exhaustive Centered" algorithm is identical to the EU, except that the centered PCA error is computed for each subset. This computes the optimal centered PCA but has exponential running time. (Our implementation.)

IU. The "Iterative Uncentered" algorithm implements a "k-means" style iteration to minimize the error E_{ru} as given by Equation (3). See the discussion following Equation (4). (Our implementation.)

IC. The "Iterative Centered" algorithm implements a "k-means" style iteration to minimize the error E_{rc} as given by Equation (5). See the discussion following Equation (6). (Our implementation.)

A*opt. The "A*optimal" algorithm is from [19]. It is equivalent to EU, but performs optimal A* search instead of exhaustive search. (Implementation from authors.)

A*greedy. The "A*greedy algorithm is from [19]. It performs non optimal greedy search instead of the exhaustive search of A*optimal, and can be applied to large datasets. (Implementation from the authors.)

HRPCA. The "High-dimensional Robust PCA" algorithm is from [23]. (Implementation from the authors.)

RobPCA. The " Robust PCA" algorithm is from [11]. (Part of R software.)

Cop. The "Coherence Pursuit" algorithm is from [18]. (Implementation from the authors.)

OP. The "Outlier Pursuit" algorithm is from [24, 25]. Unlike the algorithms above it does not take the number of outliers as input but computes it as part of the output. (Implementation from the authors.)

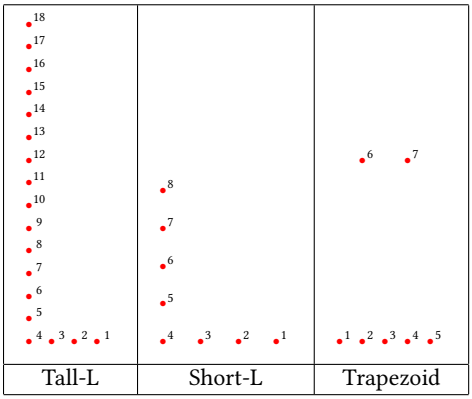
Experiments with toy datasets

We created three toy datasets as shown in Figure 6. The challenge is to compute the first principal component of robust centered PCA with the specified number of outliers. In all three cases the optimal solution has zero error.

The easiest case is Tall-L (3 outliers), where centered PCA is similar to the optimal robust centered PCA. The Short-L dataset (3 outliers) is harder, since centered PCA does not give results similar to the optimal robust centered PCA. The Trapezoid dataset (2 outliers) is the hardest, as centered PCA is orthogonal to the optimal robust centered PCA.

To compute the outliers *without* the bias method we first center each dataset and then apply the algorithms with the rank parameter r=1, and the specified number of outliers. To compute the outliers *with* the bias method we append a bias to the data and then apply the algorithms with the rank parameter r=2, and the specified number of outliers. **We wish to emphasize that the algorithms were not modified. Only the data was modified.**

The outliers computed by the various algorithms are shown in Figure 6. We mark the optimal result with ✓, near optimal with ✓ , and the others with ✗.



The toy datasets.
The three outliers in Tall-L are 1,2,3.
The three outliers in Short-L are 1,2,3.
The two outliers in Trapezoid are 6,7.

	bias	Tall-L	Short-L	Trapezoid	notes
EU	n	1,2,4 ✗	1,4,5 ✗	1,5 ✗	
	y	1,2,3 ✓	1,2,3 ✓	6,7 ✓	3 1
EC	n	1,2,3 ✓	1,2,3 ✓	6,7 ✓	1
	y	n/a	n/a	n/a	2
IU	n	1,2,4 ✗	1,4,5 ✗	1,5 ✗	
	y	1,2,3 ✓	1,4,5 ✗	1,5 ✗	
IC	n	1,2,3 ✓	1,4,5 ✗	1,5 ✗	
	y	n/a	n/a	n/a	2
A*opt	n	1,2,4 ✗	1,4,5 ✗	1,5 ✗	
	y	1,2,3 ✓	1,2,3 ✓	6,7 ✓	3 1
A*greedy	n	1,2,4 ✗	1,4,5 ✗	1,5 ✗	
	y	1,2,3 ✓	1,2,3 ✓	1,2 ✗	
HRPCA	n	18,16,4 ✗	1,6,8 ✗	6,7 ✓	5
	y	2,17,18 ✗	3,6,7 ✗	7,3 ✗	5
RobPCA	n	1,2,3 ✓	1,2 ✓	6,7 ✓	
	y	n/a	n/a	n/a	2
Cop	n	1,9,10 ✗	1,4,5 ✗	1,5 ✗	
	y	1,2,4 ✗	1,7,8 ✗	6,7 ✓	
OP	n	1,17,18 ✗	1,7,8 ✗	6,7 ✓	4
	y	16,17,18 ✗	1,7,8 ✗	6,7 ✓	4

1. The algorithm guarantees optimal robust centered PCA.
2. Internal centering nullifies the bias method.
3. Optimal uncentered becomes optimal centered.
4. We used $\lambda = 3/(7\sqrt{k})$ as recommended in [24].
5. Randomized output. Results change in consecutive runs.

Fig. 6. Results of various algorithms on the the toy datasets.

Some algorithms (EC, IC, RobPCA) center the data internally. This replaces the bias value with 0, and the bias method has no effect on the output. Improvements were obtained for EU, IU, A*opt, A*greedy, and Cop. Only the results of HRPCA became worse.

Experiments with real datasets

First experiment. We describe experiments with datasets from the UCI repository. In each case the algorithm was applied to the data twice: the first time without the bias method, and the second time with the bias method. The results are shown in Table 1. Observe that the bias method almost always improves the errors of IU, A*greedy, and Cop. Algorithms EU, EC, and A*opt were too slow to perform these experiments. Algorithms EC, IC, and RobPCA center the data internally and are not affected by the bias method. We did not include results for HRPCA because its results change in each run.

Second experiment. Additional insight is obtained by plotting the outliers. In each case we run the algorithms to compute outliers for robust PCA with two principal components. These components define a plane in the m dimensional space, and the outliers are points away from that plane. To visualize the results we display the location of each point by projecting it on the plane determined by the first and the third principal components, enabling observations of points

	bias	Iris (4 × 150)		Glass (9 × 214)		Ionosphere (34 × 351)		Wdbc (30 × 569)	
		k:r=20:2	k:r=50:3	k:r=20:2	k:r=50:3	k:r=20:2	k:r=50:10	k:r=20:2	k:r=50:10
IU	n	8.2017	0.4428	126.2292	16.6203	1339.81	225.0202	138,682	**11.6750**
	y	**8.1978**	**0.4309**	**123.0943**	**13.9111**	**1337.23**	**223.7732**	**133,242**	11.8466
A*greedy	n	8.2017	0.4342	115.9149	16.4645	1339.74	220.5231	138,347	10.9778
	y	**8.1591**	**0.4124**	**113.1375**	**13.3456**	**1336.66**	**218.4259**	**132,561**	**10.7827**
Cop	n	11.8950	1.0501	169.5034	26.6959	1548.92	288.5246	438,594	19.4834
	y	**8.3079**	**0.4460**	**114.4764**	**13.9341**	**1336.81**	**225.7976**	**133,141**	**11.1065**

Table 1. Errors of robust PCA algorithms with and without the bias method. The column k:r specifies the number of outliers (k) and the PCA rank (r).

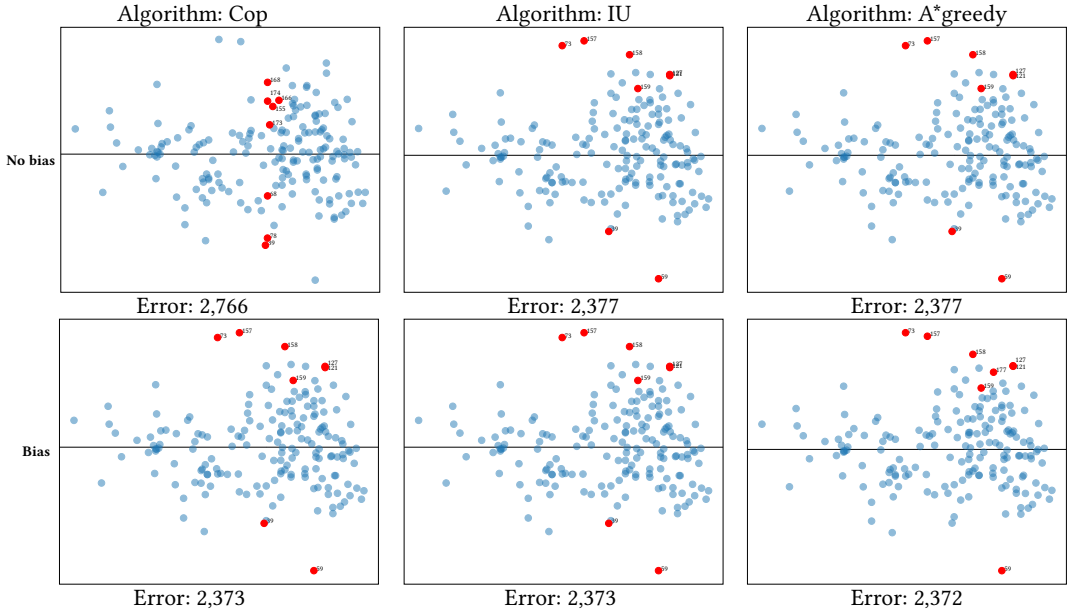


Fig. 7. Dataset: Wine. $k : r = 8 : 2$. Horizontal axis: projections on the first principal component. Vertical axis: projections on the third principal component. Red points are the outliers.

outside of the 2D plane of the first and second principal components. Our intention is to show that the outliers computed with the bias method are more "meaningful", and removing them gives a more accurate model than the removal of those computed without the bias method, even though the data is initially centered. The results obtained are shown in Figure 7. Observe that for the Cop algorithm (first column in Figure 7), with the bias method the outliers appear to be more away from the plane, shown as a thin horizontal line, when compared to the case with no bias. For the IU algorithm (second column in Figure 7), the outliers detected are same regardless of whether the bias is used. For the A*greedy algorithm (third column in Figure 7), using the bias method, there is only one outlier different from when no bias used. The improvements tend to be small.

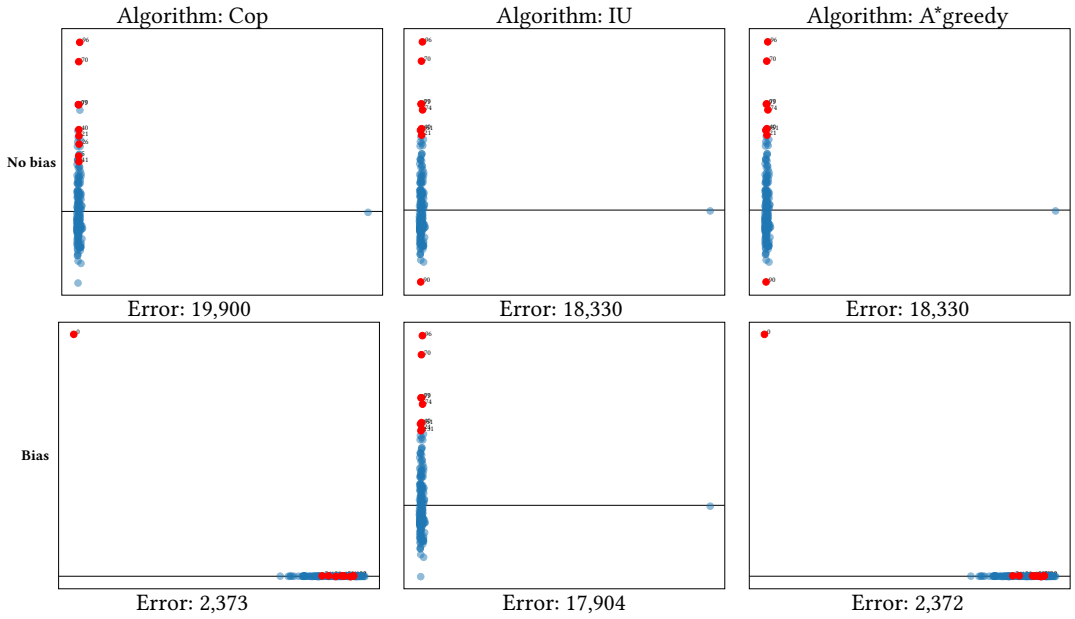


Fig. 8. Dataset: Wine + a noisy point. $k : r = (8+1) : 2$. Horizontal axis: projections on the first principal component. Vertical axis: projections on the third principal component. Red points are the outliers.

Third experiment. As observed in the first two experiments the difference between centered and uncentered PCA on datasets from the UCI repository is small. The reason may be that bad outliers were removed from the datasets before they were put in the repository. To experiment with datasets with significant outliers we create datasets as follows:

> *A single random noisy point of large coordinate values is added to a dataset from the UCI repository. The result is called "the corrupted dataset".*

In this setting we show that robust centered PCA (implemented with the bias method) does significantly better than robust uncentered PCA (the original algorithm). Results are shown in figures 8 and 9. The noisy point is shown in Figure 8 but not in Figure 9, to allow better viewing of the other points. We describe experiments with three algorithms: Cop, IU, and A*greedy. Without the bias method all three algorithms fail to detect the noisy point as an outlier and gave large errors (see first row in Figure 8). As shown in the first row of Figure 9 the detected outliers are not far away from the plane defined by the first two principal components. With the bias method the Cop and A*greedy successfully detect the noisy point as an outlier (see second row in Figure 8). In addition, the computed errors are significantly smaller with bias than without bias. For example, for Cop, using the bias method gives error 2,373, on the other hand, without bias, the error is 19,900 (see first column in Figure 8). As shown in the second row of Figure 9 the detected outliers appear far away from the plane formed by the first two principal components.

6 CONCLUDING REMARKS

This paper describes the bias method for improving algorithms that compute robust PCA by outlier detection. The improvement comes from proper centering that many robust PCA algorithms apparently do suboptimally.

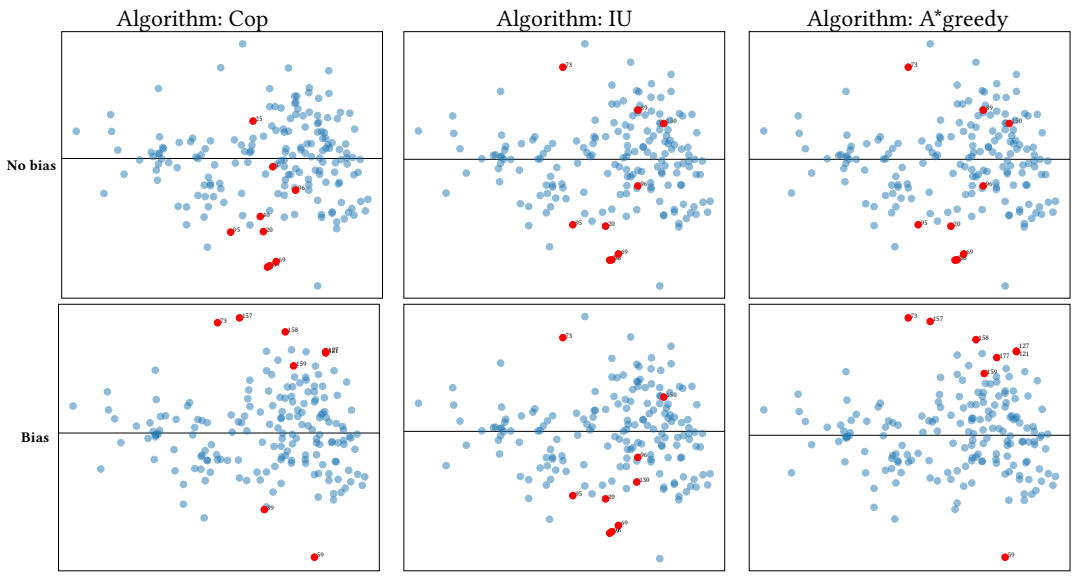


Fig. 9. Dataset: Wine + a noisy point. The noisy data point is not shown to zoom in.

Given an algorithm, a dataset, the desired number of principal components r and the desired number of outliers, the algorithm is not directly performed with the dataset. Instead, a large (bias) value is added to each data point and the algorithm is applied to compute r+1 principal components. The algorithm then identifies outliers in the modified data, and the robust PCA is obtained by computing the centered PCA of the non-outliers.

This technique retains favorable properties of the underlying algorithm, such as running time, accuracy, and the ability to run efficiently on sparse data. In particular, applying the bias method to the algorithm of [19] gives the only nontrivial algorithm we are aware of that computes the optimal robust centered PCA.

We find it somewhat surprising that this highly useful method was not previously discovered. It improves the results of many algorithms, and for other algorithms the output is unchanged. It is very rare that applying it reduces the algorithm accuracy.

We believe that the bias method has the potential of simplifying other computational tasks that involve PCA. One example is one-pass algorithms for computing PCA, as the method converts any uncentered one-pass algorithm into a centered one pass algorithm.

7 APPENDIX: CORRECTNESS OF THE BIAS METHOD

In this appendix we prove that the accuracy of the bias method is increased with the increase of the bias value. Our proof relies on a theorem by Cadima and Jolliffe [3] (given as a corollary to their Theorem 2):

Theorem (Cadima and Jolliffe): Let B be the second moments matrix of the uncentered data, let μ be the data mean, and let C be the covariance matrix. If $\mu/\|\mu\|$ is an eigenvector of B then all its other eigenpairs are also eigenpairs of C.

The main idea behind our approach is the observation that the theorem hypothesis becomes (approximately) true if a large bias is appended to each data point. To analyze the bias method we need the notion of "approximation for sufficiently large values of the bias b". It is defined as follows:

Definition A: We write $p(b) \approx q(b)$ if for any ϵ>0 there is b_ϵ such that $|p(b) - q(b)| < \epsilon$ for all $b > b_\epsilon$. We also write "p approximates q" if $p(b) \approx q(b)$.

The definition above shows that $\lim_{b\to\infty} |p(b) - q(b)| = 0$, and we are interested in the approximations obtained for finite values of b.

Notation. Let $X = (x_1 \dots x_n)$ be the data matrix, let $\mu = \frac{1}{n}\sum_i x_i$ be the data mean, and let $B = \frac{1}{n}\sum_i x_i x_i^T$ be the data second moments matrix. The covariance matrix is given by: $C = \frac{1}{n}\sum_i (x_i - \mu)(x_i - \mu)^T$. Let $\{\lambda_i, u_i\}$ be the eigenpairs of C with $\lambda_i \geq \lambda_{i+1}$. Create $X_b = (x_1^b \dots x_n^b)$ by appending a bias b for each vector: $x_i^b = \begin{pmatrix} x_i \\ b \end{pmatrix}$, so that X_b is $(m+1)\times n$. The column mean of X_b is $\mu_b = \begin{pmatrix} \mu \\ b \end{pmatrix}$. The corresponding $(m+1)\times(m+1)$ matrix of second moments is:

$$B_b = \frac{1}{n}\sum_i^n x_i^b (x_i^b)^T = \begin{pmatrix} B & b\mu \\ b\mu^T & b^2 \end{pmatrix} \tag{7}$$

Simple algebra shows that the corresponding covariance matrix is:

$$C_b = \frac{1}{n}\sum_i^n (x_i^b - \mu_b)(x_i^b - \mu_b)^T = \begin{pmatrix} C & 0 \\ 0 & 0 \end{pmatrix} \tag{8}$$

Let $\{\lambda_i^b, u_i^b\}$ be the eigenpairs of B_b for $i = 1 \dots m$, arranged so that $\lambda_i^b \geq \lambda_{i+1}^b$. Define the m vector v_i and the scalar w_i by: $\begin{pmatrix} v_i \\ w_i \end{pmatrix} = u_i^b$.

Lemma A1: (Recall that $u_1^b = \begin{pmatrix} v_1 \\ w_1 \end{pmatrix}$ is the eigenvector of B_b corresponding to its largest eigenvalue.) For a sufficiently large value of b:

$$\text{Part 1. } w_1 \approx \frac{b}{\sqrt{b^2 + \|\mu\|^2}} \quad \text{Part 2. } v_1 \approx \frac{\mu}{\sqrt{b^2 + \|\mu\|^2}}$$

Proof: From the Courant Fischer theorem (e.g., [7]) the vector v_1 and the scalar w_1 minimize the error $E(v_1, w_1)$ defined as follows:

$$E(v_1, w_1) = \min_{a_i} \sum_i \| \begin{pmatrix} x_i \\ b \end{pmatrix} - a_i \begin{pmatrix} v_1 \\ w_1 \end{pmatrix} \|^2 = \min_{a_i} \sum_i \|x_i - a_i v_1\|^2 + (b - a_i w_1)^2 \tag{9}$$

For sufficiently large value of b the rightmost term dominates and it is minimized by $a_i = b/w_1$. Substituting this in (9) gives:

$$E(v_1, w_1) \approx \sum_i \|x_i - \frac{b}{w_1} v_1\|^2$$

Since v_1 and w_1 form an eigenvector they must satisfy: $\|v_1\|^2 + w_1^2 = 1$. To minimize the above error subject to this constraint we use the method of Lagrange multipliers. The Lagrangian is:

$$L(v_1, w_1, \alpha) = \sum_i \|x_i - \frac{b}{w_1} v_1\|^2 + \alpha(\|v_1\|^2 + w_1^2 - 1) \tag{10}$$

Taking derivatives of (10) with respect to v_1 and equating to 0 gives: $2n(-b/w_1)(\mu - \frac{b}{w_1}v_1) + 2\alpha v_1 = 0$. Therefore, the vectors v_1 and μ are approximately linearly dependent: $v_1 \approx t\mu$. Substituting this in the constraint and solving for t we get:

$$t \approx \frac{\sqrt{1-w_1^2}}{\|\mu\|}, \quad \text{so that:} \quad v_1 \approx t\mu \approx \frac{\sqrt{1-w_1^2}}{\|\mu\|}\mu \tag{11}$$

To prove Part 1 we take derivatives of (10) with respect to w_1 and equate to 0. This gives:

$$2bnv_1^T\mu/w_1^2 - 2nb^2\|v_1\|^2/w_1^3 + 2\alpha w_1 = 0$$

For sufficiently large value of b the right most term can be ignored. After multiplying by w_1^3 and simplifying this gives:

$$w_1 v_1^T \mu \approx b\|v_1\|^2$$

Combining this with (11) we get the following equation in w_1:

$$w_1\|\mu\| \approx b\sqrt{1-w_1^2}.$$

Solving this equation for w_1 gives the formula in Part 1 of the lemma. Substituting that solution in (11) and simplifying gives the formula in Part 2 of the lemma. ■

Theorem A: (Recall that $u_i^b = \begin{pmatrix} v_i \\ w_i \end{pmatrix}$ is the eigenvector of B_b corresponding to the i'th largest eigenvalue λ_i^b.) For all i>1 the following approximations hold: $w_i \approx 0$, and $\{\lambda_i^b, v_i\}$ is approximately the $i-1$'th eigenpair of C.

Proof: From Lemma A1:

$$u_1^b = \begin{pmatrix} v_1 \\ w_1 \end{pmatrix} \approx \frac{1}{\sqrt{b^2 + \|\mu\|^2}} \begin{pmatrix} \mu \\ b \end{pmatrix} = \frac{\mu_b}{\|\mu_b\|}$$

Since this (approximately) satisfies the hypothesis of the Cadima and Jolliffe theorem it follows that for i>1 the eigenpairs of B_b are also (approximately) eigenpairs of C_b. It remains to show that with proper clipping they are also eigenpairs of C. From (8) it follows that if $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ is an eigenvector of C_b where z_1 is an m-vector and z_2 is a scalar then

$$C_b z = \begin{pmatrix} Cz_1 \\ 0 \end{pmatrix} = \lambda z = \begin{pmatrix} \lambda z_1 \\ \lambda z_2 \end{pmatrix}$$

This shows that $Cz_1 = \lambda z_1$ and $z_2 = 0$, which completes the proof. ■

7.1 Estimating the value of b

To obtain an approximation of b we examine the proof of Lemma A1. It requires that the eigenvector written as $\begin{pmatrix} v_1 \\ w_1 \end{pmatrix}$ corresponds to the largest eigenvalue of B_b. As shown, the other eigenvalues of B_b are the same as those of C_b. Therefore, we should have $\lambda_{\max}(B_b) > \lambda_{\max}(C_b)$, where $\lambda_{\max}(B_b)$ is the largest eigenvalue of B_b and $\lambda_{\max}(C_b)$ is the largest eigenvalue of C_b. Indeed, the proof of Lemma A1 shows that this is guaranteed for sufficiently large value of b, but we can derive a more accurate condition:

Lemma A2: Using the notation of Lemma A1, if $v_1^T\mu \geq 0$ and $\|X\|_F \leq w_1 b$ then $\lambda_{\max}(B_b) > \lambda_{\max}(C_b)$.

Proof: With $u_1^b = \begin{pmatrix} v_1 \\ w_1 \end{pmatrix}$ and using (7) we have:

$$\lambda_{\max}(B_b) = (u_1^b)^T B_b u_1^b = v_1^T B v_1 + w_1^2 b^2 + 2 w_1 b v_1^T \mu$$

Dropping the positive first term and the nonnegative third term and using the assumptions of Lemma A2 gives: $\lambda_{\max}(B_b) > w_1^2 b^2 \geq \|X\|_F^2$. The other relations that we need are: $\|X\|_F^2 \geq \|X_c\|_F^2$, $\|X_c\|_F^2 \geq \lambda_{\max}(B_c)$, and $\lambda_{\max}(B_c) = \lambda_{\max}(C)$. The first two are straightforward. In the third $\lambda_{\max}(C)$ is the largest eigenvalue of C, The desired equality was proved part of the proof of the theorem. Concatenating these inequalities and equality gives the desired result. ■

The first condition in Lemme 2 is likely to hold since according to Lemma A1 the vectors v_1, μ have approximately the same direction for sufficiently large b. Therefore, we should select $b \geq \frac{1}{w_1}\|X\|_F$, which suggests that b should be measured in units of $\|X\|_F$, as we do in Section 3.1.

REFERENCES

[1] T. Bouwmans, A. Sobral, S. Javed, S. K. Jung, and E. Zahzah. 2017. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset. *Computer Science Review* 23 (2017), 1–71.

[2] C. Burges. 2010. *Dimension Reduction: A Guided Tour.* Now Publishers Inc., Hanover, MA, USA.

[3] J. Cadima and I. Jolliffe. 2009. On Relationships Between Uncentred and Column-Centred Principal Component Analysis. *Pakistan Journal of Statistics* 25, 4 (10 2009), 473–503.

[4] E. J. Candes, X. Li, Y. Ma, and J. Wright. 2011. Robust principal component analysis? *Journal of the ACM* 58, 3 (May 2011), 11:1–11:37.

[5] A. Frank and A. Asuncion. 2010. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[6] N. Gillis and S. A. Vavasis. 2018. On the Complexity of Robust PCA and l1-Norm Low-Rank Matrix Approximation. *Mathematics of Operations Research* 43, 4 (2018), 1072–1084.

[7] G. H. Golub and C. F. Van-Loan. 2013. *Matrix Computations* (fourth ed.). Johns Hopkins University Press, Baltimore.

[8] V. Gray. 2017. *Principal Component Analysis: Methods, Applications and Technology.* Nova Science Publishers, Incorporated, New York.

[9] N. Halko, P. G. Martinsson, and J. A. Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* 53, 2 (2011), 217–288.

[10] B. He, G. Wan, and H. Schweitzer. 2020. A Bias Trick for Centered Robust Principal Component Analysis. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'20).* AAAI Press, California, in press.

[11] M. Hubert and S. Engelen. 2004. Robust PCA and classification in biosciences. *Bioinformatics* 20, 11 (02 2004), 1728–1736.

[12] J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. Feiner, and K. Akeley. 2013. *Computer Graphics: Principles and Practice* (3 ed.). Addison-Wesley, Upper Saddle River, NJ.

[13] I. T. Jolliffe. 2002. *Principal Component Analysis* (second ed.). Springer-Verlag, New York.

[14] G. Lerman and T. Maunu. 2018. An Overview of Robust Subspace Recovery. *Proc. IEEE* 106, 8 (2018), 1380–1410.

[15] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert. 2017. Algorithm 971: An Implementation of a Randomized Algorithm for Principal Component Analysis. *ACM Trans. Math. Software* 43, 3 (Jan. 2017), 28:1–28:14.

[16] M. Minsky and S. Papert. 1988. *Perceptrons* (expanded ed.). The MIT Press, Cambridge, Massachusetts.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing* (third ed.). Cambridge University Press, Cambridge.

[18] M. Rahmani and G. K. Atia. 2017. Coherence Pursuit: Fast, Simple, and Robust Principal Component Analysis. *IEEE Transactions on Signal Processing* 65, 23 (Dec 2017), 6260–6275. https://doi.org/10.1109/TSP.2017.2749215

[19] S. Shah, B. He, C. Maung, and H. Schweitzer. 2018. Computing Robust Principal Components by A* Search. *International Journal on Artificial Intelligence Tools* 27, 7 (November 2018).

[20] G. W. Stewart and J. Sun. 1990. *Matrix Perturbation Theory.* Academic Press, Boston.

[21] N. Vaswani and P. Narayanamurthy. 2018. Static and dynamic robust PCA and matrix completion: A review. *Proc. IEEE* 106, 8 (2018), 1359–1379.

[22] R. Vidal, Y. Ma, and S. S. Sastry. 2016. *Generalized Principal Component Analysis.* Springer-Verlag, New York.

[23] H. Xu, C. Caramanis, and S. Mannor. 2013. Outlier-Robust PCA: The High-Dimensional Case. *IEEE Transactions on Information Theory* 59, 1 (January 2013), 546–572.

[24] H. Xu, C. Caramanis, and S. Sanghavi. 2010. Robust PCA via outlier pursuit. In *Advances in Neural Information Processing Systems.* MIT Press, Cambridge, MA, USA, 2496–2504.

[25] H. Zhang, Z. Lin, C. Zhang, and E. Y. Chang. 2015. Exact Recoverability of Robust PCA via Outlier Pursuit with Tight Recovery Bounds. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence.* AAAI Press, California, 3143–3149.