A Lookahead Algorithm for Robust Subspace Recovery When Irrelevant Data Abound

Guihong Wan,[1] **Haim Schweitzer**[1]

[1] The University of Texas at Dallas
Guihong.Wan@utdallas.edu, HSchweitzer@utdallas.edu

Abstract

A common task in the analysis of experimental data is to compute an approximate embedding of the data in a low dimensional subspace. The standard algorithm for computing this subspace is the well known Principal Component Analysis (PCA). The PCA can be extended to cases where a small portion of the data is viewed as "outliers" that can be ignored, allowing the remaining data ("inliers") to be more tightly embedded. In this paper we consider the extreme case where the fraction of irrelevant data that should be ignored may be very large. In this case the statistical properties of the inliers may be very different from the statistical properties of the entire data. We demonstrate experimentally that previously proposed algorithms typically do not provide a satisfactory solution. They are either very slow or give low accuracy solutions. We propose a new algorithm that can successfully address this problem. The main idea is to rank each point by looking ahead and evaluating the change in the global PCA error when an inlier is converted into an outlier. We show that this lookahead procedure can be implemented efficiently, producing an accurate algorithm with running time not much above the running time of standard PCA algorithms.

1 Introduction

Principal Component Analysis (PCA) is a common technique for embedding data in a low dimensional subspace. See, e.g., (Jolliffe 2002; Burges 2010; Cabral et al. 2013; Vidal, Ma, and Sastry 2016; Gray 2017). Until recently the task of computing PCA on large data was considered a challenge, but recent algorithmic developments that use randomization enable effective computation of PCA even for very large datasets. See, e.g., (Halko, Martinsson, and Tropp 2011; Halko et al. 2011; Musco and Musco 2015; Li et al. 2017; Clarkson and Woodruff 2017).

Unfortunately, the subspace model computed by PCA is known to be sensitive to outliers. It is also known that it can be made significantly more accurate if it is allowed to ignore a small fraction of the data, considered to be outliers. For recent surveys and overviews that discuss many variants of these robust algorithms see (Vaswani et al. 2018; Lerman and Maunu 2018b; Vaswani and Narayanamurthy 2018). The third reference reviews techniques that consider outliers as partially corrupt observations, while the second reviews techniques that consider each point as either an outlier or an inlier. In this paper we follow the same interpretation of outliers as in the second reference.

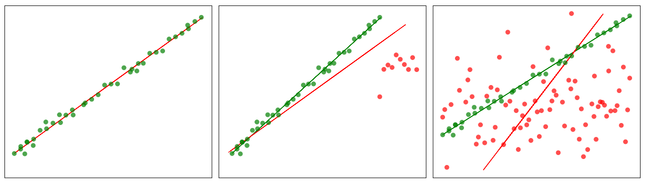
Figure 1: green points are inliers and red points are outliers. Left panel: no outliers, handled by classical PCA. Middle panel: few outliers, handled by Robust PCA / Subspace Recovery algorithms. Right panel: more outliers than inliers, addressed in this paper.

Typically the term "outliers" refers to a small portion of the data that does not follow the same pattern as most of the data. We are interested in situations where a significant portion of the data is to be taken as irrelevant, but we still refer to the irrelevant portion of the data as "outliers", and to the relevant portion as "inliers". However, we observe that the algorithmic challenge of dealing with large fractions of outliers is quite different from handling a small fraction of outliers. The first issue is running time, which may significantly increase with the number of outliers. The second issue is accuracy. One may expect that a small number of outliers will not cause a significant change in the statistical properties of the data. By contrast, if there are many outliers they may mask the statistical properties of the inliers. Indeed, evaluating the performance of previously proposed robust subspace recovery algorithms we observed significant deterioration in accuracy when increasing the outlier fraction.

As an example consider the location of the data mean (center). With a small number of outliers one may assume that the data mean and the inliers mean are very close. But this assumption is not reasonable if the inliers are just a small fraction of the data.

Figure 1 illustrates the case we are interested in. The left panel shows data that can be easily handled by standard PCA, and the red line shows the direction of the first principal component. If PCA is applied to the data in the middle

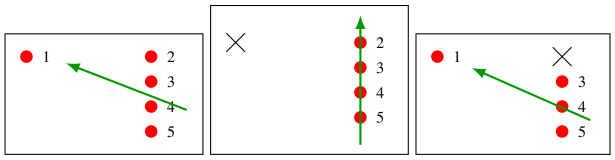

Figure 2: the lookahead idea. Left panel: PCA of the entire data. Middle panel: PCA of the data without point 1, model error is 0. Right panel: PCA of the data without point 2, model error is 0.26.

```
Input: $X$: the $m \times n$ data matrix.
   $r$: the desired number of principal components.
Output: $V$ and $\mu$.
1. $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$.
2. $C = \sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$.
3. Compute the eigendecomposition of $C$ and create $V$ from
   the $r$ eigenvectors with the largest eigenvalues.
```

Figure 3: The basic algorithm for computing the PCA

frame the result is the red line, which does not reveal the approximate linear dependency between the green points. Robust Subspace Recovery (RSR) algorithms will typically produce the correct green line for this data. The only difference between the second and the third frame is in the number of outliers. When this number increases, as shown in the third frame, RSR algorithms may fail to recover the green line as the solution.

Our approach

As in other studies a fundamental part in our approach is to assign a value to each point, indicating how likely it is to be an outlier. We differ from other algorithms by the method in which this value is computed. Observing the points in Figure 1 it appears that the distance of a point from the PCA model is a good candidate for such a measure. Indeed, this distance corresponds to the "reconstruction error" (to be defined in Section 2), a common criterion for evaluating outliers. However, as we show next, the reconstruction error may not be a good choice in situations where the given PCA estimate is inaccurate.

Consider the 5 points in the left panel of Figure 2, where the initial rank-1 PCA that was computed from all 5 points is shown as a green arrow. Even though we expect point 1 to be detected as an outlier, its reconstruction error is 0.1, the smallest among all 5 points. The largest reconstruction error is 2.2, for point 2. The approach we propose is to replace the search for points that have large reconstruction error with the search for points that, when removed, reduce the average reconstruction error of all the other points. In the middle panel of Figure 2 we show the PCA obtained when point 1 is removed. In this case the reconstruction error of all points is 0, so that the model error is 0. In the right panel we show the PCA obtained when point 2 is removed. The model error computed as the average of all reconstruction errors is 0.26. Therefore, point 1 is more likely to be an outlier than point 2, or any other point.

Clearly, performing this lookahead procedure for each point and evaluating its outlier likelihood as the model error obtained "if" it is selected as an outlier is more powerful than relying on the immediate reconstruction error, but it appears to be much more expensive. Our technical contribution is showing that these lookahead error estimates can be computed very efficiently by exploiting special matrix structure. Specifically, we show that the desired errors can be computed from certain eigenvalues, and that these eigenvalues can be computed efficiently by rank-1 modification of centered matrices. Thus, our main contributions are:

- An algorithm for fast computation of eigenvalues of centered matrices by rank 1 modification.
- An algorithm for robust subspace recovery that uses the above fast computation of eigenvalues.
- Experimental evaluation with the above algorithm showing it to be typically more accurate than the current state of the art, with similar running time.

Paper organization

The reconstruction errors of PCA and Robust PCA are defined in Section 2. In particular it is shown that the reconstruction error can be calculated as the sum of certain eigenvalues. Our lookahead algorithm is described in sections 4 and 6. The key idea for reducing the complexity of our approach is to exploit rank-1 updates of centered matrices. In Section 5 we prove that this relation exists. Experimental results are described in Section 7.

2 PCA variants and their error

2.1 Standard PCA

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. For a given $r \le \min\{m, n\}$ the PCA computes a linear mapping from the m dimensional x_i to the r dimensional y_i. The mapping is specified in terms of the matrix V with r orthogonal columns and the mean μ:

$$y_i = V^T(x_i - \mu), \quad x_i \approx V y_i + \mu, \quad \mu = \frac{1}{n}\sum_i x_i \qquad (1)$$

The reconstruction errors for x_i and for the entire data matrix X are defined as follows:

$$e_i = \|x_i - \mu - V y_i\|^2, \; E(X) = \sum_{i=1}^{n} e_i \qquad (2)$$

It is known (e.g., (Jolliffe 2002)) that the matrix V which minimizes E has as its columns the r eigenvectors corresponding to the r largest eigenvalues of the scaled covariance matrix C. This shows that the PCA can be computed by the simple algorithm shown in Figure 3.

A straightforward implementation of this PCA algorithm has running time of $O(m^2 n)$, where the dominant part is Step 2. State-of-the-art algorithms such as (Halko, Martinsson, and Tropp 2011; Halko et al. 2011; Musco and Musco

2015; Li et al. 2017; Clarkson and Woodruff 2017) can reduce the running time to $O(rmn)$. Our results rely heavily on a simple relationship between the eigenvalues of C and the reconstruction error that we state as a theorem.

Theorem 1:

$$E(X) = \sum_{i=r+1}^{n} \lambda_i(C) = \text{trace}\{C\} - \sum_{j=1}^{r} \lambda_j(C) \quad (3)$$

Proof: Let X_c be the "centered" matrix obtained from X by subtracting μ from each column. Then $C = XcXc^T$. Using the relationship between the Frobenius norm and the trace we get the following expression for the error in (2):

$$\begin{aligned} E(X) &= \|X_c - VV^T X_c\|_F^2 = \|(I - VV^T)X_c\|_F^2 \\ &= \text{trace}\{(I - VV^T)X_c X_c^T (I - VV^T)\} \\ &= \text{trace}\{(I - VV^T)C(I - VV^T)\} \\ &= \text{trace}\{C - VV^T C - CVV^T + VV^T CVV^T\} \\ &= \text{trace}\{C\} - \text{trace}\{VV^T C\} - \text{trace}\{CVV^T\} \\ &\quad + \text{trace}\{VV^T CVV^T\} \\ &= \text{trace}\{C\} - \text{trace}\{V^T CV\} \qquad \text{(note1)} \\ &= \text{trace}\{C\} - \sum_{j=1}^{r} \lambda_j(C) \end{aligned}$$

The simplification in the line marked with (note1) was obtained using the cyclic invariant property of the trace which implies that trace$\{VV^T C\}$, trace$\{CVV^T\}$, and trace$\{VV^T CVV^T\}$, are all equal to = trace$\{V^T CV\}$. ■

2.2 Robust PCA / Robust Subspace Recovery

Let O be the outlier subset, containing indices in the range $[1, n]$. Let I be the inlier subset, containing all the indices of points that are not outliers. The analogous formulas to (1) and (2) are:

$$\begin{aligned} &\mu = \frac{1}{n - |O|} \sum_{i \notin O} x_i, \quad y_i = V^T(x_i - \mu) \\ &e_i = \|x_i - \mu - Vy_i\|^2, \quad E(X) = \sum_{i \notin O} e_i \end{aligned} \quad (4)$$

Observe that it is easy to find the minimizer V when O is given, as the PCA of the inliers. It is also easy to compute O when V, μ and the number of outliers are given. Suppose k is the number of outliers, then O is the index set of the k points with largest e_i. Alternating these two steps in a "k-means" style converges quickly to a local minimum which unfortunately can be much worse than the global minimum. This algorithm that we call KMeans-Style Robust PCA is shown in Figure 4. A similar algorithm would get as input V, μ, k instead of O, k and would start with line 4 of the algorithm in Figure 4. The algorithm typically runs a small number of iterations (2-3) and its accuracy is not competitive with the top algorithms. But it is also easy to verify that it can never increase the reconstruction error.

```
  Input:
  $X$: the $m \times n$ data matrix.
  $r$: the desired number of principal components.
  $O$: an estimate to $j$ outliers.
  Output: $V$, $\mu$, and a new subset of $j$ outliers in $O$.
  Initialization: Set current error value to infinity.
1 repeat
2     Set old error value to the value of current error.
3     Compute $V, \mu$ as the rank-$r$ PCA of the inliers.
4     Compute $e_i = ||(x_i - \mu) - VV^T(x_i - \mu)||^2$ for
        all $X$ columns.
5     Replace the columns of $O$ by the $j$ columns of $X$
        with the largest $e_i$.
6     Set new error value = $\sum_{x_i \notin O} e_i$.
7 until new error value = old error value;
```

Figure 4: The KMeans-Style Robust PCA algorithm. As a formula: $O = \text{KM}(O, X, r)$.

Correctness proof for the KMeans-Style algorithm. Let I be the inliers: $I = X \setminus O$. The PCA error can be written as follows, where V is constrained to be of rank r with orthogonal columns:

$$\begin{aligned} E(I) &= \min_V ||(I - \mu\mathbf{1}^T) - VV^T(I - \mu\mathbf{1}^T)||_F^2 \\ &= \min_V \sum_{i \notin O} e_i(V) \end{aligned}$$

where $\mu = \frac{1}{n-|O|} \sum_{i \notin O} x_i$, and $e_i(V) = ||(x_i - \mu) - VV^T(x_i - \mu)||^2$. The error minimized by the algorithm is $E(I)$ subject to the additional constraint $|O| = k$. From the first equality it is clear that Line 3 minimizes $E(I)$ with respect to V when I is fixed. From the second equality it is clear that lines 4-6 minimize $E(I)$ with respect to the selection in O when V is fixed. This shows that $E(I)$ cannot increase during the iterations. It cannot decrease forever since the number of size-k subsets is finite. Therefore the algorithm converges to a local minimum. ■

3 Previous work

Most studies describe algorithms with running time not much worse than computing the PCA. A different approach attempting combinatorial search was taken in (Shah et al. 2018), where the authors view outlier detection as a search problem, and apply the weighted A^* algorithm to solve it. The result is more accurate than other algorithms (with proper choice of parameters it is optimal), but the running time makes it impractical for large datasets.

Many studies view the data as coming from a fixed distribution, and attempt to detect outliers as data points at the margins of the distribution (e.g., (Roberts 1999; Scheirer et al. 2011; Hubert and Engelen 2004; Xu, Caramanis, and Sanghavi 2012; Zhang et al. 2015)).

A recent review of the current state of the art (see (Lerman and Maunu 2018b)) identifies many algorithms as "filter outliers". See, e.g., (Lei Xu and Yuille 1995; You, Robinson, and Vidal 2017; Chen and Lerman 2009; Soltanolkotabi, Candes et al. 2012; Rahmani and Atia 2017; Rahmani and Li 2019). The idea is to associate a value with each point indicating how likely it is to be an outlier. These values are then used in various ways to identify outliers. For example, the value can be the reconstruction error in (4), or how likely the point is to lie on a low-dimensional subspace shared with other points. Another approach that uses randomization is described in (Xu, Caramanis, and Mannor 2013), where at each iteration, a point is removed with probability proportional to its variance in the current direction. A deterministic version of the algorithm was later developed in (Feng, Xu, and Yan 2012).

4 The proposed lookahead algorithm

Our lookahead algorithm can be considered a filter algorithm as discussed in Section 3. We begin by describing it without specifying the filter values associated with points. This is followed by specifying the filter values. An algorithm for computing the filter values efficiently is described in Section 6.

4.1 Top view of the lookahead algorithm

Let O_1, O_2 be two outlier sets such that $|O_1| = |O_2|$. (Both sets have the same number of outliers.) Suppose we have access to a function f such that:

$$f(O_1, X, r) > f(O_2, X, r) \Rightarrow \\ O_1 \text{ "appears to be" better than } O_2$$

Using f we propose an iterative algorithm for computing k outliers. A single iteration is given below:

```
Input: A subset $O$ of $j$ outliers. ($n-j$ inliers.)
  $k$, the number of outliers. A ratio parameter $0 \leq \alpha \leq 1$.
  $X$, the data matrix. $r$, desired PCA rank.
1. For $i = 1 ... n-j$ create the children subsets $O_i$ by
   adding the inlier column $i$ to $O$. $f_i = f(O_i, X, r)$.
2. Compute $c$ from $\alpha$ by the following formula:
        $c = \alpha(k - j - 1) + 1$        (5)
Output: the union of the $c$ children with the largest $f_i$.
As a formula: output = Update($O, k, \alpha, X, r$).
```

Combining this with the improvement produced by the KMeans-Style (see Figure 4) gives the entire algorithm:

```
Input: $1 \leq k \leq n$,  $0 \leq \alpha \leq 1$,  $X$,  $r$.
Output: A subset $O$ of $k$ outliers.
Initialization: $O = \emptyset$ (the empty set).
Updates: while $|O| < k$ do:
    $O$=Update($O, k, \alpha, X, r$),  $O$=KM($O, X, r$).
```

The value of α affects both the accuracy and the running time of the algorithm. Increasing α would in general result in a reduction in accuracy and a faster running time. The most accurate choice is α=0 which gives c=1. This requires k updates and $O(nk)$ filter evaluations. The fastest choice is α=1 which gives $c = k-j$. This requires 1 update and $O(n)$ filter evaluations. For a constant value of α the number of updates is $O(\log k)$ and the number of filter evaluations is $O(n \log k)$. For correctness first observe that the algorithm terminates because the "Update" step always returns a bigger outlier set than its input and KM does not change the number of outliers. To see that at termination $|O|=k$ observe that the formula (5) guarantees $|O| \leq k$.

4.2 The filter values

A natural choice for f is the reconstruction error e_i as given in (4). However, as shown in the example in Figure 2 it may be very inaccurate. Instead, we propose to use the model reconstruction error after the point is removed. Suppose the outlier set O is known. The lookahead error of an inlier point x_i with respect to O is the reconstruction error E as defined in (4) of the outlier set O_i, obtained by adding x_i to O.

$$f_i = f(O_i, X, r) = E(Z_i)$$

where the columns of Z_i are those in X but not in O_i, and $E(Z_i)$ is defined in (4). A direct evaluation of the lookahead error as defined above requires that a PCA algorithm is applied to the set of inliers corresponding to O_i. This is clearly impractical. We proceed to show how to calculate lookahead errors efficiently.

5 Rank-1 modifications

The efficient calculation of filter values by our algorithm is based on efficient eigenvalue estimation that relies on a rank-1 modification relation between the parent and its children. In this section we establish this relation.

The filter values that are needed for the "Update" steps in Section 4.1 require the reconstruction errors of children. Let p be the number of inliers at the parent level.

Z :	the $m \times p$ matrix of the inliers.
B :	$= ZZ^T$.
μ :	the column mean of Z.
Z^c :	the "centering" of Z. μ subtracted from columns.
C :	$= Z^c(Z^c)^T$.

Suppose a child is constructed by removing x_i from Z.

Z_i :	the $m \times (p-1)$ matrix of the remaining outliers.
B_i :	$= Z_i Z_i^T$.
μ_i :	the column mean of Z_i.
Z_i^c :	the "centering" of Z_i. μ_i subtracted from columns.
C_i :	$= Z_i^c(Z_i^c)^T$.

We proceed to show that C_i can be obtained as a rank-1 modification of C. To the best of our knowledge this was not previously observed.

Theorem 2: Define: $y_i = x_i - \mu$, and $\beta = \frac{p}{p-1}$. Then:

$$C_i = C - \beta y_i y_i^T \tag{6}$$

Proof: The following relations are known:

$$1: \; C = B - p\mu\mu^T, \quad 2: \; C_i = B_i - (p-1)\mu_i\mu_i^T$$

$$3: \; B_i = B - x_i x_i^T, \quad 4: \; \mu_i = \frac{p\mu - x_i}{p-1}$$

For 1,2 see, e.g., (Cadima and Jolliffe 2009). 3,4 are easy to verify. We can now use 1 to eliminate B in 3, use the result to eliminate B_i in 2, and use 4 to eliminate μ_i in 2. This leaves 2 with the variables C_i, C, x_i, μ. Simplifying gives the desired relation. ■
A similar result relates the traces of C and C_i:

Corollary (of Theorem 2):

$$\text{trace}\{C_i\} = \text{trace}\{C\} - \beta \|y_i\|^2 \qquad (7)$$

Proof: Apply trace to both sides of (6). ■

6 Complexity of the lookahead algorithm

To complete the description of the lookahead algorithm as given in Section 4 we need to describe how to compute the filter values. As we show the complexity of computing $O(n)$ filter values is $O(rdn)$, where r is the desired PCA rank and d is the rank of the data matrix which can be reduced by the user.

The computational tool that we use here is the well known technique for computing eigenvalues of a matrix perturbed by rank-1 modification. See, e.g., (Bunch, Nielsen, and Sorensen 1978). It can be stated as follows:

Rank-1 modification: Let A be a symmetric matrix of rank d with a known eigendecomposition. Let x be a vector, and $\beta \geq 0$ a scalar. Suppose $\tilde{A} = A - \beta x x^T$. Then there is an algorithm that computes each eigenvalue of $\tilde{A}$ in $O(d)$ cost.

Using the same notation as in Section 5 the filter values that need to be calculated are the reconstruction errors of Z_i^c. From Theorem 1 it follows that this reconstruction error can be calculated from the eigenvalues of C_i. Each reconstruction error can be computed from r eigenvalues (and a trace) where r is the desired PCA rank. Applying Theorem 2 and its corollary it follows that the cost of computing these r eigenvalues is $O(rd)$.

To further elaborate on the eigendecompositions performed by the lookahead algorithm we discuss the three locations where eigenvectors and/or eigenvalues are calculated.

1. Root factorization of the data matrix X of size $m \times n$. Using current state-of-the-art algorithms this factorization can be computed with d, a user specified rank parameter, and its running time is $O(dmn)$. If d is not specified it is taken as $\min\{m, n\}$. We view this as an initial dimensionality reduction step. We assume that the results are stored either in memory or on the hard drive.
2. Factorization at each parent. At each parent we need the eigendecomposition computed from the inlier subset at that node. With the dimensionality reduction performed at the root this takes $O(d^2 n)$.
3 Factorization for each child. As discussed in the beginning of this section this takes $O(rd)$.

Let p be the number of parents that the lookahead algorithm expands. The total number of children becomes $O(pn)$, and

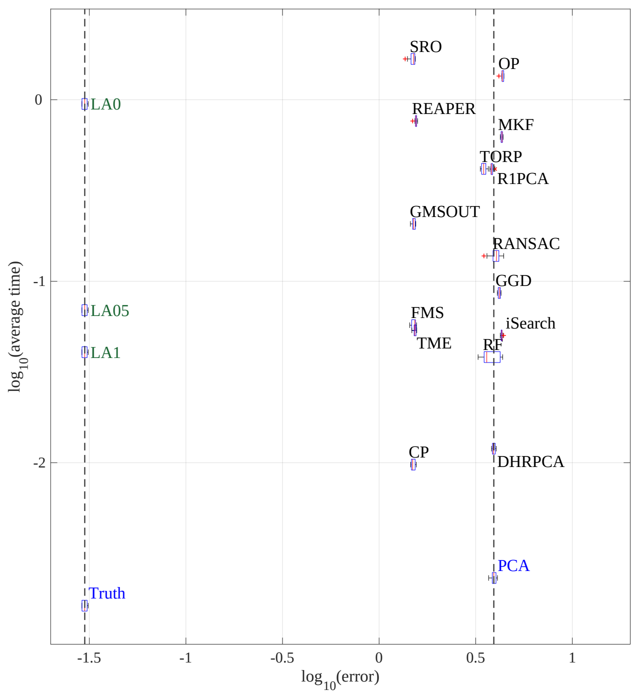

Figure 5: A boxed plot for one of the Haystack experiments. Outlier fraction: 0.6. Outlier mean: 1.0 (in all coordinates). Values were averaged over 10 runs.

the total running time complexity of the lookahead algorithm is: $O(dmn + pd^2n + prdn)$ which is the same as $O(dmn+pd^2n)$. In the table below we show these results results for the three choices of the ratio parameter α that were analyzed in Section 4.1.

α value	complexity	large $k = O(n)$
0	$O(dmn + kd^2n)$	$O(d^2n^2)$
1	$O(dmn + d^2n)$	$O(dmn)$
$0 < \alpha < 1$	$O(dmn + d^2n \log k)$	$O(d^2n \log n)$

The running time of the KMeans-Style algorithm as shown in Figure 4 is $O(trdn)$, where t is the number of iterations it takes the algorithm to converge. Even though t is typically very small (2-3 in our experiments) we make sure the algorithm runs no more than 5 iterations. Thus, using it at each parent adds to the overall complexity a term of $O(prdn)$ which can be ignored.

7 Experimental results

.
The experiments that we describe follow closely the methodology used in the recent overview of the field (Lerman and Maunu 2018b). We refer to this reference as the **overview**. The **overview** describes two artificial datasets to be used for comparing algorithms. We experimented with these datasets and added experiments with some standard datasets that are being used for evaluating machine learning algorithms.

When describing the results we refer to our lookahead algorithm with the initials LA followed by the α value. Thus,

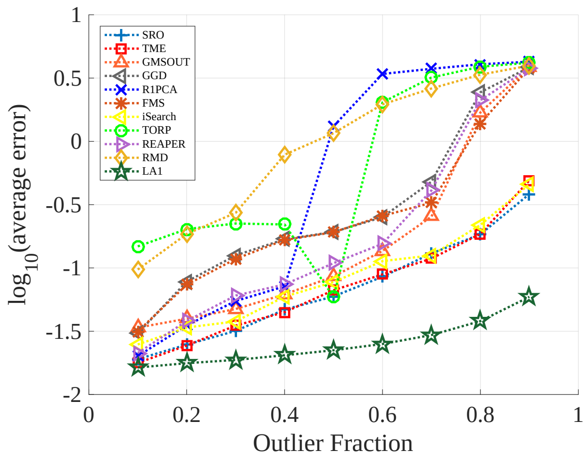

Figure 6: Error versus outlier fraction for the Haystack experiment. Outlier mean: 0.

LA0 means the algorithm with α=0, and LA05 means the algorithm with α=0.5. The comparison was with other algorithms that made their code available and are listed below: SRO (You, Robinson, and Vidal 2017), iSearch (Rahmani and Li 2019), GGD (Maunu, Zhang, and Lerman 2019), FMS (Lerman and Maunu 2018a), CP (Rahmani and Atia 2017), TME (Zhang 2016), OP (Xu, Caramanis, and Sanghavi 2012), MKF (Zhang, Szlam, and Lerman 2009), DHRPCA (Feng, Xu, and Yan 2012), REAPER (Lerman et al. 2015), TORP (Cherapanamjeri, Jain, and Netrapalli 2017), R1PCA (Ding et al. 2006), GMSOUT (Zhang and Lerman 2014), RANSAC (Fischler and Bolles 1981), RF (Hardt and Moitra 2013). Additional information about these algorithms is available in the citations above. Most of them are also discussed in **overview**. When using the artificial datasets we add "Truth" as the ground truth result, and "PCA" for the results obtained with (non-robust) standard PCA.

7.1 Experiments with the Haystack Model

The Haystack model is commonly used to evaluate subspace recovery algorithms. See, e.g., (Rahmani and Atia 2017; Lerman and Maunu 2018a,b; Rahmani and Li 2019). The model first selects a random subspace in r dimensions. the inliers are created by drawing from a zero mean Gaussian distribution of points on the model. The outliers are generated from a zero mean isotropic Gaussian distribution. Additional zero mean Gaussian noise is then added to both inliers and outliers.

Following **overview** 400 points are generated in a 200 dimensional space with model rank r=10. We use the same Haystack model as in **overview** with one exception: the mean of outliers is nonzero, and set to be (the same) constant in all coordinates. After all the inliers and the outliers are generated as explained above the entire dataset is centered.

Since the ground truth is known the algorithm performance can be measured by how accurate is the recovered

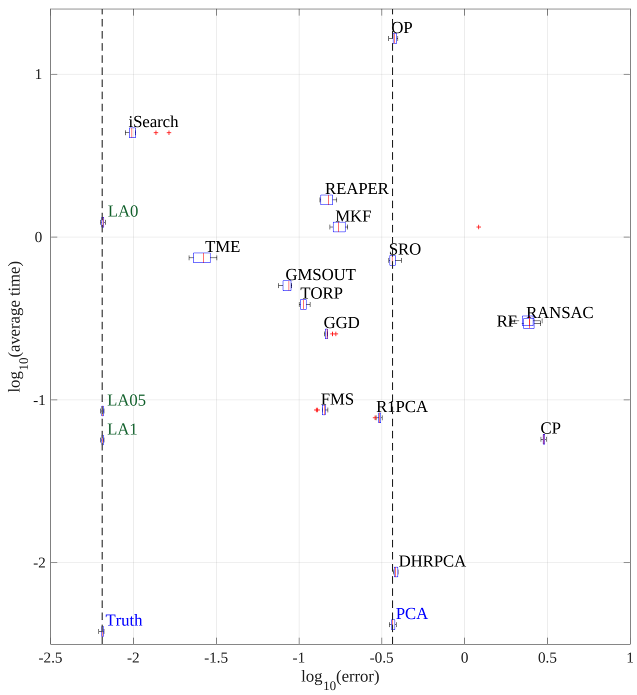

Figure 7: A boxed plot for one of the Blurryface experiments. Outlier fraction: 0.6. Values were averaged over 10 runs.

subspace. Following **overview** we take the error as the squared principal angles between the ground truth subspace and the recovered subspace.

Running the experiments we observed a huge decline in the accuracy of current state-of-the-art algorithms with the nonzero mean of outliers. A boxed plot of one of the experiments is shown in Figure 5. The vertical axis in the plot corresponds to speed, and the horizontal axis to accuracy. We observe that the lookahead variants are very accurate, but none of the other algorithms are. In fact, many have similar accuracy to the (non robust) standard PCA. In terms of speed LA0 is among the slowest, but LA1 is clearly above average. (The LA algorithms use dimensionality reduction with $d = 40$).

More information about how the accuracy changes with the increase of outlier percentage can be obtained from the plot in Figure 6. Observe the rapid increase in the error measurements which is very distinct even with the logarithmic scale of the plot. There is also an increase in the error of LA1, but its advantage over other algorithms clearly increases with the increase in outlier percentage.

7.2 Experiments with the Blurryface model

The blurryface model is another artificial dataset that was proposed by **overview** to evaluate robust subspace recovery algorithms. The goal is to recover a 9-dimensional subspace corresponding to the manifold of a group of images under various illumination changes. Here the ground truth subspace is created from a particular group, and the inliers are

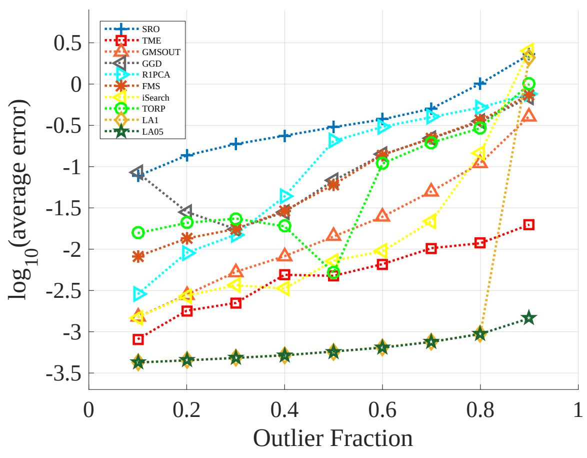

Figure 8: Error versus outlier fraction for Blurryface.

generated randomly distributed in that subspace. The outliers are sampled from other groups. The resulting data set has 500 points in a 400 dimensional space.

A boxed plot of one of the experiments is shown in Figure 7. As in the previous boxed plot the vertical axis corresponds to speed, and the horizontal axis to accuracy. We observe that the lookahead variants are again very accurate. while the accuracy of some other algorithms is even worse than the (non robust) standard PCA. (The LA algorithms use dimensionality reduction with $d = 40$).

More information about how the accuracy changes with the increase of outlier percentage can be obtained from the plot in Figure 8. LA05 and LA1 are distinctly better than the other algorithms until the 80% point where the accuracy of LA1 deteriorates. The reason for that is not clear to us and requires further study.

7.3 Experiments on real data

We ran additional experiments on datasets that are common in machine learning studies. Since the ground truth is not known for these datasets we report instead the reconstruction error as discussed in Section 2.

Table 1 shows results on several datasets that were obtained from the UCI repository. The algorithms being compared are the Lookahead with several different parameters and the top performers among the other algorithms. In all tested cases all variants of the lookahead algorithm outperform all other algorithms.

Table 2 shows results on several big datasets that are publicly available and sometimes used in the evaluation of big data machine learning algorithms. The results indicate that there is little difference in accuracy between the algorithm variants, but a huge difference in running time.

$r:k$	$\alpha:0$	$\alpha:0.3$	$\alpha:0.5$	$\alpha:1$	iSearch	TME	SRO
Geographical Original of Music ($m = 68$, $n = 1,059$)							
10:200	13,567	**13,564**	**13,564**	13,569	13,635	13,629	14,454
10:700	**3,096**	3,097	3,103	3,114	3,224	3,202	3,401
YearPredictionMSD ($m = 90$, $n = 515,345$)							
10:200	**6.161e11**	**6.161e11**	**6.161e11**	**6.161e11**	ArrayLimit	6.164e11	-
10:1e4	-	**4.879e11**	**4.879e11**	**4.879e11**	ArrayLimit	4.885e11	-
CNAE ($m = 856$, $n = 1,080$)							
20:200	**2,088**	**2,088**	**2,088**	2,089	2,090	2,152	2,1100
20:700	**377**	**377**	378	379	387	393	409
Covtype ($m = 54$, $n = 581,012$)							
15:5e3	-	**2.9248e5**	**2.9248e5**	**2.9248e5**	ArrayLimit	2.9322e5	-
30:5e3	-	**31,738**	**31,738**	31,755	ArrayLimit	31,887	-

Table 1: Comparison of reconstruction error on real datasets. “-” indicates that no results were produced after running for 30 minutes.

dataset	$d:r:k$		$\alpha:0.3$	$\alpha:0.5$	$\alpha:1$
Day1 $20,000\times3,231,957$	200:30:100	error	106,741	106,335	107,458
		runtime	3929s	2943s	1763s
	200:30:2e4	error	-	482	484
		runtime	-	5282s	1940s
Sift $128\times1,000,000$	70:20:100	error	30,399	30,399	30,399
		runtime	262s	176s	81s
	70:20:5e5	error	10,168	10,168	10,680
		runtime	2922s	1569s	121s

Table 2: Results on big datasets. The error values for “Sift” are divided by the number of points.

8 Concluding remarks

Robust subspace recovery techniques can be motivated by the need to recover PCA structure in data that may be contaminated by a small number of outliers. Situations where the outlier fraction is larger than the number of inliers are of a different flavor. Here one attempts to discover a pattern among a fraction of the data, where statistics obtained from the entire data may not be useful.

In this paper we describe the lookahead algorithm that performs one step lookahead for estimating the likelihood that a point may be an outlier. We showed experimentally that the lookahead strategy performs well even when the number of outliers dominates the number of inliers. This by itself is not surprising. What we found surprising is that it is possible to efficiently implement the lookahead strategy to achieve running time not much slower than standard PCA. The key idea was to show that the lookahead error can be computed from eigenvalues (Theorem 1), and that these eigenvalues can be computed efficiently using rank-1 modification techniques. We believe that the key observation that certain scaled covariance matrices are related to each other by rank-1 modification (Theorem-2) is new.

Evaluating the asymptotic complexity of the proposed algorithm gives the same complexity as PCA for the fastest variant, an increase by a log factor for other practical variants, and $O(d^2n^2)$ for the most accurate variant. Thus, the most accurate variant is much more expensive than PCA. This appears to suggest that the algorithm is impractical. However, as shown in our experiments it performs quite well on datasets that are not too big.

References

Bunch, J. R.; Nielsen, C. P.; and Sorensen, D. C. 1978. Rank-One Modification of the Symmetric Eigenproblem. *Numer. Math.* 31: 31–48.

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

Cabral, R.; Torre, F. D. L.; Costeira, J. P.; and Bernardino, A. 2013. Unifying Nuclear Norm and Bilinear Factorization Approaches for Low-Rank Matrix Decomposition. In *2013 IEEE International Conference on Computer Vision*, 2488–2495.

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

Chen, G.; and Lerman, G. 2009. Spectral curvature clustering (SCC). *International Journal of Computer Vision* 81(3): 317–330.

Cherapanamjeri, Y.; Jain, P.; and Netrapalli, P. 2017. Thresholding Based Outlier Robust PCA. In Kale, S.; and Shamir, O., eds., *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, volume 65 of *Proceedings of Machine Learning Research*, 593–628. PMLR.

Clarkson, K. L.; and Woodruff, D. P. 2017. Low-Rank Approximation and Regression in Input Sparsity Time. *Journal of the ACM* 63(6): 54:1–54:45.

Ding, C.; Zhou, D.; He, X.; and Zha, H. 2006. R1-PCA: rotational invariant L 1-norm principal component analysis for robust subspace factorization. In *Proceedings of the 23rd international conference on Machine learning*, 281–288.

Feng, J.; Xu, H.; and Yan, S. 2012. Robust PCA in High-Dimension: A Deterministic Approach. In *Proceedings of the 29th International Conference on Machine Learning*, 1827–1834. ISBN 9781450312851.

Fischler, M. A.; and Bolles, R. C. 1981. Random Sample Consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6): 381–395.

Gray, V. 2017. *Principal Component Analysis: Methods, Applications and Technology*. Mathematics Research Developments. New York: Nova Science Publishers, Incorporated. ISBN 9781536108897.

Halko, N.; Martinsson, P.; Shkolnisky, Y.; and Tygert, M. 2011. An Algorithm for the Principal Component Analysis of Large Data Sets. *SIAM Journal of Scientific Computing* 33(5): 2580–2594.

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

Hardt, M.; and Moitra, A. 2013. Algorithms and Hardness for Robust Subspace Recovery. In Shalev-Shwartz, S.; and Steinwart, I., eds., *COLT*, volume 30 of *JMLR Workshop and Conference Proceedings*, 354–375. JMLR.org.

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

Jolliffe, I. T. 2002. *Principal Component Analysis*. New York: Springer-Verlag, second edition.

Kuang-Chih Lee; Ho, J.; and Kriegman, D. J. 2005. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(5): 684–698.

Lei Xu; and Yuille, A. L. 1995. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Transactions on Neural Networks* 6(1): 131–143.

Lerman, G.; and Maunu, T. 2018a. Fast, robust and non-convex subspace recovery. *Information and Inference: A Journal of the IMA* 7(2): 277–336.

Lerman, G.; and Maunu, T. 2018b. An Overview of Robust Subspace Recovery. *Proceedings of the IEEE* 106(8): 1380–1410.

Lerman, G.; McCoy, M. B.; Tropp, J. A.; and Zhang, T. 2015. Robust computation of linear models by convex relaxation. *Foundations of Computational Mathematics* 15(2): 363–410.

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

Maunu, T.; Zhang, T.; and Lerman, G. 2019. A well-tempered landscape for non-convex robust subspace recovery. *Journal of Machine Learning Research* 20(37): 1–59.

Musco, C.; and Musco, C. 2015. Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. In *NIPS'15*, 1396–1404. Cambridge, MA, USA: MIT Press.

Rahmani, M.; and Atia, G. K. 2017. Coherence Pursuit: Fast, Simple, and Robust Principal Component Analysis. *IEEE Transactions on Signal Processing* 65(23): 6260–6275. ISSN 1053-587X. doi:10.1109/TSP.2017.2749215.

Rahmani, M.; and Li, P. 2019. Outlier Detection and Robust PCA Using a Convex Measure of Innovation. In *NIPS'19*, 14200–14210.

Roberts, S. J. 1999. Novelty detection using extreme value statistics. *IEE Proceedings - Vision, Image and Signal Processing* 146(3): 124–129.

Scheirer, W. J.; Rocha, A.; Micheals, R. J.; and Boult, T. E. 2011. Meta-Recognition: The Theory and Practice of Recognition Score Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(8): 1689–1695.

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

Soltanolkotabi, M.; Candes, E. J.; et al. 2012. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics* 40(4): 2195–2238.

Vaswani, N.; Bouwmans, T.; Thierry, J.; Sajid, J.; and Narayanmurthy, P. 2018. Robust Subspace Learning: Robust PCA, Robust Subspace Tracking, and Robust Subspace Recovery. *IEEE Signal Process. Mag.* 35(4): 32–55.

Vaswani, N.; and Narayanamurthy, P. 2018. Static and Dynamic Robust PCA and Matrix Completion: A Review. *Proceedings of the IEEE* 106(8): 1359–1379.

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

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

Xu, H.; Caramanis, C.; and Sanghavi, S. 2012. Robust PCA via Outlier Pursuit. *IEEE Transactions on Information Theory* 58(5): 3047–3064.

You, C.; Robinson, D. P.; and Vidal, R. 2017. Provable self-representation based outlier detection in a union of subspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3395–3404.

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

Zhang, T. 2016. Robust subspace recovery by Tyler's M-estimator. *Information and Inference: A Journal of the IMA* 5(1): 1–21.

Zhang, T.; and Lerman, G. 2014. A novel M-estimator for robust PCA. *The Journal of Machine Learning Research* 15(1): 749–808.

Zhang, T.; Szlam, A.; and Lerman, G. 2009. Median k-flats for hybrid linear modeling with many outliers. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 234–241. IEEE.