Accelerated Combinatorial Search for Outlier Detection with Provable Bounds on Sub-Optimality

Guihong Wan, Haim Schweitzer
Department of Computer Science, The University of Texas at Dallas
800 W. Campbell Road, Richardson, Texas 75080
Guihong.Wan@utdallas.edu, HSchweitzer@utdallas.edu

Abstract

Outliers negatively affect the accuracy of data analysis. In this paper we are concerned with their influence on the accuracy of Principal Component Analysis (PCA). Algorithms that attempt to detect outliers and remove them from the data prior to applying PCA are sometimes called Robust PCA, or Robust Subspace Recovery algorithms. We propose a new algorithm for outlier detection that combines two ideas. The first is "chunk recursive elimination" that was used effectively to accelerate feature selection, and the second is combinatorial search, in a setting similar to A*. Our main result is showing how to combine these two ideas. One variant of our algorithm is guaranteed to compute an optimal solution according to some natural criteria, but its running time makes it impractical for large datasets. Other variants are much faster and come with provable bounds on sub-optimality. Experimental results show the effectiveness of the proposed approach.

1 Introduction

An important challenge in the analysis of data is the design of a simple model that fits the data. In practical situations this requires handling data outliers. Typically, the data model can be made significantly more accurate if it is allowed to ignore a small fraction of the data items, considered to be outliers. See, e.g., (Aggarwal 2016; Hodge and Austin 2004).

The particular data model that we discuss in this paper is the one produced by Principal Component Analysis (PCA). Consider for example the data shown in Figure 1, consisting of 9 points. In this case, rank-1 PCA gives a bad model for all 9 points, but a good model for the 8 non-outliers. Observe the huge effect of a single outlier.

PCA is arguably the most widely used dimension reduction technique, with a variety of applications. See, e.g., (Jolliffe 2002; Burges 2010; Cabral et al. 2013; Vidal, Ma, and Sastry 2016; Boutsidis, Woodruff, and Zhong 2016; Gray 2017). Recent algorithmic approaches that use randomization give algorithms that can easily handle large datasets (e.g., (Halko, Martinsson, and Tropp 2011; Halko et al. 2011; Musco and Musco 2015; Li et al. 2017)). However, computing robust variants that ignore outliers is still a big challenge.

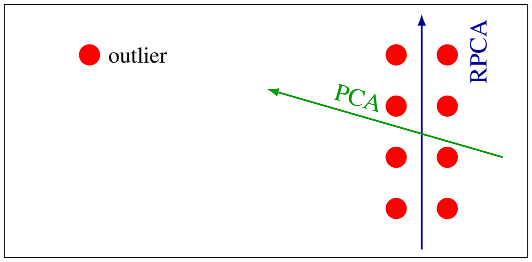

Figure 1: The direction of the dominant principal component computed from 9 points with one outlier. The direction labeled PCA (green arrow) was computed from the entire data (9 points). The direction labeled RPCA (blue arrow) was computed from the 8 non-outliers.

There is a large number of studies on outlier detection and removal specifically for computing robust PCA. The algorithms are sometimes called "Robust Principal Component Analysis" (RPCA), or, alternatively, "Robust Subspace Recovery" (RSR). A recent review paper is (Lerman and Maunu 2018b).

One way to solve the RPCA problem is to first filter outliers and then fit a subspace to the remaining data by using classical PCA. 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 2010; Zhang et al. 2015)). Another common approach is to rank the data points based on some criterion, and detect outliers as points with low scores. See, e.g., (Chen and Lerman 2009; Soltanolkotabi, Candes et al. 2012; Rahmani and Atia 2017; You, Robinson, and Vidal 2017; Rahmani and Li 2019). For example, the "Cop" algorithm (Rahmani and Atia 2017) computes the scores from the Gram matrix of the normalized data. The score for a data point is the norm of the corresponding row in the Gram matrix. The resulting algorithm runs very fast but requires a large amount of memory. The "R-graph" algorithm (You, Robinson, and Vidal 2017) measures the points based on the self-expressiveness property. The scores are computed by solving an elastic net minimization problem. The "Innovation" of each point is used to rank the points in (Rahmani and Li 2019). The authors refer to the algorithm as "iSearch".

Algorithm	time	memory
Cop (Rahmani and Atia 2017)	$O(mn^2)$	$O(n^2{+}mn)$
R-graph (You et al. 2017)	$O(mn^2{+}n^3)$	$O(n^2{+}mn)$
iSearch (Rahmani and Li 2019)	$O(mn^2)$	$O(mn)$
FMS (Lerman and Maunu 2018a)	$O(T \cdot rmn)$	$O(mn)$
fastest Shah (Shah et al. 2018)	$O(krmn)$	$O(mn)$
fastest Chunk-A^* (this paper)	$O(rmn)$	$O(mn)$

Table 1: Complexity of various algorithms. n is the number of data items. m is the dimension of each item. r is the number of principal components. k is the number of outliers. T is the number of iterations.

A different approach was taken in (Shah et al. 2018). The authors view outlier detection as a graph search problem, and apply the weighted A^* algorithm to solve it. The goal of the search is to minimize the PCA error. As reported in (Shah et al. 2018) the algorithm comes with an accuracy parameter called ϵ. With $\epsilon = 0$ the algorithm is guaranteed to terminate with an optimal outlier selection. With larger ϵ values the algorithm terminates with outliers that are typically less accurate than the optimal, but the running time is much faster. Unfortunately, as we show in Section 7, even with $\epsilon = \infty$ the algorithm is slow, and cannot be applied to large datasets.

As we show experimentally in Section 7, current state-of-the-art algorithms that compute outliers encounter difficulties when applied to large datasets with hundreds of thousands/millions of data points. Most algorithms run too slow or require an unacceptably large amount of memory. Table 1 shows the complexity of some recent algorithms. We improve on previously proposed algorithms in terms of both runtime and accuracy. The time complexity of the fastest variant of our proposed Chunk-A^* is $O(k/c \cdot rmn)$, where c is the chunk size (introduced later). The memory complexity is $O(mn)$. The approach we take is outlined below.

Our Approach

Our goal is to design a scalable algorithm capable of detecting outliers in large datasets. A useful algorithm for the closely related feature selection problem (Guyon and Elisseeff 2003) is "backward elimination", that requires a measure of the importance of each individual feature. Such a measure is typically called a "filter". Using filters, a typical implementation of backward elimination for selecting k features to be eliminated from among n features is as follows:

> *Run k iterations. In the jth iteration, rank all remaining features ($n{-}j{+}1$ features left) according to their filter values and eliminate the least important feature.*

Observe that this requires roughly nk filter evaluations which may be impractical. Instead, classical implementations of this approach (for example the well known SVM-RFE algorithm (Guyon et al. 2002)) eliminate a chunk of multiple features in each iteration. A chunk consists of a fraction of the features that are ranked as the least important. For sufficiently big chunks this reduces the number of filter evaluations to $O(n)$. We refer to this approach as the "Chunk-RFE", for "Chunk Recursive Feature Elimination".

Our main result is showing how to wrap a combinatorial search framework around the Chunk-RFE. This combines the speed of Chunk-RFE with the accuracy of the combinatorial search approach. The resulting algorithm that we call "Chunk-A^*" has two parameters that control the balance between speed and accuracy: the chunk size that we denote by c, and the ϵ parameter of the weighted A^*.

Our Results

We describe an outlier detection algorithm for detecting k outliers. (k is assumed to be known.) The algorithm error is defined as the PCA error of modeling the data without the outliers (see Section 2). The algorithm accuracy and running time are controlled by the two parameters ϵ, c, where $0 \le \epsilon \le \infty$ is the optimality parameter, and $1 \le c \le k$ is the chunk size. In general, larger values of ϵ and c result in a faster running time, and smaller values give more accurate results.

In addition to identifying outliers, the algorithm computes sub-optimality bounds on how close the results are to the optima. We are not aware of any other outlier detection algorithm that computes such bounds on sub-optimality.

The following are special cases that we prove:

- If $\epsilon = 0$ the algorithm terminates with an optimal outlier selection regardless of the value of the chunk size c.
- If $c = 1$ the algorithm is identical to (Shah et al. 2018). Still, even in this case, our algorithm has the advantage of producing sub-optimality bounds which are not computed by (Shah et al. 2018).

Paper Organization

The paper is organized as follows. The PCA error of modeling the data is defined in Section 2. The proposed algorithm is described in Section 3. The three variants of the algorithm along with a correctness proof are given in Section 4. An intuitive algorithm is introduced to further improve the results in Section 5. The sub-optimality bound is discussed in Section 6. Experimental results are shown in Section 7.

2 The Error of Modeling Data by PCA

Let $X = (x_1 \dots x_n)$ be the data matrix of size $m{\times}n$, where n is the number of data items and m is the dimension of each item. The PCA computes a linear mapping from the m dimensional x_i to the r dimensional y_i, where $r{\le}\min(m,n)$. The inverse of this mapping gives an approximate reconstruction of x_i from y_i. We denote the reconstruction error of x_i by e_i. As in the classical development of PCA (e.g., (Jolliffe 2002)) the quality of the mapping can be measured by the sum of reconstruction errors as follows:

$$e_i(r) = \|x_i - VV^T x_i\|^2, \quad E(r) = \sum_{i=1}^{n} e_i(r).$$

It is known (e.g., (Jolliffe 2002)) that the columns of V can be computed as the r eigenvectors of the matrix $B = XX^T$ corresponding to its r largest eigenvalues. A known result about matrix eigenvalues gives an explicit expression to this

```
Algorithm 1: The Chunk-$A^*$ algorithm.
   Input:
   $X$, a data matrix of $n$ columns.
   $r$, the desired number of principal components.
   $k$, the desired number of outliers.
   $f(S)$, a "filter" function. Here $S$ is an outlier subset,
   and $f(S)$ is an estimate to the PCA error if the
   outliers in $S$ are included in the solution subset.
   $c(S)$, a criterion for determining the chunk size.
   $r_d$ (optional), the rank for dimensionality reduction.
   Default: $r_d = \min(m, n)$.
   Output: a subset $S$ of $k$ outliers.
   Preprocessing: dimensionality reduction (if desired).
   Data Structures: Two global lists of subsets: the
                    fringe list $F$, and the closed list $C$.
   Initialization: Put the empty subset into $F$.
1  while $F$ is nonempty do
2      Pick $S_i$ with the smallest $f(S_i)$ value from $F$.
         Ties are resolved in favor of the larger size($S_i$).
3      if $S_i$ contains $k$ outliers then
4          Stop and return $S_i$ as the solution subset.
5      else
6          Create $U_i$ as the list of all subsets that can be
             obtained by adding a single column to $S_i$.
7          Remove from $U_i$ all the subsets that are in $C$.
8          Compute $f(S_j)$ for each subset $S_j \in U_i$.
9          Select chunk size $c(S_i)$ satisfying:
             $1 \leq c(S_i) \leq k-\text{size}(S_i)$.
10         Compute the subset $\overline{S}_j$ as the union of the
             $c(S_i)$ subsets in $U_i$ with the smallest $f(S_j)$.
11         Add $\overline{S}_j$ to $F$ and $C$.
12         Add to $F$ and $C$ all the subsets $S_j \in U_i$ that
             are not used in the creation of $\overline{S}_j$.
13     end
14 end
```

error in terms of the eigenvalues of B. Suppose $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ are the eigenvalues of B, and V is $m \times r$ then:

$$E(r) = \sum_{j=r+1}^{m} \lambda_j = \text{trace}(B) - \sum_{j=1}^{r} \lambda_j.$$

Suppose X is partitioned into a subset of outliers S and the reminder inliers subset $\tilde{X} = X \setminus S$. The desired rank-$r$ PCA is computed from the inliers. Accordingly, the PCA error is computed from the inliers subset:

$$\begin{aligned} &e_i(S,r) = \|x_i - \tilde{V}\tilde{V}^T x_i)\|^2, \\ &E(S,r) = \sum_{i \notin S} e_i(S,r) = \sum_{j=r+1}^{m} \tilde{\lambda}_j = \text{trace}(\tilde{B}) - \sum_{j=1}^{r} \tilde{\lambda}_j, \end{aligned} \tag{1}$$

where $\tilde{\lambda}_j$ and $\tilde{V}$ are computed from $\tilde{B} = \tilde{X}\tilde{X}^T$.

3 The Chunk-A^* Algorithm

In this section we describe our algorithm in terms of a general filter function that will be defined later. We refer to it as the Chunk-A^*. It is closely related to standard combinatorial search algorithms such as the A^* (Pearl 1984; Russell and Norvig 2010; He et al. 2019). The major difference is that previous studies do not have the Chunk-RFE part, as shown in lines 9-11 of Algorithm 1.

The goal of the algorithm is to compute a subset of k outliers, minimizing the "filter" function f. It maintains in F a list of subsets that need to be further evaluated and in C a list of visited subsets that need not to be added into F again. In the standard "best first" strategy, the algorithm selects the subset from F with the smallest f value to be expanded. However, unlike standard combinatorial search algorithms in our variant not all of the direct expansions of the selected subset are treated equally. Instead, the algorithm "guesses" that the best c among them are part of the solution, and packages them together in a "chunk". We refer to the "chunk" as the "super child", and the children created by adding a new column as "direct children".

The Subset Graph

The algorithm can be viewed as searching on a graph created with nodes corresponding to outlier subsets, similar to the subset graph proposed in (Arai, Maung, and Schweitzer 2015; Arai et al. 2016). With the chunk size $c = 1$, there is an edge from the subset S_i to the subset S_j if adding one column to S_i creates S_j. When $c > 1$, super children are created for each node. Two subset graphs for the matrix $X = (x_1, \ldots, x_5)$ and $k = 3$ are shown in Figure 2.

Note that all paths leading from the root to a node can be considered equivalent, since the order of outliers in a subset does not matter. For example, if the goal node $\{x_1, x_2, x_4\}$ in the right graph is found, it is irrelevant if it is reached by the path $\{\} \rightarrow \{x_1, x_2\} \rightarrow \{x_1, x_2, x_4\}$ or by the path $\{\} \rightarrow \{x_4\} \rightarrow \{x_1, x_2, x_4\}$. This property makes the chunking meaningful.

The Chunk Size

The chunk size is used in Line 9 of the algorithm. The expectation is that using big chunks would result in fast convergence of the algorithm, while small chunks would give more accurate results. As we show, our theory requires that the chunk size should satisfy the following condition:

$$1 \leq c(S_i) \leq k-\text{size}(S_i).$$

The right part $k-\text{size}(S_i)$ is the remaining number of outliers that still have to be selected. With chunk size $c = 1$, there is no acceleration at all.

The Filter Function

From Equation (1) it is clear that the subset with the smallest error can be found by exhaustively evaluating all the subsets of size k, but this is clearly impractical. Instead, we use filters within a search algorithm that evaluates only some of these subsets.

Suppose we are given a subset S_i of size $k_i < k$. We consider the error that can be obtained by completing it to size k. The best error that an outlier subset S_i can achieve is:

$$d(S_i, r, k) = \min_{\text{size}(S_i \cup T) = k} E(S_i \cup T, r). \tag{2}$$

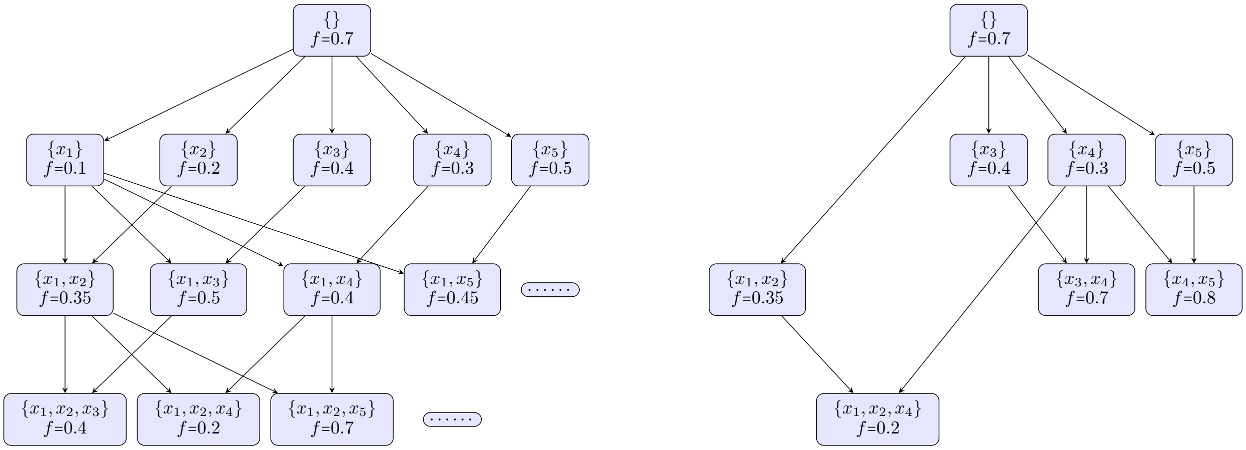

Figure 2: Examples of subset graphs with $n = 5$. The left graph is for c=1; the right graph is for c=2. For instance, when the root node is expanded, in the left graph all direct children are added into F; in the right graph, the super child $\{x_1, x_2\}$ is created by taking the union of $\{x_1\}$ and $\{x_2\}$. This allows the algorithm to quickly evaluate nodes in deeper levels.

Ideally, we may wish to use $d(S_i, r, k)$ as the filter function, but computing it is inefficient since it involves going over all subsets of size $k-k_i$. Instead, we approximate it using lower and upper bounds defined as follows. Let $\lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_m$ be the eigenvalues of the matrix $\tilde{B}$ computed from the inliers at a graph node. Define:

$$\begin{aligned} l(S_i, r, k) &= E(S_i, r + k - k_i) = \sum_{j=r+k-k_i+1}^{m} \lambda_j, \\ u(S_i, r, k) &= E(S_i, r) = \sum_{j=r+1}^{m} \lambda_j. \end{aligned} \tag{3}$$

These are similar to the functions defined in (He et al. 2019).

Lemma 1. For any subset S_i satisfying size$(S_i) \le k$:

$$l(S_i, r, k) \le \; d(S_i, r, k) \; \le u(S_i, r, k)$$

when size$(S_i) = k$ both inequalities become equalities.

The proof is based on the interlacing property of eigenvalues (e.g., (Hill and Parlett 1992; Golub and Van-Loan 2013)). Details will be given in the full version of this paper.

4 Three Variants of the Algorithm

To simplify notations we write $l_i = l(S_i, r, k)$ and $u_i = u(S_i, r, k)$. Lemma 1 shows that the best choice for a filter is “sandwiched” between l_i and u_i. Following (He et al. 2019) we express the filter function f_i as a linear combination of l_i and u_i, with three distinct options: 1. $f_i = l_i$; 2. $f_i = u_i$; and 3. $f_i = l_i + \epsilon u_i$, where $\epsilon \ge 0$. In each case the Chunk-A^* has distinct characteristics as stated in the theorems below:

Theorem 1. **(Optimal variant)** If $f_i = l_i$, then the Chunk-A^* terminates with an optimal solution.

Theorem 2. **(Greedy variant)** If $f_i = u_i$, then the Chunk-A^* is greedy and terminates after at most k node expansions.

Theorem 3. **(Suboptimal variant)** If $f_i = l_i + \epsilon u_i$, then the Chunk-A^* guarantees a solution “close” to the optimum. Let S_* be an optimal solution subset of outliers. Let $e^* = E(S_*, r)$ be the corresponding error. Let S_{**} be the solution subset of outliers found by the algorithm. Let e^{**} be the error for S_{**}. Let l_{**} be the value of $l(S_{**}, r, k)$. Let $u_{\max}$ be the largest value of $u(S_i, r, k)$ for the subsets S_i remaining in the fringe list F after the goal node is reached. Then:

$$e^{**} \le e^* + \epsilon(u_{\max} - l_{**}). \tag{4}$$

Proof of Theorems

In this section we give the technical proofs of the theorems stated in Section 4. Similar theorems were stated and proved in (He et al. 2019) for feature selection. However, their proof strategy cannot be directly applied to the outlier detection case that we study here. Our proofs use several lemmas that are not fully proved. Complete proof details will be given in the full version of this paper.

Lemma 2. The value of l_i is monotonically increasing along any path.
(The proof follows from the interlacing property of eigenvalues.)

Lemma 3. The value of u_i is monotonically decreasing along any path.
(The proof follows from the interlacing property of eigenvalues.)

Lemma 4. Consider the choice $f_i = u_i$. The f value of the super child is smaller than the f value of any of its siblings.

Lemma 5. Consider the choice $f_i = u_i$. Let n_i be the node picked at Line 2 of the algorithm. Let n_j be a child of n_i. The following three properties hold:

a. The size of S_j associated with node n_j is larger than the size of all other nodes currently in the fringe list.

b. The next node to be picked is a child of n_i.

c. If the chunk size $c > 1$, the next node to be picked is the super child of n_i.

(The proof is by induction.)

Lemma 6. Suppose Theorem 3 is false. Then for any node n_z on the path from the root to an optimal node n_* (corresponding to an optimal outlier subset S_*), the following condition holds: $f_z < f_{**}$.
(The proof is similar to the corresponding lemma in (He et al. 2019).)

Lemma 7. Let S_* be an optimal outlier subset of size k. If the algorithm always uses chunk size $c(S_i)$ satisfying: $1 \leq c(S_i) \leq k{-}\text{size}(S_i)$, then during the run of the algorithm the fringe list F always contains a subset of S_*.
(The proof is by induction.)

Proof of Theorem 1: The proof follows as a corollary of Theorem 3 with $\epsilon = 0$. ■

Proof of Theorem 2: From Lemma 5 when a node n_i is picked in Line 2 of the algorithm, one of its children will be examined next. If the chunk size $c > 1$, then the super child will be examined next. This shows that the algorithm is greedy. It detects c outliers in each iteration, and terminates after $\lceil \frac{k}{c} \rceil$ iterations. ■

Proof of Theorem 3: The proof is by contradiction. Suppose the theorem is false. From Lemma 6 it follows that all subsets on the path from the root to an optimal subset S_* have smaller f values than f_{**}. Since from Lemma 7 at any given time at least one of them is in the fringe list, they should be selected before S_{**} is selected. But this means that S_* is selected as the solution and not S_{**}. ■

Bounds

Both the Greedy variant and the Suboptimal variant are not guaranteed to produce an optimal solution. We proceed to show how to obtain a bound on how close their solution is to the optimum. The technique we use was originally proposed by (Hansen and Zhou 2007).

Consider a run of a non-optimal variant, producing the non-optimal subset S_{**}. Then $\text{size}(S_{**}) = k$, and from Lemma 1 it follows that $l_{**} = u_{**} = E(S_{**}, r)$. The value of l_{**} is related to the optimal value l_* by: $l_* \leq l_{**}$. Let B be a value satisfying: $l_{**} - l_* \leq B$. We refer to B as a bound, where a smaller B indicates a better bound, and in particular, B=0 implies an optimal solution. An important observation is that in many combinatorial search techniques it is possible to compute such values.

Let F be the fringe list after the algorithm terminates. Going over all the remaining nodes in F we can compute: $l_{\min} = \min_{S_i \in F} l_i$. Then from Lemma 2 it follows that: $l_* \geq l_{\min}$. Therefore we can take:

$$B = l_{**} - l_{\min}, \quad \text{where: } l_{\min} = \min_{S_i \in F} l_i. \tag{5}$$

5 Improving the Accuracy

We propose a simple algorithm that can improve the accuracy of the results obtained by the Chunk-A^* algorithm. The idea is similar to the well known k-means technique, and we refer to it as Algorithm 2. Running it on the results obtained by the Chunk-A^* will always decrease (never increase) the PCA error, but it typically gets "stuck" in a local minimum after a small number of iterations.

Algorithm 2: Improving Outliers

```
Input: $X$: a matrix of $n$ columns. $r$: the PCA rank.
       $S$: a subset of current outlier selection.
       $T_{\max}$: the maximum number of iterations.
Output: a subset $R$ of $X$ satisfying: $|R| = |S|$ and
        $E(R, r) \leq E(S, r)$.
Set $k = |S|$.
repeat
    Set old error = the current error.
    Compute $V$ as the rank-$r$ PCA of $X \setminus S$.
    Compute $e_i = \|x_i\|^2 - \|V^T x_i\|^2$ for all columns
      $x_i$ of $X$.
    Replace the $k$ outliers of $S$ by the $k$ columns of
      $X$ with the largest $e_i$.
    Set new error = $\sum_{x_i \notin S} e_i$.
until new error = old error or reach $T_{max}$;
Set $R = S$ and return.
```

The algorithm is motivated by the observation that the PCA error can be evaluated separately for each column of X. As in Equation (1), let V be a matrix with r orthogonal columns. The reconstruction error of x_i is given by: $e_i = \|x_i{-}VV^T x_i\|^2 = \|x_i\|^2 - \|V^T x_i\|^2$, and the PCA error can be computed as: $E(S, r){=} \sum_{i \notin S} e_i$.

The idea of Algorithm 2 is that if we know S we can calculate V using PCA, which gives optimal reduction to the reconstruction error of the columns not in S. Once V is known the best selection of k outliers from X is the columns with the k largest reconstruction errors. The PCA error reduction by this process improves the outlier selection. The complexity of the algorithm is $O(T_{\max} rmn)$, where $T_{\max}$ is the maximum number of iterations to convergence. Our experimental results show that the number of iterations is typically at most 3, and thus can be effectively viewed as a constant. Correctness proof will be given in the full version of this paper.

6 Fractional Bounds on Sub-Optimality

The bounds that we derive in this section measure how close the computed result is relative to the optimal result. For example, it may be interesting to know that the error of the computed outlier set is within 10% of the optimal selection. Let E_k be the PCA error of the outlier set S_k produced by the Chunk-A^* algorithm for input size k. Let E_k^* be the smallest possible PCA error of an outlier set of size k. The bound B_k computed from (5) satisfies: $E_k \leq E_k^* + B_k$. Let b be the fractional bound: $b = (E_k{-}E_k^*)/E_k^*$. (It is meaningful only when $E_k^* \neq 0$). After each run of the Chunk-A^* algorithm, we can compute an upper bound on b as shown in (6):

$$b = \frac{E_k - E_k^*}{E_k^*} \leq \frac{B_k}{E_k^*} \leq \frac{B_k}{E_k - B_k} = b_0. \tag{6}$$

The value of b_0 is a nontrivial bound on b, and can be calculated since both B_k, E_k are known at termination. Now consider running Algorithm 2 after the Chunk-A^*, which gives a new outlier subset with the PCA error $E_k^1 \leq E_k$. We show that the fractional bound b_0 can be improved to the fractional bound b_1 as follow:

$$b = \frac{E_k^1 - E_k^*}{E_k^*} \le \frac{E_k^1 - (E_k - B_k)}{E_k^*} \le \frac{B_k - (E_k - E_k^1)}{E_k - B_k} = b_1. \tag{7}$$

Clearly, b_1 can be computed after the run of Algorithm 2 and it is a sharper bound on b than b_0.

7 Experimental Results

We evaluate the performance of the proposed algorithm on various synthetic and real datasets. The real datasets are publicly available. The comparison is with various algorithms that made their code available, including: R-graph (You, Robinson, and Vidal 2017); iSearch (Rahmani and Li 2019); GGD (Maunu, Zhang, and Lerman 2019); FMS (Lerman and Maunu 2018a); Cop (Rahmani and Atia 2017); RMD (Goes et al. 2014); TME (Zhang 2016); OP (Xu, Caramanis, and Sanghavi 2012); MKF (Zhang, Szlam, and Lerman 2009); DHRPCA (Feng, Xu, and Yan 2012); TORP (Cherapanamjeri, Jain, and Netrapalli 2017); R1PCA (Ding et al. 2006).

Experiments on Synthetic Datasets

We model our experiments after the detailed study described in a recent survey paper (Lerman and Maunu 2018b). They use two synthetic data models: the Haystack model and the Blurryface model. Let U_* be the ground truth subspace. The error is computed as the squared principal angles between U_* and the recovered U.

The Haystack model: With fixed parameters $m = 200$, $n = 400$, and $r = 10$, inliers are drawn independently at random from $\mathbb{N}(0, U_* U_*^T / r)$, and the outliers are drawn independently at random from $\mathbb{N}(0, I/D)$. All points are perturbed by adding noise from i.i.d. $\mathbb{N}(0, 10^{-2} I)$. The entire dataset is centered at the end. One deficiency of the Haystack model is that the recovery of subspace can be easy for some algorithms, as mentioned in (Lerman and Maunu 2018b).

The Blurryface model: The motivation for the Blurryface dataset is to generate data resembling real data. The Extended Yale face database B (Kuang-Chih Lee, Ho, and Kriegman 2005) was used. The first individual's face is used to obtain the 9-dimensional subspace U_*. The inliers are generated i.i.d. $\mathbb{N}(0, c_1 U_* U_*^T / r)$ where c_1 is tuned to give inliers comparable magnitude. The outliers are sampled from other faces. The resulting dataset is of size m=400, n=500. See (Lerman and Maunu 2018b) for details.

To investigate the effect of different outlier percentages on errors, 10 datasets were generated for each percentage value from 10% to 90%. The Chunk-A^* uses ϵ=∞ and several chunk sizes. The algorithm is marked as Chunk-A^* followed by the c value. Chunk-A^*-1 is Chunk-A^* with c=1, Chunk-A^*-50 is Chunk-A^* with c=50, and Chunk-A^*-k is Chunk-A^* with c=k, which is the largest allowable value of c. The number of outliers is given as input for the following algorithms: R-graph, iSearch, DHRPCA, Shah, Cop and Chunk-A^*. Parameters unspecified for other algorithms are the same as those used in (Lerman and Maunu 2018b).

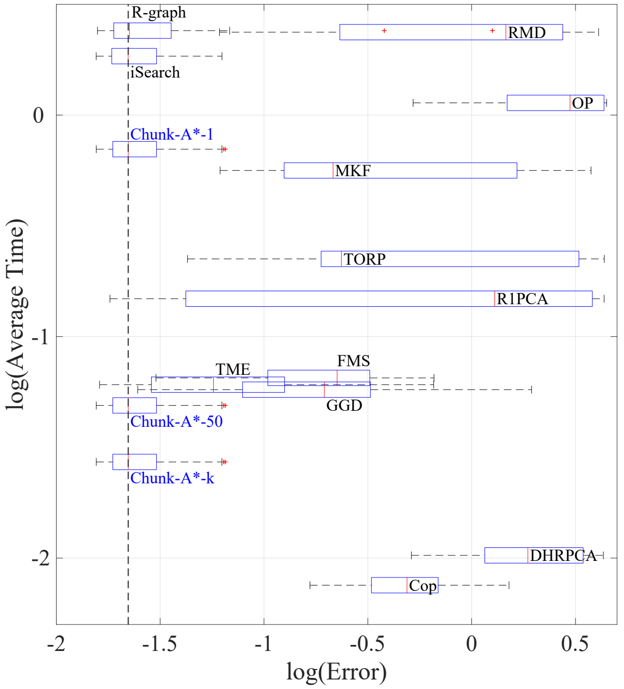

Figure 3: Comparison on the Haystack model. The horizontal axis is the boxplot of errors with different outlier percentages. The red vertical line in a box corresponds to the median of errors. The vertical axis is the average running time. The dashed vertical line corresponds to the median error of the ground truth. The closer to the left bottom, the better the algorithm is.

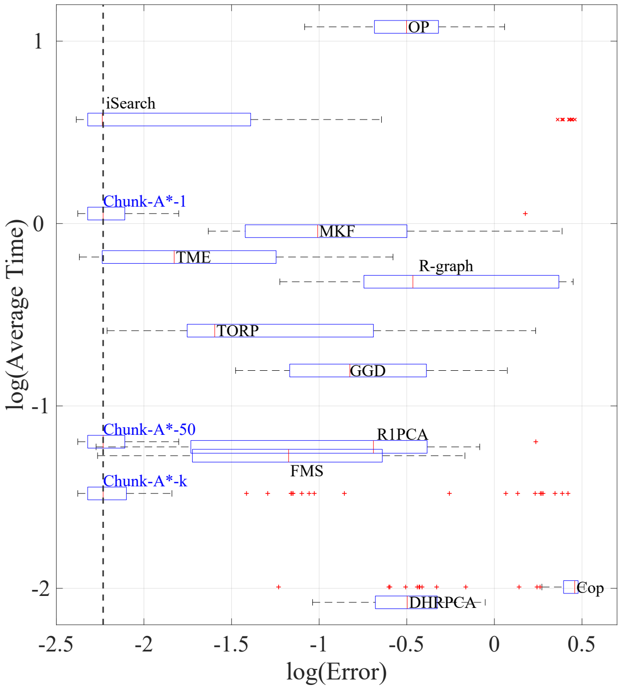

Figure 4: Comparison on the Blurryface model.

dataset	r:k	$f = l$ (optimal)	$f =$ $l + 0.2u$	$f =$ $l + 0.5u$	$f =$ $l + 1.0u$	$f = u$	$f = l + u$ bound:b_0	$f = l + 0.5u$ bound:b_0	iSearch	R-graph	TME
spectf	5:5	**290,498**	**290,498**	**290,498**	**290,498**	**290,498**	0.29	**0.09**	293,043	299,100	303,907
267 × 45	7:7	**203,556**	**203,556**	**203,556**	**203,556**	**203,556**	0.40	**0.19**	217,235	212,274	218,210
vehicle	5:5	**35,908**	**35,908**	**35,908**	36,211	36,211	0.39	**0.38**	52,553	54,686	42,234
846 × 18	10:5	**1,212**	1,242	1,242	1,242	1,580	**0.05**	**0.05**	2,122	4,140	2,354
libras	1:3	**591.43**	**591.43**	**591.43**	**591.43**	**591.43**	**0.00**	**0.00**	**591.43**	598.72	**591.43**
360 × 90	20:7	-	-	**1.03**	**1.03**	1.07	0.59	**0.22**	1.13	1.15	1.15

Table 2: Accuracy and bounds with different filter functions. The minimum errors and bounds are highlighted. "-" indicates no results after running for 5 minutes.

dataset	$r_d : r : k$		$c = 1$	$c = 0.5{*}k$	$c = k$	iSearch	R-graph	TME
Day1:sparse $20,000 \times 3,231,957$	100:30:100	error runtime	> 60min	**59,976** 884s	60,413 607s	- -	- -	- -
Sift:dense $128 \times 1,000,000$	70:15:100	error runtime	e_1 + **5.95E6** 1523s	e_1 + 5.96E6 78s	e_1 + 5.96E6 65s	ArrayLimit	> 10min	e_1 + 6.08E6 43s
YPMSD:dense $90 \times 515,345$	50:10:100	error runtime	e_2 + **2.79E8** 438s	e_2 + **2.79E8** 22s	e_2 + 2.80E8 17s	ArrayLimit	> 10min	e_2 + 3.02E8 8s
Covtype:dense $54 \times 581,012$	30:3:30	error runtime	e_3 + **8.45E7** 69s	e_3 + **8.45E7** 7s	e_3 + **8.45E7** 6s	ArrayLimit	> 10min	e_3 + 9.84E7 21s

Table 3: Greedy variant on big datasets. "-" indicates no results since the implementation does not support sparse data format. The common parts of the errors: e_1 = 4.61E10, e_2 = 6.2E11, and e_3 = 3.1E11.

The results for the Haystack model are shown in Figure 3. Our Chunk-A^* is the fastest among those which achieve high accuracy. The results for the Blurryface model are shown in Figure 4. Our Chunk-A^* is both accurate and fast.

Experiments on Real Datasets

We describe experiments with standard datasets that are commonly used for evaluating machine learning algorithms. Since the ground truth is unknown, the error is taken to be the PCA error on the inliers, as defined in Section 2.

Various filter functions: With different selections of filter functions, our algorithm produces optimal and suboptimal results. The results of different filter functions for various datasets with c=1 are shown in Table 2. Observe that: 1. With ϵ=0 (optimality guaranteed) the algorithm always produces the smallest values; 2. Larger ϵ values give solutions that are typically less accurate than the optimal; 3. Smaller ϵ values yield tighter bounds; 4. Our algorithm is significantly more accurate than other algorithms.

Bounds on sub-optimality: The suboptimal solutions given by our algorithm come with guarantees on how close the solutions are to the optima. Table 4 shows the bounds and errors with different chunk sizes, where error$_0$ is the error after running Chunk-A^*; error$_1$ is the error after running Algorithm 2. Observe that: 1. With chunking ($c > 1$), the algorithm runs faster while the error and bound values become bigger. 2. Algorithm 2 improves the results when the chunk size is big (e.g., $c \geq 0.5{*}k$).

Experiments with big datasets: Experimental results with the greedy variant applied to big datasets are shown in Table 3. Without chunking (c=1), the algorithm is much slower. R-graph and iSearch cannot work with large datasets.

$r : k : \epsilon$		c=1	c=0.2$*k$	c=0.5$*k$	c=k
5:20:0.2	b_1	**0.12**	**0.12**	**0.12**	0.14
	error$_0$	**563,313**	**563,313**	**563,313**	583,633
	error$_1$	**563,313**	**563,313**	**563,313**	577,102
	time	67s	21s	18s	17s
5:60:0.2	b_1	**0.26**	0.31	0.31	0.50
	error$_0$	**110,054**	113,716	115,004	133,020
	error$_1$	**110,054**	113,716	113,716	130,433
	time	93s	18s	16s	16s
10:70:0.1	b_1	**0.15**	0.17	0.19	0.19
	error$_0$	**46,461**	47,213	48,115	48,148
	error$_1$	**46,461**	47,213	48,115	48,148
	time	41s	17s	19s	16s
10:90:0.1	b_1	**0.18**	0.20	0.22	0.21
	error$_0$	**19,673**	19,881	20,314	20,169
	error$_1$	**19,673**	19,881	20,314	20,169
	time	417s	18s	17s	17s

Table 4: Experiments with the suboptimal variant with different chunk sizes on TechTC01 dataset (29,261 × 163).

8 Conclusions

Identifying outliers for Principal Component Analysis is an important problem in data analytics and machine learning. We propose a new algorithm for outlier detection that combines the "chunk recursive elimination" and the combinatorial search, similar to the classical A* search algorithm. We describe three variants of the algorithm. One variant is guaranteed to produce optimal solutions. Other variants are not optimal, but compare favorably with current alternatives. We also show how to compute sub-optimality bounds for this problem. Extensive experiments demonstrate the effectiveness of the proposed approach.

Acknowledgments

We would like to thank various authors who made their code available online. In particular, our experiments are closely modeled after the work described in (Lerman and Maunu 2018b) which we found to be invaluable.

References

Aggarwal, C. C. 2016. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd edition. ISBN 3319475770.

Arai, H.; Maung, C.; and Schweitzer, H. 2015. Optimal Column Subset Selection by A-Star Search. In *Proceedings of the 29th National Conference on Artificial Intelligence (AAAI'15)*, 1079–1085. AAAI Press.

Arai, H.; Maung, C.; Xu, K.; and Schweitzer, H. 2016. Unsupervised Feature Selection by Heuristic Search with Provable Bounds on Suboptimality. In *Proceedings of the 30th National Conference on Artificial Intelligence (AAAI'16)*, 666–672. AAAI Press.

Boutsidis, C.; Woodruff, D. P.; and Zhong, P. 2016. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 236–249.

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.

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. URL http://proceedings.mlr.press/v65/cherapanamjeri17a.html.

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.

Goes, J.; Zhang, T.; Arora, R.; and Lerman, G. 2014. Robust stochastic principal component analysis. In *Artificial Intelligence and Statistics*, 266–274.

Golub, G. H.; and Van-Loan, C. F. 2013. *Matrix Computations*. Johns Hopkins University Press, fourth edition.

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

Guyon, I.; and Elisseeff, A. 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research* 3: 1157–1182.

Guyon, I.; Weston, J.; Barnhill, S.; and Vapnik, V. 2002. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning* 46(1-3): 389–422.

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.

Hansen, E. A.; and Zhou, R. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research* 28: 267–297.

He, B.; Shah, S.; Maung, C.; Arnold, G.; Wan, G.; and Schweitzer, H. 2019. Heuristic Search Algorithm for Dimensionality Reduction Optimally Combining Feature Selection and Feature Extraction. In *Proceedings of the 33rd National Conference on Artificial Intelligence (AAAI'19)*, 2280–2287. AAAI Press.

Hill, Jr, R.; and Parlett, B. N. 1992. Refined interlacing properties. *SIAM journal on matrix analysis and applications* 13(1): 239–247.

Hodge, V.; and Austin, J. 2004. A Survey of Outlier Detection Methodologies. *Artificial intelligence review* 22(2): 85–126. ISSN 0269-2821. doi:10.1023/B:AIRE.0000045502.10941.a9.

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

Jolliffe, I. T. 2002. *Principal Component Analysis*. 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.

Lerman, G.; and Maunu, T. 2018a. Fast, robust and nonconvex 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.

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. *J. Mach. Learn. Res.* 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.

Pearl, J. 1984. *Heuristics : intelligent search strategies for computer*. Reading, Massachusetts: Addison-Wesley.

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 *Advances in Neural Information Processing Systems*, 14223–14233.

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

Russell, S.; and Norvig, P. 2010. *Artificial Intelligence - A Modern Approach*. Pearson Education. ISBN 978-0-13-207148-2.

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.

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

Xu, H.; Caramanis, C.; and Sanghavi, S. 2010. Robust PCA via outlier pursuit. In *Advances in Neural Information Processing Systems*, 2496–2504.

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. 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.; 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.