# Unsupervised Feature Selection by Pareto Optimization*

**Chao Feng,[1] Chao Qian,[1] Ke Tang[2]**
[1]Anhui Province Key Lab of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China
[2]Shenzhen Key Lab of Computational Intelligence, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
chaofeng@mail.ustc.edu.cn, chaoqian@ustc.edu.cn, tangk3@sustc.edu.cn

## Abstract

Dimensionality reduction is often employed to deal with the data with a huge number of features, which can be generally divided into two categories: feature transformation and feature selection. Due to the interpretability, the efficiency during inference and the abundance of unlabeled data, unsupervised feature selection has attracted much attention. In this paper, we consider its natural formulation, column subset selection (CSS), which is to minimize the reconstruction error of a data matrix by selecting a subset of features. We propose an anytime randomized iterative approach POCSS, which minimizes the reconstruction error and the number of selected features simultaneously. Its approximation guarantee is well bounded. Empirical results exhibit the superior performance of POCSS over the state-of-the-art algorithms.

## Introduction

In machine learning and data mining applications, we often encounter the input data with very high dimensionality, bringing certain challenges. Many feature transformation techniques have been proposed for dimensionality reduction, e.g., principal component analysis (Jolliffe 2011), singular value decomposition (SVD) (Golub and Reinsch 1971) and autoencoders (Hinton and Salakhutdinov 2006), to name a few. By combining existing features, these methods transform the data into a low dimensional subspace, which can capture the cardinal information of the original data. However, the new feature representation is difficult to interpret, and projecting the input features into the reduced space often requires matrix multiplication, which reduces the inference efficiency. When keeping the semantic meaning of the features is important, feature selection, which selects a subset of features instead of transforming them, is more appealing. Since unlabeled data is often very abundant, much attention has been drawn to the unsupervised case.

Unsupervised feature selection can be explored from different perspectives. In this paper, we focus on a natural formulation, column subset selection (CSS), which is to minimize the reconstruction error of a data matrix based on the subset of selected features. Formally, given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ (where $m$ is the number of instances and $n$ is the number of features) and a positive integer $k$, the goal is to select at most $k$ columns of $\mathbf{A}$ (forming a matrix $\mathbf{S}$) minimizing $f(\mathbf{S}) = \|\mathbf{A}-\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2$, where $\mathbf{S}^+$ is the Moore-Penrose inverse matrix of $\mathbf{S}$. That is, it is to find at most $k$ columns of $\mathbf{A}$ that captures as much of $\mathbf{A}$ as possible w.r.t. the Frobenius norm. Note that $\mathbf{S}\mathbf{S}^+$ denotes the projection matrix onto the span of the set $\mathbf{S}$ of columns. This problem is known to be UG-Hard (Çivril 2014), and its NP-completeness has been proved only recently (Shitov 2017). Many algorithms with bounded approximation guarantees have been proposed, as briefly reviewed in the following.

## Related Work

The CSS problem was first studied within the numerical linear algebra community. Many algorithms based on rank-revealing QR (RRQR) factorization (Chan and Hansen 1992) were proposed, e.g., (Gu and Eisenstat 1996; Pan and Tang 1999; Hoog and Mattheij 2007). It has been proved that any algorithm for computing RRQR factorization of a matrix $\mathbf{A}$ can provide solutions of the CSS problem with approximation guarantees. Let $\mathbf{A}_k$ denote the best rank-$k$ approximation to $\mathbf{A}$ obtained via SVD, and let $\mathbf{S}_{opt}$ denote an optimal solution of the CSS problem. Note that the approximation guarantee obtained by this kind of algorithms is w.r.t. $\|\mathbf{A} - \mathbf{A}_k\|_F$, which is obviously a lower bound on the truly optimal function value, i.e., $\|\mathbf{A} - \mathbf{S}_{opt}\mathbf{S}_{opt}^+\mathbf{A}\|_F$.

Several randomized sampling algorithms were also proposed for solving the CSS problem, including subspace sampling (Drineas, Mahoney, and Muthukrishnan 2008), volume sampling (Guruswami and Sinop 2012) and leverage sampling (Cohen et al. 2015). Their idea is to randomly select columns with probability proportional to some statistic. The theoretical results on these algorithms mainly showed a trade-off between the number of columns chosen, the approximation bound w.r.t. $\|\mathbf{A} - \mathbf{A}_k\|_F$ and the success probability of the algorithm.

In (Boutsidis, Mahoney, and Drineas 2009), the authors proposed a two-stage algorithm by combining the above two techniques. In the randomized stage, the algorithm selects $O(k \log k)$ columns according to some probability distribution that depends on the top $k$ right singular vectors of $\mathbf{A}$. In the deterministic stage, it returns exactly $k$ columns from the

*This work was supported by the National Key Research and Development Program of China (2017YFC0804002), the NSFC (61603367, 61672478) and the YESS (2016QNRC001).

selected $O(k \log k)$ columns by applying the deterministic column selection algorithm based on RRQR factorization. This two-stage algorithm can achieve an approximation ratio of $O(k\sqrt{\log k})$ w.r.t. $\|\mathbf{A}-\mathbf{A}_k\|_F$ with a high probability.

In (Civril and Magdon-Ismail 2012), a deterministic algorithm based on the sparse approximation of the SVD of $\mathbf{A}$ was proposed. It first computes the top $k$ left singular vectors of $\mathbf{A}$, and then selects columns from $\mathbf{A}$ to approximate the space spanned by these vectors, scaled by the singular values. It has been shown that this algorithm can achieve an approximation ratio of $(1+\epsilon)$ w.r.t. $\|\mathbf{A}-\mathbf{A}_k\|_F$ after selecting $O(k \log k/\epsilon^2)$ columns.

The standard greedy algorithm was later used to solve the CSS problem (Farahat, Ghodsi, and Kamel 2011). It iteratively selects one column, whose inclusion can reduce the objective $f(\mathbf{S})$ the most. To investigate its theoretical performance, Bhaskara et al. (2016) studied the equivalent problem that maximizes $\|\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2$, and proved an approximation ratio of $(1-\epsilon)$ w.r.t. the optimal function value $\|\mathbf{S}_{opt}\mathbf{S}_{opt}^+\mathbf{A}\|_F^2$ after selecting $O(k/\epsilon)$ columns.

Recently, Ordozgoiti et al. (2016) proposed a simple local search algorithm, which starts from $k$ randomly selected columns, and iteratively replaces one selected column with the best among $n-k$ unselected columns, until no improvement can be yielded. This algorithm has been shown empirically to outperform other state-of-the-art methods, and an approximation bound w.r.t. $\|\mathbf{S}_{opt}\mathbf{S}_{opt}^+\mathbf{A}\|_F^2$ has also been provided (Ordozgoiti, Canaval, and Mozo 2018).

All the above algorithms are approximation algorithms. In (Arai, Maung, and Schweitzer 2015), an approach using the A$^*$ heuristic search algorithm is guaranteed to find the optimum. However, due to the NP-hardness of the problem, it can effectively select only a small number of columns from small data sets. A similar approach using the weighted A$^*$ algorithm thus has been proposed (Arai et al. 2016), which can run faster, but without the optimal guarantee.

## Our Contribution

In this paper, we propose a new method based on Pareto optimization (Qian, Yu, and Zhou 2015) for the CSS problem, briefly called POCSS. The idea of POCSS is to first reformulate the original CSS problem as a bi-objective minimization problem that minimizes the given objective $f(\mathbf{S})$ and the number of columns of $\mathbf{S}$ simultaneously, then employ a randomized iterative procedure to solve it, and finally select the best solution with at most $k$ columns from the produced set of solutions. In each iteration of POCSS, it needs to evaluate the objective value of a newly generated solution, which is time-consuming. We derive the recursive relation between the objective values by adding one column into a matrix or deleting one column, based on which the algorithm is computationally feasible.

Theoretically, we prove that POCSS using polynomial time can find a solution $\widehat{\mathbf{S}}$ with at most $k$ columns such that $\|\widehat{\mathbf{S}}\widehat{\mathbf{S}}^+\mathbf{A}\|_F^2 \geq (1-e^{-\gamma}) \cdot \|\mathbf{S}_{opt}\mathbf{S}_{opt}^+\mathbf{A}\|_F^2$, where $\gamma$ is the submodularity ratio (Das and Kempe 2011) characterizing how close the function $\|\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2$ is to submodular. Furthermore, we prove that using slightly more than $k$ columns, i.e., $16k/(\epsilon\sigma)$, POCSS can achieve an approximation ratio of $(1-\epsilon)$ w.r.t. $\|\mathbf{S}_{opt}\mathbf{S}_{opt}^+\mathbf{A}\|_F^2$, where $\epsilon > 0$ is a constant and $\sigma$ denotes the smallest squared singular value of the normalized $\mathbf{S}_{opt}$, i.e., each column of $\mathbf{S}_{opt}$ is scaled to a unit vector.

The experimental results on 10 real-world data sets clearly show the superiority of POCSS over the state-of-the-art algorithms, including Two-stage (Boutsidis, Mahoney, and Drineas 2009), AprxSVD (Civril and Magdon-Ismail 2012), Greedy (Farahat, Ghodsi, and Kamel 2011), IterFS (Ordozgoiti, Canaval, and Mozo 2016) and WA$^*$ (Arai et al. 2016).

The rest of the paper first introduces the studied problem, and then presents the proposed method, its theoretical analysis and empirical study. Finally we conclude this paper.

## Unsupervised Feature Selection

We first give some notations that will be used in the paper.

- $[n]$: set $\{1, 2, \cdots, n\}$.
- $\mathbf{0}$: all-zeros vector.
- $\|\cdot\|_2$: $\ell_2$-norm of a vector.
- $\mathbf{0}_{m,n}$: zero matrix of size $m \times n$.
- $\mathbf{I}_n$: identity matrix of size $n$.
- $\mathbf{C}_{ij}$: entry of the $i$-th row and $j$-th column of matrix $\mathbf{C}$.
- $\mathbf{C}_{i:}$: $i$-th row of matrix $\mathbf{C}$.
- $\mathbf{C}_{:i}$: $i$-th column of matrix $\mathbf{C}$.
- $\mathbf{C}^T$: transpose of matrix $\mathbf{C}$.
- $\mathbf{C}^+$: Moore-Penrose inverse of matrix $\mathbf{C}$.
- $tr(\cdot)$: trace of a square matrix.
- $\|\cdot\|_F$: Frobenius norm of a matrix.
- $|\cdot|$: number of columns of a matrix.

Unsupervised feature selection can be naturally characterized by the CSS problem in Definition 1. It is to select at most $k$ columns from all the $n$ columns of a matrix $\mathbf{A}$ to best approximate $\mathbf{A}$. The goodness of approximation is measured by the sum of squared errors between the original matrix $\mathbf{A}$ and the approximation $\mathbf{S}\mathbf{S}^+\mathbf{A}$ based on the selected columns of $\mathbf{S}$. Note that $\mathbf{S}\mathbf{S}^+$ denotes the projection matrix onto the space spanned by the columns of $\mathbf{S}$.

**Definition 1** (Column Subset Selection (CSS))**.** *Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and a positive integer $k < n$, it is to find a submatrix $\mathbf{S}$ of $\mathbf{A}$ with at most $k$ columns that minimizes $\|\mathbf{A}-\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2$, that is,*

$$\arg\min_{\mathbf{S}:\text{ a submatrix of }\mathbf{A}} \|\mathbf{A}-\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2 \quad s.t. \quad |\mathbf{S}| \leq k.$$

For the ease of theoretical treatment, this minimization problem is often equivalently transformed into a maximization problem:

$$\arg\max_{\mathbf{S}:\text{ a submatrix of }\mathbf{A}} \|\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2 \quad s.t. \quad |\mathbf{S}| \leq k.$$

The equivalency can be verified by

$$\begin{aligned}
&\|\mathbf{A}-\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2 = tr((\mathbf{A}-\mathbf{S}\mathbf{S}^+\mathbf{A})^T(\mathbf{A}-\mathbf{S}\mathbf{S}^+\mathbf{A}))\\
&= tr(\mathbf{A}^T\mathbf{A}-\mathbf{A}^T\mathbf{S}\mathbf{S}^+\mathbf{A}) = tr(\mathbf{A}^T\mathbf{A})-tr(\mathbf{A}^T\mathbf{S}\mathbf{S}^+\mathbf{A})\\
&= tr(\mathbf{A}^T\mathbf{A}) - tr(\mathbf{A}^T(\mathbf{S}\mathbf{S}^+)^T\mathbf{S}\mathbf{S}^+\mathbf{A})\\
&= tr(\mathbf{A}^T\mathbf{A}) - \|\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2,
\end{aligned}$$

where the second and the fourth equalities are derived by the properties of $\mathbf{S}^+$, i.e., $(\mathbf{S}\mathbf{S}^+)^T = \mathbf{S}\mathbf{S}^+$ and $\mathbf{S}^+\mathbf{S}\mathbf{S}^+ = \mathbf{S}^+$. Note that $tr(\mathbf{A}^T\mathbf{A})$ is a constant. In our theoretical analysis, we will consider maximization.

## The Proposed Approach

In this section, we propose a Pareto optimization based approach for the CSS problem, POCSS. Note that Pareto optimization is a general framework that uses bi-objective optimization as an intermediate step to solve single-objective optimization problems. It has been successfully applied to solve the general subset selection problem (Friedrich and Neumann 2015; Qian, Yu, and Zhou 2015; Qian et al. 2017a; 2017b) as well as the problems of multiset selection (Qian et al. 2018b), $k$-subsets selection (Qian et al. 2018a) and sequence selection (Qian, Feng, and Tang 2018).

We use a binary vector $\boldsymbol{s} \in \{0,1\}^n$ to denote a submatrix $\mathbf{S}$ of $\mathbf{A}$. Each element of $\boldsymbol{s}$ corresponds to one column of $\mathbf{A}$. For $1 \leq i \leq n$, the $i$-th bit $s_i = 1$ means that the $i$-th column of $\mathbf{A}$ is included into $\mathbf{S}$; otherwise, the $i$-th column of $\mathbf{A}$ is not selected. Note that the columns of $\mathbf{S}$ are in the order of their appearance in $\mathbf{A}$. We will not distinguish $\boldsymbol{s} \in \{0,1\}^n$ and its corresponding submatrix $\mathbf{S}$ for notational simplicity.

POCSS reformulates the original problem, i.e., Definition 1, as a bi-objective minimization problem

$$\arg\min_{\boldsymbol{s}\in\{0,1\}^n} \quad \big(f(\boldsymbol{s}),\ |\boldsymbol{s}|\big)$$

where $f(\boldsymbol{s}) = \|\mathbf{A} - \boldsymbol{s}\boldsymbol{s}^+\mathbf{A}\|_F^2$. That is, POCSS minimizes the original objective function and the number of selected columns simultaneously.

In the bi-objective setting, both the two objective values have to be considered for comparing two solutions $\boldsymbol{s}$ and $\boldsymbol{s}'$. $\boldsymbol{s}$ *weakly dominates* $\boldsymbol{s}'$ (i.e., $\boldsymbol{s}$ is *better* than $\boldsymbol{s}'$, denoted as $\boldsymbol{s} \preceq \boldsymbol{s}'$) if $f(\boldsymbol{s}) \leq f(\boldsymbol{s}') \wedge |\boldsymbol{s}| \leq |\boldsymbol{s}'|$; $\boldsymbol{s}$ *dominates* $\boldsymbol{s}'$ (i.e., $\boldsymbol{s}$ is *strictly better*, denoted as $\boldsymbol{s} \prec \boldsymbol{s}'$) if $\boldsymbol{s} \preceq \boldsymbol{s}'$ and either $f(\boldsymbol{s}) < f(\boldsymbol{s}')$ or $|\boldsymbol{s}| < |\boldsymbol{s}'|$. But if neither $\boldsymbol{s}$ is better than $\boldsymbol{s}'$ nor $\boldsymbol{s}'$ is better than $\boldsymbol{s}$, they are *incomparable*.

The procedure of POCSS is described in Algorithm 1. In the optimization process, we use the archive $P$ to maintain the non-dominated solutions produced so-far, and $f(P) = \{f(\boldsymbol{s}) \mid \boldsymbol{s} \in P\}$ is the set of objective values of the solutions in $P$. The algorithm starts from the all-zeros solution $\mathbf{0}$ representing the empty matrix (line 1), and then iteratively tries to improve the quality of the solutions in $P$ (lines 4-15). In each iteration, a solution $\boldsymbol{s}$ randomly selected from the current $P$ is used to generate a new solution $\boldsymbol{y}$ by flipping each bit of $\boldsymbol{s}$ with probability $1/n$ (lines 5-6); $\boldsymbol{y}$ is then evaluated (line 7); if $\boldsymbol{y}$ is not dominated by any previously archived solution (line 8), it will be added into $P$, and meanwhile those solutions weakly dominated by $\boldsymbol{y}$ will be removed (lines 9-11). Note that the domination-based comparison makes $P$ always contain incomparable solutions.

POCSS repeats for $T$ iterations. The value of $T$ could influence the quality of the produced solution. Their relationship will be made clear in the theoretical analysis, and we will use the theoretically derived $T$ value in the experiments. After running $T$ iterations, the best solution (i.e., having the smallest $f$ value) satisfying the size constraint in $P$ is selected as the final solution (line 16).

**Algorithm 1** POCSS Algorithm

**Input**: matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, cardinality constraint $k \in [n]$
**Parameter**: the number $T$ of iterations
**Output**: a matrix $\boldsymbol{s}$ comprised by at most $k$ columns of $\mathbf{A}$
**Process**:

1: Let $\boldsymbol{s} = \mathbf{0}$.
2: Let $P = \{\boldsymbol{s}\}$, $f(P) = \{f(\boldsymbol{s})\}$, $P^+ = \{\boldsymbol{s}^+\}$.
3: Let $t = 0$.
4: **while** $t < T$ **do**
5: Select $\boldsymbol{s}$ from $P$ uniformly at random.
6: $\boldsymbol{y} :=$ flip each bit of $\boldsymbol{s}$ with probability $1/n$.
7: $(f(\boldsymbol{y}), \boldsymbol{y}^+) = Evaluate(s, f(\boldsymbol{s}), \boldsymbol{s}^+, \boldsymbol{y})$.
8: **if** $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \prec \boldsymbol{y}$ **then**
9: $Q = \{\boldsymbol{z} \mid \boldsymbol{z} \in P, \boldsymbol{y} \preceq \boldsymbol{z}\}$.
10: $P = (P \setminus Q) \cup \{\boldsymbol{y}\}$.
11: $f(P) = (f(P) \setminus \{f(\boldsymbol{z}) \mid \boldsymbol{z} \in Q\}) \cup \{f(\boldsymbol{y})\}$.
12: $P^+ = (P^+ \setminus \{\boldsymbol{z}^+ \mid \boldsymbol{z} \in Q\}) \cup \{\boldsymbol{y}^+\}$.
13: **end if**
14: $t = t + 1$.
15: **end while**
16: **return** $\arg\min_{\boldsymbol{s}\in P, |\boldsymbol{s}|\leq k} f(\boldsymbol{s})$

### Acceleration

Note that in each iteration of POCSS, we need to evaluate the objective value of a newly generated solution $\boldsymbol{y}$, i.e., computing $f(\boldsymbol{y}) = \|\mathbf{A} - \boldsymbol{y}\boldsymbol{y}^+\mathbf{A}\|_F^2$, which is very time-consuming. For a matrix $\mathbf{S} \in \mathbb{R}^{m\times k}$, the time of computing $\mathbf{S}^+$ is $O(k^2m)$; then computing $\mathbf{S}\mathbf{S}^+\mathbf{A}$ requires $2kmn$ time; finally we need to compute $\|\cdot\|_F^2$, the time of which is $mn$. Thus, computing $f(\mathbf{S})$ directly costs time $O(kmn)$, which is expensive.

To accelerate this evaluation process, we build the recursive relation between the objective values by deleting one column from a matrix or inserting one column into a matrix, as shown in Lemmas 1-2. Thus, $f(\boldsymbol{y})$ can be computed from $f(\boldsymbol{s})$ by recursively deleting columns $\{\mathbf{A}_{:j} \mid s_j = 1 \wedge y_j = 0\}$ and then inserting columns $\{\mathbf{A}_{:j} \mid s_j = 0 \wedge y_j = 1\}$. From Lemmas 1-2, we can see that the recursive relation on $f$ relies on the Moore-Penrose inverse of the current matrix. Thus, we also need to update the Moore-Penrose inverse matrix accordingly, as shown in Lemmas 3-4. During the running process of POCSS, we use a set $P^+$ to maintain the Moore-Penrose inverse matrix for each solution in $P$, i.e., $P^+ = \{\boldsymbol{s}^+ \mid \boldsymbol{s} \in P\}$. $P^+$ will be updated accordingly in line 12 of Algorithm 1. Note that for the empty matrix $\boldsymbol{s} = \mathbf{0}$, $\boldsymbol{s}^+$ is also an empty matrix and $f(\boldsymbol{s}) = \|\mathbf{A}\|_F^2$.

In the following lemmas, we assume that $\mathbf{S}$ is a full column rank matrix, i.e., $rank(\mathbf{S}) = |\mathbf{S}|$. Their proofs are provided in the Appendix.

**Lemma 1.** *$\widehat{\mathbf{S}}$ is the matrix generated by deleting the $i$-th column of $\mathbf{S}$. Let $\boldsymbol{\rho} = ((\mathbf{S}^+)_{i:})^T$ and $\boldsymbol{\beta} = \mathbf{A}^T\boldsymbol{\rho}$. Then,*

$$f(\widehat{\mathbf{S}}) = f(\mathbf{S}) + \|\boldsymbol{\rho}\|_2^{-2}\boldsymbol{\beta}^T\boldsymbol{\beta}.$$

**Lemma 2.** *$\widehat{\mathbf{S}}$ is the matrix generated by inserting the $j$-th column of $\mathbf{A}$ into $\mathbf{S}$ as the $i$-th column. Let $\mathbf{E}_{:j} = \mathbf{A}_{:j} -$*

**Algorithm 2** Evaluation Subprocedure

**Input**: $\mathbf{S}$, $f(\mathbf{S})$, $\mathbf{S}^+$ and $\mathbf{Y}$
**Output**: $f(\mathbf{Y})$ and $\mathbf{Y}^+$
**Process**:

1: $f_{cur} = f(\mathbf{S})$, $\mathbf{S}^+_{cur} = \mathbf{S}^+$.
2: $X_{delete} = \{\mathbf{A}_{:j} \mid \mathbf{A}_{:j} \in \mathbf{S}, \mathbf{A}_{:j} \notin \mathbf{Y}\}$.
3: $X_{insert} = \{\mathbf{A}_{:j} \mid \mathbf{A}_{:j} \in \mathbf{Y}, \mathbf{A}_{:j} \notin \mathbf{S}\}$.
4: **for** each column $\mathbf{A}_{:j}$ in $X_{delete}$ **do**
5: $\quad\boldsymbol{\rho} = ((\mathbf{S}^+_{cur})_{i:})^T$ (where $\mathbf{A}_{:j}$ is the $i$-th column of $\mathbf{S}$).
6: $\quad\boldsymbol{\beta} = \mathbf{A}^T\boldsymbol{\rho}$.
7: $\quad\mathbf{S} \leftarrow$ delete the $i$-th column $\mathbf{A}_{:j}$ from $\mathbf{S}$.
8: $\quad f_{cur} \leftarrow f_{cur} + \|\boldsymbol{\rho}\|_2^{-2}\boldsymbol{\beta}^T\boldsymbol{\beta}$.
9: $\quad\mathbf{S}^+_{cur} \leftarrow$ delete the $i$-th row of $\mathbf{S}^+_{cur} - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+_{cur}\boldsymbol{\rho}\boldsymbol{\rho}^T$.
10: **end for**
11: **for** each column $\mathbf{A}_{:j}$ in $X_{insert}$ **do**
12: $\quad\mathbf{E}_{:j} = \mathbf{A}_{:j} - \mathbf{S}\mathbf{S}^+_{cur}\mathbf{A}_{:j}$ and $\boldsymbol{\delta} = \mathbf{A}^T\mathbf{E}_{:j}$.
13: $\quad\mathbf{S} \leftarrow$ insert $\mathbf{A}_{:j}$ into $\mathbf{S}$ as the $i$-th column.
14: $\quad f_{cur} \leftarrow f_{cur} - \boldsymbol{\delta}^T\boldsymbol{\delta}/\delta_j$.
15: $\quad\mathbf{S}^+_{cur} \leftarrow$ insert $((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1}(\mathbf{E}_{:j})^T$ into $(\mathbf{S}^+_{cur} - ((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1}\mathbf{S}^+_{cur}\mathbf{A}_{:j}(\mathbf{E}_{:j})^T)$ as the $i$-th row.
16: **end for**
17: $f(\mathbf{Y}) = f_{cur}$, $\mathbf{Y}^+ = \mathbf{S}^+_{cur}$.
18: **return** $f(\mathbf{Y})$ and $\mathbf{Y}^+$

$\mathbf{S}\mathbf{S}^+\mathbf{A}_{:j}$ *(i.e., the residual column vector of* $\mathbf{A}_{:j}$ *w.r.t.* $\mathbf{S}$*),* $\mathbf{E} = \mathbf{A} - \mathbf{S}\mathbf{S}^+\mathbf{A}$ *and* $\boldsymbol{\delta} = \mathbf{A}^T\mathbf{E}_{:j}$*. Then,*

$$f(\widehat{\mathbf{S}}) = f(\mathbf{S}) - \boldsymbol{\delta}^T\boldsymbol{\delta}/\delta_j,$$

*where* $\delta_j$ *denotes the* $j$*-th entry of the vector* $\boldsymbol{\delta}$*.*

**Lemma 3.** $\widehat{\mathbf{S}}$ *is the matrix generated by deleting the* $i$*-th column of* $\mathbf{S}$*. Let* $\boldsymbol{\rho} = ((\mathbf{S}^+)_{i:})^T$*. Then,* $\widehat{S}^+$ *is the matrix by deleting the* $i$*-th row of* $(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T)$*.*

**Lemma 4.** $\widehat{\mathbf{S}}$ *is the matrix generated by inserting the* $j$*-th column of* $\mathbf{A}$ *into* $\mathbf{S}$ *as the* $i$*-th column. Let* $\mathbf{E}_{:j} = \mathbf{A}_{:j} - \mathbf{S}\mathbf{S}^+\mathbf{A}_{:j}$*. Then,* $\widehat{\mathbf{S}}^+$ *is the matrix by inserting* $((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1}(\mathbf{E}_{:j})^T$ *into* $(\mathbf{S}^+ - ((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1}\mathbf{S}^+\mathbf{A}_{:j}(\mathbf{E}_{:j})^T)$ *as the* $i$*-th row.*

Algorithm 2 describes the procedure of computing $f(\mathbf{Y})$ and $\mathbf{Y}^+$ of a new matrix $\mathbf{Y}$ based on the known $f(\mathbf{S})$ and $\mathbf{S}^+$ of an old matrix $\mathbf{S}$. To generate $\mathbf{Y}$ from $\mathbf{S}$, $X_{delete}$ and $X_{insert}$ record the columns that need to be deleted and inserted, respectively. According to Lemmas 1 and 3, lines 4-10 of Algorithm 2 updates the current objective value $f_{cur}$ and the current Moore-Penrose inverse matrix $\mathbf{S}^+_{cur}$ by deleting columns in $X_{delete}$. According to Lemmas 2 and 4, lines 11-16 updates $f_{cur}$ and $\mathbf{S}^+_{cur}$ by inserting columns in $X_{insert}$. After line 16, the current matrix $\mathbf{S}$ is just the matrix $\mathbf{Y}$. The algorithm returns $f_{cur}$ and $\mathbf{S}^+_{cur}$ as $f(\mathbf{Y})$ and $\mathbf{Y}^+$.

We then show how the evaluation is accelerated. For deleting one column (lines 5-9), the time is determined by computing $\boldsymbol{\beta}$ and $\mathbf{S}^+_{cur}\boldsymbol{\rho}\boldsymbol{\rho}^T$, whose time is $mn$ and $2km$, respectively. For inserting one column (lines 12-15), the time is determined by computing $\mathbf{S}\mathbf{S}^+_{cur}\mathbf{A}_{:j}$, $\boldsymbol{\delta}$ and $\mathbf{S}^+_{cur}\mathbf{A}_{:j}(\mathbf{E}_{:j})^T$, whose time is $2km$, $mn$ and $2km$, respectively. Thus, both their time is in the order of $O(mn)$. Note that in line 6 of Algorithm 1, it will flip one bit in expectation, that is, it will delete or insert one column in expectation. This implies that the evaluation by Algorithm 2 costs time $O(mn)$ in expectation, which is much smaller than the time $O(kmn)$ of directly computing $f(\mathbf{Y})$.

## Theoretical Analysis

In this section, we investigate the theoretical performance of POCSS. For the CSS problem, we consider the equivalent maximization formulation:

$$\arg\max_{\mathbf{S}:\, a\ submatrix\ of\ \mathbf{A}} \ g(\mathbf{S}) = \|\mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2 \ \ s.t.\ \ |\mathbf{S}| \le k.$$

Let $\mathbf{S}_{opt}$ denote an optimal submatrix.

The objective function $g(\mathbf{S})$ can actually be seen as a set function by treating $\mathbf{S}$ as a set of columns. Then, it can be verified that $g(\mathbf{S})$ is monotone and non-submodular. The submodularity ratio $\gamma_{\mathbf{U},l}(g) = \min_{\mathbf{L}\subseteq\mathbf{U},\mathbf{S}:|\mathbf{S}|\le l,\mathbf{S}\cap\mathbf{L}=\emptyset} \frac{\sum_{\boldsymbol{v}\in\mathbf{S}}(g(\mathbf{L}\cup\{\boldsymbol{v}\})-g(\mathbf{L}))}{g(\mathbf{L}\cup\mathbf{S})-g(\mathbf{L})}$ (Das and Kempe 2011) can be used to measure the closeness of the function $g$ to submodular. Note that $\forall\mathbf{U}\subseteq\mathbf{A}, l \ge 1 : 0 \le \gamma_{\mathbf{U},l}(g) \le 1$.

In (Qian et al. 2016), it has been proved that the Pareto optimization method using at most $2ek^2n$ expected number of iterations can achieve an approximation ratio of $(1-e^{-\gamma})$ for maximizing a general monotone set function. By using the similar proof, we can prove the approximation bound of POCSS for maximizing the function $g$ as in Theorem 1, where $\mathbb{E}[T]$ denotes the expected number of iterations. The only difference is the size of the archive $P$ during optimization. By the procedure of POCSS, we know that the solutions maintained in $P$ must be incomparable. Thus, each value of one objective can correspond to at most one solution in $P$. Since $|\mathbf{S}| \in \{0, 1, \ldots, n\}$, $|P| \le n+1$. Note that for the proof in (Qian et al. 2016), $|P| \le 2k$. Thus, the upper bound on $\mathbb{E}[T]$ changes from $2ek^2n$ to $ekn(n+1)$ accordingly.

**Theorem 1.** *For the CSS problem, POCSS with* $\mathbb{E}[T] \le ekn(n+1)$ *finds a submatrix* $\mathbf{S}$ *of* $\mathbf{A}$ *with* $|\mathbf{S}| \le k$ *and*

$$g(\mathbf{S}) \ge (1 - e^{-\gamma_{\min}}) \cdot g(\mathbf{S}_{opt}),$$

*where* $\gamma_{\min} = \min_{\mathbf{U}:|\mathbf{U}|=k-1} \gamma_{\mathbf{U},k}(g)$.

Then, we further analyze the approximation guarantee of POCSS by allowing more than $k$ columns. Our analysis needs Lemma 5, i.e., Lemma 1 of (Bhaskara et al. 2016), which shows that for any submatrix $\mathbf{T}$, there always exists one column from a better submatrix, whose insertion into $\mathbf{T}$ can bring an improvement proportional to their current distance. Let $\sigma_{\min}(\mathbf{S}) = \inf_{\|\boldsymbol{\alpha}\|_2=1} \frac{\|\mathbf{S}_{norm}\boldsymbol{\alpha}\|_2^2}{\|\boldsymbol{\alpha}\|_2^2}$, where $\mathbf{S}_{norm}$ denotes the matrix by normalizing each column of $\mathbf{S}$ to a unit vector. That is, $\sigma_{\min}(\mathbf{S})$ is the smallest squared singular value of $\mathbf{S}_{norm}$. It can be verified that $\sigma_{\min}(\mathbf{S}) \le 1$, because

$$\begin{aligned}\sigma_{\min}(\mathbf{S}) &= \inf_{\|\boldsymbol{\alpha}\|_2=1} \|\sum\nolimits_{i=1}^{|\mathbf{S}|} \alpha_i(\mathbf{S}_{norm})_{:i}\|_2^2 \\ &\le \inf_{\|\boldsymbol{\alpha}\|_2=1} \sum_{i=1}^{|\mathbf{S}|} \|\alpha_i(\mathbf{S}_{norm})_{:i}\|_2^2 = \inf_{\|\boldsymbol{\alpha}\|_2=1} \sum_{i=1}^{|\mathbf{S}|} \alpha_i^2 = 1.\end{aligned}$$

**Lemma 5.** *(Bhaskara et al. 2016) Let* $\mathbf{S}$, $\mathbf{T}$ *be two submatrices of* $\mathbf{A}$ *with* $g(\mathbf{S}) \geq g(\mathbf{T})$. *There exists one column of* $\mathbf{S}$, *inserting which into* $\mathbf{T}$ *can produce a matrix* $\mathbf{T}'$ *such that*

$$g(\mathbf{T}') - g(\mathbf{T}) \geq \sigma_{min}(\mathbf{S}) \frac{(g(\mathbf{S}) - g(\mathbf{T}))^2}{4|\mathbf{S}|g(\mathbf{S})}.$$

By Lemma 5, we prove that taking slightly more than $k$ columns, POCSS can obtain a constant approximation ratio of $(1-\epsilon)$. The proof is inspired from the analysis of the greedy algorithm, i.e., Theorem 1 in (Bhaskara et al. 2016).

**Theorem 2.** *Let* $\epsilon$ *be any positive constant. For the CSS problem, POCSS with* $\mathbb{E}[T] \leq \frac{16e}{\epsilon\sigma_{\min}(\mathbf{S}_{opt})} kn(n+1)$ *finds a submatrix* $\mathbf{S}$ *of* $\mathbf{A}$ *with* $|\mathbf{S}| \leq \frac{16k}{\epsilon\sigma_{\min}(\mathbf{S}_{opt})}$ *and*

$$g(\mathbf{S}) \geq (1-\epsilon) \cdot g(\mathbf{S}_{opt}).$$

*Proof.* We use $\sigma$ to denote $\sigma_{\min}(\mathbf{S}_{opt})$ for convenience. Let $J_{\max}$ denote the maximum integer value of $j$ such that in the archive $P$, there exists a solution $\boldsymbol{s}$ with $|\boldsymbol{s}| \leq \frac{8k}{\sigma}\sum_{i=0}^{j-1} 2^i$ and $g(\boldsymbol{s}) \geq g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^j$. Note that the vector $\boldsymbol{s}_{opt}$ corresponds to the optimal submatrix $\mathbf{S}_{opt}$. That is,

$$J_{\max} = \max\{j \in \mathbb{N}_{\geq 0} \mid \exists \boldsymbol{s} \in P, \\ |\boldsymbol{s}| \leq \frac{8k}{\sigma}\sum_{i=0}^{j-1} 2^i \wedge g(\boldsymbol{s}) \geq g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^j\}.$$

Let $N$ be the minimum integer value of $i$ such that $1/2^i \leq \epsilon$, i.e., $1/2^{N-1} > \epsilon$ and $1/2^N \leq \epsilon$. We then only need to analyze the expected number of iterations until $J_{\max} = N$, since $J_{\max} = N$ implies that there exists one solution $\boldsymbol{s}$ in $P$ satisfying that

$$|\boldsymbol{s}| \leq \frac{8k}{\sigma}\sum_{i=0}^{N-1} 2^i = \frac{8k}{\sigma}(2^N - 1) \leq \frac{16k}{\sigma}2^{N-1} \leq \frac{16k}{\epsilon\sigma}$$
$$\text{and} \quad g(\boldsymbol{s}) \geq g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^N \geq (1-\epsilon)\cdot g(\boldsymbol{s}_{opt}).$$

The initial value of $J_{\max}$ is 0, since POCSS starts from the empty matrix $\mathbf{0}$, which has $|\mathbf{0}| = 0$ and $g(\mathbf{0}) = 0$. Assume that currently $J_{\max} = j < N$. Let $\boldsymbol{s}$ be a corresponding solution with the value $j$, i.e., $|\boldsymbol{s}| \leq \frac{8k}{\sigma}\sum_{i=0}^{j-1} 2^i$ and $g(\boldsymbol{s}) \geq g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^j$. It is easy to see that $J_{\max}$ cannot decrease because deleting $\boldsymbol{s}$ from $P$ (lines 9 and 10 of Algorithm 1) implies that the newly generated solution $\boldsymbol{y} \preceq \boldsymbol{s}$, which must satisfy that $|\boldsymbol{y}| \leq |\boldsymbol{s}|$ and $g(\boldsymbol{y}) \geq g(\boldsymbol{s})$.

Then, we show that $J_{\max}$ can increase by at least 1 after at most $\frac{8ekn(n+1)}{\sigma}2^j$ iterations in expectation. By Lemma 5, $g(\boldsymbol{s}) < g(\boldsymbol{s}_{opt})$ and $|\boldsymbol{s}_{opt}| \leq k$, we know that flipping one specific 0 bit of $\boldsymbol{s}$ (i.e., inserting one specific column) can generate a new solution $\boldsymbol{y}$, which satisfies that

$$g(\boldsymbol{y}) - g(\boldsymbol{s}) \geq \sigma\frac{(g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}))^2}{4kg(\boldsymbol{s}_{opt})} > \sigma\frac{(g(\boldsymbol{s}_{opt})/2^{j+1})^2}{4kg(\boldsymbol{s}_{opt})}.$$

The last inequality is derived by $g(\boldsymbol{s}) < g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^{j+1}$, i.e., $g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}) > g(\boldsymbol{s}_{opt})/2^{j+1}$, because otherwise, it implies that $J_{\max} \geq j+1$, contradicting with our assumption $J_{\max} = j$. This process happens in one iteration with probability at least $\frac{1}{|P|}\cdot\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en(n+1)}$, where $1/|P|$ is a lower bound on the probability of selecting $\boldsymbol{s}$ in line 5 of Algorithm 1 and $\frac{1}{n}(1-\frac{1}{n})^{n-1}$ is the probability of flipping a specific bit of $\boldsymbol{s}$ while keeping other bits unchanged in line 6. Note that $|P| \leq n+1$. Thus, after at most $en(n+1)$ iterations in expectation, there must exist one solution $\boldsymbol{s}_1$ in $P$ satisfying that $|\boldsymbol{s}_1| \leq |\boldsymbol{s}| + 1 \leq \frac{8k}{\sigma}\sum_{i=0}^{j-1} 2^i + 1$ and $g(\boldsymbol{s}_1) \geq g(\boldsymbol{s}) + \sigma\frac{(g(\boldsymbol{s}_{opt})/2^{j+1})^2}{4kg(\boldsymbol{s}_{opt})} \geq g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^j + \sigma\frac{(g(\boldsymbol{s}_{opt})/2^{j+1})^2}{4kg(\boldsymbol{s}_{opt})}$. We then consider such a process on $\boldsymbol{s}_1$, which can make the archive $P$ contain a solution $\boldsymbol{s}_2$ with $|\boldsymbol{s}_2| \leq |\boldsymbol{s}_1| + 1 \leq \frac{8k}{\sigma}\sum_{i=0}^{j-1} 2^i + 2$ and $g(\boldsymbol{s}_2) \geq g(\boldsymbol{s}_1) + \sigma\frac{(g(\boldsymbol{s}_{opt})/2^{j+1})^2}{4kg(\boldsymbol{s}_{opt})} \geq g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^j + 2\sigma\frac{(g(\boldsymbol{s}_{opt})/2^{j+1})^2}{4kg(\boldsymbol{s}_{opt})}$. By repeating this process $l = \frac{8k}{\sigma}2^j$ times (we pessimistically assume that $J_{\max}$ does not increase, thus it holds that $g(\boldsymbol{s}_i) < g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^{j+1}$ for any $i < l$), $P$ will contain a solution $\boldsymbol{s}_l$ satisfying that

$$|\boldsymbol{s}_l| \leq \frac{8k}{\sigma}\sum_{i=0}^{j-1} 2^i + l = \frac{8k}{\sigma}\sum_{i=0}^{j} 2^i \quad \text{and}$$
$$g(\boldsymbol{s}_l) \geq g(\boldsymbol{s}_{opt}) - \frac{g(\boldsymbol{s}_{opt})}{2^j} + l\sigma\frac{(g(\boldsymbol{s}_{opt})/2^{j+1})^2}{4kg(\boldsymbol{s}_{opt})}$$
$$= g(\boldsymbol{s}_{opt}) - g(\boldsymbol{s}_{opt})/2^{j+1}.$$

This implies that $J_{\max} \geq j+1$. Thus, after at most $l \cdot en(n+1) = \frac{8ekn(n+1)}{\sigma}2^j$ expected number of iterations, $J_{\max}$ can increase by at least 1.

Thus, the expected number of iterations for increasing $J_{\max}$ from 0 to $N$ is at most

$$\sum_{j=0}^{N-1}\frac{8ekn(n+1)}{\sigma}2^j \leq \frac{16ekn(n+1)}{\sigma}2^{N-1} \leq \frac{16e}{\epsilon\sigma}kn(n+1),$$

which is just the expected number of iterations $\mathbb{E}[T]$ for achieving an approximation ratio of $(1-\epsilon)$. □

## Experiments

In this section, we empirically evaluate the effectiveness of POCSS on 10 real-world data sets[1]. Five state-of-the-art algorithms for the CSS problem are compared, including **Two-stage** (Boutsidis, Mahoney, and Drineas 2009), **AprxSVD** (Civril and Magdon-Ismail 2012), **Greedy** (Farahat, Ghodsi, and Kamel 2011; Bhaskara et al. 2016), **IterFS** (Ordozgoiti, Canaval, and Mozo 2016) and **WA**$^*$ (Arai et al. 2016). The detailed introduction of these algorithms can be seen in the related work. For Two-stage, $2k$ columns are selected in the randomized stage. For WA$^*$, we implement its best version WA$^*$-b and set the parameter $\epsilon = 0.5$. For POCSS, the number of iterations is $ekn(n+1)$ as suggested by Theorem 1. To improve its efficiency, submatrices with at least $2k$ columns are excluded in the running process, thus the number of iterations is set to $2ek^2n$.

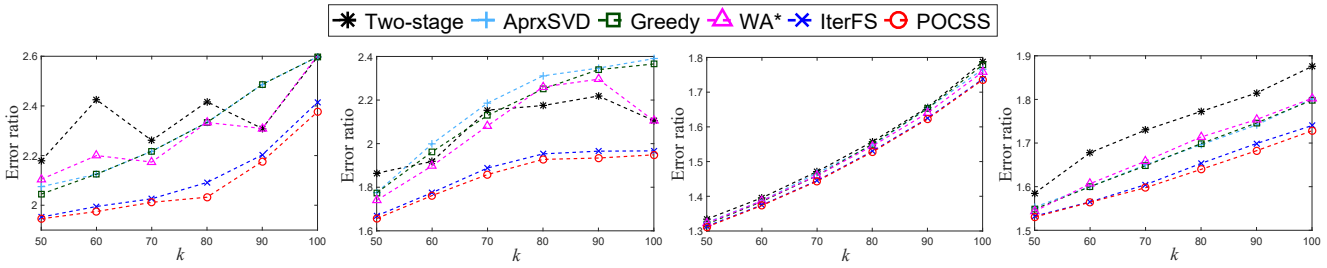
To evaluate an output submatrix $\mathbf{S}$, we measure the ratio of its reconstruction error $f(\mathbf{S})$ w.r.t. the smallest rank-$k$ approximation error obtained by SVD, i.e.,

$$\text{error ratio} = \|\mathbf{A} - \mathbf{S}\mathbf{S}^+\mathbf{A}\|_F^2 / \|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

[1]The data sets are downloaded from http://archive.ics.uci.edu/ml/ and http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. All feature vectors are normalized to unit vectors.

Table 1: The error ratio (the smaller the better) of the compared methods on 10 data sets for $k = 50$. The mean±std is reported for randomized algorithms. In each data set, the smallest values are bolded.

| data set | (#inst, #feat) | Two-stage | AprxSVD | Greedy | WA* | IterFS | POCSS |
|---|---|---|---|---|---|---|---|
| *sonar* | (208, 60) | 2.656 | 2.860 | 2.852 | 2.785 | **2.524±0.000** | **2.524±0.000** |
| *mediamill* | (30993, 120) | 2.179 | 2.075 | 2.045 | 2.105 | 1.952±0.027 | **1.945±0.022** |
| *ComCri* | (2215, 147) | 1.265 | 1.197 | 1.195 | 1.203 | 1.197±0.004 | **1.194±0.027** |
| *musk* | (7074, 168) | 1.868 | 1.775 | 1.774 | 1.740 | 1.669±0.013 | **1.656±0.012** |
| *dna* | (2000, 180) | 1.334 | 1.325 | 1.320 | 1.323 | 1.313±0.002 | **1.311±0.000** |
| *Arrhythmia* | (452, 279) | 1.734 | 1.560 | 1.551 | 1.555 | 1.535±0.006 | **1.520±0.018** |
| *scene* | (1211, 294) | 1.585 | 1.553 | 1.548 | 1.544 | 1.533±0.002 | **1.530±0.003** |
| *RlCTs* | (53500, 386) | 1.535 | 1.432 | 1.433 | 1.446 | 1.452±0.051 | **1.420±0.003** |
| *madelon* | (2000, 500) | 1.123 | 1.057 | **1.056** | **1.056** | **1.056±0.000** | **1.056±0.000** |
| *sEMG* | (1800, 2500) | 1.259 | 1.226 | 1.224 | 1.224 | 1.218±0.001 | **1.216±0.001** |
| average rank | | 5.7 | 4.55 | 3.3 | 3.8 | 2.45 | 1.2 |



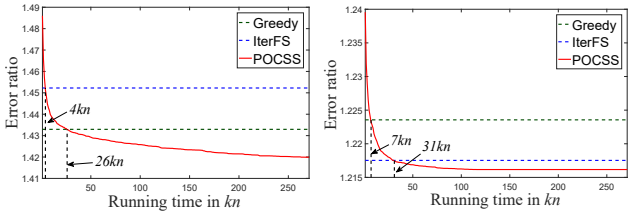

(a) *mediamill* (b) *musk* (c) *dna* (d) *scene*

Figure 1: The comparison for cardinality constraints $k \in \{50, 60, \ldots, 100\}$ (error ratio: the smaller the better).

The error ratio is obviously always larger than 1. The smaller it is, the better. For POCSS, Two-stage and IterFS, which are randomized algorithms, we repeat the run 10 times independently and report the average results.

The results for $k = 50$ are shown in Table 1. POCSS achieves the smallest error ratio on all the data sets. Note that we do not report the standard deviations of Two-stage, because they are always 0, as observed in (Ordozgoiti, Canaval, and Mozo 2016). We compute the rank of each method on each data set as in (Demšar 2006), which are averaged in the last row of Table 1. We can observe the order "POCSS $<$ IterFS $<$ Greedy $<$ WA* $<$ AprxSVD $<$ Two-stage", which are consistent with "IterFS $<$ Greedy $<$ Two-stage" in (Ordozgoiti, Canaval, and Mozo 2016) as well as "WA* $<$ Two-stage" in (Arai et al. 2016). Note that although IterFS is overall the second best, its performance is not very stable. For example, on the *RlCTs* data set, IterFS is worse than AprxSVD, Greedy and WA*. Figure 1 examines the influence of the cardinality constraint $k$. We can observe that POCSS is consistently better than other methods.

Considering the running time, measured by the number of objective function evaluations, POCSS is set to use the theoretical upper bound $2ek^2n$, i.e., the worst-case time. We also empirically examine how effective POCSS is in practice. For the two data sets *RlCTs* and *sEMG* with $k = 50$, we plot the curve of the error ratio over the running time for POCSS and select Greedy and IterFS as the baselines. We




(a) *RlCTs* (b) *sEMG*

Figure 2: Performance v.s. running time of POCSS.

can observe from Figure 2 that the time of POCSS to obtain a better performance is much less than the worst-case time $2ek^2n \approx 271kn$, implying that POCSS can be efficient in practice.

## Conclusion

In this paper, we study the CSS problem for unsupervised feature selection, and propose the new algorithm POCSS by Pareto optimization. POCSS employs a randomized iterative procedure to minimize the reconstruction error and the number of selected features simultaneously. We prove that POCSS can achieve a good approximation guarantee. Empirical results also show its excellent performance.

# Appendix

To prove Lemma 1, which shows the recursive relation on $f(\mathbf{S})$ by deleting one column from $\mathbf{S}$, we first analyze the recursive relation on $\mathbf{E} = \mathbf{A} - \mathbf{S}\mathbf{S}^+\mathbf{A}$.

**Lemma 6.** *$\widehat{\mathbf{S}}$ is the matrix generated by deleting the $i$-th column of $\mathbf{S}$. Let $\boldsymbol{\rho} = ((\mathbf{S}^+)_{i:})^T$. Then,*

$$\widehat{\mathbf{E}} = \mathbf{A} - \widehat{\mathbf{S}}\widehat{\mathbf{S}}^+\mathbf{A} = \mathbf{E} + \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A}.$$

*Proof.* According to Lemma 3, $\widehat{\mathbf{S}}^+$ is the matrix generated by deleting the $i$-th row of $(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T)$. Thus,

$$\begin{aligned}\widehat{\mathbf{S}}\widehat{\mathbf{S}}^+ &= \mathbf{S}(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T) - \mathbf{S}_{:i}(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T)_{i:} \\ &= \mathbf{S}(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T),\end{aligned}$$

where the last equality is by $(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T)_{i:} = \boldsymbol{\rho}^T - \|\boldsymbol{\rho}\|_2^{-2}\boldsymbol{\rho}^T\boldsymbol{\rho}\boldsymbol{\rho}^T = \boldsymbol{\rho}^T - \boldsymbol{\rho}^T = \mathbf{0}$. Then, we have

$$\begin{aligned}\widehat{\mathbf{E}} &= \mathbf{A} - \widehat{\mathbf{S}}\widehat{\mathbf{S}}^+\mathbf{A} = \mathbf{A} - (\mathbf{S}\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T)\mathbf{A} \\ &= \mathbf{E} + \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A}.\end{aligned}$$ □

Based on Lemma 6 and $f(\mathbf{S}) = \|\mathbf{E}\|_F^2$, we can update the objective $f$ when deleting one column from a matrix.

**Proof of Lemma 1.**

$$\begin{aligned}f(\widehat{\mathbf{S}}) &= \|\widehat{\mathbf{E}}\|_F^2 = tr(\widehat{\mathbf{E}}^T\widehat{\mathbf{E}}) \\ &= tr((\mathbf{E}^T + \|\boldsymbol{\rho}\|_2^{-2}\mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+)(\mathbf{E} + \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A})) \\ &= tr(\mathbf{E}^T\mathbf{E}) + 2\|\boldsymbol{\rho}\|_2^{-2}tr(\mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\mathbf{E}) \\ &\quad + \|\boldsymbol{\rho}\|_2^{-4}\cdot tr(\mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A}),\end{aligned}$$

where the third equality is by Lemma 6 and $(\mathbf{S}\mathbf{S}^+)^T = \mathbf{S}\mathbf{S}^+$. Since $\mathbf{S}$ is assumed to have full column rank, $\mathbf{S}^+\mathbf{S} = \mathbf{I}_{|\mathbf{S}|}$. Then, we have

$$\begin{aligned}\mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\mathbf{E} &= \mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+(\mathbf{A} - \mathbf{S}\mathbf{S}^+\mathbf{A}) \\ &= \mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\mathbf{A} - \mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\mathbf{A} = \mathbf{0}_{n,n}.\end{aligned}$$

Since $\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+ = (\mathbf{S}^+)_{i:}\mathbf{S}\mathbf{S}^+ = (\mathbf{S}^+\mathbf{S})_{i:}\mathbf{S}^+ = (\mathbf{I}_{|\mathbf{S}|})_{i:}\mathbf{S}^+ = (\mathbf{S}^+)_{i:} = \boldsymbol{\rho}^T$, we have

$$\begin{aligned}\mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A} &= \mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{S}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A} \\ &= \mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A} = \|\boldsymbol{\rho}\|_2^2\mathbf{A}^T\boldsymbol{\rho}\boldsymbol{\rho}^T\mathbf{A} = \|\boldsymbol{\rho}\|_2^2\boldsymbol{\beta}\boldsymbol{\beta}^T.\end{aligned}$$

Applying the above two equations to $f(\widehat{\mathbf{S}})$, we get $f(\widehat{\mathbf{S}}) = tr(\mathbf{E}^T\mathbf{E}) + \|\boldsymbol{\rho}\|_2^{-2}tr(\boldsymbol{\beta}\boldsymbol{\beta}^T) = f(\mathbf{S}) + \|\boldsymbol{\rho}\|_2^{-2}\boldsymbol{\beta}^T\boldsymbol{\beta}$. □

To prove Lemma 2, which shows the recursive relation on $f(\mathbf{S})$ by inserting one column into $\mathbf{S}$, we use the recursive relation on $\mathbf{E}^T\mathbf{E}$, which has already been derived in Corollary 3 of (Farahat, Ghodsi, and Kamel 2011).

**Proof of Lemma 2.** Let $\widehat{\mathbf{E}} = \mathbf{A} - \widehat{\mathbf{S}}\widehat{\mathbf{S}}^+\mathbf{A}$. According to Corollary 3 in (Farahat, Ghodsi, and Kamel 2011), we get

$$\widehat{\mathbf{E}}^T\widehat{\mathbf{E}} = \mathbf{E}^T\mathbf{E} - \mathbf{E}^T\mathbf{E}_{:j}(\mathbf{E}^T\mathbf{E}_{:j})^T / \left((\mathbf{E}^T)_{j:}\mathbf{E}_{:j}\right).$$

We then show that $\mathbf{E}^T\mathbf{E}_{:j}$ is just $\boldsymbol{\delta}$. By $(\mathbf{S}\mathbf{S}^+)^T = \mathbf{S}\mathbf{S}^+$ and $\mathbf{S}^+\mathbf{S} = \mathbf{I}_{|\mathbf{S}|}$, we get

$$\begin{aligned}\boldsymbol{\delta} &= \mathbf{A}^T\mathbf{E}_{:j} = (\mathbf{E}^T + \mathbf{A}^T\mathbf{S}\mathbf{S}^+)\mathbf{E}_{:j} = \mathbf{E}^T\mathbf{E}_{:j} + \mathbf{A}^T\mathbf{S}\mathbf{S}^+\mathbf{E}_{:j} \\ &= \mathbf{E}^T\mathbf{E}_{:j} + \mathbf{A}^T\mathbf{S}\mathbf{S}^+(\mathbf{A}_{:j} - \mathbf{S}\mathbf{S}^+\mathbf{A}_{:j}) \\ &= \mathbf{E}^T\mathbf{E}_{:j} + \mathbf{A}^T\mathbf{S}\mathbf{S}^+\mathbf{A}_{:j} - \mathbf{A}^T\mathbf{S}\mathbf{S}^+\mathbf{S}\mathbf{S}^+\mathbf{A}_{:j} = \mathbf{E}^T\mathbf{E}_{:j}.\end{aligned}$$

It is also clear that $(\mathbf{E}^T)_{j:}\mathbf{E}_{:j} = \delta_j$. Thus, we have

$$\begin{aligned}f(\widehat{\mathbf{S}}) &= \|\widehat{\mathbf{E}}\|_F^2 = tr(\widehat{\mathbf{E}}^T\widehat{\mathbf{E}}) = tr(\mathbf{E}^T\mathbf{E}) - tr(\boldsymbol{\delta}\boldsymbol{\delta}^T)/\delta_j \\ &= f(\mathbf{S}) - \boldsymbol{\delta}^T\boldsymbol{\delta}/\delta_j.\end{aligned}$$ □

Let $\widetilde{\mathbf{S}}$ and $\widehat{\mathbf{S}}$ denote the matrix by zeroing-out and deleting the $i$-th column of $\mathbf{S}$, respectively. Lemma 7, i.e., Proposition 1 in (Ordozgoiti, Canaval, and Mozo 2016), gives the Moore-Penrose inverse of $\widetilde{\mathbf{S}}$. By further investigating the relation between the Moore-Penrose inverse of $\widetilde{\mathbf{S}}$ and $\widehat{\mathbf{S}}$, we can prove Lemma 3, which shows the recursive relation on $\mathbf{S}^+$ by deleting one column from a matrix.

**Lemma 7.** *(Ordozgoiti, Canaval, and Mozo 2016) $\widetilde{\mathbf{S}}$ is the matrix resulting from zeroing-out the $i$-th column of $\mathbf{S}$, i.e., the $i$-th column of $\widetilde{\mathbf{S}}$ is comprised by all zeros. Let $\boldsymbol{\rho} = ((\mathbf{S}^+)_{i:})^T$. Then, $\widetilde{\mathbf{S}}^+ = \mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T$.*

**Proof of Lemma 3.** Since the columns of $\widetilde{\mathbf{S}}$ and $\widehat{\mathbf{S}}$ span the same space, their projection matrices are the same. That is, $\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^+ = \widehat{\mathbf{S}}\widehat{\mathbf{S}}^+$. Let $\mathbf{B}$ denote the matrix by deleting the $i$-th row of $\widetilde{\mathbf{S}}^+$. Then, $\widetilde{\mathbf{S}}\widetilde{\mathbf{S}}^+ = \widehat{\mathbf{S}}\mathbf{B}$, since the $i$-th column of $\widetilde{\mathbf{S}}$ contains all zeros. Thus, $\widehat{\mathbf{S}}\mathbf{B} = \widehat{\mathbf{S}}\widehat{\mathbf{S}}^+$, implying that $\mathbf{B} = \widehat{\mathbf{S}}^+\widehat{\mathbf{S}}\widehat{\mathbf{S}}^+ = \widehat{\mathbf{S}}^+$. According to Lemma 7, $\widehat{\mathbf{S}}^+$ is the matrix by deleting the $i$-th row of $(\mathbf{S}^+ - \|\boldsymbol{\rho}\|_2^{-2}\mathbf{S}^+\boldsymbol{\rho}\boldsymbol{\rho}^T)$. Thus, the lemma holds. □

Lemma 4 shows the recursive relation on $\mathbf{S}^+$ by inserting one column into a matrix. Our proof idea is to first analyze the Moore-Penrose inverse of the resulting matrix when the inserting position is the last column, and then analyze the influence on the Moore-Penrose inverse by switching two columns of a matrix.

**Proof of Lemma 4.** We first insert the $j$-th column of $\mathbf{A}$ into $\mathbf{S}$ as its last column, and compute the Moore-Penrose inverse of the resulting matrix $[\mathbf{S}\ \ \mathbf{A}_{:j}]$. For notational simplicity, let $\boldsymbol{v} = \mathbf{A}_{:j}$ and $\mathbf{B} = \mathbf{S}^T\mathbf{S}$.

$$\begin{aligned}[\mathbf{S}\ \ \boldsymbol{v}]^+ &= \left(\begin{bmatrix}\mathbf{S}^T\\ \boldsymbol{v}^T\end{bmatrix}[\mathbf{S}\ \ \boldsymbol{v}]\right)^{-1}\begin{bmatrix}\mathbf{S}^T\\ \boldsymbol{v}^T\end{bmatrix} = \begin{bmatrix}\mathbf{B} & \mathbf{S}^T\boldsymbol{v}\\ \boldsymbol{v}^T\mathbf{S} & \boldsymbol{v}^T\boldsymbol{v}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{S}^T\\ \boldsymbol{v}^T\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{B}^{-1} + \tau\mathbf{B}^{-1}\mathbf{S}^T\boldsymbol{v}\boldsymbol{v}^T\mathbf{S}\mathbf{B}^{-1} & -\tau\mathbf{B}^{-1}\mathbf{S}^T\boldsymbol{v}\\ -\tau\boldsymbol{v}^T\mathbf{S}\mathbf{B}^{-1} & \tau\end{bmatrix}\begin{bmatrix}\mathbf{S}^T\\ \boldsymbol{v}^T\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{B}^{-1}\mathbf{S}^T + \tau\mathbf{B}^{-1}\mathbf{S}^T\boldsymbol{v}\boldsymbol{v}^T\mathbf{S}\mathbf{B}^{-1}\mathbf{S}^T - \tau\mathbf{B}^{-1}\mathbf{S}^T\boldsymbol{v}\boldsymbol{v}^T\\ -\tau\boldsymbol{v}^T\mathbf{S}\mathbf{B}^{-1}\mathbf{S}^T + \tau\boldsymbol{v}^T\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{S}^+ + \tau\mathbf{S}^+\boldsymbol{v}\boldsymbol{v}^T\mathbf{S}\mathbf{S}^+ - \tau\mathbf{S}^+\boldsymbol{v}\boldsymbol{v}^T\\ -\tau\boldsymbol{v}^T\mathbf{S}\mathbf{S}^+ + \tau\boldsymbol{v}^T\end{bmatrix},\end{aligned}$$

where the first equality is by the definition of the Moore-Penrose inverse of a full column rank matrix, the third equality is by block matrix inversion and

$$\begin{aligned}\tau &= (\boldsymbol{v}^T\boldsymbol{v} - \boldsymbol{v}^T\mathbf{S}\mathbf{B}^{-1}\mathbf{S}^T\boldsymbol{v})^{-1} \\ &= ((\boldsymbol{v} - \mathbf{S}\mathbf{S}^+\boldsymbol{v})^T(\boldsymbol{v} - \mathbf{S}\mathbf{S}^+\boldsymbol{v}))^{-1} = ((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1},\end{aligned}$$

and the fourth equality is by block matrix multiplication. Since $\boldsymbol{v}^T\mathbf{S}\mathbf{S}^+ = (\mathbf{S}\mathbf{S}^+\boldsymbol{v})^T = \boldsymbol{v}^T - (\mathbf{E}_{:j})^T$, we have

$$[\mathbf{S}\ \ \boldsymbol{v}]^+ = \begin{bmatrix}\mathbf{S}^+ - ((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1}\mathbf{S}^+\boldsymbol{v}(\mathbf{E}_{:j})^T\\ ((\mathbf{E}_{:j})^T\mathbf{E}_{:j})^{-1}(\mathbf{E}_{:j})^T\end{bmatrix}. \quad (1)$$

Then, we show that if $\widetilde{\mathbf{C}}$ is the matrix generated by switching the $i$-th and $j$-th columns of a matrix $\mathbf{C}$, $\widetilde{\mathbf{C}}^+$ is the matrix that swaps the $i$-th row and $j$-th row in $\mathbf{C}^+$. Let $\mathbf{T}$ denote the elementary matrix obtained by swapping the $i$-th row and $j$-th row of the identity matrix $\mathbf{I}_{|\mathbf{C}|}$. Then, $\widetilde{\mathbf{C}} = \mathbf{CT}$. According to Corollary 2.3 in (Cline and Greville 1970), we get

$$\begin{aligned}\widetilde{\mathbf{C}}^+ &= (\mathbf{CT})^+ = (\mathbf{C}^+\mathbf{CT})^+(\mathbf{CTT}^+)^+ \\ &= (\mathbf{I}_{|\mathbf{C}|}\mathbf{T})^+(\mathbf{CI}_{|\mathbf{C}|})^+ = \mathbf{T}^+\mathbf{C}^+ = \mathbf{TC}^+.\end{aligned}$$

Since $\mathbf{TC}^+$ is just swapping the $i$-th row and $j$-th row in $\mathbf{C}^+$, our claim holds.

Note that the matrix $\widehat{\mathbf{S}}$ we are to analyze is generated by inserting the $j$-th column of $\mathbf{A}$ into $\mathbf{S}$ as the $i$-th column. Thus, by combining Eq. (1) (where $\boldsymbol{v} = \mathbf{A}_{:j}$) with the above claim, the lemma holds. □

## References


Arai, H.; Maung, C.; Xu, K.; and Schweitzer, H. 2016. Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In *AAAI*, 666–672.

Arai, H.; Maung, C.; and Schweitzer, H. 2015. Optimal column subset selection by A-star search. In *AAAI*, 1079–1085.

Bhaskara, A.; Rostamizadeh, A.; Altschuler, J.; Zadimoghaddam, M.; Fu, T.; and Mirrokni, V. 2016. Greedy column subset selection: New bounds and distributed algorithms. In *ICML*, 2539–2548.

Boutsidis, C.; Mahoney, M. W.; and Drineas, P. 2009. An improved approximation algorithm for the column subset selection problem. In *SODA*, 968–977.

Çivril, A. 2014. Column subset selection problem is UG-hard. *Journal of Computer and System Sciences* 80(4):849–859.

Chan, T. F., and Hansen, P. C. 1992. Some applications of the rank revealing QR factorization. *SIAM Journal on Scientific and Statistical Computing* 13(3):727–741.

Civril, A., and Magdon-Ismail, M. 2012. Column subset selection via sparse approximation of SVD. *Theoretical Computer Science* 421:1–14.

Cline, R. E., and Greville, T. N. E. 1970. An extension of the generalized inverse of a matrix. *SIAM Journal on Applied Mathematics* 19(4):682–688.

Cohen, M. B.; Elder, S.; Musco, C.; Musco, C.; and Persu, M. 2015. Dimensionality reduction for $k$-means clustering and low rank approximation. In *STOC*, 163–172.

Das, A., and Kempe, D. 2011. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *ICML*, 1057–1064.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7:1–30.

Drineas, P.; Mahoney, M.; and Muthukrishnan, S. 2008. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications* 30(2):844–881.

Farahat, A. K.; Ghodsi, A.; and Kamel, M. S. 2011. An efficient greedy method for unsupervised feature selection. In *ICDM*, 161–170.

Friedrich, T., and Neumann, F. 2015. Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evolutionary Computation* 23(4):543–558.

Golub, G. H., and Reinsch, C. 1971. Singular value decomposition and least squares solutions. In *Linear Algebra*. 134–151.

Gu, M., and Eisenstat, S. C. 1996. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing* 17(4):848–869.

Guruswami, V., and Sinop, A. K. 2012. Optimal column-based low-rank matrix reconstruction. In *SODA*, 1207–1214.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hoog, F. D., and Mattheij, R. 2007. Subset selection for matrices. *Linear Algebra and its Applications* 422(2):349 – 359.

Jolliffe, I. 2011. Principal component analysis. In *International Encyclopedia of Statistical Science*. 1094–1096.

Ordozgoiti, B.; Canaval, S. G.; and Mozo, A. 2016. A fast iterative algorithm for improved unsupervised feature selection. In *ICDM*, 390–399.

Ordozgoiti, B.; Canaval, S. G.; and Mozo, A. 2018. Iterative column subset selection. *Knowledge and Information Systems* 54(1):65–94.

Pan, C.-T., and Tang, P. T. P. 1999. Bounds on singular values revealed by QR factorizations. *BIT Numerical Mathematics* 39(4):740–756.

Qian, C.; Shi, J.-C.; Yu, Y.; Tang, K.; and Zhou, Z.-H. 2016. Parallel Pareto optimization for subset selection. In *IJCAI*, 1939–1945.

Qian, C.; Shi, J.-C.; Yu, Y.; Tang, K.; and Zhou, Z.-H. 2017a. Subset selection under noise. In *NIPS*, 3562–3572.

Qian, C.; Shi, J.-C.; Yu, Y.; and Tang, K. 2017b. On subset selection with general cost constraints. In *IJCAI*, 2613–2619.

Qian, C.; Shi, J.-C.; Tang, K.; and Zhou, Z.-H. 2018a. Constrained monotone $k$-submodular function maximization using multi-objective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation* 22(4):595–608.

Qian, C.; Zhang, Y.; Tang, K.; and Yao, X. 2018b. On multiset selection with size constraints. In *AAAI*, 1395–1402.

Qian, C.; Feng, C.; and Tang, K. 2018. Sequence selection by Pareto optimization. In *IJCAI*, 1485–1491.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2015. Subset selection by Pareto optimization. In *NIPS*, 1774–1782.

Shitov, Y. 2017. Column subset selection is NP-complete. *arXiv:1701.02764*.