2016 IEEE 16th International Conference on Data Mining

A Fast Iterative Algorithm for Improved Unsupervised Feature Selection

Bruno Ordozgoiti, Sandra Gómez Canaval, Alberto Mozo
Departamento de Sistemas Informáticos, Universidad Politécnica de Madrid
Calle Alan Turing, s/n, 28031 Madrid
bruno.ordozgoiti@upm.es

***Abstract*—Dimensionality reduction is often a crucial step for the successful application of machine learning and data mining methods. One way to achieve said reduction is feature selection. Due to the impossibility of labelling many data sets, unsupervised approaches are frequently the only option. The column subset selection problem translates naturally to this purpose, and has received considerable attention over the last few years, as it provides simple linear models for data reconstruction. Existing methods, however, often achieve approximation errors that are far from the optimum. In this paper we present a novel algorithm for column subset selection that consistently outperforms state-of-the-art methods in approximation error. We present a series of key derivations that allow an efficient implementation, making it comparable in speed and in some cases faster than other algorithms. We also prove results that make it possible to deal with huge matrices, which has strong implications for other algorithms of this type in the big data field. We validate our claims through experiments on a wide variety of well-known data sets.**

I. Introduction

Over the last few years, machine learning and data mining have proved to be useful for many applications such as automated classification, forecasting and anomaly detection. However, data are often high dimensional, which poses certain challenges. First, this entails what is known as the curse of dimensionality, which makes learning tasks more difficult. Second, high dimensionality sometimes has a dramatic impact on performance.

Many techniques can be used for dimensionality reduction, such as principal component analysis [1], the singular value decomposition [2] or autoencoders [3]. These methods transform the data into a new, lower dimensional subspace which captures the majority of the information of the original records. This representation, however, can be difficult to interpret for domain experts, and requires all features to be collected for their subsequent storage in compressed form. To overcome thiese issues, feature selection methods can be employed [4]. These techniques eliminate redundant or uninformative features in favor of the ones considered to be the most relevant in some regard, which results in a lower-dimensional and hence more manageable data set.

There exist various methods for supervised feature selection, tipically trying to keep the features that bear more information about the target values [5] [6] [7]. Supervised approaches, however, require labelled datasets, and the results are tied to one specific learning objective. An alternative is the use of unsupervised feature selection methods, which need no labels and can be used in conjunction with any learning algorithm. The column subset selection problem [8], [9] provides a framework that translates naturally to unsupervised feature selection, and can be formulated as follows.

The research leading to these results has received funding from the European Union under the FP7 grant agreement n. 619633 (project ONTIC) and H2020 grant agreement n. 671625 (project CogNet).

Definition 1. *Column Subset Selection Problem (CSSP). Given a matrix $A \in \mathbb{R}^{m\times n}$ and a positive integer k smaller than the rank of A, let $\mathcal{A}_k$ denote the set of $m \times k$ matrices comprised of k columns of A. Find C such that*

$$C = \underset{X\in\mathcal{A}_k}{\operatorname{argmin}} \|A - XX^+A\|_F \tag{1}$$

where X^+ is the Moore-Penrose pseudoinverse of X.

The solution to the CSSP is the subset of columns with which we can best rebuild the rest of our matrix using linear combinations of the chosen ones. This can be seen as a form of low rank approximation constraining the basis to be formed by a subset of columns of the original matrix. It is therefore akin to the singular value decomposition, but the models it produces retain the original features, which can be useful in certain domains. The CSSP is believed to be NP-Hard, so no existing algorithm can find the solution efficiently.

Some of the first methods that relate to this problem are the algorithms for rank-revealing QR factorizations, which were motivated by the need to solve ill-conditioned least squares problems [10], [11]. Given a matrix A, these methods seek a permutation matrix Π such that the QR factorization of $A\Pi$

$$A\Pi = Q\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

results in a lower-right block of R with a small spectral norm $\|R_{22}\|_2$, revealing the numerical rank of A [12], [13]. Algorithms of this kind –or others following similar criteria pertaining to the singular values– are thus capable of identifying or discarding numerically redundant sets of columns in rank-defficient matrices [8], [14], but do not provide the solution to the CSSP in general. This problem has received considerable attention during the last few years, leading to a variety of algorithms and theoretical results. We discuss some of the most significant contributions in section II.

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.116

In this paper we propose a novel approach to find approximate solutions to the CSSP. It is based on the following key idea: some well-known methods attempt to find good column subsets by choosing features one by one, or by examining the properties of individual features. Our method, on the contrary, starts from a random subset and updates feature choices taking the entirety of the subset into account, yielding significantly better results than other state-of-the-art algorithms. Since a straightforward implementation of this approach would be very inefficient, we derive a series of non-trivial optimizations that make it possible to draw subsets of tens or hundreds of features in a few seconds or miliseconds. Even though our algorithm can iterate to improve its initial selection, we show in our experiments that a single iteration is often enough to provide better subsets than other methods. This makes our proposal comparable in speed to some of the most efficient previous proposals, and even faster in some cases, while producing better results. Finally, we prove results that allow us to deal with huge matrices and that have implications for other algorithms of this kind in the context of big data.

The rest of this paper is structured as follows. Section II provides an overview of related research. Section III describes our approach in detail and provides relevant proofs. Section IV shows our experimental results and section V offers concluding remarks.

II. Related work

The problem of unsupervised feature selection, and in particular column subset selection, has received considerable attention during the last few years. In [15] a similarity measure is proposed to select homogeneous feature subsets, keeping only a representative of each of them at the end. In [16], an expectation-maximization based approach is proposed, along with two different criteria to evaluate the chosen subsets. Other approaches for unsupervised feature selection include [17], based on Laplacian scores; [18], which proposes a unified framework for supervised and unsupervised feature selection; and [19] which seeks the feature subset that best preserves the cluster structure of the data. Many proposals focus on the column subset selection problem (CSSP), which is the problem we tackle in this paper. A celebrated method is the CUR decomposition [20], which randomly selects rows as well as columns based on the idea of statistical leverage, i.e. rows and columns that seem to be strongly influential on optimal low-rank reconstructions are favored. The idea of statistical leverage is also employed in [9], where various candidate subsets are randomly sampled and then reduced using a rank-revealing factorization. Theoretical guarantees for this algorithm can be found in [21]. Other recent works presenting interesting theoretical results can be found in [22], [23], [24]. Interesting recent approaches include [25], which proposes feature elimination by a regularized coefficient matrix, and [26], where an algorithm based on A-star search with optimal guarantees is proposed. The fact that this algorithm is guaranteed to find optimal subsets is remarkable, although it can only do so in reasonable amounts of time for small values of k, and the problem remains intractable when the involved parameters increase slightly. In [27], a method for greedy selection of features is proposed. The approach is very efficient, although as we point out in section III it can fail in simple cases. A parallelized alternative for very high-dimensional data is offered in [28].

Among the discussed methods, those that do not tackle the CSSP do not provide good linear approximations in general. The ones that do address on the CSSP are sometimes focused on theoretical properties and to the best of our knowledge are not always successful in practice or are not easy to implement. Certain methods, like [27] and [9], do perform well in practice but as we show in this paper, their approximations can be outperformed.

III. Proposed algorithm

A. Notation

- C^+ denotes the Moore-Penrose generalized inverse of C.
- H_k^i denotes a diagonal $k \times k$ matrix whose entries are all 1, except for element H_{ii} which is zero.
- For a matrix A and a set R, A_R is the submatrix of A comprised by the columns whose indices are the elements of the set R.
- $A_{i:}$ is the i-th row and $A_{:i}$ the i-th column of A.
- Given $A \in \mathbb{R}^{m \times n}$, $A \backslash i$ is a $m \times n - 1$ submatrix of A resulting from the removal of column i.
- In our pseudocode we employ the function uniSampleWithoutReplacement(S, k), which represents a sample of $k \in \mathbb{N}$ elements drawn uniformly at random without replacement from the set S.
- In our pseudocode, for a set R and some $i \in \mathbb{N}$, if we employ the notation $R[i]$ we consider the set to be ordered and $R[i]$ to be its i-th element.
- Lowercase light letters such as f, δ denote vectors. f_i is the i-th element of vector f. Context is sufficient to distinguish vectors from scalars.
- e_i is the i-th vector of the canonical basis of the indicated dimensionality.
- $\circ$ denotes an element-wise vector multiplication operator.
- Given two matrices A and B, $(A|B)$ is the matrix resulting from appending the columns of B to A. E.g., if $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{m \times k}$, then $(A|B) \in \mathbb{R}^{m \times n+k}$ and consists of the columns of both matrices.

B. Motivation of our algorithm

We start by detailing the motivation of our method, and we do so by examining two well-known algorithms for the column subset selection problem (CSSP) in detail. First we analyze the greedy algorithm proposed in [27]. This method picks columns one by one, choosing the one that minimizes the objective function accounting for the ones chosen before. This criterion, however, does not account for the fact that more columns will be chosen afterwards. The example below illustrates how a greedy approach might fail. Let us consider the following matrix.

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1+\epsilon & 0 \\ 1 & 0 & 0 & 1+\epsilon \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

for some small but non-negligible ϵ. Let us consider that we want to pick k columns. The best choice for $k = 1$ is the first column. However, that column does not belong to the best subset for $k = 2$, which contains the second and fourth features. A greedy approach would choose the first column at the first iteration and never discard it, resulting in a sub-optimal final subset. Despite following a greedy approach, the algorithm described in [27] is very efficient and provides good approximations in practice.

Second, we examine a family of methods which exploit randomized sampling using a concept known as the leverage scores. Given a matrix $A \in \mathbb{R}^{m \times n}$, let V_k be the matrix whose columns are its top k right singular vectors. We define the i-th leverage score as

$$\|(V_k)_{i:}\|_2^2$$

Some existing methods simply sort the columns in descending order according to their leverage scores [29]. However, it is more common to use these scores to build biased probability distributions and then randomly draw various candidate column subsets. An approach belonging to this family can be found in [9], where $c > k$ columns are randomly sampled and then reduced to exactly k using a factorization with column pivoting. The leverage scores are high for a matrix column if it is similar to a left singular vector. Since V_k is a submatrix of an orthogonal matrix, high-scoring columns tend to be isolated in that regard. However, we have observed that in real datasets this isolation is rare, and fairly correlated columns can have similarly high scores. This can lead to columns with a considerable amount of redundancy being chosen in the final subset.

Our algorithm overcomes these limitations by taking into account at every step of the process that k columns are being chosen. Algorithm Prototype, shown in Figure 1, provides a high level overview of how it functions. For an input matrix $A \in \mathbb{R}^{m \times n}$ let us consider that we want to pick k columns. First, an initial subset R of k columns is chosen uniformly at random without replacement (line 2), forming a matrix $C = A_R \in \mathbb{R}^{m \times k}$. Then, the algorithm iterates until convergence as follows. For $i = 1, \ldots, k$, column i is removed from C, forming matrix $\tilde{C} \in \mathbb{R}^{m \times k-1}$, and is replaced by another column such that the objective function (1) is minimized over all possible $n - k + 1$ replacements (we don't rule out the column we removed). The algorithm converges when no single column replacement yields an improvement in the objective function anymore. This simple approach provides column subsets that outperform the algorithms discussed above.

Algorithm Prototype can find very good approximations to the CSSP objective function. However, it can be very slow

Fig. 1: Algorithm Prototype

1: **procedure** ALGORITHM PROTOTYPE(A, k)
2: $\quad R \leftarrow$ uniSampleWithoutReplacement$(\{1 \ldots n\}, k)$
3: $\quad C \leftarrow A_R$
4: $\quad$ Compute C^+
5: $\quad$ **while** not converged **do**
6: $\qquad$ **for** $i = 1, \ldots, k$ **do**
7: $\qquad\quad \tilde{C} \leftarrow CH_k^i$ ▷ Zero out column i
8: $\qquad\quad w \leftarrow \operatorname{argmin}_j \|A - (\tilde{C}|A_{:j})(\tilde{C}|A_{:j})^+ A\|_F$
9: $\qquad\quad C \leftarrow (\tilde{C}|A_{:w}) \backslash i$
10: $\qquad\quad R[i] \leftarrow w$
11: $\qquad$ **end for**
12: $\quad$ **end while**
13: $\quad$ output R
14: **end procedure**

when the values of k and n grow slightly. We now present a series of non-trivial derivations that enable the design of Algorithm IterFS, an efficient algorithm which yields the same result as Algorithm Prototype.

In [27], it is shown that when the column subset is constructed in a greedy fashion, the next minimizing column can be found fast. In our case we cannot rely on this fact, since our algorithm does not construct the subset greedily, but rather it runs by iteratively removing one column and replacing it with a new one. We nevertheless use some results proved in [27]. If we have a column subset of A, forming matrix C, the following theorem provides us with a simple criterion to identify the best single column to append to matrix C, based on the matrix $E = A - CC^+A$.

Theorem 1. *Let $A \in \mathbb{R}^{m \times n}$. For some $k \in \mathbb{N}, k <$ rank(A) let $C \in \mathbb{R}^{m \times k}$ be a matrix comprised of a subset of the columns of A. Let $E = A - CC^+A$. Then*

$$\underset{i}{\operatorname{argmin}} \|A - (C|A_{:i})(C|A_{:i})^+ A\|_F = \underset{i}{\operatorname{argmax}} \frac{\|E^T E_{:i}\|_2^2}{\|E_{:i}\|_2^2}$$

This means that if we have computed matrix E, we can easily find the best column to add. If we define $F = E^T E$, we can express this criterion as

$$\underset{i}{\operatorname{argmax}} \frac{\|F_{:i}\|_2^2}{F_{ii}} \tag{2}$$

In [27], efficient formulae are given for recomputing $\|F_{:i}\|^2$ and F_{ii} once a column has been appended to matrix C. Our algorithm, however, does not build the column subset incrementally, but it iteratively replaces each column by another. Therefore, not only does it require to update these values when a column is added from C, but also when it is removed (or equivalently zeroed out to be replaced by a different one). We now present a series of derivations that allow us to do this efficiently, involving fast updates of the Moore-Penrose generalized inverse.

C. IterFS: An efficient algorithm for column subset selection

In this section we describe and prove the necessary derivations to design Algorithm IterFS, an efficient and equivalent variant of Algorithm Prototype. The key derivations are the efficient update of the Moore-Penrose pseudoinverse of C, the subsequent efficient update of the residual matrix $E = A - CC^+A$ and the fast update of the numerator and denominator of (2) to determine the winning column at each step of the algorithm.

We first point out how to update the Moore-Penrose generalized inverse of C. In [30], efficient formulae for rank-1 udpates of this type of matrix are provided. Six different cases are considered, depending on the nature of the vectors in which the update can be decomposed. We employ this result to derive efficient updates in our context.

Proposition 1. *Let $A \in \mathbb{R}^{m\times n}$. For some $k \in \mathbb{N}, k < rank(A)$ let $C \in \mathbb{R}^{m\times k}$ be a matrix comprised of a subset of the columns of A such that $rank(C) = k$. Let $\tilde{C} \in \mathbb{R}^{m\times k}$ be the matrix resulting from zeroing-out column i in C (i.e., column i of $\tilde{C}$ is comprised uniquely of zeros). Let $\rho = ((C^+)_{i:})^T$ (the i-th row of C^+ as a column vector). Then*

$$\tilde{C}^+ = C^+ - \|\rho\|_2^{-2} C^+\rho\rho^T$$

Proof. Zeroing out the column i of matrix C can be seen as the following rank-1 update of C:

$$\tilde{C} = C + cd^T$$

where $c = C_{:i}$ and $d = -e_i$ is minus the i-th vector of the canonical basis of a k-dimensional vector space (all zeros and a -1 in position i). It is obvious that c lies in the column space of C. Additionally, $\text{rank}(C) = k$. Therefore, $d^T \in \mathbb{R}^{1\times k}$ lies in the row space of C. Let us define $\beta = 1 + d^T C^+ c$. Then

$$\beta = 1 - (C^+)_{i:}c = 1 - \left((C^TC)^{-1}C^T\right)_{i:}c = 1 - 1 = 0$$

This means that this update always corresponds to case (vi) of [30] and the generalized inverse can be updated as

$$\tilde{C}^+ = C^+ - kk^+C^+ - C^+h^+h + k^+C^+h^+kh$$

where $k = C^+c$ and $h = d^TC^+ = (C^+)_{i:} = \rho^T$. Now, we have the three following equalities:

$$\begin{aligned} kk^+C^+ =& \ C^+c(C^+c)^+C^+ = e_ie_i^TC^+ = (C^+)_{i:} \\ C^+h^+h =& \ C^+\frac{h^T}{\|h\|_2^2}h = \|\rho\|_2^{-2}C^+\rho\rho^T \\ k^+C^+h^+kh =& \ (C^+c)^+C^+(\rho^T)^+kh = kh = (C^+)_{i:} \end{aligned}$$

Hence,

$$\tilde{C}^+ = C^+ - \|\rho\|_2^{-2}C^+\rho\rho^T$$

□

Secondly, we indicate how to update the residual matrix E when a column is removed.

Proposition 2. *Let $\tilde{C} \in \mathbb{R}^{m\times k} = CH_k^i$, $E = A - CC^+A$, $\tilde{E} = A - \tilde{C}\tilde{C}^+A$, $\rho = ((C^+)_{i:})^T$. Then*

$$\tilde{E} = E + C_{:i}\rho^TA + \|\rho\|_2^{-2}\tilde{C}C^+\rho\rho^TA$$

Proof. Let $G = -\|\rho\|_2^{-2}C^+\rho\rho^T$ (Proposition 1). Then

$$\begin{aligned} \tilde{E} =& \ A - \tilde{C}\tilde{C}^+A \\ =& \ A - \tilde{C}(C^+ + G)A \\ =& \ A - \tilde{C}C^+A - \tilde{C}GA \end{aligned}$$

Since $\tilde{C}C^+A = CC^+A - C_{:i}(C^+A)_{i:}$

$$\begin{aligned} \tilde{E} =& \ E + C_{:i}(C^+A)_{i:} - \tilde{C}GA \\ =& \ E + C_{:i}\rho^TA - \tilde{C}(-\|\rho\|_2^{-2}C^+\rho\rho^T)A \end{aligned}$$

□

We now provide efficient formulae to compute (2). We define $F = E^TE, \tilde{F} = \tilde{E}^T\tilde{E}$ and the vectors

$$\begin{aligned} f =& \ (\|F_{:1}\|_2^2, \ldots, \|F_{:n}\|_2^2) \\ g =& \ (F_{11}, \ldots, F_{nn}) \\ \tilde{f} =& \ (\|\tilde{F}_{:1}\|_2^2, \ldots, \|\tilde{F}_{:n}\|_2^2) \\ \tilde{g} =& \ (\tilde{F}_{11}, \ldots, \tilde{F}_{nn}) \end{aligned}$$

Proposition 3. *Let $\delta = (\tilde{E}_{:j})^T\tilde{E}$ and $\gamma = E^TE\delta$. Then*

$$\tilde{f} = f + \|\delta\|_2^2(\delta\circ\delta)\delta_j^{-2} + 2(\gamma\circ\delta)\delta_j^{-1}$$

$$\tilde{g} = g + (\delta\circ\delta)\delta_j^{-1}$$

Proof. From corollary 3 in [27],

$$\tilde{F} = F + \frac{\delta\delta^T}{\delta_j}$$

Therefore, for $k = 1, \ldots, n$ we have

$$\begin{aligned} \tilde{f}_k =& \ \|\tilde{F}_{:k}\|_2^2 = \sum_i (F_{ik} + \frac{\delta_i\delta_k}{\delta_j})^2 \\ =& \ \sum_i F_{ik}^2 + \left(\frac{\delta_i\delta_k}{\delta_j}\right)^2 + 2F_{ik}\frac{\delta_i\delta_k}{\delta_j} \\ =& \ \|F_{:k}\|_2^2 + \frac{\delta_k^2}{\delta_j^2}\|\delta\|_2^2 + 2\sum_i F_{ik}\frac{\delta_i\delta_k}{\delta_j} \end{aligned}$$

Additionally, we have the following:

$$\sum_i F_{ik}\frac{\delta_i\delta_k}{\delta_j} = \frac{\delta_k}{\delta_j}\sum_i F_{ik}\delta_i = \frac{\delta_k}{\delta_j}(F_{:k})^T\delta = \frac{\delta_k}{\delta_j}\gamma_k$$

Therefore

$$\tilde{f}_k = \|F_{:k}\|_2^2 + \frac{\delta_k^2}{\delta_j^2}\|\delta\|_2^2 + 2\frac{\delta_k}{\delta_j}\gamma_k = f_k + \frac{\delta_k^2}{\delta_j^2}\|\delta\|_2^2 + 2\frac{\delta_k}{\delta_j}\gamma_k$$

Considering the whole vector $\tilde{f}$, we retrieve the first equality we wanted to prove. The case of $\tilde{g}$ is trivial, given that

$$\tilde{g}_k = F_{kk} + \frac{\delta_k\delta_k}{\delta_j}$$

□

The criterion to find the current best column is

$$\underset{i}{\operatorname{argmax}} \frac{\tilde{f}_i}{\tilde{g}_i}$$

Equivalent derivations yield the update formulae to use when a column is chosen and added to the subset:

$$f = \tilde{f} + \|\delta\|_2^2(\delta \circ \delta)\delta_j^{-2} - 2(\gamma \circ \delta)\delta_j^{-1}$$
$$g = \tilde{g} - (\delta \circ \delta)\delta_j^{-1}$$

We now give an efficient formula to update C^+ once the winning column of the current iteration is added.

Proposition 4. *Let $C \in \mathbb{R}^{m\times k}$ be the matrix resulting from adding column w of A to $\tilde{C}$ at position i. Let $\omega = \tilde{E}_{:w}$ Then*

$$C^+ = \tilde{C}^+ - \|\omega\|_2^{-2}(\tilde{C}^+ A_{:w}\omega^T - e_i\omega^T)$$

Proof. Adding the column $A_{:w}$ to matrix $\tilde{C}$ can be seen as the following rank-1 update:

$$C = \tilde{C} + cd^T$$

where $c = A_{:w}$ and $d = e_i$ is the i-th vector of the canonical basis of a k-dimensional vector space (all zeros and a 1 in position i).

We assume that the addition of column $A_{:w}$ increases the rank of $\tilde{C}$, which implies that c is not in the column space of $\tilde{C}$. In addition, since $\tilde{C}_{:i} = 0$, $d^T = e_i^T$ is not in the rowspace of $\tilde{C}$. Therefore, we are in case (i) of [30]. We define $\beta = 1 + d^T\tilde{C}^+c$. The generalized inverse can thus be updated as

$$C^+ = \tilde{C}^+ - ku^+ - v^+h + \beta v^+u^+$$

where $k = \tilde{C}^+c$, $h = d^T\tilde{C}^+ = (\tilde{C}^+)_{i:}$, $u = (I - \tilde{C}\tilde{C}^+)c$ and $v = d^T(I - \tilde{C}^+\tilde{C})$. Since rank$(\tilde{C})$ = $k - 1$ and $\tilde{C}_{:i} = 0$, $I - \tilde{C}^+\tilde{C}$ is comprised exclusively of zeros, except for $\tilde{C}_{ii} = 1$. Therefore, $v = e_i^T$. We therefore have

$$C^+ = \tilde{C}^+ - \frac{\tilde{C}^+c(c - \tilde{C}\tilde{C}^+c)^T}{\|c - \tilde{C}\tilde{C}^+c\|_2^2} - H + \frac{\beta e_i(c - \tilde{C}\tilde{C}^+c)^T}{\|c - \tilde{C}\tilde{C}^+c\|_2^2}.$$

where H is a zero matrix with h in its i-th row. Now, since $\tilde{C}_{:i} = 0$, then $h = (\tilde{C}^+)_{i:} = 0$ and $\beta = 1$. Also, note that $c - \tilde{C}\tilde{C}^+c = \tilde{E}_{:w} = \omega$. Therefore,

$$C^+ = \tilde{C}^+ - \|\omega\|_2^{-2}(\tilde{C}^+ A_{:w}\omega^T - e_i\omega^T)$$

□

Finally, since in this case we have added a column to the subset, we can employ the result proved in lemma 2 of [27] to update E.

$$E = \tilde{E} - \omega\omega^T\tilde{E}\|\omega\|_2^{-2} \tag{3}$$

where $\omega = \tilde{E}_{:w}$.

These derivations enable the design of Algorithm IterFS, equivalent to Algorithm Prototype but much more efficient.

Despite the efficiency of Algorithm IterFS, two problems can still arise. First, if m is sufficiently large the running time can be prohibitive and the matrix might not fit in main memory. Second, the initial computation of f and g (for (2)) requires the computation of the product $E^TE \in \mathbb{R}^{n\times n}$. If n is too large, this imposes significant memory and computational requirements. We now show how to address these two potential problems.

D. Dealing with huge matrices

We now prove two results that allow us to deal with matrices comprised of a large number of rows or a large number of columns. The next theorem implies that in order to find an approximation to the solution of the CSSP on a $m \times n$ matrix, we can operate on an $n \times n$ matrix and obtain the same result. An example of the benefits brought by this result is the following. We ran our algorithm on a commodity four-core PC with 8GB of RAM on a dataset of 11,000 rows (the USPS dataset described in section IV). Without using the result from Theorem 2 the algorithm took over 6 seconds. Using it, the running time was reduced to 0,28 seconds.

This result can also be beneficial when dealing with matrices with a huge number of rows that do no fit in main memory. These can be preprocessed fast taking advantage of distributed computing platforms such as Apache Spark [31], which provide efficient parallelized implementations of the singular value decomposition. The resulting matrix is compact and can therefore be processed by our algorithm in the main memory of a modest machine.

Theorem 2. *Let $A \in \mathbb{R}^{m\times n}$, $m > n$, rank$(A) = n$ and let $U^TAV = \Sigma$ be its singular value decomposition. Let $\mathcal{S}^{n\times k}$ be the set of $n \times k$ column sampling matrices (permuted and column-truncated identity matrices). Then*

$$\begin{aligned}
&\underset{S\in\mathcal{S}^{n\times k}}{\operatorname{argmin}} \|A - AS(AS)^+A\|_F^2 \\
=\ &\underset{S\in\mathcal{S}^{n\times k}}{\operatorname{argmin}} \|\Sigma V^T - \Sigma V^T S(\Sigma V^T S)^+\Sigma V^T\|_F^2
\end{aligned} \tag{4}$$

Proof. For some $S \in \mathcal{S}^{n\times k}$, let $C = AS$ and $\tilde{C} = \Sigma V^T S$. Since U is orthogonal and its first n columns span the column space of A, for all $S \in \mathcal{S}^{n\times k}$

$$\begin{aligned}
&\|A - CC^+A\|_F^2 = \|U^T(A - CC^+A)\|_F^2 \\
=\ &\|U^TA - U^TCC^+A\|_F^2 \\
=\ &\|\Sigma V^T - \tilde{C}C^+A\|_F^2
\end{aligned}$$

The first equality holds also for the “thin” SVD because U is column-wise orthogonal and CC^+A lies in the column space of A, and thus so does $A - CC^+A$. We have that

$$\begin{aligned}
\tilde{C}^T\tilde{C} =\ &(\Sigma V^TS)^T\Sigma V^TS = S^TV\Sigma\Sigma V^TS \\
=\ &S^TA^TUU^TAS = (AS)^TAS = C^TC
\end{aligned}$$

Now, since $C^+ = (C^TC)^{-1}C^T$, $\tilde{C}^+ = (\tilde{C}^T\tilde{C})^{-1}\tilde{C}^T$ and $\tilde{C} = \Sigma V^TS = U^TAS = U^TC$, we have that

$$\begin{aligned}
&\tilde{C}^+\Sigma V^T = (\tilde{C}^T\tilde{C})^{-1}\tilde{C}^T\Sigma V^T = (C^TC)^{-1}\tilde{C}^T\Sigma V^T \\
=\ &(C^TC)^{-1}(U^TC)^T\Sigma V^T = (C^TC)^{-1}C^TU\Sigma V^T = C^+A
\end{aligned}$$

Which implies that

$$\begin{aligned} & \|A-(AS)(AS)^{+}A\|_F^2 = \|A-CC^{+}A\|_F^2 \\ = & \ \|\Sigma V^T - \tilde{C}C^{+}A\|_F^2 = \|\Sigma V^T - \tilde{C}\tilde{C}^{+}\Sigma V^T\|_F^2 \\ = & \ \|\Sigma V^T - \Sigma V^T S(\Sigma V^T S)^{+}\Sigma V^T\|_F^2 \end{aligned}$$

Note that $\text{rank}(A) = n$ and $n < m$, so $\text{rank}(C) = k$ and $(C^TC)^{-1}$ exists. Therefore, the equality in (4) holds for all values of S, which means that the minimum will be attained at the same argument. □

Applying Theorem 2: In order to apply this theorem, if the input matrix A is such that $m \gg n$, then A should be replaced with ΣV^T on entry.

The following theorem implies that we do not need to compute the product E^TE explicitly, which could impose prohibitive memory requirements for large values of n, i.e. large numbers of columns.

Theorem 3. *For some matrix A and a column subset C, let $E = A - CC^{+}A$ and let $E = U\Sigma V^T$ be its singular value decomposition. Let $f_i = \|E^TE_{:i}\|_2^2$, $g_i = \|E_{:i}\|_2^2$. Then*

$$f_i = \|(\Sigma^2 V^T)_{:i}\|_2^2$$
$$g_i = \|(\Sigma V^T)_{:i}\|_2^2$$

Proof.

$$\begin{aligned} f_i = & \ \|(E^TE)_{:i}\|_2^2 = \|(V\Sigma U^T U\Sigma V^T)_{:i}\|_2^2 \\ = & \ \|(V\Sigma^2 V^T)_{:i}\|_2^2 = \|(\Sigma^2 V^T)_{:i}\|_2^2 \end{aligned}$$

The last equality holds because V is orthogonal. On the other hand,

$$\begin{aligned} g_i = & \ \|E_{i:}\|_2^2 = (E_{:i})^T E_{:i} = (V\Sigma U^T)_{i:}(U\Sigma V^T)_{:i} \\ = & \ (V\Sigma)_{i:}(\Sigma V^T)_{:i} = \|(\Sigma V^T)_{:i}\|_2^2 \end{aligned}$$

□

This result is especially advantageous when the rank of our data matrix is much smaller than the number of columns, a circumstance that arises frequently in certain domains such as bioinformatics or image processing.

Applying Theorem 3: This theorem can be applied to our algorithm as follows. If the input matrix A is such that $m \ll n$, $E = A - CC^{+}A$ and $E = U\Sigma V^T$ is the singular value decomposition of E, then lines 3 and 4 of Algorithm IterFS should be replaced by

$$F \leftarrow \Sigma^2 V^T$$
$$f_i \leftarrow \|(\Sigma^2 V^T)_{:i}\|_2^2 \text{ for } i = 1 \dots n$$
$$g_i \leftarrow \|(\Sigma V^T)_{:i}\|_2^2 \text{ for } i = 1 \dots n$$

E. Complexity analysis

The running time of our algorithm is generally mostly determined by the complexity of the loop. The main operations to be performed are the updates of C^{+} and E (when removing and when adding a column) and the computation of δ and γ. The computation of the pseudoinverse generally takes $O(mk^2)$ time. However, taking advantage of propositions 1 and 4 this can be done in $O(mk)$ time, while E is updated twice in $O(mn)$ time. On the other hand, δ and γ can be computed in $O(mn)$ time, and f and g in $O(n)$. We therefore have that the necessary operations in the loop take $O(2mk + 3mn + 4n) = O(mn)$ time (note that k is always smaller than the rank of the input matrix). Since these operations are run for each column of the candidate subset, each iteration requires $O(mnk)$ operations. If $m \gg n$, using Theorem 2 the complexity of the loop can be reduced to $O(n^2k)$ at the cost of an $O(mn^2)$ operation at the beginning of the algorithm. Since existing implementations of the SVD are very efficient and often run in parallel, this can yield significant benefits in practice. The product E^TE at the beginning takes $O(mn^2)$, but if $n > m$ this can be reduced to $O(m^2n)$ using Theorem 3.

IV. Experimental results

In order to evaluate the efficiency and effectiveness of our algorithm, we have tested it on various benchmark datasets, comparing it to other well-known algorithms for the column subset selection problem. Algorithms such as the ones presented in [17] [18] [19], e.g., do not explicitly target the same objective function as our method, generally yielding much worse linear approximations and therefore not being comparable in this regard. We run the algorithms on a 12-core machine with 32 GB of RAM. We employ the following algorithms for comparison:

- **Two-stage**: The two-stage algorithm described in [9]. As suggested by the authors, we draw 40 candidate subsets and pick the best one. We use our own Matlab implementation.
- **GreedyFS**: The algorithm described in [27]. We employ the Matlab code provided by the authors [1].
- **IterFS**: Our method[2].
- **IterFS-1st**: Our method, limited to one iteration.

A. Datasets

We employ the following datasets: the Columbia University Image library (**COIL20**)[3] [32]; the **ORL** database of faces [4] provided by the AT&T Laboratories Cambridge [33]; the Sheffield face database (**UMIST**) provided by the University of Sheffield[5]; the **Binary Alpha** Digits dataset[6]; the Extended Yale Face database B (**YaleB**) [7] [34], [35]; the **USPS** hand-written digits dataset, taking both the training and the test set[8];

[1] http://www.afarahat.com/code

[2] https://github.com/brunez/IterFS

[3] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[4] http://www.cl.cam.ac.uk/research/dtg /attarchive/facedatabase.html

[5] http://www.sheffield.ac.uk/eee/research/iel/research/face

[6] http://www.cs.nyu.edu/∼roweis/data.html

[7] http://vision.ucsd.edu/∼leekc/ExtYaleDatabase /ExtYaleB.html

[8] www.iro.umontreal.ca/∼lisa/twiki/bin/view.cgi /Public/PublicDatasets

Fig. 2: Algorithm IterFS

procedure ITERFS(A, k)
  $R \leftarrow$ uniSampleWithoutReplacement($1 \ldots n, k$)
    ▷ Optionally apply theorem 2 or 3
  $F \leftarrow E^T E$
  $f_i \leftarrow \|F_{:i}\|_2^2$; $g_i \leftarrow F_{ii}$ for $i = 1 \ldots n$
  $C \leftarrow A_R$
  while not converged **do**
    for $i = 1, \ldots, k$ **do**
      $j \leftarrow R[i]$
      $\tilde{C} \leftarrow C H_k^i$ ▷ Zero out column i
      $\tilde{C}^+ \leftarrow C^+ - \|\rho\|_2^{-2} C^+ \rho \rho^T$ ▷ Prop. 1
      $S_1 \leftarrow C_{:i} \rho^T A$
      $S_2 \leftarrow \|\rho\|_2^{-2} \tilde{C} C^+ \rho \rho^T A$
      $\tilde{E} \leftarrow E + S_1 + S_2$ ▷ Prop. 2
      $\delta \leftarrow \tilde{E}_{:j}^T \tilde{E}$; $\gamma \leftarrow E^T E \delta$
      $\tilde{f} \leftarrow f + \|\delta\|_2^2 (\delta \circ \delta) \delta_j^{-2} + 2(\gamma \circ \delta) \delta_j^{-1}$
      $\tilde{g} \leftarrow g + (\delta \circ \delta) \delta_j^{-1}$ ▷ Prop. 3
      $w \leftarrow \text{argmax}_h \tilde{f}_h / \tilde{g}_h$
      $\delta \leftarrow \tilde{E}_{:w}^T \tilde{E}$; $\gamma \leftarrow \tilde{E}^T \tilde{E} \delta$
      $f \leftarrow \tilde{f} + \|\delta\|_2^2 (\delta \circ \delta) \delta_w^{-2} - 2(\gamma \circ \delta) \delta_w^{-1}$
      $g \leftarrow \tilde{g} - (\delta \circ \delta) \delta_w^{-1}$
      $C^+ \leftarrow \tilde{C}^+ - \|\omega\|_2^{-2} (\tilde{C}^+ A_{:w} \omega^T - e_i \omega^T)$
        ▷ Prop. 4
      $E \leftarrow \tilde{E} - \omega \omega^T \tilde{E} \|\omega\|_2^{-2}$ ▷ (3)
      checkConvergence()
      $R[i] \leftarrow w$
      $C \leftarrow A_R$
    end for
  end while
  Output R
end procedure

the **Online News** popularity dataset[9] [36]; the **BlogFeedback** dataset[10] [37] and the **YearPredictionMSD** dataset[11] [38]. Some of these datasets are available at the UCI machine learning repository [39]. Table I provides a summary.

All datasets are standardized to zero mean and unit variance before running the algorithms (i.e. each column is transformed by subtracting its mean from it and then dividing it by its standard deviation).

[9] https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity
[10] https://archive.ics.uci.edu/ml/datasets/BlogFeedback
[11] https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD

Dataset	Rows	Columns
COIL20	1440	1024
ORL	400	1024
UMIST	575	10304
BinaryAlpha	1404	320
YaleB	2414	1024
USPS	11100	256
OnlineNews	39644	60
BlogFeedback	52397	281
YearPredictionMSD	515345	90

TABLE I: Employed datasets.

B. *Objective function*

In order to evaluate the accuracy of our algorithm in the objective function we measure its error with respect to the best possible rank-k approximation, which can be obtained using the singular value decomposition and provides a loose lower bound for the CSSP. Specifically, given a data matrix A let A_k be its best rank-k approximation, and let C be the submatrix of A comprised of the k columns chosen by one of the considered algorithms. We define the **error ratio** as

$$\frac{\|A - CC^+ A\|_F^2}{\|A - A_k\|_F^2}$$

The closer this is to 1, the better. The error ratio is always greater than one, unless a subset of features matches the top left singular vectors, which is of course extremely unlikely. Table II reports the error ratio for all the algorithms on all datasets, for increasing values of k (the number of chosen features). In this table, TS is Two-stage, GFS is GreedyFS, IFS is IterFS and IFS-1 is IterFS limited to one iteration. Since Two-stage, IFS and IFS-1 have a random component, we report the average of 10 runs in Table II and comment on the standard deviation below. These results show how our algorithm consistently outperforms both **Two-stage** and **GreedyFS**. Remarkably, one iteration of our method is enough to provide better approximation errors than the other two algorithms in most cases.

Due to space constraints we do not include the standard deviation in Table II, although we comment on it briefly. The results of IterFS presented a small standard deviation, generally below 10^{-3}. We did observe a higher deviation in the case of the OnlineNews dataset for $k = 30, 40$, reaching values of 0,033 and 0,024 respectively. Our algorithm only showed noticeable instability in the case of the BlogFeedback dataset for values of $k > 100$, reaching values close to 0.1 and even 0.3 in one instance. We are not currently sure about the cause of this instability, though we intend to investigate it further. As expected, the standard deviation shown by IterFS-1st is slightly higher, but not significantly in general. The Two-stage algorithm shows negligible deviations in most cases. We remark that even though our method presents a certain amount of variety in its results, the produced subset is almost always better than those found by the other algorithms, regardless of

Dataset	TS	GFS	IFS-1	IFS	TS	GFS	IFS-1	IFS	TS	GFS	IFS-1	IFS	TS	GFS	IFS-1	IFS
	k=20				k=40				k=60				k=80			
COIL20	1.578	1.499	1.488	**1.459**	1.770	1.619	1.587	**1.552**	1.838	1.681	1.660	**1.618**	1.896	1.744	1.717	**1.672**
ORL	1.542	1.421	1.419	**1.387**	1.673	1.548	1.529	**1.499**	1.747	1.625	1.601	**1.567**	1.894	1.702	1.672	**1.637**
UMIST	1.697	1.469	1.449	**1.420**	1.783	1.574	1.559	**1.515**	1.823	1.637	1.622	**1.569**	1.908	1.695	1.666	**1.617**
B.Alpha	1.426	1.427	1.427	**1.411**	1.600	1.564	1.559	**1.540**	1.657	1.637	1.617	**1.594**	1.645	1.665	1.637	**1.620**
YaleB	1.672	1.400	1.385	**1.356**	1.678	1.508	1.500	**1.471**	1.817	1.583	1.581	**1.546**	1.867	1.657	1.642	**1.608**
USPS	1.479	1.412	1.401	**1.392**	1.641	1.536	1.539	**1.512**	1.670	1.682	1.654	**1.628**	1.707	1.795	1.752	**1.702**
OnlineNews	1.357	1.291	1.302	**1.290**	1.846	1.797	1.746	**1.729**	-	-	-	-	-	-	-	-
BlogFeedback	1.281	1.058	**1.054**	1.055	1.175	1.092	**1.087**	**1.087**	1.238	1.124	1.119	**1.116**	1.348	1.153	**1.144**	1.150
YearPredMSD	1.418	1.350	1.342	**1.338**	1.653	1.608	1.578	**1.562**	1.856	1.865	1.796	**1.766**	2.004	2.126	1.907	**1.891**
	k=100				k=120				k=140				k=160			
COIL20	1.914	1.790	1.762	**1.713**	1.896	1.834	1.795	**1.742**	1.912	1.869	1.828	**1.778**	1.976	1.897	1.864	**1.808**
ORL	1.924	1.777	1.743	**1.695**	2.032	1.853	1.818	**1.766**	2.140	1.931	1.891	**1.838**	2.240	2.012	1.966	**1.903**
UMIST	2.017	1.748	1.718	**1.665**	2.024	1.792	1.758	**1.700**	2.074	1.830	1.786	**1.733**	2.118	1.865	1.819	**1.756**
B.Alpha	1.680	1.671	1.659	**1.636**	1.687	1.689	1.680	**1.654**	1.722	1.724	1.707	**1.686**	1.767	1.763	1.742	**1.717**
YaleB	1.913	1.724	1.701	**1.665**	1.962	1.782	1.753	**1.709**	1.944	1.829	1.797	**1.749**	2.012	1.862	1.826	**1.783**
USPS	1.808	1.889	1.817	**1.756**	1.923	1.974	1.905	**1.848**	2.017	2.025	1.979	**1.931**	2.135	2.127	2.078	**2.044**
BlogFeedback	1.413	1.179	**1.168**	1.197	1.626	**1.202**	1.240	1.209	1.817	1.227	**1.226**	1.228	2.089	1.261	**1.230**	1.261

TABLE II: Approximation error with respect to the best rank-k approximation.

the initial random subset.

C. Running time

We evaluate the running time of our algorithm compared to the other methods. Since our algorithm iterates until no further improvement is achieved, its running times can vary noticeably from one execution to another. Since a single iteration is often enough to outperform the other algorithms (Table II), we limit our algorithm to just the first iteration. Figure 3 shows the results. In the case of Two-stage, when $m > n$ we take advantage of Theorem 2 to significantly speed up the second phase. Our algorithm is as fast as GreedyFS in the case of USPS and faster in BlogFeedback, OnlineNews and YearPredictionMSD. In some cases the higher sensitivity of our method to the value of k is significant, as is the case of Coil20, ORL and YaleB. In other instances, though noticeable, the impact of this sensitivity is not so notable, as is the case of UMIST and BinaryAlpha and our algorithm can still select tens of features in a few tens of miliseconds. In the case of the former, our algorithm is faster for small values of k. The sensitivity to the value of k seems to be particularly noticeable when the input matrix is large in both dimensions.

D. Reconstruction capabilities

Since our algorithm achieves good approximation errors, we evaluate its capabilities to reconstruct single data instances. Given a data instance $x \in \mathbb{R}^n$, which is a row of the input matrix $A \in \mathbb{R}^{m \times n}$, we consider the vector $x' \in \mathbb{R}^k$ comprised only of the features chosen by the studied algorithms (i.e., x' is a row of the submatrix of chosen columns C). The complete instance x can be approximated as

$$x \approx x' C^+ A. \tag{5}$$

We pick various instances from the Yale faces dataset[12] with the following criterion. For $k = 10, 20, 30, 40, 50$ we compute the approximation error for the whole dataset using our algorithm (IterFS). We compute the per-instance average approximation error (i.e. the total error divided by the number of instances) and pick the individual instance whose approximation error (i.e. the sum of squared errors between x and the value approximated by (5)) is closest to this average. This can be interpreted as the instance that is closest to the expected approximation error if one were to pick a random data point.

Figure 4 shows the reconstruction achieved for five such images. Each row shows one image as reconstructed by Two-Stage, GreedyFS, IterFS, the SVD and the original image. Below each image the reconstruction error with respect to the best rank-k approximation is shown. It might be shocking that in one instance the error attained by our algorithm is better than the one yielded by the SVD. Though unusual, this is entirely possible. The SVD yields the best approximation of the entire matrix, but an individual row might be approximated better using a different model. It should be noted that even though our algorithm yields better matrix approximations, it does not necessarily approximate all individual instances better than the other two algorithms. In this dataset, however, our algorithm consistently outperforms Two-Stage and GreedyFS when it comes to the number of individual images that are approximated better than the other two. In Table III we report, for different values of k, the number of instances that each algorithm approximates better than the other two. We run IterFS and Two-Stage 10 times and report the average and half the standard deviation. This dataset totals 161 images.

k	Two-Stage	GreedyFS	IterFS
10	10.8 ± 2.19	67.2 ± 1.17	87 ± 1.25
20	14.4 ± 1.11	62 ± 3.13	88.6 ± 3.52
30	23.5 ± 1.53	51.6 ± 1.86	89.9 ± 1.88
40	18.8 ± 1.43	52 ± 1.93	94.2 ± 2.39
50	16.8 ± 1.74	46.7 ± 1.51	101.5 ± 2.3

TABLE III: Number of instances that each algorithm approximates better than the other two on average.

[12] http://vision.ucsd.edu/content/yale-face-database

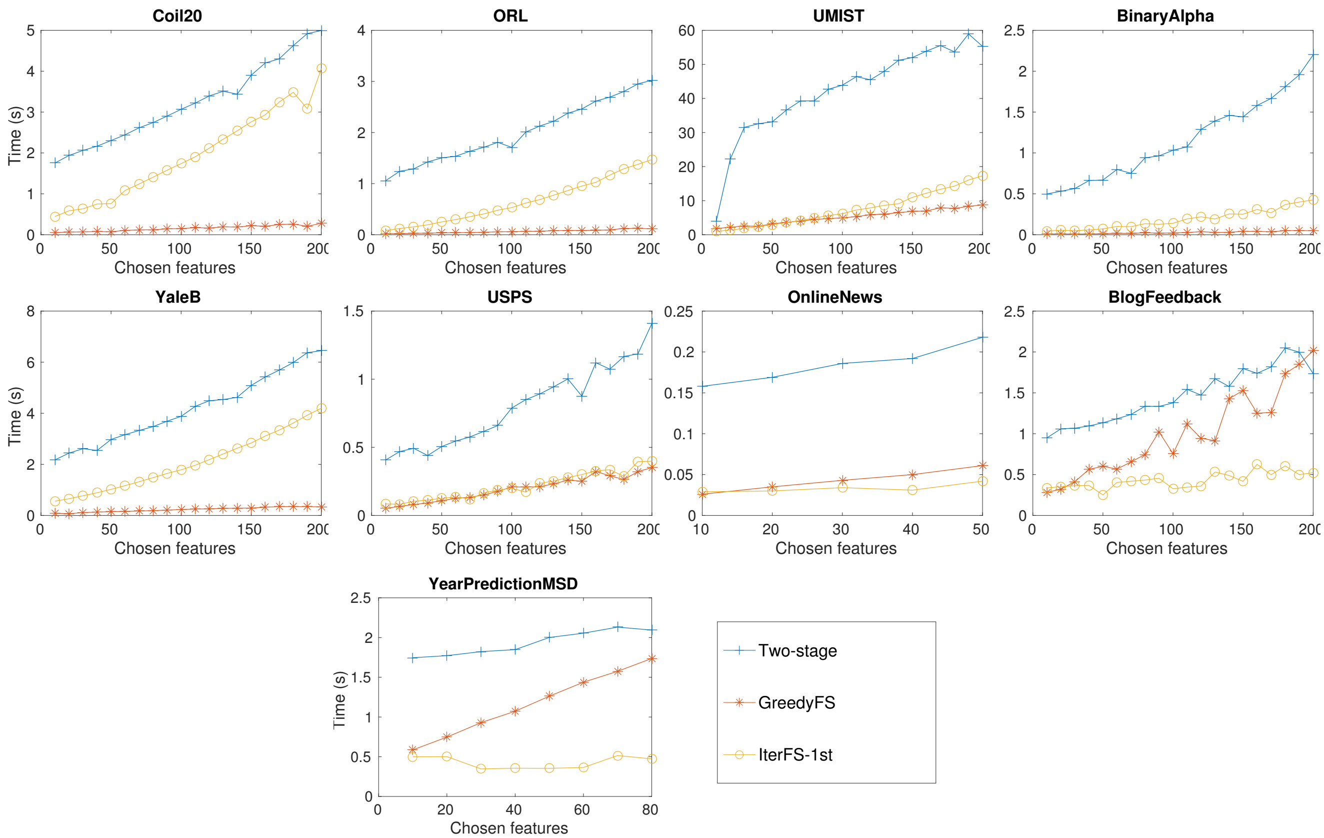

Fig. 3: Running times for one iteration of our algorithm compared to other methods.

V. Conclusions and Future Work

We have presented a novel algorithm for unsupervised feature selection based on the column subset selection problem. The algorithm is based on a new idea and optimizes features taking the whole chosen subset into account. We have also provided proofs for results that enable the application of this algorithm (and others) to huge data sets regardless of their number of rows. We have shown experimentally that our algorithm consistently outperforms state-of-the-art methods for the column subset selection problem, both in full matrix and in single instance approximation errors. In addition, our method is comparable in speed to very efficient existing algorithms, and faster in some cases. There are several directions that would be interesting to explore. First, the efficiency of our algorithm could be improved by devising parallelized strategies. Second, its application to regularized formulations of the CSSP might result in robust models that work better on unseen data. It would also be interesting to apply these ideas to non-linear criteria for feature selection. Finally, our approach opens up a particularly compelling question. Can our algorithm reach the optimum of the objective function starting from different subsets? If the anwer to this question is affirmative and the number of such subsets is large in general, this could have important implications in the theory of the column subset selection problem, since it could imply a significant reduction of the search space.

References

[1] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[2] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[3] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[4] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[5] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[6] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *The Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[7] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.

[8] T. F. Chan and P. C. Hansen, "Some applications of the rank revealing qr factorization," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 3, pp. 727–741, 1992.

[9] C. Boutsidis, M. W. Mahoney, and P. Drineas, "Unsupervised feature selection for principal components analysis," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 61–69.

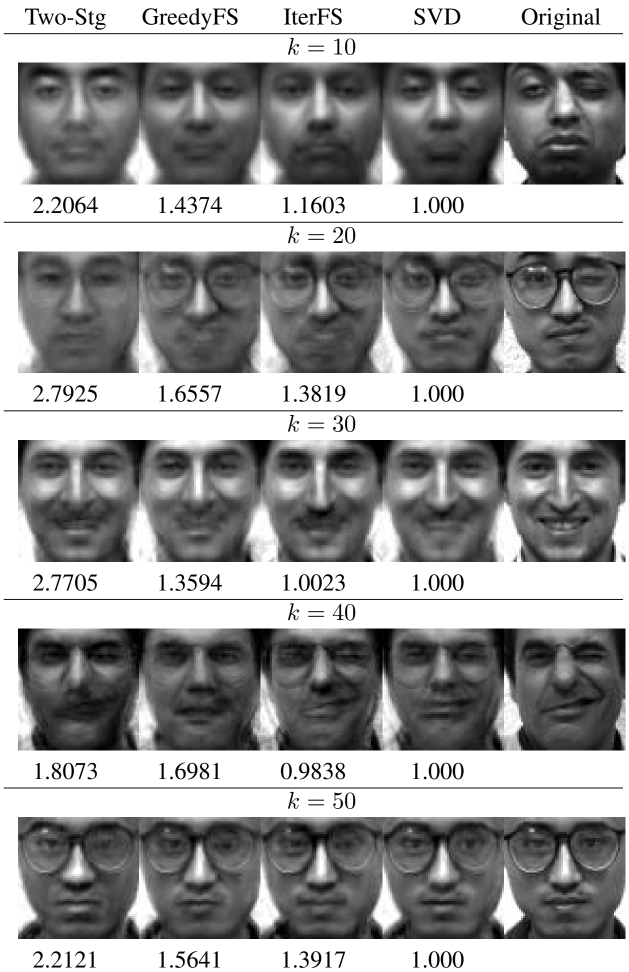

Fig. 4: Reconstructions achieved with different algorithms. Below each image is the approximation error with respect to the SVD.

[10] G. Golub, "Numerical methods for solving linear least squares problems," *Numerische Mathematik*, vol. 7, no. 3, pp. 206–216, 1965.

[11] P. Businger and G. H. Golub, "Linear least squares solutions by householder transformations," *Numerische Mathematik*, vol. 7, no. 3, pp. 269–276, 1965.

[12] T. F. Chan, "Rank revealing qr factorizations," *Linear algebra and its applications*, vol. 88, 1987.

[13] M. Gu and S. C. Eisenstat, "Efficient algorithms for computing a strong rank-revealing qr factorization," *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 848–869, 1996.

[14] L. V. Foster, "Rank and null space calculations using matrix decomposition without column interchanges," *Linear Algebra and its Applications*, vol. 74, pp. 47–71, 1986.

[15] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 301–312, 2002.

[16] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," *The Journal of Machine Learning Research*, vol. 5, pp. 845–889, 2004.

[17] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in neural information processing systems*, 2005, pp. 507–514.

[18] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 1151–1157.

[19] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 333–342.

[20] M. W. Mahoney and P. Drineas, "Cur matrix decompositions for improved data analysis," *Proceedings of the National Academy of Sciences*, vol. 106, no. 3, pp. 697–702, 2009.

[21] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *Proc. of the 20th Annual ACM-SIAM Symp. on Discrete Algorithms*. Soc. for Industrial and Applied Mathematics, 2009, pp. 968–977.

[22] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal column-based matrix reconstruction," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 687–717, 2014.

[23] V. Guruswami and A. K. Sinop, "Optimal column-based low-rank matrix reconstruction," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2012, pp. 1207–1214.

[24] S. Paul, M. Magdon-Ismail, and P. Drineas, "Column selection via adaptive sampling," in *Advances in Neural Information Processing Systems*, 2015, pp. 406–414.

[25] P. Zhu, W. Zuo, L. Zhang, Q. Hu, and S. C. Shiu, "Unsupervised feature selection by regularized self-representation," *Pattern Recognition*, vol. 48, no. 2, pp. 438–446, 2015.

[26] H. Arai, C. Maung, and H. Schweitzer, "Optimal column subset selection by a-star search," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[27] A. K. Farahat, A. Ghodsi, and M. S. Kamel, "An efficient greedy method for unsupervised feature selection," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 161–170.

[28] A. K. Farahat, A. Elgohary, A. Ghodsi, and M. S. Kamel, "Distributed column subset selection on mapreduce," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 171–180.

[29] D. Papailiopoulos, A. Kyrillidis, and C. Boutsidis, "Provable deterministic leverage score sampling," in *Proceedings of the 20th ACM SIGKDD*. ACM, 2014, pp. 997–1006.

[30] C. D. Meyer, Jr, "Generalized inversion of modified matrices," *SIAM Journal on Applied Mathematics*, vol. 24, no. 3, pp. 315–323, 1973.

[31] Z. et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. of the 9th USENIX conf. on Networked Systems Design and Implementation*. USENIX Assoc., 2012, pp. 2–2.

[32] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-20)," Columbia University, Tech. Rep. CUCS-005-96, February 1996.

[33] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*. IEEE, 1994, pp. 138–142.

[34] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 643–660, 2001.

[35] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 684–698, 2005.

[36] K. Fernandes, P. Vinagre, and P. Cortez, "A proactive intelligent decision support system for predicting the popularity of online news," in *Progress in Artificial Intelligence*. Springer, 2015, pp. 535–546.

[37] K. Buza, "Feedback prediction for blogs," in *Data analysis, machine learning and knowledge discovery*. Springer, 2014, pp. 145–152.

[38] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[39] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml