Heuristic Search Algorithm for Dimensionality Reduction Optimally Combining Feature Selection and Feature Extraction

Baokun He, Swair Shah, Crystal Maung, Gordon Arnold, Guihong Wan, Haim Schweitzer
(Baokun.He, swair, gordon.arnold, Guihong.Wan, HSchweitzer)@utdallas.edu, Crystal.Maung@gmail.com
Department of Computer Science, The University of Texas at Dallas
800 W. Campbell Road, Richardson, TX 75080

Abstract

The following are two classical approaches to dimensionality reduction: 1. Approximating the data with a small number of features that exist in the data (feature selection). 2. Approximating the data with a small number of arbitrary features (feature extraction). We study a generalization that approximates the data with both selected and extracted features. We show that an optimal solution to this hybrid problem involves a combinatorial search, and cannot be trivially obtained even if one can solve optimally the separate problems of selection and extraction. Our approach that gives optimal and approximate solutions uses a "best first" heuristic search. The algorithm comes with both an *a priori* and an *a posteriori* optimality guarantee similar to those that can be obtained for the classical weighted A* algorithm. Experimental results show the effectiveness of the proposed approach.

1 Introduction

The representation of data in terms of a small number of features is a fundamental tool in data analysis. The compact representation allows for efficient manipulation, and may reveal relations in the data that are harder to identify. We study the unsupervised case, where a typical criterion of quality for the representation is the accuracy with which the data can be reconstructed from the compact representation.

Let m be the number of data items, each specified in terms of n features, so that the data can be viewed as the matrix X of m rows and n columns. A compact representation with r features is given by a matrix V of size $m \times r$, with $r \leq n$. The reconstruction of X from V is computed by $X \approx VA$, where A is the $r \times n$ coefficients matrix. We note that the matrix VA is of rank r, so that X is being approximated by a rank r matrix. Conversely, any rank r matrix can be expressed as the product VA, and thus gives a compact representation in terms of r features.

1.1 Previous work and the current state of the art

Studies of dimensionality reduction distinguish between the case where the columns of V must also be columns of X (feature selection), and the case in which this constraint is not enforced (feature extraction). We review these two approaches and then propose to combine them. We show that applying selection followed by extraction or vice versa does not give the optimal hybrid representation.

```
   Input: the matrix $X$, the integer $r$.
   Output: the $r$ vectors $v_1, \ldots, v_r$.
1  Compute the matrix $B = XX^T$.
2  $v_1, \ldots, v_r$ are the top $r$ eigenvectors of $B$.
```

Figure 1: The algorithm for optimal feature extraction

Let Θ be a matrix norm, then the error of approximating the matrix X by VA is given by:

$$X \approx VA, \quad \text{error} = \Theta(X - VA) \qquad (1)$$

Feature extraction. The well-known algorithm for optimal feature extraction is shown in Fig.1. See, e.g., (Jolliffe 2002; Li et al. 2017). Applications of this algorithm include the technique of principal component analysis (**PCA**), which is arguably the most popular feature extraction technique. With recent advances in numerical techniques for computing eigenvectors (e.g., (Halko, Martinsson, and Tropp 2011; Li et al. 2017)) the algorithm in Fig.1 can be implemented efficiently even for large amounts of data.

Among the topics of current research are attempts to minimize the approximation error (1) in norms that are not unitarily invariant. This turns out to be very challenging. In particular, minimizing the entry-wise l_1 or l_0 norms is expected to improve the robustness of the estimation, but unfortunately the problem formulated in these norms turns out to be NP-hard. See, e.g., Gillis (2018), Song (2017), and Bringmann (2017). For the more general case of entrywise l_p norms see Chierichetti (2017).

Feature selection. In feature selection the columns of V are constrained to be columns of X. This is sometimes known as the Column Subset Selection Problem (**CSSP**). See, e.g. (Golub and Van-Loan 2013). Even though the approximation obtained by feature selection is worse than the approximation obtained by feature extraction, there are advantages of feature selection that make it the preferred choice in many situations. For example:

• Unlike feature selection, the results obtained by feature extraction are "notoriously difficult to interpret in terms of the underlying data" (Drineas, Mahoney, and Muthukrishnan 2008).

- Selected features generalize better than extracted features in machine learning tasks (Guyon and Elisseeff 2003).
- Functions computed from extracted features depend on all the features and are typically more expensive to evaluate than functions computed from few selected features.
- Feature selection retains the data sparsity.

To describe current and previous results we need the following notation. Let $E_{\text{FE}}, E_{\text{FS}}$ be the smallest errors obtainable by feature extraction and by feature selection respectively. Consider an algorithm α that produces a selection S from the matrix X. Its error is given by $E_\alpha(S,X) = \min_A \Theta(X - SA)$. For such algorithm one can define:

$$p_\alpha(X) = \frac{E_\alpha(S,X)}{E_{\text{FE}}}, \quad p_\alpha = \max_X p_\alpha(X)$$

Then the value of p_α indicates the estimation quality in the worst-case (e.g., Boutsidis (2009), Golub (2013)). The motivation behind this definition is that for any algorithm α and a matrix X we have: $1 \leq \frac{E_\alpha(S,X)}{E_{\text{FE}}} \leq p_\alpha$. Therefore, small values of p_α imply better worst-case performance. For example, Deshpande (2010) showed that for the Frobenius norm error in selecting r features $p_\alpha = \sqrt{r+1}$. Thus, we say that an algorithm α is optimal if $E_\alpha(S,X)$ is the smallest possible, and it is worst-case optimal if its p_α is the smallest possible.

Unsupervised feature selection formulated as CSSP has attracted a lot of attention, with the first algorithm (pivoted QR) being developed more than 50 years ago (Businger and Golub 1965). Recent results improve the accuracy, the running time, and the number of passes (e.g., (Paul, Magdon-Ismail, and Drineas 2015; Maung and Schweitzer 2013)). The problem was recently proved NP-hard (Shitov 2017). There are, however, polynomial algorithms that are worst-case optimal, and nontrivial optimal algorithms that run much faster than exhaustive search.

Numerical linear algebra studies focus on algorithms for minimizing the Spectral norm. The deterministic algorithm with the best worst-case error can be found in (Gu and Eisenstat 1996). A randomized algorithm with an improved worst-case accuracy for the Spectral norm is described in (Boutsidis, Mahoney, and Drineas 2009). The theoretical computer science community produced worst-case optimal and near optimal randomized algorithms for the Frobenius norm. These include, among others, (Deshpande et al. 2006; Guruswami and Sinop 2012). A worst-case optimal deterministic algorithm for the Frobenius norm is given in (Deshpande and Rademacher 2010; Guruswami and Sinop 2012).

The algebraic approach taken by most researchers was shown effective in deriving worst-case optimal algorithms, but so far has not produced optimal algorithms. Recent studies using classical AI tools of combinatorial search were used to derive optimal and near optimal algorithms in the Frobenius norm. See Arai (2015; 2016).

Hybrid low rank representation. As discussed above feature extraction and feature selection each have unique advantages and disadvantages. A hybrid representation that includes both extracted and selected features was previously proposed in (Kneip and Sarda 2011) and (Wang 2012). The main idea is that feature extraction works well in situations where the features are highly correlated, while feature selection works well in situations where the data is uncorrelated. Therefore, these studies apply feature extraction to remove the correlated components and follow it by feature selection. As we show this approach is not optimal.

1.2 Our results

In our model we fix both the number of selected features and the number of extracted features, and attempt to perform selection and extraction to minimize the approximation error in various norms. We show that the optimal combination of extraction and selection cannot be obtained by separate optimal algorithms for selection and extraction and requires a combinatorial search. To the best of our knowledge we are the first to make this observation.

The model we propose has r_1 selected features and r_2 extracted features. The approximation of X is given by:

$$X \approx SA_1 + VA_2, \text{ error} = \min_{A_1,A_2} \Theta(X - SA_1 - VA_2) \quad (2)$$

where S consists of r_1 columns from X and the r_2 columns of V are unconstrained. We refer to the representation in (2) as the "Hybrid Low Rank", or **HLR**. Our main result is an algorithm that computes HLR for any unitarily invariant error criteria. Observe that the HLR has simple feature extraction and simple feature selection as special cases.

The algorithm. An obvious approach to obtain a hybrid low rank representation is to start with the selection of r_1 features and follow it with the extraction of r_2 features. Another alternative is to have the order of selection and extraction reversed. However, it turns out (see Section 2.1) that *neither of these approaches is optimal*. Instead, we propose to use variants of a "best first" heuristic search to find optimal and near optimal HLR solutions.

The algorithm that we develop is based on the combinatorial approach to feature selection described in Arai (2016). The authors define a search graph for subsets, and use variants of A* to find a solution. The key to their algorithm is the introduction of heuristic functions that use eigenvalues. We show that the solution to the HLR can be found in a similar way, but with different heuristic functions.

The main contributions.

- A heuristic search algorithm for computing optimal and near optimal hybrid low rank (HLR).
- *A priori* and *a posteriori* bounds for these algorithms.
- Since feature selection is a special case of the HLR (r_2=0), our HLR algorithm can also be used for optimal feature selection in all unitarily invariant norms. In particular this gives the first optimal feature selection algorithm for the spectral norm and for the nuclear norm.

2 Hybrid Low Rank representations

To simplify expressions related to matrices that are sometimes used as sets of columns we use the following notation: For two matrices A, B with the same number of rows we write $A \subset B$ to indicate that the columns of A are a subset of the columns of B. We write $|A|$ for the number of columns in A, and $[A|B]$ for the matrix consisting of the columns of A followed by the columns of B.

Let Θ be an error criterion. We consider the following approximation errors:

$$\boxed{\begin{aligned} E_{\text{FE}}(X,r) &= \min_{V,A} \Theta(X - VA) \\ &\quad \text{subject to } |V| = r \\ E_{\text{FS}}(X,r) &= \min_{S,A} \Theta(X - SA) \\ &\quad \text{subject to } S \subset X,\, |S| = r \\ E_{\text{HLR}}(X,r_1,r_2) &= \min_{S,A_1,V,A_2} \Theta(X - SA_1 - VA_2) \\ &\quad \text{subject to } S \subset X,\, |S| = r_1,\, |V| = r_2 \end{aligned}} \tag{3}$$

From (3) it easily follows that with $r = r_1 + r_2$ we have:

$$E_{\text{FE}}(X,r) \le E_{\text{HLR}}(X,r_1,r_2) \le E_{\text{FS}}(X,r) \tag{4}$$

Thus, one would expect the HLR to have some desired properties of feature selection combined with some desired properties of feature extraction. For example, r_1 of the HLR features are easy to interpret (as in feature selection), and only r_2 of them are hard to interpret (as in feature extraction).

2.1 Greedy HLR is not Optimal

Suppose we are given a black box algorithm that computes optimal selection, and another black box algorithm that computes optimal extraction. We show by example that one cannot perform optimal selection followed by optimal extraction, or vice versa, to compute the optimal HLR. Consider the following two matrices:

$$X_1 = \begin{pmatrix} 100 & 0 & 1 \\ 0 & 1 & 100 \\ 0 & 100 & 50 \end{pmatrix}, \quad X_2 = \begin{pmatrix} 20 & 0 & 12 \\ -5 & 0 & 100 \\ 10 & 30 & 0 \end{pmatrix}$$

The goal for both matrices is to optimally select one column and extract one feature ($r_1 = 1, r_2 = 1$). If optimal selection of one feature is applied to X_1, the best selection (in Frobenius norm) is Column 3 (the error is $E_{\text{FS}}(X_1, 1)$=133.9). Combining the selection of Column 3 with an optimally extracted single feature reduces the error to 89.0. This, however, is not optimal. The selection of Column 1 with one extracted feature reduces the error to $E_{\text{HLR}}(X_1, 1, 1)$=77.4 which is optimal. This shows that optimal selection followed by optimal extraction does not guarantee the optimal HLR.

Similarly, if optimal extraction of one feature is applied to X_2 followed by optimal selection, the error is reduced to 20.44. The selection in this case is Column 2. This is not optimal since it is possible to extract a feature followed by the selection of Column 3 and reduce the error to $E_{\text{HLR}}(X_2, 1, 1)$=18.8. This shows that optimal extraction followed by optimal selection does not guarantee the optimal HLR.

3 HLR by heuristic search

A recent paper (Arai et al. 2016) has shown how to solve CSSP with the weighted A* algorithm for the Frobenius norm. They create a graph of subsets and perform the search on that graph. We use the same graph to convert the HLR into a graph search problem and study the performance of graph search algorithms for this problem. We propose two heuristics in a standard "best-first" setting. The first heuristic, that we call u, is an upper bound on the optimal HLR value. As we show, selecting graph nodes according to u gives a fast greedy algorithm.

The second heuristic, that we call f, is a lower bound on the optimal HLR value. We prove that using f by itself gives an algorithm that is guaranteed to find the optimal solution. Experimental results show that the algorithm runs much faster than exhaustive search (and produces the same results).

We linearly combine f and u to create the following heuristic: $f' = f + \epsilon u$. This gives a much faster algorithm than using f by itself. This is similar to the weighted A* approach, and we prove that the solution found by our algorithm comes with guaranteed bounds on its accuracy.

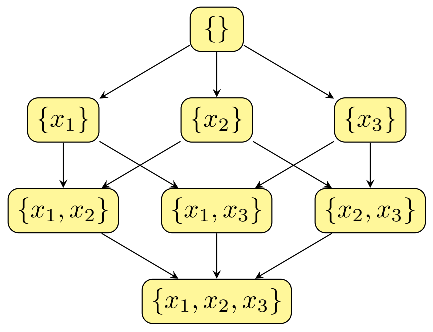

Figure 2: Example of the subsets graph

3.1 The subsets graph

The subsets graph is created with nodes corresponding to column subsets. There is an edge from subset S_i to subset S_j if adding one column to S_i creates S_j. The graph generated for the matrix $X = (x_1, x_2, x_3)$ is shown in Fig.2.

Even though a subset graph is not a tree, it has two properties that are typically associated with trees. The first property is that it has a root, corresponding to the empty subset. The second is that all paths leading from the root to a node can be considered equivalent. For example, if the goal node $\{x_1, x_3\}$ is found, it is irrelevant if it is reached by the path $\{\} \to \{x_1\} \to \{x_1, x_3\}$ or by the path $\{\} \to \{x_3\} \to \{x_1, x_3\}$. This is similar to the case of a tree where the choice of path leading to a node is irrelevant since there is a unique path leading from the root to any node.

3.2 The heuristic search algorithm

The algorithm in Figure 3 performs the search for the optimal HLR. It is similar to the standard "best-first" algorithm except for the following notable difference. The standard graph search algorithm updates a node in the fringe if a better path to it is found (in Line 8 of the algorithm, when n_j is in the fringe). In our algorithm there is no such update.

```
Input: $X$,$r_1$,$r_2$, and a heuristic function $f'(n)$.
Output: a subset $S$ of selected columns.
Data Structures: Each node $n_i$ keeps the subset $S_i$,
                 and $f'_i$. Two global lists: the fringe
                 list $L$, and the closed nodes list $C$.
Initialization: Put an empty subset into $L$.
while $L$ is nonempty do
    Pick $n_i$ with the smallest $f'_i$ from $L$. Ties are
      resolved in favor of the larger $|S_i|$ (depth).
    if $S_i$ contains $r_1$ columns then
        Stop and return $S_i$ as the solution subset.
    else
        Add $n_i$ to $C$.
        for each child $n_j$ of $n_i$ do
            if $n_j$ is not in $C$ or $L$ then
                Compute $f'_j$ from $X$, $S_j$, $r_1$, and $r_2$.
                Put $n_j$ with its corresponding $f'_j$ in $L$.
            end
        end
    end
end
```

Figure 3: The best-first search algorithm.

As explained in Section 3.1, all paths to the same subset are equivalent, and the value of the node depends only on the subset and not on the path leading to the subset.

3.3 Heuristic functions

The HLR is defined in terms of X, r_1, r_2. At each node n_i the subset S_i and its size $k_i = |S_i|$ are known. Recall that the error E_{HLR} is the smallest error of approximating X by a selection of r_1 columns and the best possible additional r_2 unconstrained vectors. The function d_i at node n_i is defined as the smallest error of approximating X by a selection of r_1 columns that include S_i and the best possible additional r_2 unconstrained vectors. This is the value at the best goal node below n_i. The function u_i at node n_i is defined as the smallest error of approximating X by the selection S_i and the best possible additional r_2 unconstrained vectors. The function f_i at node n_i is defined as the smallest error of approximating X by the selection S_i and the best possible additional $r_1 + r_2 - k_i$ unconstrained vectors, where $k_i = |S_i|$.

$$\begin{aligned}
&E_{\text{HLR}}(X, r_1, r_2) = \min_{S,A_1,V,A_2} \Theta(X - SA_1 - VA_2) \\
&\qquad \text{subject to } \; S \subset X, |S| = r_1, |V| = r_2 \\
&d_i = d(n_i, r_1, r_2) = \min_{S,A_1,V,A_2} \Theta(X - SA_1 - VA_2) \\
&\qquad \text{subject to } \; S_i \subset S \subset X, |S| = r_1, |V| = r_2 \\
&u_i = u(n_i, r_2) = \min_{A_1,V,A_2} \Theta(X - S_iA_1 - VA_2) \\
&\qquad \text{subject to } \; |V| = r_2 \\
&f_i = f(n_i, r_1, r_2) = \min_{A_1,V,A_2} \Theta(X - S_iA_1 - VA_2) \\
&\qquad \text{subject to } \; |V| = r_1 + r_2 - k_i
\end{aligned} \tag{5}$$

Observe that E_{HLR} and d_i cannot be calculated efficiently since the optimal selection S is unknown. By contrast, u_i and f_i use only the partial selection available at the given node, and as shown later can be computed efficiently. Clearly, the best heuristic choice for the algorithm is $f'_i = d_i$. But since it cannot be efficiently calculated we consider other choices using f_i and u_i. The motivation behind these choices is that both f_i and u_i can be viewed as approximations of d_i, as shown in Proposition 1.

Proposition 1: For each node n_i:

$$f(n_i, r_1, r_2) \le d(n_i, r_1, r_2) \le u(n_i, r_2)$$

and at a goal node (where k_i=r_1) the inequalities become equalities.

Proof:

- To see that $f_i \le d_i$ observe that both f_i and d_i use the same number of vectors ($r_1 + r_2$), and both use the k_i vectors in S_i. The rest of the vectors are unconstrained in f_i but partially constrained in d_i. This proves the left hand side inequality.
- To see that $d_i \le u_i$, let S_i, V be the vector subsets that are used to calculate u_i. The minimum in the definition of d_i includes the subsets S_i and V (and additional vectors). This proves the right hand side inequality.
- If $k_i = r_1$ then the definitions of f_i and u_i are identical.

■

4 The three variants of the algorithm

Proposition 1 shows that the optimal heuristic d_i is "sandwiched" between f_i and u_i. We consider three different options for running the algorithm. The first is the choice $f'_i = u_i$, the second is the choice $f'_i = f_i$, and the third takes f'_i between f_i and u_i. Specifically, for the third choice we observe that taking $f'_i = (1-\beta)f_i + \beta u_i$ with $0 \le \beta \le 1$ is equivalent to taking $f'_i = f_i + \epsilon u_i$ with $\epsilon = \frac{1-\beta}{\beta}$, $\epsilon \ge 0$.

The Greedy HLR algorithm: $f'_i = u_i$. We prove in Theorem 1 that using f'_i=u_i gives a greedy algorithm that examines exactly r_1 nodes before terminating with a solution.

The Optimal HLR algorithm: $f'_i = f_i$. We prove in Theorem 2 using f'_i=f_i gives an algorithm that is guaranteed to find the optimal solution.

The Suboptimal HLR algorithm: $f'_i = f_i + \epsilon u_i$. We prove in Theorem 3 that using f'_i=f_i+ϵu_i guarantees a solution "close" to the optimum.

Bounds on the distance between the optimum and the solution can be calculated *a priori* before the algorithm is executed, and *a posteriori*, after the algorithm terminates.

4.1 Proofs

Theorem 1: With the choice $f'_i = u_i$, the algorithm terminates after examining r_1 nodes.

Theorem 2: With the choice $f'_i = f_i$, the algorithm terminates with an optimal solution. The optimal solution error is $E_{\text{HLR}}(X, r_1, r_2)$.

Theorem 3: Let n_* be an optimal solution node for the HLR. Let $e^* = E_{\text{HLR}}(X, r_1, r_2)$ be the error at n_*. Suppose the algorithm is using $f'_i = f_i + \epsilon u_i$, with $\epsilon \geq 0$. Let n_{**} be the goal node found by the algorithm. Let e^{**} be the error at n_{**}, and f_{**} be the value of f at n_{**}. Let $u_{\max}$ be the largest value of u in the nodes remaining at the Fringe list after the goal node is reached. Then:

$$e^{**} \leq e^* + \epsilon(u_{\max} - f_{**}) \tag{6}$$

Lemma 1: f_i is monotonically increasing along any path.
Proof of Lemma 1: Suppose n_j is a child of n_i, so that $S_j = [S_i|x]$, where x is the added column. We need to show:

$$f(n_j, r_1, r_2) = \min_{|V_j|=r_1+r_2-k_i-1} \min_A \Theta(X - [S_i|x|V_j]A)$$
$$\geq \min_{|V_i|=r_1+r_2-k_i} \min_A([S_i|V_i]A) = f(n_i, r_1, r_2)$$

This follows because the minimum on the right hand side has one unconstrained vector that is constrained on the left hand side. ■

Lemma 2: The value of u_i is monotonically decreasing along any path.
Proof of Lemma 2: We need to show that if n_j is the child of n_i then $u_j \leq u_i$. From the definition in (5) the right hand side reduces the error with the subset S_i while the left hand side reduces the error with the subset S_j, which includes S_i and one additional column. Clearly, the additional column can only reduce the error. ■

Lemma 3: Consider the choice $f'_i = u_i$. Let n_i be the node picked at Line 2 of the algorithm. Let n_j be a child of n_i. The following two properties hold:
a. The depth $|S_j|$ of n_j is larger than the depth of all other nodes currently in the fringe.
b. The next node to be picked is a child of n_i.
Proof of Lemma 3: The proof is by induction. Property a follows trivially from Property b. To prove Property b observe that from Lemma 2, u_i is monotonically decreasing (non-increasing) along any path. Therefore, the f' values of the children of n_i will be no greater than the f' values of all the nodes currently in the fringe. Property a guarantees that the tie breaker will always be decided in favor of a child, so that the child of n_i will be selected next. ■

Lemma 4: Suppose Theorem 3 is false. Then for any node n_z on the path from the root to n_* the following condition holds: $f'_z < f'_{**}$.
Proof of Lemma 4: The falsehood of Theorem 3 can be written as follows: $e^{**} > e^* + \epsilon(u_{\max} - e^{**})$. Since both n_* and n_{**} are goal nodes Proposition 1 implies: $e^{**} = f_{**} = u_{**}$ and $e^* = f_* = u_*$. Using this and some algebra it can be shown that an equivalent falsehood condition is: $f_{**} > f_* + \frac{\epsilon}{1+\epsilon}(u_{\max} - f_*)$. The lemma can now be proved as follows:

$$\begin{aligned} f'_{**} &= f_{**} + \epsilon u_{**} = (1+\epsilon)f_{**} && (c_1) \\ &> (1+\epsilon)f_* + \epsilon(u_{\max} - f_*) = f_* + \epsilon u_{\max} && (c_2) \\ &\geq f_z + \epsilon u_{\max} \geq f_z + \epsilon u_z = f'_z && (c_3) \end{aligned}$$

c_1 : from the definition of f'. c_2 : from the equivalent falsehood assumption. c_3 : from Lemma 1 $f_* > f_z$. ■

Proof of Theorem 1: The proof follows trivially from Lemma 3. ■

Proof of Theorem 2: The proof follows as a corollary of Theorem 3 with $\epsilon = 0$. ■

Proof of Theorem 3: If the theorem is false then from Lemma 4 it follows that all nodes on the path from the root to n_* have smaller f'_i values than f'_{**}. Since at any given time at least one of them is in the fringe list, they should all be selected before n_{**} is selected. But this means that n_* is selected as the solution and not n_{**}. ■

4.2 *A priori* and *a posteriori* bounds

Both the Greedy HLR and the Suboptimal HLR are not guaranteed to produce the optimal solution. We proceed to show how to obtain bounds on how close their solution is to the optimal. We call a bound *a priori* if it can be calculated before the run of the algorithm and *a posteriori* if it can only be calculated after the run of the algorithm.

Consider a run of a nonoptimal algorithm producing the nonoptimal value of f_{**}, while the optimal value is f_*. The value of f_{**} can be bounded as follows:

$$f_{**} \leq f_* + B, \quad B \geq f_{**} - f_*$$

We refer to the value of B as a bound, where a smaller B indicates a better bound, and $B = 0$ implies an optimal solution.

The *a posteriori* bounds that we describe require the examination of the fringe list after the run of the algorithm. In particular we compute the following two values from the fringe list:

$$f_{\min} = \min_{n_i \in F} f_i, \quad u_{\max} = \max_{n_i \in F} u_i$$

In addition, the *a posteriori* bounds use the value f_{**} at the (nonoptimal) goal node.

From Lemma 1 it follows that $f_* \geq f_{min}$, so that $B_1 = f_{**} - f_{\min}$ is an *a posteriori* bound for all variants of the algorithm.

Greedy HLR. Greedy HLR has the following *a priori* bound: $B_2 = u_{\text{root}} - f_{\text{root}}$. This bound follows from Proposition 1. The only *a posteriori* bound of Greedy HLR is B_1.

Suboptimal HLR. Suboptimal HLR has the following *a priori* bound: $B_3 = \epsilon u_{\text{root}}$. This bound follows from Theorem 3 and Lemma 2 by observing that:

$$\epsilon(u_{\max} - f_{**}) \leq \epsilon u_{root}$$

Suboptimal HLR has two *a posteriori* bounds: B_1 and $B_4 = \epsilon(u_{\max} - f_{**})$. Clearly, its effective bound is the minimum of the two.

4.3 Using *a posteriori* bound to improve the result

A paper by Thayer and Ruml (2008) shows how to use the *a posteriori* bounds to improve the output of the classic weighted A* algorithm. The idea is to run the weighted A* algorithm to convergence, and then identify the node in the fringe list that affects the bound the most. That node is

```
Input: $X$,$r_1$,$r_2$, $\epsilon$, $T$.
Output: a subset $S$ of selected columns.
1  Start with an empty fringe $F$ and a Closed list $C$.
2  for $t = 1, \ldots, T$ do
3      Run either the Greedy HLR or the Suboptimal HLR
         to convergence, using $F$ and $C$.
4      Go over the fringe $F$, identify the nodes $n_{b1}$ and
         $n_{b4}$, and compute the values of $B_1, B_4$.
           $n_{b1} = \arg\min_{n_i \in F} f_i, \quad n_{b4} = \arg\max_{n_i \in F} u_i$
5      if $B_1 < B_4$ then
6          Expand $n_{b1}$.
7      else
8          Expand $n_{b4}$.
9      end
10 end
```

Figure 4: Optimistic Search Algorithm

then expanded, its children are added to the fringe, and the weighted A* algorithm continues with the new fringe. Typically, a single iteration of this algorithm would either improve goal node or improve the *a posteriori* bound. Fig.4 describes the algorithm in detail.

5 Relationship to previous work

In this section we discuss the relationship between the algorithm presented here and classical work on the weighted A* algorithm. We also compare our work to the results of (Arai et al. 2016).

There are many similarities between our model and the classical weighted A* graph search algorithm (e.g. (Pearl 1984)). The most important one is introduction of the heuristic function f with the following three key properties: 1. f is a lower bound on the true value at the goal. 2. f is monotonically increasing. 3. At a goal node the value of f is the value that one attempts to minimize. Although a heuristic function is also introduced in the classical theory of (weighted) A* search, its definition is entirely different. On the other hand, there is no function in our setting that corresponds naturally to the functions g (distance from the root) or h (heuristic) in the classical theory. Similarly, there is no natural function in the classical theory that corresponds to the function u in our setting.

The similarity in the properties of f makes our suboptimality proofs similar to the classical proofs of weighted A* suboptimality (e.g. (Pearl 1984)). However, since the heuristic functions used here are different from those used in graph search, one cannot use the classical proofs "as is" and apply them to our case. In particular, our Lemma 1 has a corresponding lemma in the classical theory, and our proof idea of Lemma 4 is similar (but not identical) to the classical theory. However, there is no correspondence to our Proposition 1 (right hand side), Lemma 2, and Theorem 1. The bound obtained in Theorem 3 is also different. The result for the classical weighted A* algorithms are in terms of a relative bound, while the guarantees in our case are in terms of an additive bound. Still, the similarity between the approaches enables us to map ideas that were developed in the classical theory to our setting. We demonstrated this with the Optimistic Search Algorithm that can be applied almost verbatim in our case. (The only difference is the exact formulas for the *a posteriori* bounds.)

Our work is motivated by the study described in Arai (2016). The main difference is that our results are for the HLR, and do not use any norm specific assumptions. By contrast, the Arai proofs are for the CSSP which is a special case of the HLR, and they make use of the Frobenius norm assumption.

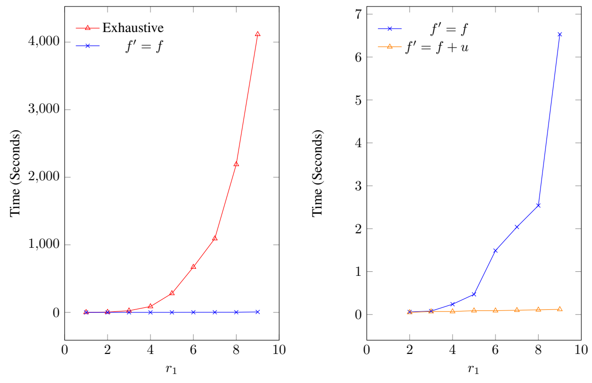

Figure 5: Run-time results HLR on the dataset *vehicle*. x-axis shows r_1 and r_2=$10-r_1$. Error criterion is the Schatten p-Norm with p=0.25.

6 Experimental Results

Efficiently computing f_i and u_i. The optimality proof does not use any properties of the error criterion Θ. However, an efficient computation requires the norms to be unitarily invariant. From the definition of f_i, u_i in (5) the challenge is the computation of the coefficient matrices A_1, A_2, and the unconstrained matrix V. For all unitarily invariant norms A_1, A_2 can be calculated with a pseudo inverse, and the matrix V by calculating eigenvectors. For other norms it is not immediately clear how to compute these values. Specifically, for the entry-wise l_0, l_1 these calculations are known to be NP-hard. See Gillis (2018).

Running time. Fig.5 shows running-time on the dataset *vehicle*. The left panel shows that the algorithm with $f_i' = f_i$ is significantly faster than exhaustive search. The right panel shows that using $f_i + u_i$ runs much faster than f_i.

Optimal feature selection. As discussed in Section 1.2 feature selection is a special case of the HLR. Our algorithm is the first nontrivial algorithm for optimal feature selection for unitarily invariant error criteria besides Frobenius. The results for various norms are shown in Table 1. We do not include the results when algorithms run more than five minutes. They are compared with two algorithms. The column ARSS shows results obtained by the algorithm of Zhu (2015). Their algorithm cannot be used to compute the

Error Criterion	r_1	$f' = f$	$f' =$	$f' =$	$f' =$	$f' = u$	$f' = u$		ARSS	GE
			$f + 0.2u$	$f + 0.4u$	$f + 0.8u$		*a priori*	*a posteriori*		
vehicle dataset ($m = 846$, $n = 18$)										
Nuclear	5	**1399.20**	1402.64	1569.49	1569.49	1569.49	24490.7	270.83	3465.75	-
Spectral	5	**247.58**	326.12	326.12	326.12	326.12	19600.32	82.66	-	248.58
Nuclear	10	**466.85**	520.18	520.18	520.18	520.18	25371.7	105.55	1682.16	-
Spectral	10	**112.19**	138.80	144.99	144.99	148.60	19744.0	48.85	-	131.68
spectf dataset ($m = 267$, $n = 45$)										
Nuclear	5	**3814.14**	**3814.14**	3816.69	3817.42	3821.42	8334.75	435.57	4598.65	-
Spectral	5	**252.69**	257.91	290.60	290.60	280.45	6841.58	78.09	-	348.24
Nuclear	15	-	-	2297.04	**2292.79**	**2292.79**	9850.00	457.51	3091.28	-
Spectral	15	-	152.12	**151.83**	165.41	1883.42	6938.17	82.43	-	154.62
libras dataset ($m = 360$, $n = 90$)										
Nuclear	4	**68.44**	68.53	68.53	68.48	71.55	135.88	11.03	91.79	-
Spectral	4	**8.558**	9.954	9.954	9.954	13.182	84.62	4.80	-	11.863
Nuclear	30	-	**6.134**	6.185	6.322	6.322	189.90	1.89	8.235	-
Spectral	30	-	**0.343**	0.351	0.351	0.712	92.80	0.50	-	0.4211

Table 1: Accuracy comparison under Nuclear norm and Spectral norm. The minimum error is highlighted.

Spectral norm, so we use the algorithms of Gu and Eisenstat (1996) instead.

$Norm$	r_1	r	l_0 error	l_1 error
spectf dataset ($m = 267$, $n = 45$)				
Nuclear	1	30	0.693	1.16
	3	30	0.671	1.15
	5	30	0.647	1.13
$p = 0.25$	1	30	0.691	1.16
	3	30	0.675	1.16
	5	30	0.649	1.14
vehicle dataset ($m = 846$, $n = 18$)				
Frobenius	1	10	0.562	0.92
	5	10	0.472	0.831
	9	10	0.342	0.71
$p = 0.4$	1	10	0.562	0.92
	5	10	0.465	0.819
	9	10	0.342	0.71

Table 2: Reduction in l_0 and l_1 entrywise norms with increased r_1

Minimizing entry-wise l_0 and l_1 norms. As discussed in Section 1.1 a current topic of interest is the computation of low rank representation minimizing entrywise l_0 and l_1 norms. We found experimentally that feature selection typically gives lower errors for entry-wise l_0 and l_1 norms than feature extraction, though feature extraction performs better in terms of the unitarily invariant norm used as the error criterion. The hybrid low rank approach allows us to balance this trade-off, reducing the unitarily invariant norm while at the same time reducing the error in the l_0 and/or l_1 norms. Table 2 shows that for a fixed r, increasing r_1 indeed reduces the entry-wise l_0 and l_1 norms.

Experiments with big sparse data. We describe experiments with the Greedy HLR algorithm applied to the

r_1	*Bound*	$r_2 = 0$	$r_2 = 5$	$r_2 = 10$
100	*a priori*	530.37	122.62	100.46
	a posteriori	0.19	0.15	0.13
	solution error	21354.43	15511.81	11278.12
120	*a priori*	1606.61	430.05	391.93
	a posteriori	0.24	0.16	0.12
	solution error	7056.89	4445.01	2909.46
140	*a priori*	9410.57	4065.81	9779.79
	a posteriori	0.28	0.17	0.07
	solution error	1205.26	470.98	116.84

Table 3: Greedy HLR on TechTC01 data with relative bounds

TechTC dataset. The matrix size in this case is 163×29261. This means that the algorithm selection is from 29261 features. Exhaustive search algorithms are clearly not practical in this case. (For example, there are approximately 10^{288} subsets of selecting 100 features out of 29261, which is significantly more than the number of atoms in the universe.)

The results are shown in Table 3. The value of the bounds is given as the ratio between the bounds and the errors at the goal node.

Experiments with the Optimistic Search Algorithm. We do experiments with the Optimistic Search Algorithm, as discussed in Section 4.3. The results are shown in Table 4. Observe that the solution error does not change, but the relative error bound is being reduced (slightly) with additional iterations.

7 Concluding remarks

This paper introduces the "Hybrid Low Rank" (HLR) representation of a matrix as a low rank matrix representation that uses both selected features and extracted features. It was shown that an optimal HLR representation cannot be obtained by first selecting features and then extracting features, or vice versa. Instead, it requires a combinatorial search.

$r_1 : r_2$	Iterations			solution error
	1	10	100	
42:0	0.09173	0.09171	0.09156	273585.83
42:5	0.06124	0.06122	0.06105	214917.10
42:10	0.04434	0.04434	0.04416	170169.98
5:0	0.04592	0.04528	0.03664	2.02e6
5:5	0.01126	0.01033	0.00854	1.16e6
5:10	0.00388	0.00385	0.00299	8.25e5

Table 4: Relative *a posteriori* bounds of the Greedy HLR with Optimistic Search Algorithm on the TechTC01 dataset

An algorithm that uses the "best-first" heuristic search approach was described. Three variants, optimal, suboptimal and greedy, were described, This heuristic search technique allows us to compute *a priori* and *a posteriori* bounds, which show how close the results are to the optimal solution. *A priori* bounds can be computed before the run of the algorithm. *A posteriori* bounds can be computed after termination. The paper also shows how to use the *a posteriori* bound to improve the solution accuracy.

A short abstract describing some of the results in this paper appears in (Shah et al. 2018b). Similar ideas were also used to derive new algorithms for robust PCA. See (Shah et al. 2017; 2018a).

References

Arai, H.; Maung, C.; Xu, K.; and Schweitzer, H. 2016. Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In *AAAI'16*, 666–672.

Arai, H.; Maung, C.; and Schweitzer, H. 2015. Optimal column subset selection by A-Star search. In *AAAI'15*, 1079–1085.

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

Bringmann, K.; Kolev, P.; and Woodruff, D. P. 2017. Approximation algorithms for ℓ_0-low rank approximation. In *NIPS'17*. Curran Associates, Inc.

Businger, P., and Golub, G. H. 1965. Linear least squares solutions by Householder transformations. *Numer. Math.* 7:269–276.

Chierichetti, F.; Gollapudi, S.; Kumar, R.; Lattanzi, S.; Panigrahy, R.; and Woodruff, D. P. 2017. Algorithms for ℓ_p low-rank approximation. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 806–814. PMLR.

Deshpande, A., and Rademacher, L. 2010. Efficient volume sampling for row/column subset selection. In *FOCS*, 329–338. IEEE Computer Society Press.

Deshpande, A.; Rademacher, L.; Vempala, S.; and Wang, G. 2006. Matrix approximation and projective clustering via volume sampling. *Theory of Computing* 2(12):225–247.

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

Gillis, N., and Vavasis, S. A. 2018. On the complexity of robust pca and l1-norm low-rank matrix approximation. *Mathematics of Operations Research* in press.

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

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

Guruswami, V., and Sinop, A. K. 2012. Optimal column-based low-rank matrix reconstruction. In Rabani, Y., ed., *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, 1207–1214. SIAM.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3:1157–1182.

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

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

Kneip, A., and Sarda, P. 2011. Factor models and variable selection in high-dimensional regression analysis. *The Annals of Statistics* 39(5):2410–2447.

Li, H.; Linderman, G. C.; Szlam, A.; Stanton, K. P.; Kluger, Y.; and Tygert, M. 2017. Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Trans Math Softw.* 43(3):28:1–28:14.

Maung, C., and Schweitzer, H. 2013. Pass-efficient unsupervised feature selection. In *Advances in Neural Information Processing Systems (NIPS)*, volume 26, 1628–1636.

Paul, S.; Magdon-Ismail, M.; and Drineas, P. 2015. Column selection via adaptive sampling. In *NIPS'15*. Curran Associates, Inc. 406–414.

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

Shah, S.; He, B.; Maung, C.; and Schweitzer, H. 2017. Computing robust principal components by A* search. In *IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 1042 – 1049.

Shah, S.; He, B.; Maung, C.; and Schweitzer, H. 2018a. Computing robust principal components by A* search. *International Journal on Artificial Intelligence Tools* 27(7).

Shah, S.; He, B.; Xu, K.; Maung, C.; and Schweitzer, H. 2018b. Solving generalized column subset selection with heuristic search. In *Proceedings of the 32nd National Conference on Artificial Intelligence (AAAI'18)*, 8153–8154. AAAI Press.

Shitov, Y. 2017. Column subset selection is np-complete. arXiv e-print (arXiv:1701.02764[math.CO]).

Song, Z.; Woodruff, D. P.; and Zhong, P. 2017. Low rank approximation with entrywise ℓ_1-norm error. In *STOC'17*, 688–701. New York, NY, USA: ACM.

Thayer, J. T., and Ruml, W. 2008. Faster than weighted a*: An optimistic approach to bounded suboptimal search. In *Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'08, 355–362. AAAI Press.

Wang, H. 2012. Factor profiled sure independence screening. *Biometrika* 99(1):15–28.

Zhu, F.; Fan, B.; Zhu, X.; Wang, Y.; Xiang, S.; and Pan, C. 2015. 10,000+ times accelerated robust subset selection. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 3217–3223.