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...
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 $E_{FE}, E_{FS}$ be the smallest errors obtainable by feature extraction and by feature selection respectively. Consider an algorithm $\alpha$ 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:

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

Then the value of $p_\alpha$ indicates the estimation quality in the worst-case (e.g., Boutsidis (2009), Golub (2013)). The motivation behind this definition is that for any algorithm $\alpha$ and a matrix $X$ we have: $1 \leq \frac{E_\alpha(S, X)}{E_{FE}} \leq p_\alpha$. Therefore, small values of $p_\alpha$ imply better worst-case performance. For example, Deshpande (2010) showed that for the Frobenius norm error in selecting $r$ features $p_\alpha = \sqrt{r + 1}$. With this notation we say that an algorithm $\alpha$ is optimal if $E_\alpha(S, X)$ is the smallest possible, and it is worst-case optimal if its $p_\alpha$ 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). The problem was recently proved NP-hard (Shitov 2017), and therefore there are no optimal polynomial algorithms. 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:

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

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 $\Theta$ be an error criterion. We consider the following approximation errors:

\[
\begin{align*}
E_{TE}(X, r) &= \min_{V,A} \Theta(X - VA) \\
& \text{subject to } |V| = r \\
E_{FS}(X, r) &= \min_{S,A} \Theta(X - SA) \\
& \text{subject to } S \subseteq X, |S| = r \\
E_{HLR}(X, r_1, r_2) &= \min_{S,A} \Theta(X - SA_1 - VA_2) \\
& \text{subject to } S \subseteq X, |S| = r_1, |V| = r_2
\end{align*}
\]  

(3)

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

\[
E_{TE}(X, r) \leq E_{HLR}(X, r_1, r_2) \leq E_{FS}(X, r)
\]  

(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 $E_{FS}(X_1, 1) = 133.9$). 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 $E_{HLR}(X_1, 1, 1) = 77.4$ 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 $E_{HLR}(X_2, 1, 1) = 18.8$. 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.

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.

![Figure 2: Example of the subsets graph](image)

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 $\{\} \rightarrow \{x_1\} \rightarrow \{x_1, x_3\}$ or by the path $\{\} \rightarrow \{x_3\} \rightarrow \{x_1, x_3\}$. 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 differences. 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. As explained in Section 3.1, all paths to the same subset are...
The HLR is defined in terms of $S$ as the smallest error of approximating $n$ below $d$ing $r$ing $n$. Add $6$ Stop and return $S_i$ as the solution subset.

**Initialization:** Put an empty subset into $L$. 

**while** $L$ is nonempty **do**

1. Pick $n_i$ with the smallest $f'_i$ from $L$. Ties are resolved in favor of the larger $|S_i|$ (depth).
2. **if** $S_i$ contains $r_1$ columns **then**
   - Stop and return $S_i$ as the solution subset.
3. **else**
4.   - Add $n_i$ to $C$.
5.   **for** each child $n_j$ of $n_i$ **do**
6.     **if** $n_j$ is not in $C$ or $L$ **then**
7.       Compute $f'_j$ from $X$, $S_j$, $r_1$, and $r_2$.
8.       Put $n_j$ with its corresponding $f'_j$ in $L$.
9.     **end**
10. **end**
11. **end**
12. **end**

Figure 3: The best-first search algorithm.

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 $k_i = |S_i|$.

$$E_{HLR}(X, r_1, r_2) = \min_{S, A_1, V, A_2} \Theta(X - SA_1 - VA_2)$$

subject to $S \subset X, |S| = r_1, |V| = r_2$

$$d_i = d(n_i, r_1, r_2) = \min_{S, A_1, V, A_2} \Theta(X - SA_1 - VA_2)$$

subject to $S_i \subset S \subset X, |S| = r_1, |V| = r_2$

$$u_i = u(n_i, r_2) = \min_{A_1, V, A_2} \Theta(X - S_iA_1 - VA_2)$$

subject to $|V| = r_2$

$$f_i = f(n_i, r_1, r_2) = \min_{A_1, V, A_2} \Theta(X - S_iA_1 - VA_2)$$

subject to $|V| = r_1 + r_2 - k_i$

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 $n_i$:

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

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

**Proof:**

- To see that $f_i \leq d_i$ 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 $d_i \leq u_i$, 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 $0 \leq \beta \leq 1$ is equivalent to taking $f'_i = f_i + \epsilon u_i$ with $\epsilon = \frac{1-\beta}{\beta}, \epsilon \geq 0$.

**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 $f'_i=f_i+\epsilon u_i$ 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 $E_{HLR}(X, r_1, r_2)$. 

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

**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:
\[
f(n_j, r_1, r_2) \geq \min_{[V_j]=r_1+r_2-k_i} \min_A \Theta(X - [S_i][V_j]) = f(n_i, r_1, r_2)
\]
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 \( n_j \).

**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) among 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 \( n_\ast \) the following condition holds:
\[ f'_i < f_{\ast\ast} \]

**Proof of Lemma 4:** The falsehood of Theorem 3 can be written as follows: \( e^{**} > e^* + \epsilon(u_{\max} - e^{**}) \). Since both \( n_\ast \) and \( n_{\ast\ast} \) are goal nodes Proposition 1 implies: \( e^{**} = f_{\ast\ast} = u_{\ast\ast} \) and \( e^* = f_\ast = u_\ast \). Using this and some algebra it can be shown that an equivalent falsehood condition is: \( f_{\ast\ast} > f_\ast + \frac{\epsilon}{1+\epsilon}(u_{\max} - f_\ast) \). The lemma can now be proved as follows:
\[
f_{\ast\ast} = f_{\ast\ast} + \epsilon u_{\ast\ast} = (1+\epsilon)f_{\ast\ast} - f_\ast \quad (c_1)
\]
\[
(1+\epsilon)f_{\ast\ast} + \epsilon(u_{\max} - f_\ast) = f_{\ast\ast} + \epsilon u_{\max} \quad (c_2)
\]
\[
(1+\epsilon)f_{\ast\ast} \geq f_\ast + \epsilon u_{\max} \quad (c_3)
\]
\[
e: \text{from the definition of } f' \quad e_2: \text{from the equivalent falsehood assumption. } e_3: \text{from Lemma 1 } f_\ast > f_z.
\]

**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 \( n_\ast \) have smaller \( f'_i \) values than \( f_{\ast\ast} \). Since at any given time at least one of them is in the fringe list, they should all be selected before \( n_{\ast\ast} \) is selected. But this means that \( n_\ast \) is selected as the solution and not \( n_{\ast\ast} \). ■

### 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 \( f_{\ast\ast} \), while the optimal value is \( f_\ast \). The value of \( f_{\ast\ast} \) can be bounded as follows:
\[ f_{\ast\ast} \leq f_\ast + B, \quad B \geq f_{\ast\ast} - f_\ast \]

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:
\[ f_{\min} = \min_{i \in F} f_i, \quad u_{\max} = \max_{i \in F} u_i \]

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

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

**Greedy HLR.** Greedy HLR has the following *a priori* bound: \( B_2 = u_{\text{root}} - f_\ast \). 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:
\[ \epsilon(\max - f_{\ast\ast}) \leq \epsilon u_{\text{root}} \]

Suboptimal HLR has two *a posteriori* bounds: \( B_1 \) and \( B_3 = \epsilon(u_{\max} - f_{\ast\ast}) \). 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 Rumel (2008) shows how to use the *a posteriori* bounds to improve the output of the classical 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
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.

Figure 4: Optimistic Search Algorithm

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.

6 Experimental Results

Efficiently computing \( f_i \) and \( u_i \). The optimality proof does not use any properties of the error criterion \( \Theta \). 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. x-axis shows \( r_1 \) and \( r_2=10-r_1 \). Error criterion is the Schatten \( p \)-Norm with \( p=0.25 \).

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 \( p=0.25 \).
Table 1: Accuracy comparison under Nuclear norm and Spectral norm. The minimum error is highlighted.

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

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.

Table 3: Greedy HLR on TechTC01 data with relative bounds

Experiments with big sparse data. We describe experiments with the Greedy HLR algorithm applied to the TechTC dataset. The matrix size in this case is \( 163 \times 29261 \). 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
Table 4: Relative a posteriori bounds of the Greedy HLR with Optimistic Search Algorithm on the TechTC01 dataset

<table>
<thead>
<tr>
<th>$r_1 : r_2$</th>
<th>Iterations</th>
<th>solution error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>42:0</td>
<td>0.09173</td>
<td>0.09171</td>
</tr>
<tr>
<td>42:5</td>
<td>0.06124</td>
<td>0.06122</td>
</tr>
<tr>
<td>42:10</td>
<td>0.04434</td>
<td>0.04434</td>
</tr>
<tr>
<td>5:0</td>
<td>0.04592</td>
<td>0.04528</td>
</tr>
<tr>
<td>5:5</td>
<td>0.01126</td>
<td>0.01033</td>
</tr>
<tr>
<td>5:10</td>
<td>0.00388</td>
<td>0.00385</td>
</tr>
</tbody>
</table>

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.

An algorithm that uses the “best-first” heuristic search approach was described. Three variants, optimal, suboptimal and greedy, were listed, all with the same overall structure but each using different heuristics. A priori and a posteriori bounds are provided for suboptimal and greedy variants. These two bounds show how close the results are to the optimal solution. The paper also shows how to use the a posteriori bound to improve the final output results.

The algorithm works with any unitarily invariant norm. By setting the extraction feature number to zero, the algorithm can be used to select the optimal features. This is the first optimal feature selection algorithm for the spectral norm and the nuclear norm.

References


