Heuristic Search Algorithms for Approximating One Matrix in Terms of Another Matrix

Guihong Wan,[1] **Haim Schweitzer**[1]
[1] The University of Texas at Dallas
Guihong.Wan@utdallas.edu, HSchweitzer@utdallas.edu

Abstract

We study the approximation of a target matrix in terms of few selected columns of another matrix, sometimes called "a dictionary". The approximation of a target matrix is a general setting which includes multi-target prediction, multi-label classification, and multi-variate regression among others. An optimal column selection algorithm for the special case where the target matrix has only one column is known since the 1970's, but previously proposed column selection algorithms for the general case are greedy. We propose the first nontrivial optimal algorithm for the general case, using a heuristic search setting similar to the classical A^* algorithm. We also propose practical sub-optimal algorithms in a setting similar to the classical Weighted A^* algorithm. Experimental results show that our sub-optimal algorithms compare favorably with the current state-of-the-art greedy algorithms.

1 Introduction

Let $Y = (y_1 \dots y_N)$ be a target matrix of m rows and N columns. Let $X = (x_1 \dots x_n)$ be a "dictionary" matrix of m rows and n columns. Consider selecting $k \le n$ columns from X that linearly approximate Y:

$$Y \approx SA, \tag{1}$$

where $S = (x_{s_1} \dots x_{s_k})$ is the $m{\times}k$ selection matrix, and A is the coefficients matrix of size $k{\times}N$, as illustrated in Figure 1. The quality of the selected columns in S is measured by the following error measured in the Frobenius norm:

$$E(S) = \min_A \|Y - SA\|_F^2 \tag{2}$$

The following are special cases:

- When N=1, the problem is sometimes called supervised subset selection. The criterion in (2) is equivalent to minimizing the residual sum of squares (RSS).
- When X=Y this is knows as unsupervised feature selection, or the column subset selection problem (CSSP).

Optimal solutions, as well as approximations within a constant are known to be NP-hard even for the N=1 case (Natarajan 1995; Davis, Mallat, and Avellaneda 1997; Amaldi and Kann 1998).

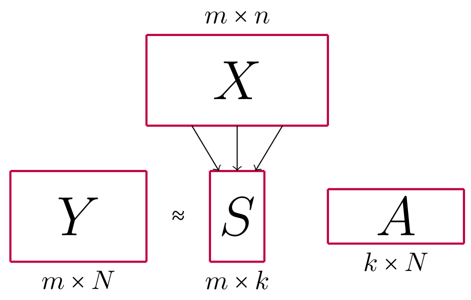

Figure 1: Illustration of the problem. The target matrix Y is approximated by linearly combining of columns in S which are selected from the dictionary matrix X.

Extensive studies were directed at the N=1 case. See, e.g., (Mallat 1999; Tropp 2004; Furnival and Wilson 1974; Zhang 2009; Qian, Yu, and Zhou 2015). Applications include signal processing (e.g., (Mallat 1999; Pati, Rezaiifar, and Krishnaprasad 1993; Neff and Zakhor 1997; Tropp 2004)) and supervised feature selection in linear regression (e.g., (Furnival and Wilson 1974; Hastie, Tibshirani, and Friedman 2009; Neff and Zakhor 1997; Zhang 2009; Qian, Yu, and Zhou 2015)). An optimal algorithm proposed in (Furnival and Wilson 1974) finds the best solution but it is not feasible for large k and large datasets.

Previously proposed approximate solutions can be categorized into roughly three main groups: forward selection, backward elimination, and convex relaxation. Forward selection sequentially adds columns that improve the quality the most. Backward elimination starts with the full selection and sequentially removes the columns that affect the quality the least. As shown in (Zhang 2009), the two techniques have limitations due to their greedy behavior. The author introduced an adaptive algorithm that combines the two ideas to alleviate the flaws. The convex relaxation approaches replaces some natural constraints (sometimes defined in terms of l_0 norm) with other convex constraints. An example is the l_1 norm is used in the Lasso technique (Tibshirani 1996). In (Qian, Yu, and Zhou 2015), the authors treated the subset selection as a bi-objective optimization problem. Their algorithm is optimal for data drawn from Exponential Decay distribution.

The case where X=Y is known as unsupervised feature selection or unsupervised column subset selection. See, e.g., (Golub and Van-Loan 2013; Wei and Billings 2007; Çivril and Magdon-Ismail 2012; Maung and Schweitzer 2013; Arai, Maung, and Schweitzer 2015; Arai et al. 2016). The algorithms proposed in (Arai, Maung, and Schweitzer 2015; Arai et al. 2016) recast this problem as a graph search problem and apply the A^* and weighted A^* to solve it. Our method is motivated by this approach. The algorithms developed in these studies can be viewed as special cases of our method.

The general case where Y is a matrix did not attract as much attention as the N= 1 case. We observe that one cannot simply apply an N=1 algorithm separately to each column of Y. The challenge of the general case is to find columns in X that can simultaneously approximate all columns in Y.

Previously proposed algorithms for the general case are greedy. They include: (Maung and Schweitzer 2015; Tropp, Gilbert, and Strauss 2006; Çivril and Magdon-Ismail 2012; Chen and Huo 2006). Some of these algorithms are generated from the greedy algorithms for N=1. For example the Simultaneous Orthogonal Matching Pursuit (SOMP) Algorithm (Tropp, Gilbert, and Strauss 2006) is generated from the Orthogonal Matching Pursuit (OMP) (Tropp 2004; Soussen et al. 2013) to handle a target matrix of N columns. Similarly, the Simultaneous Orthogonal Least Squares (SOLS) Algorithm is a direct generation from the Orthogonal Least Squares (OLS) Algorithm (Soussen et al. 2013; Blumensath and Davies 2007). See, e.g., (Chen and Huo 2006; Çivril and Magdon-Ismail 2012) for theoretical analysis of the SOLS algorithm. An algorithm established in (Çivril and Magdon-Ismail 2012) improves the SOLS algorithm in term of runtime at the cost of increased memory. In (Maung and Schweitzer 2015), the authors improved the speed of the SOLS algorithm by a recursive formulation.

We propose a heuristic search approach for solving the general problem of selecting a subset of k columns from the dictionary matrix to simultaneously approximate the entire target matrix. Our heuristic functions are based on eigenvalues of related matrices. With proper selection of heuristics our algorithm can be tuned to gives the optimal solution as well as approximate solutions. The optimal variant runs slow and works only for small k or small dictionary matrices. The suboptimal variants run much faster and produces solutions with guarantees on how close they are to the optimum.

1.1 The main contributions

- The first nontrivial algorithm guaranteed to produce an optimal solution for the general case where Y is a matrix.
- Sub-optimal algorithms that produce more accurate results than the greedy algorithms with the added benefit of guarantees on how far the solution is from an optimal solution.
- A greedy algorithm that produces similar results to known greedy algorithms and provides additional guarantees on how far the solution is from an optimal solution.

```
Input:
$Y$, a target matrix of $N$ columns.
$X$, a dictionary matrix of $n$ columns.
$k$, the desired number of selected columns from $X$.
$f(\cdot)$, a heuristic function.
Output: a subset $S$ of $k$ columns.
Data Structures: Two global lists of subsets: the
                 fringe list $F$, and the closed list $C$.
Initialization: Put the empty subset into $F$.
1  while $F$ is nonempty do
2      Pick node $n_i$ (associated with subset $S_i$ of size
         $k_i$) with the smallest heuristic $f_i$ from $F$. Ties
         are resolved in favor of the larger $k_i$.
3      if $S_i$ contains $k$ columns then
4          Stop and return $S_i$ as the solution subset.
5      else
6          for each child $n_j$ of $n_i$ do
7              if $n_j$ is not in $C$ then
8                  Put $n_j$ in $C$.
9                  Compute $f_j$.
10                 Put $\{n_j, f_j\}$ in $F$.
11             end
12         end
13     end
14 end
```

Figure 2: The search algorithm.

2 The algorithm

In this section we describe the algorithm without specifying the heuristic functions. It is a standard heuristic search procedure, as shown in Figure 2. The same search procedure (with different heuristics) was also used in (Arai, Maung, and Schweitzer 2015; Arai et al. 2016; He et al. 2019) for their solutions to the unsupervised column subset selection problem.

There are two differences between the algorithm in Figure 2 and the typical best first search procedure. The first is in Line 8, where the node is immediately inserted into the closed list, and the second is in Line 10, where the classical algorithm first checks if the node is already in the fringe, and if so decides if its value needs to be updated. The justification for these differences is that in our case all paths leading from the root to a node (a subset) are equivalent. Once a subset node is added into the fringe other nodes with the same subset (obtained through a different path) will have the exact same heuristic values and therefore can be ignored. This implies that the fringe is a subset of the closed nodes list.

This observation shows that the closed nodes list can be implemented by a hash table, and the fringe by a heap. Furthermore, unlike the standard implementation of the heap there is no need to implement deletions that are not through the heap "pop" operation.

3 The heuristic functions

In this section we specify the heuristic functions that are needed for the algorithm.

Recall that the error of approximating Y by a linear combination of k columns in S is given by (2). Let n_i be an arbitrary subset node. Let $k_i{<}k$ be the number of columns selected at n_i, and let S_i be the matrix formed by these columns. Let $\overline{k}_i{=}k{-}k_i$ be the number of columns that still need to be selected. We consider the error that can be obtained by completing S_i to size k. Let $\overline{S}_i$ of size $\overline{k}_i$ be such completion. Its error is given by:

$$e(\overline{S}_i) = E(S_i \cup \overline{S}_i) = \min_{A_i,\overline{A}_i} \|Y - S_iA_i - \overline{S}_i\overline{A}_i\|_F^2 \qquad (3)$$

In the equation above E is the error defined in (2), and $A_i, \overline{A}_i$ are arbitrary coefficient matrices. Thus, the smallest error of a solution subset that contains S_i would be:

$$d_i = \min_{\overline{S}_i} e(\overline{S}_i) \qquad (4)$$

This shows that the best choice for the heuristic value f_i in the algorithm of Figure 2 is $f_i{=}d_i$. Unfortunately, computing d_i is too expensive since it requires going over all the subsets $\overline{S}_i$ of size $\overline{k}_i$. However, we show that there is an effective computational approach to approximate d_i. We define the two approximations l_i and u_i and show that they bound d_i from below and above.

$$\begin{aligned} d_i &= \min_{A_i,\overline{A}_i,\overline{S}_i} \|Y - S_iA_i - \overline{S}_i\overline{A}_i\|_F^2 \\ u_i &= \min_{A_i} \|Y - S_iA_i\|_F^2 = \|Y - S_iA_i^*\|_F^2 = \|Y_i\|_F^2 \\ &\quad \text{where:} \quad Y_i = Y{-}S_iA_i^* \\ l_i &= \|Y_i - \overline{Y}_i\|_F^2 \\ &\quad \text{where } \overline{Y}_i \text{ is the best rank } \overline{k}_i \text{ approximation to } Y_i. \end{aligned} \qquad (5)$$

Lemma 1 *For any selection S_i of size $k_i \le k$:*

$$l_i \le d_i \le u_i \qquad (6)$$

with equality if $k_i = k$.

Proof:

- *To prove the right hand side inequality observe that for any i the value of u_i is the same as the expression for d_i if the matrix $\overline{S}_i$ is taken to be identically 0.*
- *To prove the left hand side inequality:*

$$\begin{aligned} l_i &= \|Y_i - \overline{Y}_i\|_F^2 \\ &\le \min_{\overline{S}_i,\overline{A}_i} \|Y_i - \overline{S}_i\overline{A}_i\|_F^2 \qquad \text{(note1)} \\ &= \min_{\overline{S}_i,A_i,\overline{A}_i} \|Y - S_iA_i - \overline{S}_i\overline{A}_i\|_F^2 = d_i \end{aligned}$$

  The justification for the inequality in (note1) is that the rank of $\overline{S}_i\overline{A}_i$ is at most $\overline{k}_i$.
- *It remains to show that the inequalities become equalities when $k_i{=}k$. Under this condition $\overline{k}_i{=}0$, which implies that $\overline{Y}_i = 0$. Therefore, both l_i and u_i equal $\|Y_i\|_F^2$.* ■

3.1 Three variants of the search algorithm

Clearly, the best choice for the heuristic function is $f_i = d_i$. However since it cannot be efficiently calculated we consider linear combination between l_i and u_i. Three options of heuristic function are considered. The resulting variants are stated as in the following theorems.

Theorem 1 ***(the optimal variant)*** *If $f_i = l_i$ then the algorithm is guaranteed to terminate with an optimal solution.*

Theorem 2 ***(the greedy variant)*** *If $f_i = u_i$ then the algorithm is greedy and terminates after expanding k nodes.*

Theorem 3 ***(the weighted variant)*** *If $f_i = l_i{+}\gamma u_i$, where $\gamma{\ge}0$, and the algorithm terminates with the subset S^{**} that has error of e^{**} then*

$$e^{**} \le e^* + \gamma\|Y\|_F^2$$

where e^* is the smallest possible error.

The proofs for the theorems are given in Section 5, where a sharper bound is given for Theorem 3.

4 Efficient heuristics calculation

The following theorem summarizes the computational formulas for heuristic values.

Theorem 4 *At the node n_i where the k_i columns in S_i have already been selected from X, additional $\overline{k}_i{=}k{-}k_i$ columns still need to be selected. Define: $B_i = Y_iY_i^T$, where Y_i is defined in (5). Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of B_i. Then the values of l_i, u_i defined in (5) can be calculated by:*

$$\begin{aligned} u_i &= \sum_{j=1}^{m} \lambda_j = \mathit{Trace}(B_i) \\ l_i &= \sum_{j=\overline{k}_i+1}^{m} \lambda_j = \mathit{Trace}(B_i) - \sum_{j=1}^{\overline{k}_i} \lambda_j \end{aligned} \qquad (7)$$

The proof for Theorem 4 is given in Section 5. From the formulas in Theorem 4 it is clear that only the top $\overline{k}_i$ eigenvalues of B_i need to be calculated in addition to its trace. But they have to be calculated for each node, which is impractical. We proceed to show that there is a matrix related to B_i with a special structure that enables efficient computation of these eigenvalues.

Observe that we only need heuristics for the children of the picked node (parent) at line 9 in the algorithm. At that point the parent node n_i is known. We show how to efficiently calculate the heuristics for the children by first performing an expensive eigendecomposition of the parent. However the expensive eigendecomposition of the parent can be reduced by computing eigendecomposition in the initial step. In summary, we discuss three different types of eigendecompositions. The first one is the one in the initial step, which only needs to be performed once. The second is for each parent, and the third is for the children, where only the eigenvalues are needed.

4.1 Initial Eigendecomposition

In the initial step we compute the eigenvalues and the eigenvectors of the matrices $B_y = YY^T$ and $B_x = XX^T$. The eigenvalue decomposition gives:

$$B_y = U_y D_y U_y^T, \quad B_x = U_x D_x U_x^T,$$

where U_y, U_x are the eigenvectors and D_y, D_x are diagonal matrices with the eigenvalues as the diagonal elements.

Let r_y be the rank of Y then $r_y \leq \min(m, N)$. Let r_x be the rank of X then $r_x \leq \min(m, n)$. As we show later we can ignore zero eigenvalues and their corresponding eigenvectors and replace X, Y with the following information:

$$\begin{aligned} &W_x = U_x^T X \quad \text{where } W_x \text{ is } r_x \times n. \\ &D_y \quad \text{reduced to size } r_y \times r_y. \\ &P = D_y^{\frac{1}{2}} U_y^T U_x \quad \text{of size } r_y \times r_x. \end{aligned} \tag{8}$$

With the advancement of randomized algorithms for matrix decompositions, this initial step can be performed efficiently. For example, using the algorithm described in (Halko, Martinsson, and Tropp 2011), the complexity of this initial step is $O(mNr_y + mnr_x)$.

4.2 Eigendecomposition for parent nodes

Instead of working with the matrix B_i as defined in Theorem 4 we use a related matrix H_i which has same eigenvalues as B_i. The special structure of H_i makes the calculations of these eigenvalues more efficient.

Lemma 2 *Let Q_i be an orthonormal basis of the current selection S_i (selected from X) of size k_i. Let $\tilde{S}_i$ be the corresponding selection from $W_x = U_x^T X$, and $\tilde{Q}_i$ be the orthonormal basis of $\tilde{S}_i$, then:*

$$Q_i = U_x \tilde{Q}_i.$$

Proof: *Straightforward.*

Lemma 3 *Let B_i be the matrix whose eigenvalues are used in (7) to calculate the heuristics. Let $\tilde{Q}_i$ be an orthonormal basis of $\tilde{S}_i$ of size k_i. Given D_y, P from (8) define the following $r_y \times r_y$ matrix:*

$$\begin{aligned} &H_i = D_y - Z_i Z_i^T = D_y - \sum_{j=1}^{k_i} z_j z_j^T \\ &\text{where } Z_i = P\tilde{Q}_i,\ z_i = P\tilde{q}_i. \end{aligned} \tag{9}$$

Then H_i and B_i have the same eigenvalues (and trace).

Proof: B_i can be written as follows:

$$\begin{aligned} B_i &= Y_i Y_i^T \\ &= (I - Q_i Q_i^T) YY^T (I - Q_i Q_i^T)^T \\ &= (I - Q_i Q_i^T) U_y D_y U_y^T (I - Q_i Q_i^T)^T \\ &= G_i D_y^{\frac{1}{2}} (G_i D_y^{\frac{1}{2}})^T \end{aligned}$$

where $G_i = (I - Q_i Q_i^T) U_y$. Since the order of matrix product does not affect the eigenvalues the following matrix that we call H_i has the same eigenvalues as B_i:

$$\begin{aligned} H_i &= (G_i D_y^{\frac{1}{2}})^T G_i D_y^{\frac{1}{2}} \\ &= D_y^{\frac{1}{2}} G_i^T G_i D_y^{\frac{1}{2}} \\ &= D_y^{\frac{1}{2}} U_y^T (I - Q_i Q_i^T)^T (I - Q_i Q_i^T) U_y D_y^{\frac{1}{2}} \\ &= D_y^{\frac{1}{2}} U_y^T (I - Q_i Q_i^T) U_y D_y^{\frac{1}{2}} \\ &= D_y - D_y^{\frac{1}{2}} U_y^T Q_i Q_i^T U_y D_y^{\frac{1}{2}} \\ &= D_y - D_y^{\frac{1}{2}} U_y^T U_x \tilde{Q}_i (D_y^{\frac{1}{2}} U_y^T U_x \tilde{Q}_i)^T \\ &= D_y - P\tilde{Q}_i (P\tilde{Q}_i)^T = D_y - Z_i Z_i^T \end{aligned}$$

where $P = D_y^{\frac{1}{2}} U_y^T U_x$. This completes the proof. ■

Equation (9) shows that the H_i can be computed from k_i times rank-one updates of the diagonal eigenvalue matrix, which enables specialized routines to compute its eigenpairs. See, e.g., (Bunch, Nielsen, and Sorensen 1978).

4.3 Eigenvalues for children nodes

To compute the heuristics at line 9 in the algorithm, eigenvalues for children nodes along with their traces are needed. It is the most expensive part of the algorithm. We show how to compute those values efficiently.

Let n_p be the node picked at Line 2 of the algorithm and let $\tilde{S}_p$ of size k_p be the corresponding selection. Let $\tilde{Q}_p$ be the orthonormal basis of $\tilde{S}_p$. Let H_p be the matrix computed according to (9). Let t_p be the trace of H_p. Suppose a child node n_c is created by adding a column w from W_x to $\tilde{S}_p$. From (9) the associated matrix H_c for the child can be computed by:

$$H_c = H_p - z_c z_c^T \tag{10}$$

where $z_c = P\tilde{q}_c$, $\tilde{q}_c = \bar{w}_c / \|\bar{w}_c\|$, and $\bar{w}_c = (I - \tilde{Q}_p \tilde{Q}_p^T) w$. The key observation here is that H_c is a rank-one modification of H_p. In this case the eigenvalues of the updated matrix can be computed efficiently in $O(kr_y)$ from the eigendecomposition of H_p. See, e.g., (Bunch, Nielsen, and Sorensen 1978). Since computing z_c takes $O(kr_x)$ this adds up to $O(k(r_x + r_y)n)$ for computing the heuristic values for all children of a parent node.

5 Correctness

In this section we give the technical proofs of theorems. The tool we use here is the interlacing property of eigenvalues. We use the following special case described in Chapter 8 of (Golub and Van-Loan 2013):

Theorem 5 *Suppose $\tilde{Z} = Z - cc^T$ where $Z \in \mathbf{R}^{d \times d}$ is symmetric, and $c \in \mathbf{R}^d$. Let $\lambda_i(\cdot)$ be the i^{th} eigenvalue of a matrix, then:*

$$\lambda_{i+1}(Z) \leq \lambda_i(\tilde{Z}) \leq \lambda_i(Z).$$

As an immediate corollary we get:

$$\sum_{i=\tau+1}^{d} \lambda_i(Z) \leq \sum_{i=\tau}^{d} \lambda_i(\tilde{Z}) \text{ where } 1 \leq \tau \leq d. \tag{11}$$

5.1 Proofs of the three variants

Our proofs have similar structure to the proofs in (He et al. 2019). When both matrices X and Y are same their result is a special case of our result. However, the tools used in (He et al. 2019) may not be poweful enough to prove our result, which relies on the interlacing property. We need following lemmas.

Lemma 4 *If n_j is a child of n_i then $l_i \leq l_j$.*

Proof: Let S_i be the subset associated with node n_i of size k_i, and set $\overline{k}_i = k - k_i$. We need to show that if $S_j \supset S_i$ (so that S_j is a child of S_i) then l_j is greater than l_i. Set $Z = Y_i Y_i^T$, $\tau = \overline{k}_i$. Then the lemma follows from the inequality in Equation (11). ■

Lemma 5 *If n_j is a child of n_i then $u_j \leq u_i$.*

Proof: The child node n_j is created by adding a column into the selection S_i at node n_i. Clearly, the additional column can only reduce the error. ■

Lemma 6 *Consider the choice $f_i = u_i$. Let n_i be the node picked at Line 2 of the algorithm. The following two properties hold:*

a. *For each child n_j of n_i the selection size $|S_j|$ is larger than the selection size of all nodes currently in the fringe.*
b. *The next node to be picked will be a child of n_i.*

Proof: The proof is by induction. Property **a** follows trivially from Property **b**. To prove Property **b** observe that from Lemma 5, u_i is monotonically decreasing 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. Sine ties are are also resolved in favor of a child, then the child of n_i will be selected next. ■

Instead of proving Theorem 3 directly we prove a stronger version with a tighter bound:

Theorem 6 ***(the tighter weighted variant)*** *If $f_i = l_i + \gamma u_i$, where $\gamma \geq 0$, and the algorithm terminates at the node n_{**} with the subset S^{**} that has error of e^{**} then*

$$e^{**} \leq e^* + \gamma(u_{\max} - l_{**})$$

where e^* is the smallest possible error, l_{**} is the value of the lower bound for n_{**}, and $u_{\max}$ be the largest value of the upper bound for the nodes remaining in the fringe after the goal node is reached.

Lemma 7 *Suppose Theorem 6 is false. Then for any node n_z on the path from the root to n_* the following condition holds: $f_z < f_{**}$.*

Proof: The falsehood of Theorem 6 can be written as follows: $e^{**} > e^* + \gamma(u_{\max} - e^{**})$. Since both n_* and n_{**} are goal nodes their corresponding subsets S^* and S^{**} are both of size k. Lemma 1 implies: $e^{**} = l_{**} = u_{**}$ and $e^* = l_* = u_*$. Using this and some algebra it can be shown that an equivalent falsehood condition is: $l_{**} > l_* + \frac{\gamma}{1+\gamma}(u_{\max} - l_*)$.

The lemma can now be proved as follows:

$$\begin{aligned} f_{**} &= l_{**} + \gamma u_{**} = (1+\gamma) l_{**} && (c_1) \\ &> (1+\gamma) l_* + \gamma(u_{\max} - l_*) && (c_2) \\ &= l_* + \gamma u_{\max} \geq l_z + \gamma u_{\max} && (c_3) \\ &\geq l_z + \gamma u_z = f_z \end{aligned}$$

c_1 : from the definition of f. c_2 : from the equivalent falsehood assumption. c_3 : from Lemma 1 $l_* > l_z$. ■

Proof of Theorem 2: From Lemma 6, when a node n_i is picked at Line 2 of the algorithm, one of its children will be examined next. This shows that the algorithm is greedy, and terminates after examining k nodes. ■

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

Proof of Theorem 6: If the theorem is false then from Lemma 7 it follows that all nodes on the path from the root to S^* have smaller f values than f_{**}. Since at any given time at least one of them is in the fringe, they should all be selected before S^{**} is selected. But this means that S^* is selected as the solution and not S^{**}. ■

Proof of Theorem 3: The proof follows from Theorem 6 by observing that $u_{\max} - l_{**} \leq u_{\max} \leq u_{\text{root}} = \|Y\|_F^2$.

Proof of Theorem 4: Recall the relation between the trace and the Frobenius norm. For any matrix M, $\|M\|_F^2 = \text{trace}\{MM^T\}$. For the upper bound: $u_i = \|Y_i\|_F^2$, so that $u_i = \text{trace}\{Y_i Y_i^T\} = \text{trace}\{B_i\}$.
To prove the formula for the lower bound we observe that the best rank $\overline{k}_i$ approximation to Y_i can be written as: $\overline{Y}_i = UU^T Y_i$. See, e.g., (Golub and Van-Loan 2013). Here U is formed by the first $\overline{k}_i$ eigenvectors of $Y_i Y_i^T$. Therefore:

$$\begin{aligned} l_i &= \|Y_i - \overline{Y}_i\|_F^2 = \|(I - UU^T)Y_i\|_F^2 \\ &= \text{trace}\{(I - UU^T) Y_i Y_i^T (I - UU^T)\} \\ &= \text{trace}\{(I - UU^T) B_i (I - UU^T)\} \\ &= \text{trace}\{B_i - UU^T B_i - B_i UU^T + UU^T B_i UU^T\} \\ &= \text{trace}\{B_i\} - \text{trace}\{UU^T B_i\} - \text{trace}\{B_i UU^T\} \\ &\quad + \text{trace}\{UU^T B_i UU^T\} \\ &= \text{trace}\{B_i\} - \text{trace}\{U^T B_i U\} && \text{(note1)} \\ &= \text{trace}\{B_i\} - \sum_{j=1}^{\overline{k}_i} \lambda_j(B_i) \end{aligned}$$

The simplification in the line marked with (note1) was obtained using the cyclic invariant property of the trace which implies that $\text{trace}\{UU^T B_i\}$, $\text{trace}\{B_i UU^T\}$, and $\text{trace}\{UU^T B_i UU^T\}$, are all equal to = $\text{trace}\{U^T B_i U\}$. ■

6 Bounds on Sub-optimality

Both the greedy variant and the weighted variant are not guaranteed to produce the optimal solution. We proceed to show how to obtain bound on how close their solution is to

k	f=l	f=l+u	f=l+5u	f=u	Forward	Backward	Leaps	OMP	FoBa	POSS	ISOLS
eunite2001 X:367×16 Y:267×1											
5	**1.154e6**	**1.154e6**	**1.154e6**	1.169e6	1.169e6	1.161e6	**1.154e6**	1.509e6	1.329e6	1.226e6	1.169e6
10	**0.920e6**	**0.920e6**	0.931e6	1.035e6	1.035e6	**0.920e6**	**0.920e6**	1.043e6	1.043e6	1.013e6	1.035e6
spectf X:267×44 Y:267×1											
5	**38.64**	**38.64**	**38.64**	39.62	39.62	39.20	**38.64**	40.52	40.52	38.87	39.62
7	**37.73**	**37.73**	38.10	38.36	38.36	38.41	**37.73**	38.74	37.83	**37.73**	38.36
libras X:360×90 Y:360×1											
3	**5192.12**	**5192.12**	**5192.12**	**5192.12**	**5192.12**	5333.00	**5192.12**	5334.94	5334.94	5194.66	**5192.12**
5	**4723.07**	**4723.07**	4778.88	4796.08	4796.08	5086.07	**4723.07**	4953.25	4953.25	4777.82	4796.08
duke breast cancer X:$44 \times 7,129$ Y:44×1											
2	**14.67**	**14.67**	14.94	14.94	error	error	error	14.94	14.94	16.07	14.94
5	-	**4.92**	5.14	5.14	error	error	error	5.36	5.36	5.14	5.14

Table 1: Accuracy comparison for the case $N = 1$. The minimum errors are highlighted. No results (-) is shown if the runtime is longer than 30 minutes. The "error" means that there is an error thrown out during the run of the algorithm. Our optimal variant ($f = l$) and **Leaps** are guaranteed to produce a best selection. Our weighted variant ($f = l + \gamma u$) are more accurate than other non-optimal algorithms.

the optimal. The technique we use was originally proposed by (Thayer and Ruml 2008).

Consider a run of a non-optimal algorithm producing the non-optimal selection S^{**}. Then size$(S^{**}) = k$, and from Lemma 1 it follows that $l_{**} = u_{**} = E(S^{**})$. The value of l_{**} is related to the optimal value $l_* = u_* = E(S^*)$ by: $l_* \le l_{**}$. Let b be a value satisfying: $l_{**} \le l_* + b$, or, equivalently $b \ge l_{**} - l_*$. We refer to b as a bound, where a smaller b indicates a better bound, and in particular $b = 0$ implies an optimal solution. An important observation is that in combinatorial search one can always compute such values. Let F be the fringe list after the algorithm terminates. Going over all the remaining nodes in the fringe list we can compute: $l_{\min} = \min_{n_i \in \text{Fringe}} l_i$. From Lemma 4 it follows that $l_* \ge l_{min}$, so that we can take:

$$b = l_{**} - l_{\min}, \quad \text{where: } l_{\min} = \min_{n_i \in \text{Fringe}} l_i. \qquad (12)$$

7 Experimental results

In this section we describe experiments on various datasets that are publicly available.

For the case where N=1, we compare the proposed algorithms with the following methods: **Leaps** (Furnival and Wilson 1974); **Forward** (Hastie, Tibshirani, and Friedman 2009); **Backward** (Hastie, Tibshirani, and Friedman 2009); **OMP** (Mallat 1999); **FoBa** (Zhang 2009); **POSS** (Qian, Yu, and Zhou 2015).

For the general case (N≥1), we compare our algorithms the following algorithms: **SOMP** (Tropp, Gilbert, and Strauss 2006); **SOLS** (Chen and Huo 2006); **CM** (Çivril and Magdon-Ismail 2012); **ISOLS** (Maung and Schweitzer 2015), the exact version is used. The results for **SOLS**, **CM** and **ISOLS** are same. The results for **ISOLS** are shown.

The implementations used for **Leaps**, **Forward**, and **Backward** are the functions in the R library: leaps. Other implementations are made available by the authors. Experiments are conducted on iMac (macOS Catalina) with Processor 4GHz Quad-Core Intel i7 and Memory 32 GB.

7.1 Comparison

As discussed in Section 3, with different selection of the heuristic function our algorithm produces optimal, suboptimal and greedy results. We compare those results with the current state-of-the-art algorithms for the $N = 1$ case and the general case $N \ge 1$.

Comparison for the $N = 1$ case: The results for various datasets are shown in Table 1. As expected, our optimal variant ($f = l$) gives results same as the results produced by **Leaps**. But our optimal variant also works when $N > 1$. The errors for the greedy variant ($f = u$) are same as **Forward** and **ISOLS**. The errors for the weighted variant ($f = l + \gamma u$) are between the optimal solution and the greedy solution. They are more accurate than other non-optimal algorithms.

Comparison for the general case: The results for various datasets are shown in Table 2. The first three datasets are split evenly and experimented without intercept. The remaining multi-target/label datasets are tested with intercept. Our optimal variant ($f = l$) is guaranteed to produce optimal solutions. The errors for the greedy variant ($f = u$) are same as the results of **ISOLS**. The errors for $f = l + \gamma u$ are between the optimal solution and the greedy solution.

7.2 The effect of the parameter γ

We investigate the influence of the parameter γ on the accuracy and runtime. The results are shown in Figure 3. The red curve corresponds to the change of errors as the increase of the parameter γ. The blue curve is for the change of the runtime.Observe that as the increase of γ, the error goes up from an optimal solution to a greedy solution, while the runtime decreases.

8 Concluding remarks

In this paper we study a common setting where several columns of one matrix, the dictionary, are selected to simultaneously approximate all the columns of another matrix. While there are many algorithms that were developed for the case where the target matrix has a single column, we

k	$f{=}l$	$f{=}l{+}u$		$f{=}l{+}2u$		$f{=}l{+}10u$		$f{=}u$		SOMP	ISOLS
	error	error	bound	error	bound	error	bound	error	bound		
libras X:360 × 45 Y:360 × 46											
3	**6,010**	**6,010**	**0**	**6,010**	0.947	**6,010**	0.949	6,169	0.95	6,047	6,169
5	**5,587**	**5,587**	**0.941**	5,594	0.987	5,623	0.987	5,686	0.987	5,632	5,686
spectf X:267 × 22 Y:267 × 23											
5	**423,909**	428,524	**0.635**	433,697	0.639	433,697	0.639	433,697	0.639	431,650	433,697
10	**374,453**	377,282	0.842	377,282	0.842	377,282	0.842	377,282	0.842	384,156	377,282
duke breast cancer X:44 × 3565 Y:44 × 3565											
5	-	**62,040**	**0.071**	**62,040**	**0.071**	62,191	0.074	62,191	0.074	62,976	62,191
25	-	**18,040**	**0.159**	18,700	0.189	18,700	0.189	18,700	0.189	213,17	18,700
scm20d X:8966 × 61 Y:8966 × 16											
5	**5.727e9**	**5.727e9**	**0.727**	5.786e9	0.748	6.007e9	0.758	6.007e9	0.758	6.159e9	6.007e9
7	-	**5.205e9**	**0.882**	5.239e9	0.883	5.367e9	0.886	5.367e9	0.886	5.24e9	5.367e9
mediamill X:43907 × 120 Y:43907 × 101											
5	**116,292**	**116,292**	**0**	**116,292**	0.395	116,899	0.398	117,894	0.403	118,744	117,894
10	-	-	-	-	-	**112,975**	0.579	113,931	0.583	114,403	113,931
oes97 X:334 × 263 Y:334 × 16											
5	-	**2.120e9**	**0.597**	2.126e9	0.598	2.126e9	0.598	2.126e9	0.598	2.498e9	2.126e9
7	-	**1.704e9**	**0.756**	1.708e9	0.757	1.708e9	0.757	1.708e9	0.757	2.002e9	1.708e9

Table 2: Accuracy comparison for the case $N > 1$. Our optimal variant ($f = l$) is guaranteed to produce a best selection. The suboptimal results come with bounds. The bounds are normalized by the solution errors.

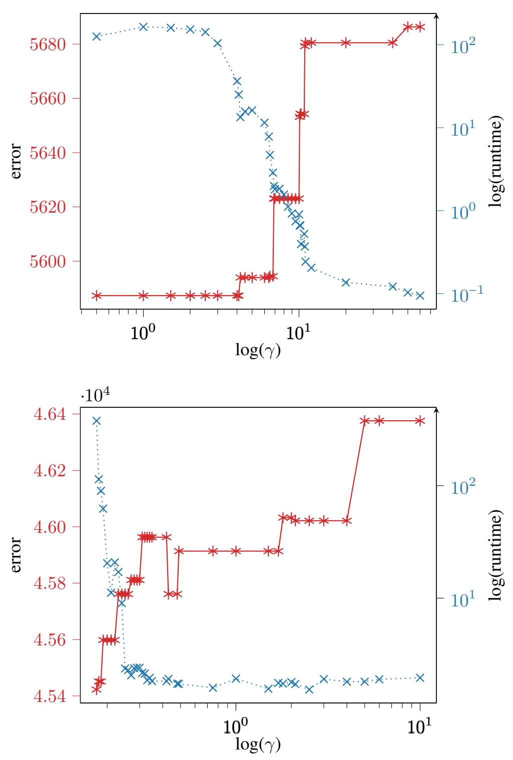

Figure 3: The effect of γ on accuracy and runtime for the general case. The upper panel: libras dataset $N = 46, k = 5$. The lower panel: duke breast cancer $N = 3565, k = 10$.

are only aware of greedy algorithms for the general setting. We observe that the general setting does not reduce to the single column case, since the challenge is to select columns that simultaneously approximate all the target columns.

The algorithm that we develop are similar to the weighted A^* algorithm. We show that with 0 assigned to a weight parameter the algorithm is guaranteed to be optimal, but its running time is very slow. Other nonzero choices give practical algorithms that beat the accuracy of the current state of the art greedy algorithms.

In addition to producing a solution our algorithm produces bounds on how far the solution is from the optimal. The quality of these bounds are data dependent. In some cases it provides no useful information, but in other cases it shows that computed result to be very close to the optimal. This is the case even for the greedy variant of our algorithm. It produces the same result as other known algorithms but gives the additional information of a bound.

While there is a significant similarity between our algorithm and the classical theory of A^* we are not aware of direct applications of A^* to the problem discussed here. In particular, our upper bound does not seem to have a parallel in the general theory.

References

Amaldi, E.; and Kann, V. 1998. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science* 209(1–2): 237–260.

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

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

Blumensath, T.; and Davies, M. E. 2007. On the Difference Between Orthonormal Matching Pursuit and Orthogonal Least Squares. Technical report, University of Edinburgh.

Bunch, J. R.; Nielsen, C. P.; and Sorensen, D. C. 1978. Rank-One Modification of the Symmetric Eigenproblem. *Numer. Math.* 31: 31–48.

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

Chen, J.; and Huo, X. 2006. Theoretical Results of Sparse Representations of Multiple Measurement Vectors. *IEEE Transactions on Signal processing* 54(12): 4634–4643.

Davis, G.; Mallat, S.; and Avellaneda, M. 1997. Adaptive greedy approximations. *Constructive approximation* 13(1): 57–98.

Furnival, G. M.; and Wilson, R. W. 1974. Regressions by Leaps and Bounds. *Technometrics* 16(4): 499–511.

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

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.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Statistical Learning*, section 3.3.1. Springer, second edition.

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

Mallat, S. 1999. *A wavelet Tour of Signal Processing*. Academic Press.

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

Maung, C.; and Schweitzer, H. 2015. Improved Greedy Algorithms for Sparse Approximation of a Matrix in terms of Another Matrix. *IEEE Transactions on Knowledge and Data Engineering* 27(3): 769–780.

Natarajan, B. K. 1995. Sparse approximate solutions to linear systems. *SIAM Journal of Computing* 25(2): 227–234.

Neff, R.; and Zakhor, A. 1997. Very low bit-rate video coding based on matching pursuits. *IEEE Transactions on Circuits and Systems for Video Technology* 7(1): 158–171.

Pati, Y. C.; Rezaiifar, R.; and Krishnaprasad, P. S. 1993. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, volume 1, 40–44. ISSN 1058-6393.

Qian, C.; Yu, Y.; and Zhou, Z. 2015. Subset Selection by Pareto Optimization. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 1774–1782. Curran Associates, Inc.

Soussen, C.; Gribonval, R.; Idier, J.; and Herzet, C. 2013. Joint k-Step Analysis of Orthogonal Matching Pursuit and Orthogonal Least Squares. *IEEE Transactions on Information Theory* 59(5): 3158–3174.

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. California: AAAI Press.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.* 58(1): 267–288.

Tropp, J. A. 2004. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory* 50(10): 2231–2242.

Tropp, J. A.; Gilbert, A. C.; and Strauss, M. J. 2006. Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. *Signal Processing* 86(3): 572–588.

Wei, H.; and Billings, S. A. 2007. Feature Subset Selection and Ranking for Data Dimensionality Reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(1): 162–166.

Zhang, T. 2009. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In *Advances in Neural Information Processing Systems*, 1921–1928.