A Fast Algorithm for Simultaneous Sparse Approximation

Guihong Wan(✉) and Haim Schweitzer

Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA
{Guihong.Wan,HSchweitzer}@utdallas.edu

Abstract. Simultaneous sparse approximation problems arise in several domains, such as signal processing and machine learning. Given a dictionary matrix X of size $m \times n$ and a target matrix Y of size $m \times N$, we consider the classical problem of selecting k columns from X that can be used to linearly approximate the entire matrix Y. The previous fastest nontrivial algorithms for this problem have a running time of $O(mnN)$. We describe a significantly faster algorithm with a running time of $O(km(n+N))$ with accuracy that compares favorably with the slower algorithms. We also derive bounds on the accuracy of the selections computed by our algorithm. These bounds show that our results are typically within a few percentage points of the optimal solution.

Keywords: Simultaneous sparse approximation · Multiple measurement vectors · Multi-target regression · Spectral pursuit

1 Introduction

Sparse approximation, or alternatively, sparse representation, has attracted significant attention in fields of signal processing, image processing, and machine learning (e.g., [16,27,33]). It originally arose in the study of linear regression in which a target vector is approximated by a linear combination of several selected features to foster interpretability and avoid overfitting.

The problem that we discuss in this paper is an extension of the classical sparse approximation problem to multiple targets. The goal is to identify a small number of "atoms" (columns of the dictionary matrix) that can be used to linearly approximate all the columns of another "target" matrix. This formulation is referred to as simultaneous sparse approximation (e.g., [17,25]). It has various applications, such as supervised feature selection for multi-target regression [34], human action recognition in videos [11], multi-sensor image fusion [31], and hyperspectral image unmixing [28].

We present an efficient selection algorithm with a running time that depends linearly on the number of columns of the relevant matrices. This significantly improves on previous algorithms which have a running time depending on the product of these parameters.

K. Karlapalem et al. (Eds.): PAKDD 2021, LNAI 12714, pp. 42–54, 2021.
https://doi.org/10.1007/978-3-030-75768-7_4

The main idea is to select columns from the dictionary matrix whose span is close to the dominant spectral components of the target matrix. The following two-stage procedure is proposed. In the first stage, a greedy algorithm is used to select k columns from the dictionary matrix. In each iteration, a column that is most related to the first left eigenvector of the residual target matrix is selected. In the second stage, an iterative "bidirectional selection" algorithm is used to improve the results produced in the first stage.

1.1 Problem Formulation

The problem of simultaneous sparse approximation can be stated as follows. 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 (n is typically very large). Consider an approximation of Y in terms of X:

$$Y \approx X\tilde{A}, \tag{1}$$

where $\tilde{A}$ is the coefficient matrix of size $n \times N$. The approximation in Eq. (1) is considered a sparse approximation if only a small number of rows of $\tilde{A}$ are nonzero. See, e.g., [4,19,24,25]. Let $k \leq n$ be the number of nonzero rows in $\tilde{A}$, then the approximation in (1) can be written as:

$$Y \approx SA, \tag{2}$$

where $S = (x_{s_1} \dots x_{s_k})$ is the $m \times k$ selection matrix consisting of the k columns of X corresponding to the k nonzero rows of $\tilde{A}$, and A is the coefficient matrix of size $k \times N$, created from the k nonzero rows of $\tilde{A}$. Thus, the goal of the problem described in (2) is to select k columns from X such that all the columns of Y can be simultaneously approximated by a linear combination of the selected columns in S. The quality of the selected columns in S is measured by the following (Frobenius norm) error:

$$E(S) = \min_{A} \|Y - SA\|_F^2, \quad \text{where } S \text{ is a subset of } X \text{ columns.} \tag{3}$$

The following cases are special sub-problems:

- When $N = 1$, the problem is called sparse approximation, where Y is a single vector and X is a dictionary. It is also called supervised subset selection when Y corresponds to labels for the data matrix X. See, e.g., [9,20].
- When $X = Y$, the problem is known as "unsupervised feature selection" if each column corresponds to a data feature. It is known as "representative selection" if each column corresponds to a data point. See, e.g., [1,13,30].

Optimal solutions for the optimization in (3), as well as approximate solutions within a constant are known to be NP-hard even when $N = 1$ [8,20].

1.2 Related Work

Extensive studies were directed at the $N = 1$ case (e.g., [9,18,21,24,32]). Applications include signal processing e.g., [18,24] and supervised feature selection in linear regression e.g., [9,21,32]. An optimal algorithm proposed in [9] finds the best solution, but it is not feasible for large k and large datasets. Previously proposed approximate solutions can be roughly divided into three 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 [32], these two techniques have limitations due to their greedy behavior. The author of [32] introduced an adaptive algorithm that combines these two ideas to alleviate the flaws. The convex relaxation approaches replace some natural constraints (sometimes defined in terms of the l_0 norm) with convex constraints. An example is the l_1 norm used in the Lasso technique [23]. In [21], the authors treated the subset selection as a bi-objective optimization problem. Their algorithm is optimal for data drawn from Exponential Decay distribution but not for the general case.

The case where $X = Y$ is known as the unsupervised feature selection or unsupervised column subset selection. See, e.g., [1,2,4,10,14,30]. Recently, a fast algorithm [30] was proposed, which greedily selects columns that are closest to the first left eigenvector of a residual matrix. Another recent study in [14] proposes to iteratively improve a selection using a bi-directional stepwise refinement. Both these studies address the $X = Y$ case. Our method is motivated by these two recent studies, generalizing them to the supervised multiple-target case. We show that the speed advantage of this approach over other approaches becomes bigger for large N.

In the multi-target case, the target matrix Y has $N > 1$ columns. We observe that one cannot simply apply an $N = 1$ algorithm separately to each column of Y. The challenge is to find columns in X that can simultaneously approximate all the columns of Y. Previously proposed algorithms for this general case are typically greedy. See, e.g., [3–5,19,25]. Some of these algorithms are generated from the $N = 1$ case. For example, the Simultaneous Orthogonal Matching Pursuit (SOMP) [25] is generated from the Orthogonal Matching Pursuit (OMP) [22,24] to handle a target matrix of N columns. Similarly, the Simultaneous Orthogonal Least Squares (SOLS) is a direct extension of the Orthogonal Least Squares (OLS) [22]. See [4,5] for the analysis of the SOLS algorithm. An algorithm established in [4] improves the SOLS algorithm in terms of speed at the cost of increased memory. In [19], the authors improved the speed of the SOLS algorithm by a recursive formulation. The running time and the memory requirements of some of these algorithms are summarized in Table 1.

1.3 Our Approach

Suppose the matrix S in (3) is not constrained to be a submatrix of X. Then it is known that the k columns of the optimal solution matrix S are the left

Table 1. Complexity of various algorithms. T is the number of iterations.

Algorithms	Time complexity	Memory complexity
SOMP [25]	$O(kmnN)$	$O(m(N+k))$
S-SBR [3]	$O(Tkmn N)$	$O(m(N+k))$
SOLS [6]	$O(kmnN)$	$O(m(n+N))$
CM [4]	$O(nN(m+k))$	$O(m(n+N))+nN$
ISOLS [19]	$O(mnN)$	$O(km)+2n$
SPXY (this work)	$O(km(n+N))$	$O(m(n+N))$

eigenvectors of Y corresponding to the k largest singular values (see e.g., [10]). Our algorithmic approach is based on this result.

Motivated by [30] and [14], we propose a two-stage algorithm that we call the Spectral Pursuit for the matrices X and Y (SPXY). It runs significantly faster than previously proposed algorithms, and its accuracy compares very favorably with the current state of the art. We also show how to derive a bound on how far the selection computed by SPXY is from the optimal solution. (Recall that the computation of the optimal solution is NP-hard.)

In the first stage we use a greedy technique to select k columns from the dictionary matrix. The algorithm runs k iterations. In each iteration the column selected is the one most similar to the left eigenvector corresponding to the largest singular value. The two matrices are then projected on the null space of the columns that were already selected. The null space of the selected columns indicates a subspace that the selected columns cannot span. This first stage algorithm is greedy, and gets "stuck" in a local minimum. In the second stage we use a bidirectional stepwise technique to further improve the results.

Similar to many other data analysis techniques, the performance of our SPXY algorithm is data-dependent. Since the general problem is NP-hard, there can be situations where one may be tempted to use other more accurate algorithms, such as exhaustive search, in the hope of significantly improving the accuracy of the result. To this end, we derive bounds on how far the results given by our algorithm are from the optima. As we show, in many practical problems our results are provably within a small percentage of the optima. Thus, in such cases, even the exhaustive search can provide almost no improvement. In summary our main contributions are as follows:

- We introduce the SPXY algorithm that has a linear running time in terms of the numbers of columns of the two matrices X and Y. This running time is significantly faster than the current state of the art that requires running time proportional to the product of the numbers of columns of X and Y.
- The accuracy of the SPXY algorithm compares favorably with the current state of the art.
- We derive bounds on how far the solutions produced by SPXY are from the optimal solutions.

2 The Proposed Algorithm

In this section, a computationally efficient algorithm is proposed for approximating the solution to the NP-hard selection problem of minimizing Eq. (3).

Algorithm 1. Spectral Pursuit XY algorithm

```
procedure SPXY(X, Y, k)
    S ← SELECT(X, Y, k).          ▷ select k columns from X to approximate Y.
    S ← IMPROVE(X, Y, k, S).      ▷ improve the selection of previous stage.
    return S
end procedure
```

Algorithm 2. The selection algorithm

```
procedure SELECT(X, Y, k)
    S = {}.
    while |S| < k do
        u = the first left eigenvector of Y corresponding to its largest singular value.
        i = index of the column from X, which most correlates with u.
        S = S ∪ {i};   q_i = x_i/‖x_i‖.
        Y = Y − q_i q_i^T Y;   X = X − q_i q_i^T X.
    return S
end procedure
```

The top view of the proposed algorithm is shown as Algorithm 1, with two stages involved. In the first stage, a greedy algorithm is introduced to select k columns from the dictionary matrix. In the next stage, an efficient non-greedy algorithm is used to improve the results.

2.1 The Selection Algorithm

We first observe that the vectors: $q_1 \dots q_k$, generated by Algorithm 2, are mutually orthogonal. Define $Q = (q_1 \dots q_k)$, where $Q \in \mathbb{R}^{m \times k}$. The projection of Y on Q can be written as $Q^T Y$. The reconstruction of Y from this projection can be written as: $\hat{Y} = QQ^T Y$. Therefore, the optimization problem (3) can be restated as:

$$Q = \text{argmin}_Q \|Y - QQ^T Y\|_F^2, \tag{4}$$

where Q is restricted to be the basis of k columns from X. As mentioned before, this is an NP-hard problem. If the constraint on Q is relaxed (it can be any matrix with size $m \times k$ with orthonormalized columns), then it is known that the minimizer of problem (4) is the k left eigenvectors of Y corresponding to the k largest singular values (e.g., [10]). We call the k eigenvectors corresponding to the k largest singular values as the first k eigenvectors.

Algorithm 3. The improvement algorithm

procedure IMPROVE(X, Y, k, S)
  $S_{\text{opt}} \leftarrow S$; $\text{error}_{\min} \leftarrow$ the current error value.
  iter $= 0$.
  while a stopping criterion is not met **do**
    $i = \text{mod}(\text{iter}, k)$.
    $Q_{\bar{i}} =$ basis of columns in S except the ith column of S.
    $R_Y = Y - Q_{\bar{i}} Q_{\bar{i}}^T Y, \quad R_X = X - Q_{\bar{i}} Q_{\bar{i}}^T X$.
    $u =$ the first left eigenvector of R_Y.
    $j =$ index of the most correlated column in R_X with u.
    $q_j =$ the normalized jth column of R_X; error $= \|R_Y\|_F^2 - \|q_j^T R_Y\|_F^2$.
    if error $< \text{error}_{\min}$ **then:**
      $S[i] = j$. ▷ Improvement
      $S_{\text{opt}} \leftarrow S$; $\text{error}_{\min} =$ error.
    iter += 1.
  end while
  return S_{opt}
end procedure

In the selection algorithm, we modify (4) into two sub-problems. The first one relaxes the constraint that Q must be the basis of k columns from X. This relaxation makes finding the solution tractable at the expense of resulting in a solution that may not correspond to columns in X. To fix this problem, we introduce the second sub-problem that reimposes the underlying constraint that selects the column that has highest correlation with the vector computed in the first sub-problem. These two sub-problems are formulated as follows:

$$\begin{aligned} u &= \text{argmin}_u \|Y - uu^T Y\|_F^2, \ s.t. \ \|u\| = 1, \\ i &= \text{argmin}_i |u^T q_i|, \end{aligned} \tag{5}$$

where $q_i = x_i / \|x_i\|$. The resulting i is the index of the selected column. The first sub-problem is equivalent to computing the first left eigenvector of Y (e.g., [10]). After solving for u (which is not necessarily one of the columns in X), we find the column that matches u the most (has the least angle with u).

After selecting the first column x_i, we project all columns onto the orthogonal space of this selected column. This forms the residual matrices: $Y = Y - q_i q_i^T Y$, $X = X - q_i q_i^T X$. In the future iterations, we solve problem (5) on the residual matrices. This process continues until k columns are selected.

Algorithm 2 shows the selection algorithm. To simplify notation, we do not distinguish between the selection matrix and the selection indexes. To evaluate the algorithm complexity we observe that there are several recently proposed efficient algorithms for computing eigenvectors. In particular, the randomized eigendecomposition algorithm [12] can be used to compute u in $O(mN)$ time. With this we conclude that the time complexity of Algorithm 2 is $O(km(n+N))$, and its memory complexity is $O(m(n+N))$.

2.2 The Improvement Algorithm

Since Algorithm 2 is greedy we propose to improve its result by iteratively revising the selection as long as it can be locally improved. Similar to the selection algorithm in Sect. 2.1, we modify the problem into two sub-problems. The first one is built upon the assumption that $k-1$ columns from X are already selected and the objective is to select next best column. The sub-problems are formulated as:

$$\begin{aligned} u_i &= \text{argmin}_u \|Y - Q_{\bar{i}} Q_{\bar{i}}^T Y - uu^T Y\|_F^2, \ s.t. \ \|u\| = 1, \\ j &= \text{argmin}_j |u_i^T q_j|. \end{aligned} \tag{6}$$

Here $Q_{\bar{i}}$ is the basis of $S_{\bar{i}}$ obtained by removing the ith selection in S. The first sub-problem is equivalent to finding the first left eigenvector of the residual matrix of Y: $R_Y = Y - Q_{\bar{i}} Q_{\bar{i}}^T Y$. The residual matrix of X is given by: $R_X = X - Q_{\bar{i}} Q_{\bar{i}}^T X$, and $q_j = r_X^j / \|r_X^j\|$ is the normalized jth column of R_X. The index j found in this way is the new selection which will replace the ith selection in S.

The improvement algorithm is shown as Algorithm 3. The stopping criterion that we use is a pre-defined maximum number of iterations. The algorithm can be efficiently implemented by using the rank-one update for QR factorization [7] with complexity per iteration of $O(km(n+N))$, and memory complexity of $O(m(n+N))$. In our implementation we limit the algorithm to run no more than 30 iterations. The algorithm terminates if there is no improvement in 5 iterations. The convergence behavior is studied in Sect. 5.

3 Fractional Bound

We proceed to show how to obtain nontrivial bound on how close the computed result is to the optimal solution. As described in (4) the algorithms minimize the following error:

$$\begin{aligned} &E(S) = E(Q) = \|Y - QQ^T Y\|_F^2 = \|Y\|_F^2 - \|Q^T Y\|_F^2 = \|Y\|_F^2 - G(S), \\ &\text{where } Q \text{ is an orthogonal basis of } S, \text{ and } G(S) = \|Q^T Y\|_F^2. \end{aligned}$$

Since $\|Y\|_F^2$ is independent of S, the minimization of $E(S)$ is equivalent to the maximization of $G(S)$. Let E_{opt} be the smallest possible error, and let G_{opt} be the largest possible value of $G(S)$. They are related by: $E_{\text{opt}} = \|Y\|_F^2 - G_{\text{opt}}$. We define the fractional bound in terms of $G(S)$ as follows:

$$\tilde{b}_f(S) = (G_{\text{opt}} - G(S))/G_{\text{opt}}.$$

A smaller $\tilde{b}_f$ value indicates a better result, and in particular, $\tilde{b}_f = 0$ implies an optimal solution. Unfortunately both G_{opt} and E_{opt} are unknown. However, we observe that $\tilde{b}_f(S)$ is monotonically increasing as a function of G_{opt}. This implies that any nontrivial upper bound of G_{opt} can be used to obtain a nontrivial estimate of $\tilde{b}_f(S)$ as follows: If $G_{\text{upper}} \geq G_{\text{opt}}$ then:

$$\tilde{b}_f(S) = \frac{G_{\text{opt}} - G(S)}{G_{\text{opt}}} = 1 - \frac{G(S)}{G_{\text{opt}}} \leq 1 - \frac{G(S)}{G_{\text{upper}}}.$$

Let U_k be the matrix computed from the first k left eigenvectors of Y. Then for any S we have: $E(U_k) \leq E(S)$, so that for any S, $G(U_k) = \|Y\|_F^2 - E(U_k) \geq G(S)$. This shows that $G(U_k) \geq G_{\text{opt}}$, and gives the following formula for provable fractional bound:

$$b_f = 1 - G(S)/G(U_k). \tag{7}$$

4 Robustness

The sparse approximation algorithms are vulnerable to outliers. The span of outliers usually covers a bigger subspace, which may not be desirable to be represented. In our algorithm, at each iteration, we compute the first left eigenvector of the residual target matrix to guide the selection. If this eigenvector is robust, outliers in the target matrix X will be automatically rejected. We show that this eigenvector is the most robust spectral component against perturbation of matrix the Y. Additionally, we can use robust principal component analysis algorithms (e.g., [15,26]) to compute this eigenvector.

Consider the second moment matrix of Y: $B = YY^T = \sum_{i=1}^{N} y_i y_i^T$. The first eigenvector of this matrix is same as the first left eigenvector of Y. If an outlier is added, it will perturb this matrix. The following lemma proven in [30] shows the robustness of ith eigenvector of B against the perturbation.

Lemma 1. Let B be a symmetric matrix with rank r, $(\lambda_1, \ldots, \lambda_r)$ be its eigenvalues (in decreasing order) and $(u_1, \ldots, u_r)$ be the corresponding eigenvectors. Then: $\frac{\|\partial u_i\|_2}{\|\partial B\|_F} \leq \sqrt{\sum_{j \neq i} \frac{1}{(\lambda_i - \lambda_j)^2}}$.

Define the sensitivity of the ith eigenvector as: $s_i = \sqrt{\sum_{j \neq i} \frac{1}{(\lambda_i - \lambda_j)^2}}$. It is known that s_1 is smaller than other $s_i, \forall i > 1$ if the gap between consecutive eigenvalues is monotonously decreasing [30].

5 Experimental Results

We describe experiments on various datasets that are publicly available, and compare the proposed algorithm with the following algorithms: SOMP [25]; S-SBR [3]; SOLS [6]; CM [4]; ISOLS (the exact version is used) [19]. Besides, the results of the random selection are shown. The error and bound are defined in (3) and (7), respectively. In experimental results, they are shown in percentage: error $= \frac{E(S)}{\|Y\|_F^2} * 100$; bound $= b_f * 100$. The computational efficiency and the selection accuracy of our algorithm are demonstrated.

5.1 Quantitative Comparison

In the first experiment, we evenly split the datasets into two matrices, serving as X and Y. The results are shown in Table 2. In the second experiment, we randomly split the datasets with the proportion of X:Y = 3:1. The results are shown in Table 3. We choose to split the datasets, since the learning of the dictionary matrix is another big topic and task-dependent (e.g., [29]). As mentioned in [19], the results of SOLS, CM, and ISOLS are exactly same (different in terms of speed). We show the results of ISOLS, as it is the fastest among these three algorithms. The parameter γ for S-SBR is 0, since k is known in our case.

Observe that our algorithm is much faster than other algorithms. The running time of ISOLS almost does not change as the increase of k. However, its initial step can be very expensive for big dense datasets. Taking the Duke breast cancer dataset in Table 2 as an example, its initial step takes 8.94 s. The overall running time for our algorithm is less than 1 s.

Table 2. Comparison when data is split with proportion to 1:1. "–" indicates that the algorithm runs more than 30 min without results.

k	Random	SOMP [25]	S-SBR [3]	ISOLS [19]	SPXY (this work)		
	Error	Error/time (s)	Error/time (s)	Error/time (s)	Error	Bound	Time (s)
Duke breast cancer X:44 × 3,565 Y:44 × 3,565							
10	40.3	30.7/0.73	29.6/3.83	29.6/9.13	**29.1**	**4.2%**	**0.13**
20	22.4	18.5/1.42	16.7/9.97	16.7/9.37	**16.3**	**3.1%**	**0.16**
30	11.3	9.4/2.15	**7.9**/18.04	**7.9**/9.41	8.0	**1.8%**	**0.24**
YearPredictionMSD X:91 × 257,672 Y:91 × 257,673							
10	9.2	–	–	–	**5.6**	**0.9%**	**28**
20	5.3	–	–	–	**2.7**	**0.6%**	**37**
30	2.8	–	–	–	**1.5**	**0.5%**	**45**
Sift X:128 × 500,000 Y:128 × 500,000							
10	36.3	–	–	–	**27.2**	**6.1%**	**76**
20	24.7	–	–	–	**19.8**	**5.5%**	**97**
30	21.7	–	–	–	**15.4**	**5.2%**	**117**
Sift:transpose X:500,000 × 128 Y:500,000 × 128							
10	**60.3**	–	–	–	60.5	**49%**	**61**
20	60.4	–	–	–	**59.7**	**53%**	**159**
30	59.4	–	–	–	**59.3**	**54%**	**150**

The errors given by our algorithm are typically smaller or similar to the errors of ISOLS. Additionally, our errors come with bounds indicating how close the solutions are to the optima. The bounds are usually within 10%. The reason for high bound values on Sift:transpose dataset is that X cannot explain Y well, since the errors are very big and do not change as the increase of k. It is further discussed in Sect. 5.2. The errors given by SOMP are much bigger. One of the reasons is that SOMP does not update the matrix X on the null-space in each iteration. Clearly, other algorithms are not practical for big datasets.

Table 3. Comparison when data is split with proportion to 3:1.

k	Random	SOMP [25]	S-SBR [3]	ISOLS [19]	SPXY (this work)		
	Error	Error/time (s)	Error/time (s)	Error/time (s)	Error	Bound	Time (s)
Duke breast cancer X:44 × 5,347 Y:44 × 1,783							
10	44.9	31.4/0.84	30.3/4.85	30.3/6.89	**29.8**	**4.5%**	**0.15**
20	24.3	18.5/1.65	17.0/11.56	17.0/7.15	**16.9**	**3.4%**	**0.22**
30	12.4	9.5/2.46	8.0/20.01	8.0/7.29	**7.9**	**1.6%**	**0.26**
YearPredictionMSD X:91 × 386,508 Y:91 × 128,873							
10	11.0	–	–	–	**5.4**	**0.7%**	**23**
20	5.3	–	–	–	**2.7**	**0.6%**	**31**
30	2.9	–	–	–	**1.4**	**0.4%**	**37**
Sift X:128 × 750,000 Y:128 × 250,000							
10	38.9	–	–	–	**26.8**	**5.7%**	**63**
20	27.3	–	–	–	**19.7**	**5.5%**	**110**
30	19.9	–	–	–	**15.1**	**4.9%**	**134**
Extended Yale Face B X:32,256 × 1,488 Y:32,256 × 496							
10	23.6	18.6/214	18.0/176	18.0/433	**17.6**	**7.4%**	**18**
20	18.2	15.1/432	**13.9**/552	**13.9**/362	14.1	**8.2%**	**39**
30	15.5	12.8/649	**11.7**/1105	**11.7**/372	12.1	**8.1%**	**53**

5.2 Correlation Values of Algorithm 2

In the first stage, Algorithm 2 greedily selects the column with highest correlation value in each iteration. Figure 1 shows the correlation values of the selected columns as the run of Algorithm 2 for various datasets (split with proportion to 1:1) with $k = 30$. Generally, as the increase of the iteration, the maximum correlation value between the first left eigenvector of the residual matrix of Y and the normalized residual columns of X decreases. For the transpose of Sift dataset, the correlation value for the first iteration is 0.78, but for later iterations, they are very small (almost 0). It shows that X cannot explain Y well.

5.3 Convergence of Algorithm 3

In this experiment, we investigate the convergence of Algorithm 3 in the second stage. Here we set the maximum number of iterations to be 50 without early termination. The results are shown in Fig. 2 and Fig. 3. The red lines correspond to errors or bounds given in the first stage. As expected, the algorithm improves errors and bounds sharply at the beginning.

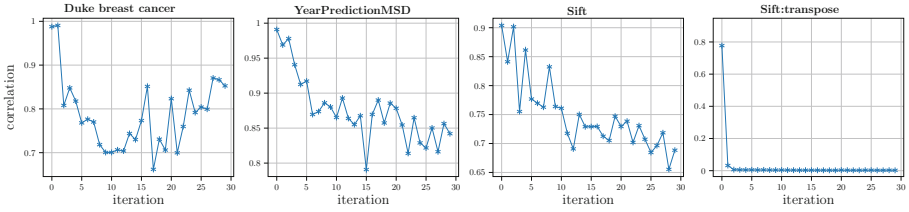

Fig. 1. Correlation values of selected columns for various datasets.

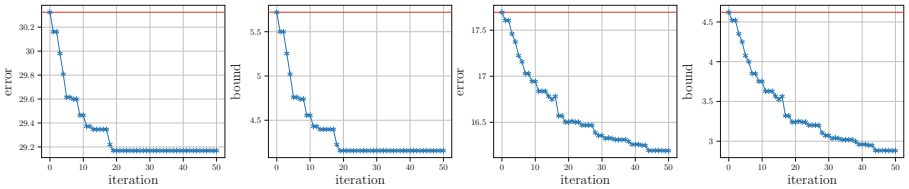

Fig. 2. Convergence of Algorithm 3 on Duke breast cancer dataset (1:1). The left two plots are for $k = 10$. The right two plots are for $k = 20$.

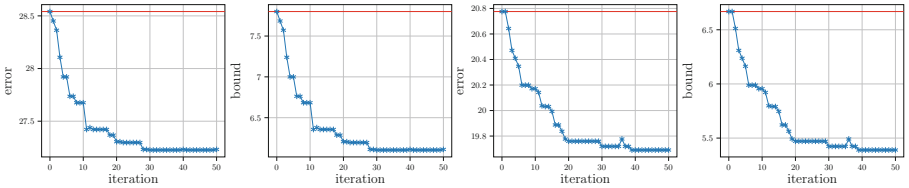

Fig. 3. Convergence of Algorithm 3 on Sift dataset (1:1). The left two plots are for $k = 10$. The right two plots are for $k = 20$.

6 Conclusion

The problem discussed in this paper, simultaneously approximating one entire matrix in terms of a small number of selected columns from another matrix, is well-known, and appears to have many practical applications.

A novel two-stage selection algorithm, referred to as Spectral Pursuit for X and Y (SPXY), is presented, which selects columns capturing the spectral characteristics of the target matrix. What we found surprising is that it is possible to efficiently implement the algorithms with linear complexity w.r.t. the size of the two matrices. We show experimentally that our algorithm can outperform the state-of-the-art methods.

In addition to producing a solution, our algorithm produces a bound on how far the solution is from the optimum. The quality of the bound is data-dependent. In some cases (e.g., when X cannot explain Y well), it is loose, but in other cases, it shows that the computed result is very close to the optimal solution.

References

1. Arai, H., Maung, C., Schweitzer, H.: Optimal column subset selection by A-star search. In: AAAI 2015, pp. 1079–1085. AAAI Press (2015)
2. Arai, H., Maung, C., Xu, K., Schweitzer, H.: Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In: AAAI 2016 (2016)
3. Belmerhnia, L., Djermoune, E.H., Brie, D.: Greedy methods for simultaneous sparse approximation. In: 22nd European Signal Processing Conference (2014)
4. Çivril, A., Magdon-Ismail, M.: Column subset selection via sparse approximation of SVD. Theoret. Comput. Sci. **421**, 1–14 (2012)

5. Chen, J., Huo, X.: Theoretical results of sparse representations of multiple measurement vectors. IEEE Trans. Signal Process. **54**(12), 4634–4643 (2006)
6. Cotter, S.F., Rao, B.D., Engen, K., Kreutz-Delgado, K.: Sparse solutions to linear inverse problems with multiple measurement vectors. ASP **53**(7), 2477–2488 (2005)
7. Daniel, J., Gragg, W., Kaufman, L., Stewart, G.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. Math. Comput. **30**, 772–795 (1976)
8. Davis, G., Mallat, S., Avellaneda, M.: Adaptive greedy approximations. Constr. Approx. **13**(1), 57–98 (1997)
9. Furnival, G., Wilson, R.: Regressions by leaps and bounds. Technometrics **42**, 69–79 (1974)
10. Golub, G.H., Van-Loan, C.F.: Matrix Computations, 4th edn. The Johns Hopkins University Press, Baltimore (2013)
11. Guha, T., Ward, R.K.: Learning sparse representations for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 1576–1588 (2011)
12. Halko, N., Martinsson, P., Tropp, J.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**, 217–288 (2011)
13. He, B., Shah, S., Maung, C., Arnold, G., Wan, G., Schweitzer, H.: Heuristic search algorithm for dimensionality reduction optimally combining feature selection and feature extraction. In: AAAI 2019, pp. 2280–2287. AAAI Press, California (2019)
14. Joneidi, M., et al.: Select to better learn: fast and accurate deep learning using data selection from nonlinear manifolds. In: CVPR, pp. 7819–7829 (2020)
15. Lerman, G., Maunu, T.: Fast, robust and non-convex subspace recovery. Inf. Inference J. IMA **7**(2), 277–336 (2018)
16. Mairal, J., Bach, F., Ponce, J.: Sparse Modeling for Image and Vision Processing. Foundations and Trends in Computer Graphics and Vision (2014)
17. Malioutov, D., Cetin, M., Willsky, A.: A sparse signal reconstruction perspective for source localization with sensor arrays. IEEE Trans. Signal Process. **53**, 3010–3022 (2005)
18. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press, Cambridge (1999)
19. Maung, C., Schweitzer, H.: Improved greedy algorithms for sparse approximation of a matrix in terms of another matrix. IEEE TKDE **27**(3), 769–780 (2015)
20. Natarajan, B.: Sparse approximate solutions to linear systems. SIAM J. Comput. **24**, 227–234 (1995)
21. Qian, C., Yu, Y., Zhou, Z.: Subset selection by pareto optimization. In: Advances in Neural Information Processing Systems. Curran Associates, Inc. (2015)
22. Soussen, C., Gribonval, R., Idier, J., Herzet, C.: Joint k-step analysis of orthogonal matching pursuit and orthogonal least squares. IEEE Trans. Inf. Theory **59**, 3158–3174 (2013)
23. Tibshirani, R.: Regression shrinkage and selection via the Lasso. J. R. Stat. Soc. B **58**, 267–288 (1996)
24. Tropp, J.A.: Greed is good: algorithmic results for sparse approximation. IEEE Trans. Inf. Theory **50**(10), 2231–2242 (2004)
25. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Algorithms for simultaneous sparse approximation. Part I: greedy pursuit. Signal Process. **86**(3), 572–588 (2006)
26. Wan, G., Schweitzer, H.: A new robust subspace recovery algorithm (student abstract). In: the 35th National Conference on Artificial Intelligence (AAAI) (2021)
27. Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T., Yan, S.: Sparse representation for computer vision and pattern recognition. Proc. IEEE **98**, 1031–1044 (2010)

28. Xu, X., Shi, Z.: Multi-objective based spectral unmixing for hyperspectral images. ISPRS J. Photogramm. Remote Sens. **124**, 54–69 (2017)
29. Xu, Y., Li, Z., Yang, J., Zhang, D.: A survey of dictionary learning algorithms for face recognition. IEEE Access **5**, 8502–8514 (2017)
30. Zaeemzadeh, A., Joneidi, M., Rahnavard, N., Shah, M.: Iterative projection and matching: finding structure-preserving representatives and its application to computer vision. In: CVPR 2019 (2019)
31. Zhang, Q., Liu, Y., Blum, R., Han, J., Tao, D.: Sparse representation based multi-sensor image fusion for multi-focus and multi-modality images: a review. Inf. Fusion **40**, 57–75 (2018)
32. Zhang, T.: Adaptive forward-backward greedy algorithm for sparse learning with linear models. In: Advances in Neural Information Processing Systems (2009)
33. Zhang, Z., Xu, Y., Yang, J., Li, X., Zhang, D.: A survey of sparse representation: algorithms and applications. IEEE Access **3**, 490–530 (2015)
34. Zhu, X., Hu, R., Lei, C., Thung, K.H., Zheng, W., Wang, C.: Low-rank hypergraph feature selection for multi-output regression. World Wide Web **22**(2), 517–531 (2017). https://doi.org/10.1007/s11280-017-0514-5