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@utdallas.edu, 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 the running time of $O(mnN)$. We describe a significantly faster algorithm with the 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 from 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., [15, 26, 32]). 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 as simultaneous sparse approximation (see, e.g., [16, 24]). It has various applications such as supervised feature selection for multi-target regression [33], human action recognition in videos [10], multi-sensor image fusion [30], and hyperspectral image unmixing [27].

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

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 which 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 $Y = (y_1, \ldots, y_N)$ be a "target" matrix of m rows and N columns. Let $X = (x_1, \ldots, x_n)$ 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 (1) is called a sparse approximation if only a small number of rows of $\tilde{A}$ are nonzero. See, e.g., [3, 18, 23, 24]. 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 $S = (x_{s_1} \ldots x_{s_k})$ 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 the entire matrix Y can be simultaneously approximated by the linear combination of the selected columns in S. The quality of the selected columns in S is measured by the following error in the Frobenius norm:

$$E(S) = \min_A \|Y - SA\|_F^2. \tag{3}$$

The following cases are special sub-problems:

- When N=1, the problem is called sparse approximation, where Y is a signal vector, X is a dictionary. It is also called supervised subset selection in which Y corresponds to the labels for the data matrix X. See, e.g., [8, 19].
- When X=Y, it is known as unsupervised feature selection if each column corresponds to a data feature, or representative selection if each column corresponds to a data point. See, e.g., [12, 29].

Optimal solutions, as well as approximated solutions within a constant are known to be NP-hard even when N=1 [7, 19].

1.2 Related Work

Extensive studies were directed at the N=1 case (e.g., [23, 17, 8, 31, 20]). Applications include signal processing (e.g., [23, 17]) and supervised feature selection

in linear regression (e.g., [8, 31, 20]). An optimal algorithm proposed in [8] finds the best solution, but it is not feasible for large k and large datasets. Previously proposed approximate solutions can be roughly categorized 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 [31], these two techniques have limitations due to their greedy behavior. The author of [31] 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 [22]. In [20], 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 the unsupervised feature selection or unsupervised column subset selection. See, e.g., [9, 3, 1, 29, 13]. Recently, a fast algorithm [29] was proposed, which greedily selects columns that are closest to the first left eigenvector of a residual matrix. An algorithm in [13] is introduced to iteratively improve a selection using a bi-directional stepwise refinement. Our method is motivated by these two works, generalizing them to the supervised multiple-target case. In addition, we derive guarantees on the accuracy.

In the multi-target case, the target matrix is Y with $N > 1$. 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 columns in Y. Previously proposed algorithms for this general case are typically greedy. See, e.g., [24, 18, 3, 4, 2]. Some of these algorithms are generated from the $N{=}1$ case. For example, the Simultaneous Orthogonal Matching Pursuit (SOMP) [24] is generated from the Orthogonal Matching Pursuit (OMP) [23, 21] 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) [21]. See [4, 3] for the analysis of the SOLS algorithm. An algorithm established in [3] improves the SOLS algorithm in term of speed at the cost of increased memory. In [18], 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.

Algorithms	**time complexity**	**memory complexity**
SOMP [24]	$O(kmnN)$	$O(m(N+k))$
S-SBR [2]	$O(Tkmn N)$	$O(m(N+k))$
SOLS [5]	$O(kmnN)$	$O(m(n+N))$
CM [3]	$O(nN(m+k))$	$O(m(n+N))+nN$
ISOLS [18]	$O(mnN)$	$O(km)+2n$
SPXY (this work)	$O(km(n+N))$	$O(m(n+N))$

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

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 eigenvectors of Y corresponding to the k largest singular values (see e.g., [9]). Our algorithmic approach is based on this result.

Motivated by [29] and [13], 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 bound on how far is the selection computed by SPXY from the optimal solution. (Recall 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 optimum. As we show, in many practical problems our results are provably within a small percentage of the optimum. It is clear that in such cases even exhaustive search can provide almost no improvement.

In summary our main contributions are as follows:

- We introduce the SPXY algorithm that has 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 stat 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 bound indicating on how far the solution produced by SPXY is from an optimal solution.

2 The Proposed Algorithm

In this section, a computationally efficient algorithm is proposed for approximating the solution to the NP-hard selection problem (2).

The top view of the proposed algorithm is described in 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.

Algorithm 1 Spectral Pursuit XY algorithm

procedure SPXY(X, Y, k)
  $S \leftarrow$ SELECT(X, Y, k). ▷ select k columns.
  $S \leftarrow$ IMPROVE(X, Y, k, S). ▷ improve the selection in 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 the largest singular value.
    $X =$ column normalized X, obtained by dividing each column by its norm.
    $i =$ index of the column from X, which most correlates with u.
    $S = S \cup \{i\}$; $q_i =$ the ith column of X.
    $Y = Y - q_i q_i^T Y$; $X = X - q_i q_i^T X$.
  return S
end procedure

2.1 The Selection Algorithm

Projections of all columns in Y on the orthonormal basis of k columns from X, indexed by S, can be expressed by: $Q^T Y$, where $Q \in \mathbb{R}^{m \times k}$ is the basis of a set of k columns from X indexed by S. The reconstruction of Y can be written as: $\hat{Y} = QQ^T Y$. Therefore, the optimization problem (2) 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 the minimizer of problem (4) is the k left eigenvectors of Y corresponding to the k largest singular values (see e.g., [9]). We denote the k eigenvectors corresponding to the k largest singular values as the first k eigenvectors.

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 belong 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., [9]).

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.
    $R_Y = (I - Q_{\bar{i}}Q_{\bar{i}}^T)Y, \quad R_X = (I - Q_{\bar{i}}Q_{\bar{i}}^T)X$.
    $u =$ the first left eigenvector of R_Y.
    $R_X =$ column normalized R_X.
    $j =$ index of the most correlated column in R_X with u.
    $S_i = j$. ▷ Improvement
    $q_j =$ the jth column of R_X; error $= \|R_Y\|_F^2 - \|q_j^T R_Y\|_F^2$.
    if error $<$ $\text{error}_{\min}$ **then:** $S_{\text{opt}} \leftarrow S$; $\text{error}_{\min} =$ error.
    iter += 1.
  end while
  return S_{opt}
end procedure

After solving for u (which is not necessarily one of columns in X), we find the column that matches to u the most (has the least angle with u).

After selecting the first column x_i, we project all columns onto the null 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 process of the proposed selection algorithm. Without loss of generality, we do not distinguish the selection matrix and the selection indexes. The randomized eigendecomposition algorithm [11] can be used to compute u. It is a low-complexity algorithm with no parameters to be tuned. The time complexity is $O(km(n+N))$; the memory complexity is $O(m(n+N))$.

2.2 The Improvement Algorithm

The improvement algorithm facilitates revising the current selection in each iteration and escaping from local optima. Similar to the selection algorithm in Section 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 column of the current selection S. The first sub-problem is equivalent to finding the first left eigenvector of the residual matrix: $R_Y = Y - Q_{\bar{i}}Q_{\bar{i}}^T Y$. 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 jth column of normalized R_X. The found index j is the new selection which will replace the ith selection in S.

The process of the improvement algorithm is elaborated in Algorithm 3. The stopping criteria can be reaching a pre-defined maximum number of iterations. The algorithm can be efficiently implemented by using the rank-one update for QR factorization [6] with computational complexity $O(km(n{+}N))$ and memory complexity $O(m(n+N))$. The convergence behavior is studied in Section 5.

3 Fractional Bound

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

$$Q = \text{argmin}_Q \|Y - QQ^T Y\|_F^2 = \text{argmin}_Q(\|Y\|_F^2 - \|Q^T Y\|_F^2).$$

Since $\|Y\|_F^2$ is a constant, the minimization of the error is equivalent to the maximization of the explained information from Y: $Q = \text{argmax}_Q \|Q^T Y\|_F^2$.

Consider a run of the proposed algorithm producing the selection S with error E. Let E_* be the smallest possible error. The bound indicating how close the solution error to the optimal error is: $E - E_*$. The fractional bound is given by: $(E - E_*)/E_*$. But it is undefined when $E_* = 0$.

Alternatively, we define the fractional bound in terms of the explained information. Let $G = \|Q^T Y\|_F^2$ be the explained or gained information by the selection S, where Q is the basis of S. Let G_* be the optimal gain. Let b_f be the fractional bound: $b_f = (G_* - G)/G_*$. A smaller b_f value indicates a better result, and in particular $b_f = 0$ implies an optimal solution.

However G_* or E_* is unknown. Let G_{upper} be the gain if using the first k left eigenvectors of Y to replace the basis of S. As mentioned before, G_{upper} is the maximum gain we could obtain, when Q contains k vectors with no constraints (e.g., [9]). Then: $G_{\text{upper}} \geq G_* \geq G$. Now we can compute the upper bound on b_f by using G_{upper} as follows:

$$b_f = \frac{G_* - G}{G_*} = 1 - \frac{G}{G_*} \leq 1 - \frac{G}{G_{\text{upper}}}.$$

This bound is a nontrivial bound on b_f, and can be calculated. This gives a bound on optimality of the computed result.

4 Robustness

The sparse approximation algorithms are vulnerable to outliers. The span of outliers usually covers a wilder subspace, which may not be desirable to be represented. In our algorithm, in 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 Y. Additionally, we can use robust principal component analysis algorithms (e.g., [25, 14]) 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 [29] 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 [29].

5 Experimental Results

We describe experiments on various datasets that are publicly available, and compare the proposed algorithm with the following algorithms: SOMP [24]; S-SBR [2]; SOLS [5]; CM [3]; ISOLS (the exact version is used) [18]. In addition, the results of random selection are shown. 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., [28]). As mentioned in [18], the results of SOLS, CM and ISOLS are exactly same (different in term 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 significantly faster than other algorithms. The advantage of ISOLS is that its running time 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 seconds. The running time for our algorithm is less than 1 second.

In terms of accuracy, our algorithm typically produces smaller errors than other algorithms. Additionally, the errors come with bounds indicating how close the solution is to the optimum. The bounds are usually within 10%. The high bound values for Sift:transpose is because of that X cannot explain Y well, discussed in Section 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.

k	Random error	SOMP [24] error / time(s)	S-SBR [2] error / time(s)	ISOLS [18] error / time(s)	SPXY (this work) error	bound	time (s)
Duke breast cancer X:$44 \times 3,565$ Y:$44 \times 3,565$							
10	62,524	48,214 / 0.73	46,376 / 3.83	46,376 / 9.13	**45,900**	**4.3%**	**0.17**
20	33,976	29,062 / 1.42	26,186 / 9.97	26,186 / 9.37	**25,952**	**3.3%**	**0.26**
30	18,462	14,779 / 2.15	12,403 / 18.04	12,403 / 9.41	**12,271**	**1.6%**	**0.39**
YearPredictionMSD X:$91 \times 257,672$ Y:$91 \times 257,673$							
10	7.93E11	-	-	-	**4.30E11**	**1.1%**	**32**
20	3.71E11	-	-	-	**2.00E11**	**0.6%**	**59**
30	2.04E11	-	-	-	**1.11E11**	**0.5%**	**70**
Sift X:$128 \times 500,000$ Y:$128 \times 500,000$							
10	4.85E10	-	-	-	**3.52E10**	**6.0%**	**101**
20	3.87E10	-	-	-	**2.57E10**	**5.6%**	**156**
30	2.52E10	-	-	-	**1.95E10**	**4.8%**	**185**
Sift:transpose X:$500,000 \times 128$ Y:$500,000 \times 128$							
10	8.20E10	-	-	-	**7.83E10**	**49.0%**	**87**
20	7.83E10	-	-	-	**7.71E10**	**52.3%**	**159**
30	7.75E10	-	-	-	**7.67E10**	**54.4%**	**176**

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

k	Random error	SOMP [24] error / time(s)	S-SBR [2] error / time(s)	ISOLS [18] error / time(s)	SPXY (this work) error	bound	time (s)
Duke breast cancer X:$44 \times 5,347$ Y:$44 \times 1,783$							
10	29,888	24,659 / 0.84	23,746 / 4.85	23,746 / 6.89	**23,293**	**4.3%**	**0.15**
20	17,784	14,516 / 1.65	13,319 / 11.56	13,319 / 7.15	**13,146**	**3.5%**	**0.22**
30	9.470	7,482 / 2.46	6,299 / 20.01	6,299 / 7.29	**6,235**	**1.7%**	**0.37**
YearPredictionMSD X:$91 \times 386,508$ Y:$91 \times 128,873$							
10	4.53E11	-	-	-	**2.10E11**	**0.9%**	**21**
20	1.57E11	-	-	-	**0.99E11**	**0.6%**	**45**
30	1.05E11	-	-	-	**0.53E11**	**0.4%**	**54**
Sift X:$128 \times 750,000$ Y:$128 \times 250,000$							
10	2.57E10	-	-	-	**1.76E10**	**6.2%**	**82**
20	1.68E10	-	-	-	**1.28E10**	**5.8%**	**161**
30	1.32E10	-	-	-	**1.28E10**	**5.8%**	**192**
Extended Yale Face B X:$32,256 \times 1,488$ Y:$32,256 \times 496$							
10	1.16E10	0.88E10 / 214	**0.85E10** / 176	**0.85E10** / 433	**0.85E10**	**7.9%**	**22**
20	0.82E10	0.71E10 / 432	**0.66E10** / 552	**0.66E10** / 362	0.67E10	**8.2%**	**49**
30	0.73E10	0.60E10 / 649	**0.55E10** / 1105	**0.55E10** / 372	0.59E10	**8.3%**	**53**

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

5.2 Correlation Values of Algorithm 2

In the first stage, Algorithm 2 greedily selects the column that has highest correlation value in each iteration. Fig. 1 shows the correlation values of the selected columns as the run of Algorithm 2 for various datasets with $k = 30$. Generally,

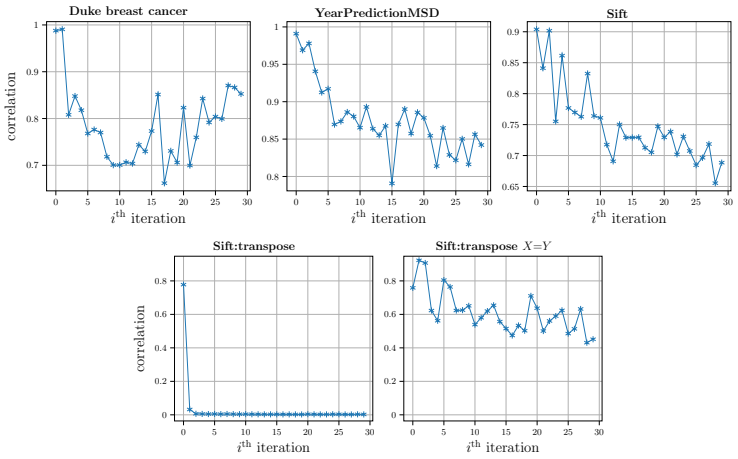

Fig. 1: Correlation values of selected columns. The largest correlation value generally goes down as the run of the selection algorithm in Algorithm 2.

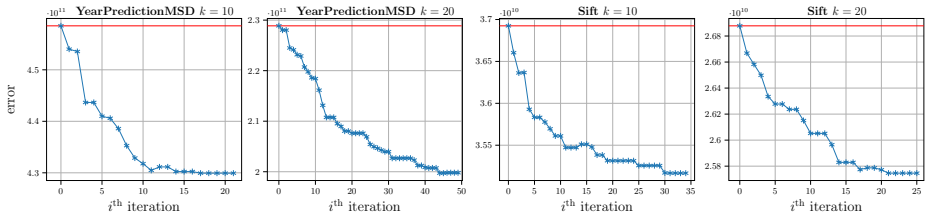

Fig. 2: Convergence of Algorithm 3. The red lines are the errors given by Algorithm 2 in the first stage. Algorithm 3 initially improves the results fast.

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). The reason is that X cannot explain Y well. This can be verified by replacing X by Y itself. The results are shown in the last plot of Fig. 1.

5.3 Convergence of Algorithm 3

In this experiment, we investigate the convergence of Algorithm 3 in the second stage. The results are shown in Fig. 2. We set the maximum number of iterations to be 50 with early termination. (The algorithm terminates if no improvements in 10 iterations.) As expected, the errors decrease sharply at the beginning; later the errors tend to be stable.

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 is able to outperform the state-of-the-art methods.

In addition to producing a solution our algorithm produces 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., Xu, K., Schweitzer, H.: Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In: Proceedings of the 30th National Conference on Artificial Intelligence (AAAI'16). pp. 666–672 (2016)
2. Belmerhnia, L., Djermoune, E.H., Brie, D.: Greedy methods for simultaneous sparse approximation. In: 22nd European Signal Processing Conference (2014)
3. Çivril, A., Magdon-Ismail, M.: Column subset selection via sparse approximation of SVD. Theoretical Computer Science **421**, 1–14 (Mar 2012)
4. Chen, J., Huo, X.: Theoretical results of sparse representations of multiple measurement vectors. IEEE Transactions on Signal processing **54**(12), 4634–4643 (2006)
5. 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)
6. Daniel, J.W., Gragg, W.B., Kaufman, L., Stewart, G.W.: Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. Mathematics of Computation **30**(136), 772–795 (1976)
7. Davis, G., Mallat, S., Avellaneda, M.: Adaptive greedy approximations. Constructive approximation **13**(1), 57–98 (1997)
8. Furnival, G.M., Wilson, R.W.: Regressions by leaps and bounds. Technometrics **16**(4), 499–511 (1974)
9. Golub, G.H., Van-Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore, fourth edn. (2013)
10. Guha, T., Ward, R.K.: Learning sparse representations for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **34** (2011)
11. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review **53**(2), 217–288 (2011)
12. 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: Proceedings of the 33rd National Conference on Artificial Intelligence (AAAI'19). pp. 2280–2287. AAAI Press, California (2019)

13. Joneidi, M., Vahidian, S., Esmaeili, A., Wang, W., Rahnavard, N., Lin, B., Shah, M.: Select to better learn: Fast and accurate deep learning using data selection from nonlinear manifolds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7819–7829 (2020)
14. Lerman, G., Maunu, T.: Fast, robust and non-convex subspace recovery. Information and Inference: A Journal of the IMA **7**(2), 277–336 (2018)
15. Mairal, J., Bach, F., Ponce, J.: Sparse modeling for image and vision processing (2014)
16. Malioutov, D., Cetin, M., Willsky, A.S.: A sparse signal reconstruction perspective for source localization with sensor arrays. IEEE transactions on signal processing **53**(8), 3010–3022 (2005)
17. Mallat, S.: A wavelet Tour of Signal Processing. Academic Press (1999)
18. Maung, C., Schweitzer, H.: 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 (March 2015)
19. Natarajan, B.K.: Sparse approximate solutions to linear systems. SIAM Journal of Computing **25**(2), 227–234 (1995)
20. Qian, C., Yu, Y., Zhou, Z.: Subset selection by pareto optimization. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 1774–1782. Curran Associates, Inc. (2015)
21. Soussen, C., Gribonval, R., Idier, J., Herzet, C.: Joint k-step analysis of orthogonal matching pursuit and orthogonal least squares. IEEE Transactions on Information Theory **59**, 3158–3174 (2013)
22. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Royal. Statist. Soc B. **58**(1), 267–288 (1996)
23. Tropp, J.A.: Greed is good: algorithmic results for sparse approximation. IEEE Transactions on Information Theory **50**(10), 2231–2242 (2004)
24. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Algorithms for simultaneous sparse approximation. part I: Greedy pursuit. Signal Processing **86**(3), 572–588 (2006)
25. Wan, G., Schweitzer, H.: A new robust subspace recovery algorithm (student abstract). In: the 35th National Conference on Artificial Intelligence (AAAI) (2021)
26. Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T.S., Yan, S.: Sparse representation for computer vision and pattern recognition. Proceedings of the IEEE **98**(6), 1031–1044 (2010)
27. Xu, X., Shi, Z.: Multi-objective based spectral unmixing for hyperspectral images. ISPRS Journal of Photogrammetry and Remote Sensing **124**, 54–69 (2017)
28. Xu, Y., Li, Z., Yang, J., Zhang, D.: A survey of dictionary learning algorithms for face recognition. IEEE access **5**, 8502–8514 (2017)
29. Zaeemzadeh, A., Joneidi, M., Rahnavard, N., Shah, M.: Iterative projection and matching: Finding structure-preserving representatives and its application to computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5414–5423 (2019)
30. Zhang, Q., Liu, Y., Blum, R.S., Han, J., Tao, D.: Sparse representation based multi-sensor image fusion for multi-focus and multi-modality images: A review. Information Fusion **40**, 57–75 (2018)
31. Zhang, T.: Adaptive forward-backward greedy algorithm for sparse learning with linear models. In: Advances in Neural Information Processing Systems (2009)
32. Zhang, Z., Xu, Y., Yang, J., Li, X., Zhang, D.: A survey of sparse representation: algorithms and applications. IEEE access **3**, 490–530 (2015)
33. 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) (2019)