FAST NONNEGATIVE MATRIX FACTORIZATION: AN ACTIVE-SET-LIKE METHOD AND COMPARISONS[*]

JINGU KIM[†] AND HAESUN PARK[†]

Abstract. Nonnegative matrix factorization (NMF) is a dimension reduction method that has been widely used for numerous applications including text mining, computer vision, pattern discovery, and bioinformatics. A mathematical formulation for NMF appears as a non-convex optimization problem, and various types of algorithms have been devised to solve the problem. The alternating nonnegative least squares (ANLS) framework is a block coordinate descent approach for solving NMF, which was recently shown to be theoretically sound and empirically efficient. In this paper, we present a novel algorithm for NMF based on the ANLS framework. Our new algorithm builds upon the block principal pivoting method for the nonnegativity-constrained least squares problem that overcomes a limitation of the active set method. We introduce ideas that efficiently extend the block principal pivoting method within the context of NMF computation. Our algorithm inherits the convergence property of the ANLS framework and can easily be extended to other constrained NMF formulations. Extensive computational comparisons using data sets that are from real life applications as well as those artificially generated show that the proposed algorithm provides state-of-the-art performance in terms of computational speed.

Key words. nonnegative matrix factorization, nonnegativity-constrained least squares, block principal pivoting method, active set method, lower rank approximation, dimension reduction

AMS subject classifications. 15A23, 62H25, 65F30, 65K05

1. Introduction.

Nonnegative matrix factorization (NMF) [28, 22] has attracted much attention during the past decade as a dimension reduction method in machine learning and data mining. NMF is considered for high-dimensional data in which each element has a nonnegative value, and it provides a low rank approximation formed by factors whose elements are also nonnegative. Due to the nonnegativity, the factors of low rank approximation give a natural interpretation: Each data item can be explained by an additive linear combination of physically-meaningful basis components [22]. Numerous successful applications of NMF were reported in areas including text mining [30, 35], computer vision [25], and bioinformatics [3, 16, 8] among others.

NMF can be mathematically formulated in the following way. Given an input matrix $A \in \mathbb{R}^{m \times n}$, in which each element is nonnegative, and an integer $k < \min\{m, n\}$, NMF aims to find two factors $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ with nonnegative elements such that

$$A \approx WH. \tag{1.1}$$

The goodness of the approximation in Eqn. (1.1) can be measured in various ways including Frobenius norm, Kullback-Leibler divergence [23], and Bregman divergence [9, 24]. In this paper, we focus on the most common choice, which is Frobenius norm. Factors W and H are then found by solving the optimization problem:

$$\min_{W \geq 0, H \geq 0} f(W, H) = \frac{1}{2} \left\| A - WH \right\|_F^2 , \tag{1.2}$$

[*]An earlier version of this paper appeared in a conference proceeding: Jingu Kim and Haesun Park. Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM'08), Pisa, Italy, Dec. 2008. [19]

[†]School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: {jingu,hpark}@cc.gatech.edu

where the inequalities $W \geq 0$ and $H \geq 0$ mean that all the elements of W and H are nonnegative. The problem in Eqn. (1.2) is a non-convex optimization with respect to variables W and H, and it is shown that solving NMF is NP-hard [34]. Hence, one only hopes to find a local minimum in practice.

Many algorithms have been developed for solving Eqn. (1.2). More than a decade ago, Paatero and Tapper [28] initially proposed NMF (to be precise, *positive* matrix factorization in their terms), but their algorithms [28, 27] did not rigorously deal with the nonnegativity constraints. Lee and Seung popularized NMF with their seminal work [22]. Their multiplicative updating algorithm [23] has been one of the most commonly used for NMF, but some issues related to its performance [26, 15, 17] and problems with convergence [11] were reported. In [1], a simple algorithm that solves an unconstrained least squares problem at every iteration was devised, but it also has difficulty with convergence. Recently, several algorithms based on the alternating nonnegative least squares (ANLS) framework [26, 15, 17] were introduced with good performance. These algorithms possess a good convergence property because every limit point produced by the ANLS framework is a stationary point [26]. An algorithm that updates each column of W and each row of H per iteration, called the hierarchical alternating least squares (HALS) algorithm, was also introduced recently [6, 5]. The HALS algorithm was independently studied in [13], and a convergence property of HALS with a modification to stabilize the algorithm was proved in [10].

In this paper, we introduce a new and fast algorithm for NMF using the block principal pivoting method in the ANLS framework. Previous NMF algorithms using the ANLS framework include the active set method [17], the projected gradient method [26], and the projected quasi-Newton method [15]. The names tell how each algorithm solves the nonnegativity-constrained least squares (NNLS) subproblem. The projected gradient and the projected quasi-Newton methods apply techniques from unconstrained optimization with modifications for nonnegativity constraints. The active set method searches for the optimal active and passive sets of variables by maintaining working sets as candidates. At each iteration, an unconstrained least squares problem is solved, and the working sets are updated based on its result. The block principal pivoting method [31, 14], which we call an *active-set-like* method due to its similarity with the active set method, also follows this framework, but it overcomes a limitation of the active set method. Unlike the active set method, in which typically only one variable is exchanged between working sets, the block principal pivoting method allows the exchanges of multiple variables with a goal of finding the optimal active and passive sets faster. In this paper, we adopt the block principal pivoting method in NMF computation. We introduce ideas that improve the block principal pivoting method in the context of NMF and then build a new algorithm for NMF.

Thorough experimental comparisons among several NMF algorithms, including the one proposed in this paper, will follow the introduction of the new algorithm. The ANLS-based algorithms [26, 15, 17] and the HALS algorithm [5] appeared recently, and there has not been other papers with substantial comparisons such as the ones we offer in this paper. The comparisons were performed on text, image, and synthetic data sets, and the results are demonstrated showing the relative computational efficiency of various NMF algorithms. The proposed new algorithm exhibits state-of-the-art performance allowing only HALS to be comparable. We also show a condition under which the proposed method outperforms the HALS algorithm. This paper extends a previous conference version in [19]: The presentation of our algorithm is substantially improved, and more importantly, all the experimental results are updated with much more extensive comparisons and their interpretations.

The rest of this paper is organized as follows. In Section 2, the ANLS framework for NMF and related background materials are introduced. In Section 3, our new algorithm for NMF as well as its extensions are described. Implementation details and experimentation settings are shown in Section 4, and comparison results are demonstrated in Section 5. We conclude the paper in Section 6 with discussion.

2. Alternating Nonnegative Least Squares Framework for NMF. We begin by reviewing the ANLS framework for solving Eqn. (1.2). In the ANLS framework, the variables are divided into two groups in a straightforward manner, and then the two groups are updated in turn. The framework is summarized as follows.

1. Initialize $H \in \mathbb{R}^{m\times k}$ with nonnegative elements.
2. Repeat solving the following problems until a stopping criterion is satisfied:

$$\min_{W\geq 0} \left\| H^T W^T - A^T \right\|_F^2 \tag{2.1a}$$

   where H is fixed, and

$$\min_{H\geq 0} \left\| WH - A \right\|_F^2 \tag{2.1b}$$

   where W is fixed.
3. The columns of W are normalized to unit L_2-norm and the rows of H are scaled accordingly.

Alternatively, one may initialize W first and iterate Eqn. (2.1b) then Eqn. (2.1a). Note that each subproblem is an instance of the nonnegativity-constrained least squares (NNLS) problem. Although the original problem in Eqn. (1.2) is non-convex, the subproblems in Eqns. (2.1) are convex problems, for which optimal solutions can be found.

It is important to observe that the NNLS problems in Eqns. (2.1) have special characteristics. NMF is a dimension reduction algorithm, which is applied to high-dimensional data. The original dimension is very large, e.g., several thousands or more, and the reduced dimension is small, e.g., on the order of tens. Therefore, the coefficient matrix $W \in \mathbb{R}^{m\times k}$ in Eqn. (2.1b) is typically very long and thin ($m \gg k$), and the coefficient matrix $H^T \in \mathbb{R}^{n\times k}$ in Eqn. (2.1a) is also long and thin ($n \gg k$) in general. At the same time, the matrices W^T and H of unknown variables in Eqns. (2.1a) and (2.1b), respectively, are flat and wide for the same reason. Figure 2.1 illustrates these characteristics. These observations are critical in designing an efficient algorithm for the subproblems in Eqns. (2.1), and we will revisit this point in later sections.

For the convergence of any NMF algorithm based on the ANLS framework, it is important to find the optimal solutions of Eqns. (2.1) at each iteration. The ANLS framework is a two-block coordinate-descent algorithm, and a result by Grippo and Sciandrone [12] states that any limit point of the sequence of optimal solutions of two-block subproblems is a stationary point. Thus, the ANLS framework has a good optimization property that every limit point generated from the framework is a stationary point for the problem in Eq. (1.2). In non-convex optimization, most algorithms only guarantee the stationarity of the limit point. In the alternating least squares (ALS) algorithm [1], in contrast, the subproblems are solved as an unconstrained least squares problem, and then every negative element is set to zero. In

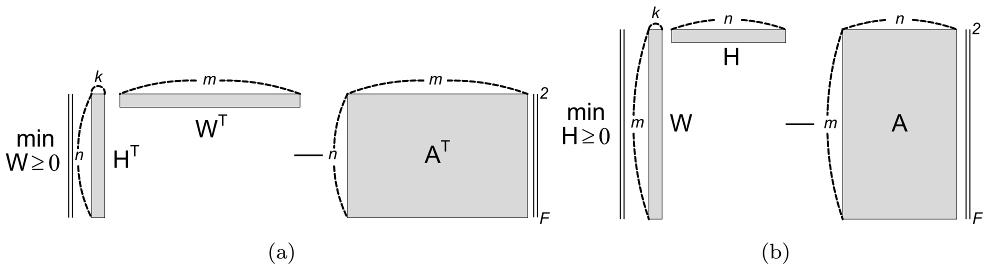

Fig. 2.1: Typical structure of the NNLS problems arising in NMF computation.

this case, it is difficult to analyze convergence because the algorithm updates, at each iteration, with a solution that is not optimal for the subproblem.

In order to design an NMF algorithm based on the ANLS framework, one has to devise a specific method to solve the subproblems in Eqns. (2.1). A classic algorithm for the NNLS problem is the active set method by Lawson and Hanson [21]. The active set method searches for the optimal active and passive sets by exchanging a variable between two working sets. Note that if we know the passive (i.e., strictly positive) variables of the solution in advance, then an NNLS problem can be easily solved by a simple unconstrained least squares procedure on the passive variables. Although the Lawson and Hanson's algorithm has been a standard for the NNLS problems [1], it is extremely slow when used for NMF in a straightforward way. Faster versions of the algorithm were recently developed by Bro and De Jong [2] and Van Benthem and Keenan [33], and Kim and Park utilized them in their NMF algorithm [17].

A major limitation of the active set method is that the variables of working sets are exchanged satisfying the nonnegativity of the solution vector while making sure that the objective function decreases after each iteration. As a result, typically only one variable is exchanged between working sets per iteration, slowing down the algorithm when the number of unknowns is large. Methods based on iterative optimization schemes such as the projected gradient method due to Lin [26] and the projected quasi-Newton method due to Kim et al. [15] are free of the above limitation. These algorithms are modified from techniques in unconstrained optimization by providing specialized rules to choose step length and projecting the solution to the feasible nonnegative orthant at every iteration.

The block principal pivoting method overcomes the limitation of the active set method in a different fashion. We now describe this method in detail.

3. Block Principal Pivoting Method.

In this section, we present the block principal pivoting method for the NNLS problems and the NMF algorithm based on the method. We will first review the block principal pivoting method that was developed for the NNLS problems with a single right-hand side [14] and then introduce methods that improve upon this to handle multiple right-hand sides efficiently.

[1]Lawson and Hanson's algorithm is adopted as a MATLAB function *lsqnonneg*.

3.1. NNLS with a single right-hand side vector. For the moment, let us focus on the NNLS problem with a single right-hand side vector formulated as

$$\min_{x \geq 0} \|Cx - b\|_2^2 , \tag{3.1}$$

where $C \in \mathbb{R}^{p \times q}$, $b \in \mathbb{R}^{p \times 1}$, $x \in \mathbb{R}^{q \times 1}$, and $p \geq q$. The subproblems in Eqns. (2.1) can be decomposed into several independent instances of Eqn. (3.1) with respect to each right-hand side vector. Thus, an algorithm for Eqn. (3.1) is a basic building block for an algorithm for Eqns. (2.1).

The Karush-Kuhn-Tucker optimality condition for Eqn. (3.1) is written as follows:

$$y = C^T Cx - C^T b \tag{3.2a}$$
$$y \geq 0 \tag{3.2b}$$
$$x \geq 0 \tag{3.2c}$$
$$x_i y_i = 0, \ i = 1, \cdots, q. \tag{3.2d}$$

We assume that matrix C has full column rank. In this case, matrix $C^T C$ is positive definite, and the problem in Eqn. (3.1) is strictly convex. Then, a solution x that satisfies the conditions in Eqns. (3.2) is the optimal solution of Eqn. (3.1). Problems in the form of Eqns. (3.2) are known as linear complementarity problems.

We divide the index set $\{1, \cdots, q\}$ into two groups F and G such that $F \cup G = \{1, \cdots, q\}$ and $F \cap G = \emptyset$. Let x_F, x_G, y_F, and y_G denote the subsets of variables with corresponding indices, and let C_F and C_G denote the submatrices of C with corresponding column indices. Initially, we assign

$$x_G = 0 \text{ and } y_F = 0.$$

That is, all the elements of x_G and y_F are set as zero. Then, x and y always satisfy Eqn. (3.2d) for any values of x_F and y_G. Now, we compute x_F and y_G using Eqn. (3.2a) and check whether the computed values of x_F and y_G satisfy Eqns. (3.2b) and (3.2c). The computation of x_F and y_G is done as follows:

$$x_F = \arg\min_{x_F} \|C_F x_F - b\|_2^2 \tag{3.3a}$$
$$y_G = C_G^T (C_F x_F - b). \tag{3.3b}$$

One can first solve for x_F in Eqn. (3.3a) and substitute the result into Eqn. (3.3b). We call the pair (x_F, y_G) a complementary basic solution if it is obtained by Eqns. (3.3).

If a complementary basic solution (x_F, y_G) satisfies $x_F \geq 0$ and $y_G \geq 0$, then it is called *feasible*. In this case, current x is the optimal solution of Eqn. (3.1), and the algorithm terminates. Otherwise, a complementary basic solution (x_F, y_G) is *infeasible*, and we need to update F and G by exchanging variables for which Eqn. (3.2b) or Eqn. (3.2c) does not hold. Formally, we define the following index set

$$V = \{i \in F : x_i < 0\} \cup \{i \in G : y_i < 0\}, \tag{3.4a}$$

and then a variable x_i with $i \in V$ is called an *infeasible variable*. Now, choose a non-empty subset $\hat{V} \subset V$. Then, F and G are updated by the following rules:

$$F = (F - \hat{V}) \cup (\hat{V} \cap G) \tag{3.5a}$$
$$G = (G - \hat{V}) \cup (\hat{V} \cap F). \tag{3.5b}$$

Algorithm 1 Block principal pivoting method for the NNLS problem with a single right-hand side vector

```
Input: $C \in \mathbb{R}^{p \times q}$ and $b \in \mathbb{R}^p$
Output: $x (\in \mathbb{R}^{q \times 1}) = \arg\min_{x \geq 0} \|Cx - b\|_2^2$
 1: Initialize $F = \emptyset$, $G = \{1, \cdots, q\}$, $x = 0$, $y = -C^T b$, $\alpha = 3$, and $\beta = q + 1$
 2: Compute $x_F$ and $y_G$ by Eqns. (3.3).
 3: while $(x_F, y_G)$ is infeasible do
 4:     Compute $V$ by Eqns. (3.4).
 5:     If $|V| < \beta$, set $\beta = |V|$, $\alpha = 3$, and $\hat{V} = V$.
 6:     If $|V| \geq \beta$ and $\alpha \geq 1$, set $\alpha = \alpha - 1$ and $\hat{V} = V$.
 7:     If $|V| \geq \beta$ and $\alpha = 0$, set $\hat{V}$ by Eqn. (3.6).
 8:     Update $F$ and $G$ by Eqns. (3.5).
 9:     Update $x_F$ and $y_G$ by Eqns. (3.3).
10: end while
```

The size $|\hat{V}|$ represents how many variables are exchanged per iteration. If $|\hat{V}| > 1$, then the algorithm is called a *block* principal pivoting algorithm; if $|\hat{V}| = 1$, then the algorithm is called a *single* principal pivoting algorithm. The active set method can be understood as an instance of single principal pivoting algorithms. The algorithm repeats this procedure until the number of infeasible variables (i.e., $|\hat{V}|$) becomes zero.

In order to speed up the search procedure, one usually uses $\hat{V} = V$, which we call the *full exchange rule*. The full exchange rule means that we exchange all variables of F and G that do not satisfy Eqns. (3.2), and the rule accelerates computation by reducing the number of iterations required until termination. However, contrary to the active set method in which the variable to exchange is carefully selected to reduce the objective function, the full exchange rule may lead to a cycle and fail to find an optimal solution although it occurs rarely. To ensure finite termination, we need to employ a backup rule, which uses the following exchange set for Eqns. (3.5):

$$\hat{V} = \{i : i = \max\{i \in V\}\}. \tag{3.6}$$

The backup rule, where only the infeasible variable with the largest index is exchanged, is a single principal pivoting rule. This simple exchange rule guarantees a finite termination: Assuming that matrix C has full column rank, the exchange rule in Eq. (3.6) returns the solution of (3.2) in a finite number of iterations [14].

Combining the full exchange rule and the backup rule, the block principal pivoting method for the NNLS problem with a single right-hand side is obtained as summarized in Algorithm 1. Because the backup rule is much slower than the full exchange rule, it is used only if the full exchange rule does not work well. Finite termination of Algorithm 1 is achieved by controlling the number of infeasible variables. In Algorithm 1, variable α is used as a buffer on the number of the full exchange rules that may be tried. If the full exchange rule increases the number of infeasible variables, then α is reduced by one. Once α becomes zero, the backup rule is used until it makes the number of infeasible variables smaller than the lowest value achieved so far, which is stored in β. This has to occur in a finite number of steps because the backup rule has a finite termination property. As soon as the backup rule achieves a new lowest number of infeasible variables, we return to the full exchange rule. We used three as the default value of α, which means that we can try the full exchange rule up to three times until it reduces the number of infeasible variables. Since the number

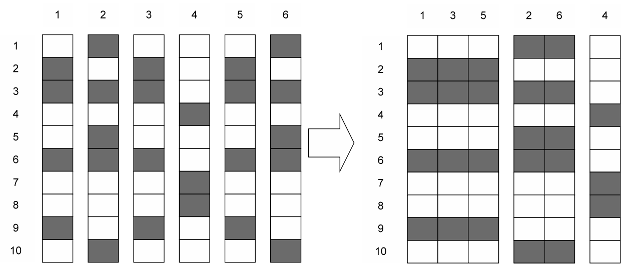

Fig. 3.1: An example of the grouping of right-hand side vectors when $q = 10$ and $r = 6$. Dark cells indicate variables with indices in F, which need to be computed by Eqn. (3.8). By grouping the columns that have a common F set, i.e., columns $\{1, 3, 5\}$,$\{2, 6\}$ and $\{4\}$, we can avoid redundant computation for Cholesky factorization in solving Eqn. (3.8).

of infeasible variables is systematically reduced, the algorithm terminates in a finite number of steps.

The block principal pivoting method was shown very efficient for the NNLS problems [31, 4]. In principle, the computational cost of the block principal pivoting method would depend on how often the full exchange rule fails so that the backup rule has to be activated. In our extensive tests in this paper, the backup rule appearance was not observed, suggesting that the full exchange rule works well in practice. Since the full exchange rule allows the exchanges of multiple variables between F and G, the block principal pivoting method becomes much faster than the active set algorithm, as we report in Section 5.

One might relate the two sets, F and G, of the block principal pivoting method to the passive and active sets in the active set method. However, they are not necessarily identical. In the active set method, the solution is sought while satisfying the condition $x \geq 0$, so a variable x_i in which i is in the passive set is required to satisfy $x_i \geq 0$ in every iteration. In the block principal pivoting method, on the other hand, a variable x_i with $i \in F$ can be of any sign except for the final iteration. Therefore, the block principal pivoting method does not require an initial solution with $x \geq 0$ while the active set method does.

3.2. NNLS with multiple right-hand side vectors. Now suppose we need to solve the following NNLS problem:

$$\min_{x \geq 0} \|CX - B\|_F^2 , \tag{3.7}$$

where $C \in \mathbb{R}^{p\times q}$, $B \in \mathbb{R}^{p\times r}$, and $X \in \mathbb{R}^{q\times r}$. It is possible to simply run Algorithm 1 for each right-hand side vector $b_1, \cdots, b_r$ where $B = [b_1, \cdots, b_r]$ since the columns of X do not depend on each other. However, this approach is not computationally efficient, and we will explain how we obtain an efficient algorithm for the multiple right-hand side case using the ideas from [2] and [33] in the context of the block principal pivoting method.

In Algorithm 1, the major computational burden is from the need to compute x_F and y_G as shown in Eqns. (3.3). We can solve Eqn. (3.3a) by a normal equation

$$C_F^T C_F x_F = C_F^T b, \tag{3.8}$$

Algorithm 2 Block principal pivoting method for the NNLS with multiple right-hand side vectors. X_{F_j} and Y_{G_j} represents the subsets of j-th column of X and Y indexed by F_j and G_j, respectively.

```
Input: $C \in \mathbb{R}^{p\times q}$,$B \in \mathbb{R}^{p\times r}$
Output: $X(\in \mathbb{R}^{q\times r}) = \arg\min_{x\geq 0} \|CX - B\|_F^2$
 1: Compute $C^T C$ and $C^T B$.
 2: Initialize $F_j = \emptyset$ and $G_j = \{1, \cdots, q\}$ for all $j \in \{1, \cdots, r\}$. Set $X = 0$, $Y = -C^T B$, $\alpha(\in \mathbb{R}^r) = 3$, and $\beta(\in \mathbb{R}^r) = q + 1$.
 3: Compute $X_{F_j}$ and $Y_{G_j}$ for all $j \in \{1, \cdots, r\}$ by Eqns. (3.3) using column grouping.
 4: while any $(X_{F_j}, Y_{G_j})$ is infeasible do
 5:     Find the indices of columns in which the solution is infeasible: $I = \{j : (X_{F_j}, Y_{G_j}) \text{ is infeasible}\}$.
 6:     Compute $V_j$ for all $j \in I$ by Eqns. (3.4).
 7:     For all $j \in I$ with $|V_j| < \beta_j$, set $\beta_j = |V_j|$, $\alpha_j = 3$ and $\hat{V}_j = V_j$.
 8:     For all $j \in I$ with $|V_j| \geq \beta_j$ and $\alpha_j \geq 1$, set $\alpha_j = \alpha_j - 1$ and $\hat{V}_j = V_j$.
 9:     For all $j \in I$ with $|V_j| \geq \beta_j$ and $\alpha_j = 0$, set $\hat{V}_j$ by Eqn. (3.6).
10:     Update $F_j$ and $G_j$ for all $j \in I$ by Eqns. (3.5).
11:     Update $X_{F_j}$ and $Y_{G_j}$ for all $j \in I$ by Eqns. (3.3) using column grouping.
12: end while
```

and Eqn. (3.3b) can be rewritten as

$$y_G = C_G^T C_F x_F - C_G^T b. \tag{3.9}$$

We need $C_F^T C_F$, $C_F^T b$, $C_G^T C_F$, and $C_G^T b$ for solving Eqns. (3.8) and (3.9), and these matrices and vectors vary throughout iterations because F and G are updated in each iteration.

The improvements are closely related with the observations mentioned in Section 2. First, note that for the NNLS problems arising from NMF, matrix C is typically very long and thin, and computing $C_F^T C_F$, $C_F^T b$, $C_G^T C_F$, and $C_G^T b$ is computationally expensive. However, we can compute $C^T C$ and $C^T B$ in the beginning and reuse them in later iterations. As $C_F^T C_F$, $C_F^T b$, $C_G^T C_F$, and $C_G^T b$ can be retrieved as submatrices of $C^T C$ and $C^T B$ for any F and G, we can avoid expensive matrix-matrix multiplications. Since the column size of C is small for the NNLS problems arising from NMF, storage needed for $C^T C$ and $C^T B$ will also be small. This idea is also applicable to the single right-hand side case, but its impact is more dramatic in the multiple right-hand side case.

Another improvement comes from the fact that matrix X is typically flat and wide in the NNLS problems for NMF. Suppose we simultaneously run Algorithm 1 for many right-hand side vectors. In each iteration, we have index sets F_j and G_j for each column $j \in \{1, \cdots, r\}$, and we must compute x_{F_j} and y_{G_j} using Eqns. (3.8) and (3.9). The idea is to find groups of columns that share the same index sets F_j and G_j and solve Eqn. (3.8) for the columns in the same group. By doing so, we avoid repeated computation for Cholesky factorization in solving Eqn. (3.8). Figure 3.1 illustrates this grouping idea. Note that if X is flat and wide, which is the case for the NNLS problems in NMF, more columns are likely to share their index sets F_j and G_j, allowing us to obtain bigger speed-up. We summarize the improved block principal pivoting method for multiple right-hand sides in Algorithm 2.

3.3. NMF based on ANLS with block principal pivoting. The block principal pivoting method combined with the improvements is quite efficient for the NNLS problems with multiple right-hand sides. To compute NMF, we use Algorithm 2 to solve the subproblems in Eqns. (2.1). Implementation issues such as a stopping criterion are discussed in Section 4.

3.4. Block principal pivoting method for sparse and regularized NMF. Our new algorithm, which we described so far for the NMF formulation in Eqn. (1.2), can easily be extended to other variations of NMF. In some applications, it is important to obtain sparse factors in NMF. For example, when sparsity is imposed on factor H, the resulting NMF can be used as a clustering algorithm [16, 18]. NMF that imposes sparsity on factor H [16] can be formulated as

$$\min_{W \geq 0, H \geq 0} \left\{ \frac{1}{2} \|A - WH\|_F^2 + \eta \|W\|_F^2 + \beta \sum_{j=1}^{n} \|H(:,j)\|_1^2 \right\}. \tag{3.10}$$

We can solve Eqn. (3.10) by solving the following subproblems alternatingly:

$$\min_{W \geq 0} \left\| \begin{pmatrix} H \\ \sqrt{2\eta} I_k \end{pmatrix} W^T - \begin{pmatrix} A^T \\ 0_{k \times m} \end{pmatrix} \right\|_F^2$$

where I_k is a $k \times k$ identity matrix and $0_{k \times m}$ is a zero matrix of size $k \times m$, and

$$\min_{H \geq 0} \left\| \begin{pmatrix} W \\ \sqrt{2\beta} e_{1 \times k} \end{pmatrix} H - \begin{pmatrix} A \\ 0_{1 \times n} \end{pmatrix} \right\|_F^2$$

where $e_{1 \times k}$ is a row vector having every element as one and $0_{1 \times n}$ is a zero vector with length n. Similarly, one can impose sparsity on factor W or on both factors. For more details of promoting sparsity using Eqn. (3.10), see [16]. On the other hand, when W and H^T are not necessarily of full column rank, a regularized formulation may be considered [17, 29]:

$$\min_{W \geq 0, H \geq 0} \left\{ \frac{1}{2} \|A - WH\|_F^2 + \alpha \|W\|_F^2 + \beta \|H\|_F^2 \right\}. \tag{3.11}$$

As shown in [17], Eqn. (3.11) can also be recast into the ANLS framework: One iterates solving

$$\min_{W \geq 0} \left\| \begin{pmatrix} H^T \\ \sqrt{2\alpha} I_k \end{pmatrix} W^T - \begin{pmatrix} A^T \\ 0_{k \times m} \end{pmatrix} \right\|_F^2$$

and

$$\min_{H \geq 0} \left\| \begin{pmatrix} W \\ \sqrt{2\beta} I_k \end{pmatrix} H - \begin{pmatrix} A \\ 0_{k \times n} \end{pmatrix} \right\|_F^2$$

where $0_{k \times n}$ is a zero matrix of size $k \times n$ until convergence. The proposed new block principal pivoting method is applicable to the problems shown in Eqns. (3.10) and (3.11) since both of them can be recast into the ANLS framework, whose subproblems are the NNLS problems with multiple right-hand sides.

4. Implementation and Data Sets.

We describe the details of our implementation and data sets used for comparisons.

4.1. NMF algorithms compared. We compared the following algorithms for NMF. Due to space limitations, we do not present the details of other algorithms but only refer to the papers in which they are presented.

1. (ANLS-BPP) ANLS with the block principal pivoting method proposed in this paper
2. (ANLS-AS) ANLS with Kim and Park's active set method [17]
3. (ANLS-PGRAD) ANLS with Lin's projected gradient method [26]
4. (ANLS-PQN) ANLS with Kim et al.'s projected quasi-Newton method [15]
5. (HALS) Cichocki and Phan's hierarchical alternating least squares algorithm [6, 5]
6. (MU) Lee and Seung's multiplicative updating algorithm [23]
7. (ALS) Berry et al.'s alternating least squares algorithm [1]

We implemented our ANLS-BPP method in MATLAB. For ANLS-AS, we implemented two different versions. The first one is using a grouping idea for multiple right-hand sides as described in [33, 17], and we refer to this implementation as ANLS-AS-GROUP. Alternatively, the ANLS-AS method can be implemented by solving the NNLS problems with a single right-hand side separately using the updating of the Cholesky factor. We refer to this case as ANLS-AS-UPDATE. For the ANLS-PGRAD method, we used the MATLAB code written by its author, and for ANLS-PQN, we refined the code written by its authors since it was not carefully optimized for high-dimensional data, in which NMF is typically used. Our refined version of ANLS-PQN is much faster than the original one by the authors, and thus we used the refined version in our experiments. In [5], two versions of the HALS algorithm are introduced, and we used a faster version that updates all columns of W (rows of H) before proceeding to update H (W). We implemented MU and ALS, which are straightforward to implement. The implementations used in our experiments will become available in the webpages of the first[2] and the second[3] authors.

In all the algorithms, once we obtain $H^{(i)}$ and $W^{(i)}$ as the result of the i-th iteration, we used them as initial values of the $(i+1)$-th iteration. As iteration progress, the solutions $H^{(i)}$ and $W^{(i)}$ do not change much, and therefore starting from the result of the previous iteration is a good warm start for the next iteration. In particular, for the block principal pivoting method, warm starting means that only the partitioning of the indices into sets F and G from the result of the previous iteration is used instead of specific numerical values.

4.2. Stopping criterion. When an iterative algorithm is executed in practice, one needs a criterion to stop iterations. In the case of NMF, a stopping criterion can be designed to check whether a local minimum of the objective function has been reached. In practice, one usually checks whether a point is stationary, which can be checked by the following criterion suggested by Lin [26].

According to the KKT condition, (W, H) is a stationary point of Eqn. (1.2) if and only if

$$W \geq 0, H \geq 0, \tag{4.1a}$$

$$\nabla f_W = \partial f(W,H)/\partial W \geq 0, \nabla f_H = \partial f(W,H)/\partial H \geq 0, \tag{4.1b}$$

$$W .* \nabla f_W = 0, H .* \nabla f_H = 0, \tag{4.1c}$$

[2]http://www.cc.gatech.edu/~jingu

[3]http://www.cc.gatech.edu/~hpark

where $.*$ represents element-wise multiplications. Defining the projected gradient $\nabla^p f_W$ as

$$(\nabla^p f_W)_i \equiv \begin{cases} (\nabla f_W)_i & \text{if } (\nabla f_W)_i < 0 \text{ or } W_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

and $\nabla^p f_H$ similarly, the conditions in Eqns. (4.1) can be rephrased as

$$\nabla^p f_W = 0 \text{ and } \nabla^p f_H = 0.$$

We use the norm of the projected gradients defined as

$$\Delta = \sqrt{\|\nabla^p f_W\|_F^2 + \|\nabla^p f_H\|_F^2}. \tag{4.2}$$

Using this definition, the stopping criterion is

$$\frac{\Delta}{\Delta_0} \le \epsilon, \tag{4.3}$$

where Δ_0 is the value of Δ using the initial values of W and H, and ϵ is a tolerance value to choose. This criterion has been useful in determining when to stop iterations, but we found some issues with it in our experiments. We explain them in Section. 5.2.

4.3. Data sets. We have used four real-world data sets for our comparison, and their information is shown in Table 5.1. Among them, two text data sets are in sparse format. The Topic Detection and Tracking 2 (TDT2) text corpus[4] contains news articles from various sources such as NYT, CNN, and VOA in 1998. The corpus is manually labeled across 100 different topics, and it has been widely used for text mining research. From the corpus, we randomly selected 40 topics in which the number of articles in each topic is greater than 10. By counting the term frequency of each term in each document, we obtained a matrix of size $19,009 \times 3,087$. The 20 Newsgroups data set[5] is a collection of newsgroup documents in 20 different topics. We used a term-document matrix of size $26,214 \times 11,314$[6].

Two image data sets in Table 5.1 are in dense format. The facial image database by AT&T Laboratories Cambridge[7] contains 400 facial images of 40 different people with 10 images per person. Each facial image has 92×112 pixels in 8-bit gray level. The resulting matrix was of size $10,304\times400$. The CMU PIE database[8] is collection of human face under different poses, illumination conditions, and expressions. A matrix of size $4,096 \times 11,554$, from a resized version in 64×64 pixels[9], was used for our executions.

We also employed synthetic data sets for additional observations. The details of synthetic data sets are described in Section 5.3.

5. Comparison Results .

All experiments were executed in MATLAB on a Linux machine with a 2.66GHz Intel Quad-core processor and 6GB memory. The multi-threading option of MATLAB was disabled. In all the executions, all the algorithms were provided with the same initial values.

[4]`http://projects.ldc.upenn.edu/TDT2/`
[5]`http://people.csail.mit.edu/jrennie/20Newsgroups/`
[6]`http://www.zjucadcg.cn/dengcai/Data/TextData.html`
[7]`http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`
[8]`http://www.ri.cmu.edu/projects/project_418.html`
[9]`http://www.zjucadcg.cn/dengcai/Data/FaceData.html`

Table 5.1: Data sets, their sizes, the sparsity of solutions, and the grouping effects of the ANLS-AS-GROUP and the ANLS-BPP methods. Sparsities are calculated as the proportions of zero elements in each factor. See text for the description of grouping effects.

Data set	Size and Format	k	Sparsity (%)		Grouping effect (%)			
					BPP		AS-GROUP	
			W	H	W	H	W	H
TDT2	19,009×3,087 Sparse (99.58% sparsity)	10	53.1	46.4	94.3	76.2	94.4	76.2
		20	70.6	60.2	54.1	12.4	53.6	13.6
		80	85.4	77.7	40.7	3.1	28.6	3.2
		160	89.4	84	39.4	2.8	21.3	2.7
20 Newsgroups	26,214×11,314 Sparse (99.66% sparsity)	10	58.3	44.6	97.1	95.8	97	95.7
		20	67.5	53.3	48.0	19.4	53.3	48.5
		80	80.5	73.1	9.6	0.9	7.6	1.2
		160	86.8	78.8	7.4	0.7	4.6	0.8
ATNT	10,304×400 Dense	10	14.1	9.6	97.6	87.1	97.6	87.1
		20	20.8	18.8	66.4	20.3	66.4	21.8
		80	33.9	41.1	0.6	0.7	1.6	1.7
		160	32.7	61.3	0.5	0.5	0.6	0.7
PIE 64	4,096×11,554 Dense	10	27.4	14.5	91.7	96.6	91.8	96.8
		20	38.8	17.3	44.2	68.9	46.6	71.1
		80	58.7	21.3	0.8	1.5	1.4	3.3
		160	63.4	28.1	0.5	0.5	0.5	1.3

5.1. Comparison of the active set and the block principal pivoting methods. We first report observations about the ANLS-AS and the ANLS-BPP methods. As mentioned before, we implemented two versions of ANLS-AS: ANLS-AS-GROUP is based on a column grouping strategy, as used in Kim and Park [17], and ANLS-AS-UPDATE solves each right-hand side vector separately using the updating of the Cholesky factor. We compared the performance of the two versions, and we are also interested in how they perform compared to ANLS-BPP.

In Table 5.1, we present the results from the execution of ANLS-AS-GROUP and ANLS-BPP. Both methods were executed with 5 different random initializations for 100 iterations, and average results are shown in the table. Sparsities are calculated as the proportions of zero elements in each factor after 100 iterations. The grouping effect (GE) is calculated as

$$\text{GE} = \frac{\#\text{ of Cholesky factorizations that are omitted thanks to column grouping}}{\#\text{ of systems of equations needed to be solved}}.$$

Both the numerator and the denominator are summed over the 100 iterations. When GE is close to 100%, it means that significant savings are achieved; when GE is close to zero, there are only little savings. In Table 5.1, it can be seen that significant savings from grouping were observed when $k = 10$ and $k = 20$ whereas only limited savings were observed when $k = 80$ and $k = 160$.

Now let us see Figures 5.1 and 5.2 where we present the average execution time of each iteration and their accumulation. Unlike the ANLS-PGRAD and the ANLS-PQN methods, both the ANLS-AS and the ANLS-BPP methods solve the NNLS subproblems in each iteration exactly. Hence, when the same initial values are provided, solutions after each iteration from these methods are the same up to numerical rounding errors. It is therefore enough to observe the amount of time spent in each iteration for the purpose of comparing these methods. In Figure 5.1, which shows

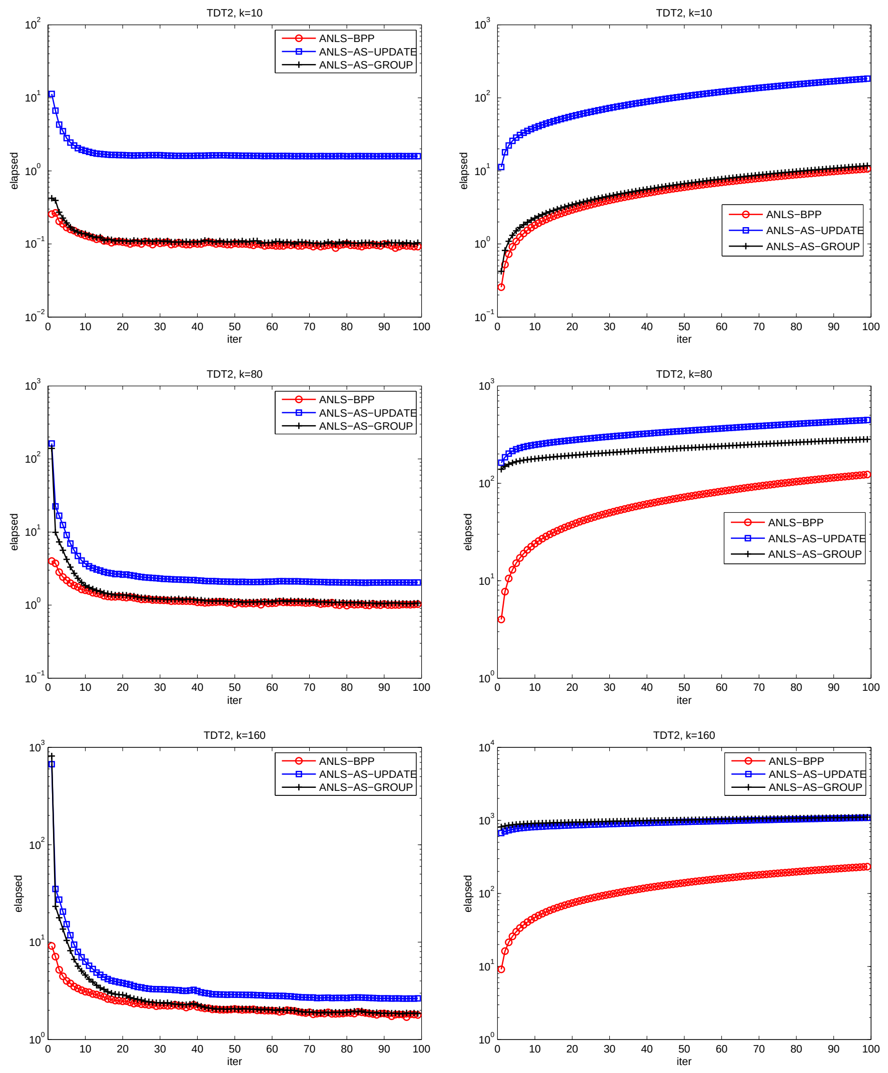

Fig. 5.1: Comparison of the active set (ANLS-AS) and the block principal pivoting (ANLS-BPP) methods on the TDT2 text data set. The left column shows the execution time of each iteration, and the right column shows cumulative execution time. Average of 5 different random initializations are shown. Top row: $k = 10$, middle row: $k = 80$, bottom row: $k = 160$.

results on the TDT2 text data set, it can be seen that ANLS-AS-UPDATE remains slower than ANLS-AS-GROUP for various k values. On the other hand, in Figure 5.2, which shows results on the ATNT image data set, the trend is a little different: While ANLS-AS-UPDATE performed slower than ANLS-AS-GROUP for small k values, ANLS-AS-UPDATE became faster than ANLS-AS-GROUP for large k values. Partial explanations of this difference are as follows.

For small k values, ANLS-AS-GROUP and ANLS-BPP achieved significant sav-

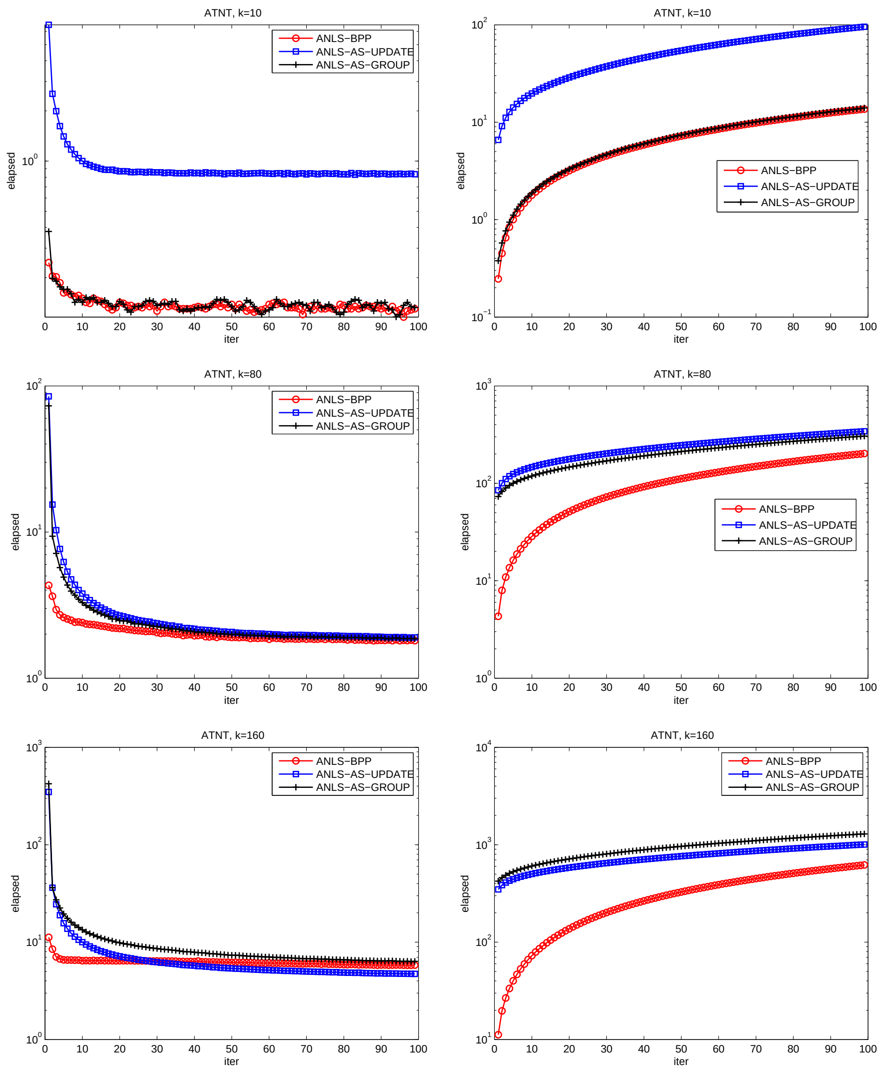

Fig. 5.2: Comparison of the active set (ANLS-AS) and the block principal pivoting (ANLS-BPP) methods on the ATNT image data set. The left column shows the execution time of each iteration, and the right column shows cumulative execution time. Average of 5 different random initializations are shown. Top row: $k = 10$, middle row: $k = 80$, bottom row: $k = 160$.

ings by grouping as can be seen from Table 5.1. Consequently, in the $k = 10$ cases of Figures 5.1 and 5.2, ANLS-AS-GROUP was significantly faster than ANLS-AS-UPDATE. For large k values, in contrast, it is generally expected that using the updating of the Cholesky factor is beneficial. For the ATNT image data set, this was the case as can be seen from the fact that ANLS-AS-UPDATE outperformed ANLS-AS-GROUP for $k = 160$. For the TDT2 text data set, however, nontrivial savings from grouping was observed even when $k = 160$. Hence, ANLS-AS-GROUP remained faster than ANLS-AS-UPDATE for all k values.

It is important to note that ANLS-BPP is either as fast as ANLS-AS-GROUP or is significantly faster than both the ANLS-AS methods. For the $k = 160$ case on the ATNT data set, the iteration cost of ANLS-AS-UPDATE becomes smaller than that of ANLS-BPP after many iterations; however, the cumulative cost of ANLS-BPP is still much smaller than that of ANLS-AS-UPDATE. This observation suggests that a hybrid method can be potentially investigated, where we employ ANLS-BPP but replace ANLS-BPP with ANLS-AS-UPDATE in later iterations only for large k values. On the other hand, since the ANLS-AS and the ANLS-BPP methods typically converge within 20 to 30 iterations, the benefit of the hybrid method is not expected to be significant in practice.

Results on the 20 Newsgroups and the PIE 64 data sets showed similar trends, and we do not present them here.

5.2. Comparison with other algorithms. We now show comparison results of ANLS-BPP with all other methods that are listed in Section 4.1. The computational cost of each iteration and the objective function reduced after each iteration in those algorithms are generally different, so a fair way to compare them would be observing "how well an algorithm minimizes the objective function in how much computation time." Figures 5.3 and 5.4 show the average execution results of 5 different random initializations. We have recorded the relative objective value $\left(\left\|A - W^{(i)}H^{(i)}\right\|_F / \|A\|_F\right)$ at the end of each iteration, and the time spent to compute the objective value is excluded from the execution time. One execution result with relative objective values measured at discrete time points gives us a piecewise-linear function, and we averaged piecewise-linear functions for different random initializations to plot in the figures. Because the first several iterations of ANLS-AS took too much time, the results of ANLS-AS are not shown.

The results on the TDT2 and the 20 Newsgroups data sets are shown in Figure 5.3. When $k = 10$, most algorithms except ANLS-PQN tended to quickly converge. When $k = 80$ or $k = 160$, it becomes clear that ANLS-BPP and HALS are the best performing algorithms among the ones we tested. The ANLS-PGRAD, the ANLS-PQN, and the MU algorithms showed a trend of convergence although the convergence was slow, but the ALS algorithm showed difficulty in convergence. Among ANLS-BPP and HALS, while HALS generally appeared to converge faster in early iterations, ANLS-BPP was the only algorithm with comparable performance.

In Figure 5.4, execution results on the ATNT and the PIE 64 data sets are presented. Similarly to previous data sets, ANLS-PGRAD, ANLS-PQN, and MU showed slow convergence, and ALS was unpredictable as the relative objective value fluctuated up and down without converging to a small value. In the $k = 160$ cases, the results of ALS are not shown because it was not able to reduce the objective value in the range of the plot. Again, it can be seen that the ANLS-BPP and the HALS algorithms are the best performing ones. Among the two, HALS showed faster convergence in the ATNT data set whereas ANLS-BPP outperformed HALS in the PIE 64 data set.

In Table 5.2, relative norms of the projected gradient defined in Eq. (4.3) after executions for specified durations are shown. It can be seen that ANLS-BPP appeared very successful in minimizing this criterion. A caveat here is that a smaller value of $\frac{\Delta}{\Delta_0}$ does not necessarily imply a smaller objective value or vice versa, as can be seen from the fact that HALS sometimes produced high values of $\frac{\Delta}{\Delta_0}$ although it often showed one of the best performance in terms of the objective value. A partial

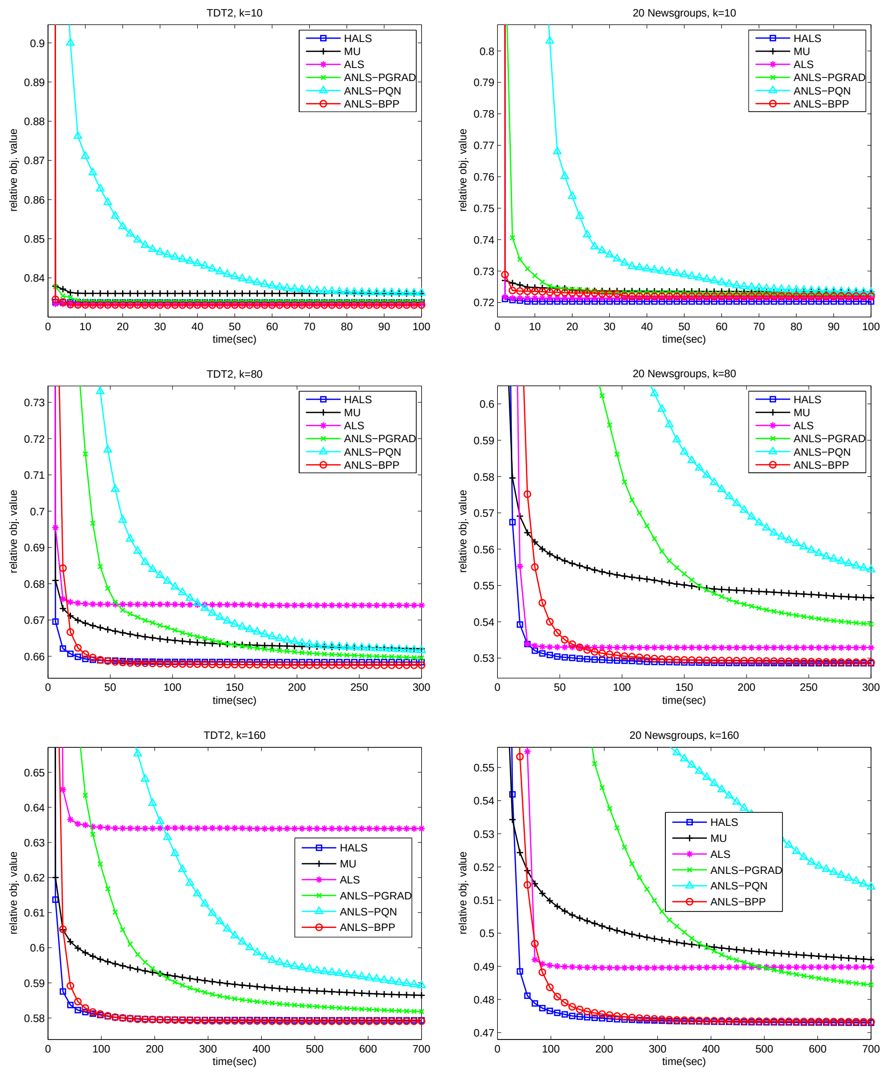

Fig. 5.3: Relative objective value ($\left\|A - W^{(i)}H^{(i)}\right\|_F / \|A\|_F$) vs. execution time on the TDT2 and the 20 Newsgroups text data sets. Average results of 5 different random initializations are shown. Left column: TDT2, right column: 20 Newsgroups, top row: k = 10, middle row: k = 80, bottom row: k = 160. See text for more details.

explanation about these results is given as follows[10]. Note that the diagonal scaling of W and H does not affect the quality of approximation: For a diagonal matrix $D \in \mathbb{R}^{k\times k}$ with positive diagonal elements, $WH = WD^{-1}DH$. However, the norm of the projected gradients in Eq. (4.2) is affected by a diagonal scaling: It is easy to check that $\left(\frac{\partial f}{\partial(WD^{-1})}, \frac{\partial f}{\partial(DH)}\right) = \left(\left(\frac{\partial f}{\partial W}\right) D, D^{-1}\left(\frac{\partial f}{\partial H}\right)\right)$. Hence, two solutions that are only different up to a diagonal scaling have the same objective function value,

[10]We are indebted to an anonymous reviewer for this explanation about the results in Table 5.2.

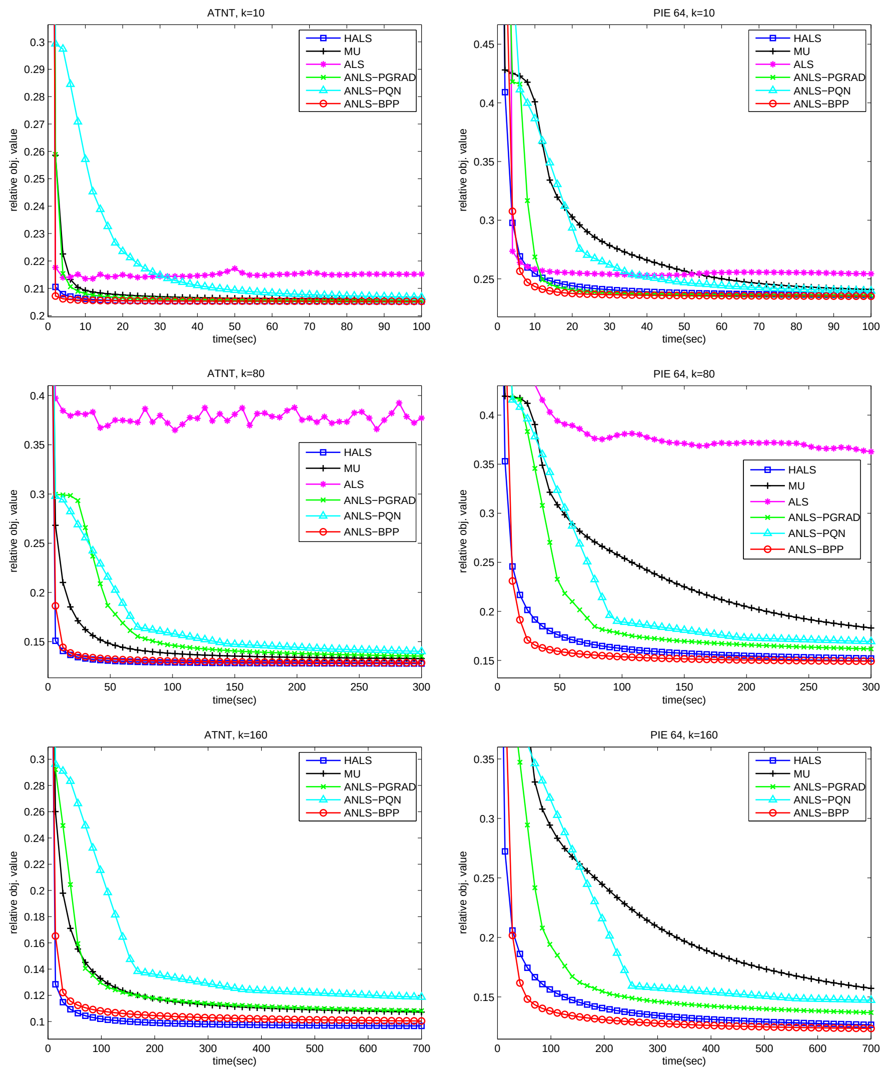

Fig. 5.4: Relative objective value ($\left\|A - W^{(i)}H^{(i)}\right\|_F / \left\|A\right\|_F$) vs. execution time on the ATNT and the PIE 64 image data sets. Average results of 5 different random initializations are shown. Left column: ATNT, right column: PIE 64, top row: k = 10, middle row: k = 80, bottom row: k = 160. See text for more details.

but they can be measured differently in terms of the norm of the projected gradients. In particular, the solution from the HALS algorithm is typically unbalanced having large elements in W and small elements in H. This can be a reason for the relatively poor evaluation of the HALS algorithm in Table 5.2. Ho [13] considered including a normalization step before computing Eq. (4.2) to avoid this issue.

Among the ANLS-based algorithms that have been actively studied recently, results in Figures 5.3 and 5.4 demonstrate that ANLS-BPP is clearly the best. Among all the NMF algorithms, the HALS algorithm showed very promising behavior as

Table 5.2: The relative norm of the projected gradient (i.e., Δ/Δ_0) after executions of specified amounts of time.

Data set	k	time	ANLS			HALS	MU	ALS
			BPP	PGRAD	PQN			
TDT2	10	100	2.64e-17	1.93e-05	3.82e-04	0.0015	0.0057	3.51e-04
	80	300	1.35e-07	4.33e-05	8.29e-05	8.04e-05	0.0013	1.61e-04
	160	700	2.40e-06	1.61e-05	9.03e-05	3.04e-05	7.47e-04	9.65e-05
20 Newsgroups	10	100	2.45e-08	8.71e-05	0.0020	0.0051	0.0058	0.0025
	80	300	4.05e-05	1.09e-04	0.0013	1.54e-04	0.0012	2.34e-04
	160	700	1.14e-05	8.40e-05	8.52e-04	5.28e-05	6.48e-04	1.33e-04
ATNT	10	100	4.12e-05	1.98e-04	0.0022	3.4601	0.0553	31.6
	80	300	2.56e-04	0.0018	0.0065	0.865	0.0136	57.6
	160	700	4.18e-04	0.0015	0.0059	0.533	0.0115	71.6
PIE 64	10	100	6.43e-04	7.42e-04	0.0065	19.3	0.437	114
	80	300	7.46e-04	0.0034	0.0045	4.32	0.0584	140
	160	700	0.0010	0.0043	0.0050	3.24	0.0369	180

it outperformed ANLS-BPP in some cases. In the following subsection, we further investigated the two algorithms using synthetic data sets.

5.3. ANLS-BPP and HALS on synthetic data sets. In order to further understand the behavior of the ANLS-BPP and the HALS algorithms, we performed experiments using synthetic data sets. Using $m = 10,000$, $n = 2,000$, and $k = 160$, we created factor matrices $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ having 50%, 90%, and 95% sparsities. We then multiplied the factors to obtain $A = WH$ upon which the ANLS-BPP and the HALS algorithms were executed.

Figure 5.5 shows the results. Our expectation was that the sparser the factors are, the better ANLS-BPP would perform compared to HALS: If the factors are sparse, the ANLS-BPP method only needs to solve for a small number of nonzero elements in W and H matrices whereas the HALS method still needs to update all the elements of W and H in each iteration. In the top row of Figure 5.5, when the sparsity of the original factors increases, ANLS-BPP method showed noticeably faster convergence than HALS. Relevant information is shown in the bottom row: The sparsity pattern of W and H obtained by ANLS-BPP quickly became close to that of the original factors that were used to create data sets, and the pattern changed only little as iteration progressed. When W and H are sparse, the cost of each iteration of ANLS-BPP decreases since only the nonzero elements needs to be updated.

6. Conclusions and Discussion. In this paper, a new algorithm for computing NMF based on the ANLS framework is proposed. The new algorithm is built upon the block principal pivoting method for the NNLS problem. The method allows exchanges of multiple variables between working sets with a goal to reach the final partitioning into the active and passive sets quickly. We improved the method to handle the multiple right-hand side case of the NNLS problems efficiently. The newly constructed algorithm inherits the convergence theory of the ANLS framework, and it can easily be extended to other constrained NMF formulations such as sparse or regularized NMF. Thorough experimental comparisons with recently developed NMF algorithms were performed using both real-world and synthetic data sets. The proposed algorithm demonstrated state-of-the-art performance allowing only the hierarchical alternating

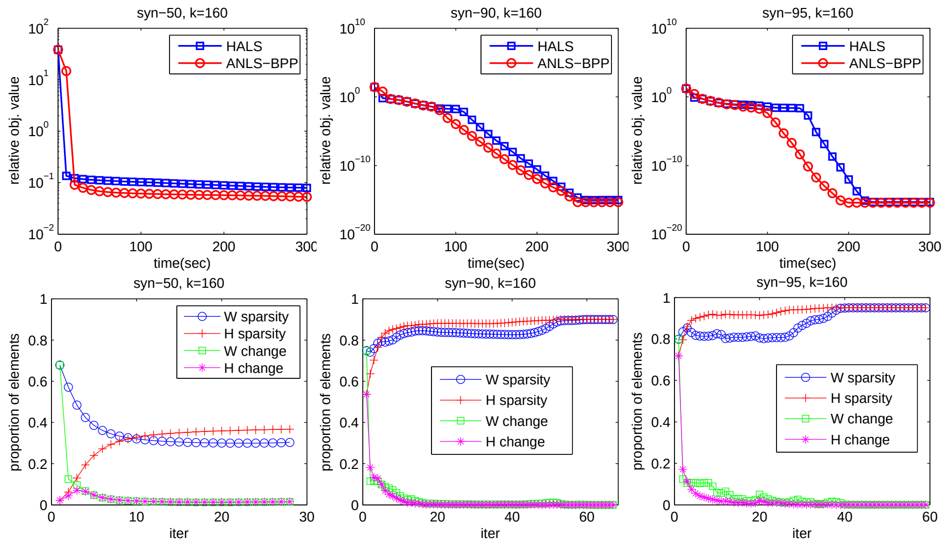

Fig. 5.5: Execution results on synthetic data sets with factors of different sparsities. The top row shows relative objective values ($\left\|A - W^{(i)}H^{(i)}\right\|_F / \|A\|_F$) with respect to execution time, and the bottom row shows sparsity patterns of $W^{(i)}$ and $H^{(i)}$ that are obtained from ANLS-BPP. 'W sparsity' and 'H sparsity' show the proportions of zero elements of $W^{(i)}$ and $H^{(i)}$, respectively, and 'W change' and 'H change' show the proportions of elements that switched between zero and nonzero in $W^{(i)}$ and $H^{(i)}$, respectively. Sparsity in original factors used to create data sets are 50% in the left figures, 90% in the middle figures, and 95% in the right figures. Average results of 5 different random initializations are shown.

least squares (HALS) algorithm to be comparable. In a test using synthetic data sets, the proposed algorithm clearly outperformed the HALS algorithm when the low-rank factors are sparse.

Although we explored various values for k (from 10 to 160), it has to be understood that all these values are much smaller than the original dimension. This trend is generally expected for a dimension reduction method, as mentioned in Section 2. We emphasize that the long-and-thin structure of the coefficient matrix and the flat-and-wide structure of the matrix with unknowns are key features that enables us to use speed-up techniques explained in Section 3.2 and thus obtain the successful experimental results of our new algorithm.

Different limitations of different algorithms can be noted. A limitation of a NMF algorithm based on the active set or the block principal pivoting method is that it may break down if matrix C in Eqn. (3.1) does not have full column rank. The regularization method mentioned in Section 3.4 can be adopted to remedy this problem making these algorithms generally applicable for the computation of NMF. Even without regularization, the algorithms behave well in practice as shown in our experiments. On the other hand, the HALS algorithm breaks down when either a column of W or a row of H becomes a zero vector in the process of iterations. As this problem indeed happens quite often, typically a small number $\epsilon \approx 1e^{-16}$ is used in places that are supposed to be zero [7, 10]. Due to this modification, the factors obtained by the

HALS algorithm are not sparse unless explicitly thresholded afterwards.

The block principal pivoting method studied in this paper can also be utilized in other classes of problems. The l_1-regularized linear regression, also known as the LASSO [32], is an important problem that has been widely studied and extended to other sparse learning techniques. The KKT conditions of the l_1-regularized linear regression appear as a linear complementarity problem with bounds, whose form is similar to Eqns. (3.2) except that the absolute values of the elements in y are bounded and the elements in x can take a negative sign. Utilizing the block principal pivoting method, we have proposed a fast algorithm for the l_1-regularized linear regression in [20].

Acknowledgments. We are very grateful to the editor and anonymous reviewers for valuable comments and suggestions based on which we were able to improve the manuscript substantially. We also thank Dr. Yuan (Alan) Qi of Purdue University for helpful comments and suggestions. This work was supported in part by the National Science Foundation grants CCF-0732318 and CCF-0808863 and the scholarship awarded to Jingu Kim by the Samsung Foundation of Culture. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons, *Algorithms and applications for approximate nonnegative matrix factorization*, Computational Statistics and Data Analysis, 52 (2007), pp. 155–173.

[2] R. Bro and S. De Jong, *A fast non-negativity-constrained least squares algorithm*, Journal of Chemometrics, 11 (1997), pp. 393–401.

[3] J. Brunet, P. Tamayo, T. Golub, and J. Mesirov, *Metagenes and molecular pattern discovery using matrix factorization*, Proceedings of the National Academy of Sciences, 101 (2004), pp. 4164–4169.

[4] J. Cantarella and M. Piatek, *tsnnls: A solver for large sparse least squares problem with non-negative variables*, ArXiv Computer Science e-prints, (2004).

[5] A. Cichocki and A.-H. Phan, *Fast local algorithms for large scale nonnegative matrix and tensor factorizations*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E92-A (2009), pp. 708–721.

[6] A. Cichocki, R. Zdunek, and S.-I. Amari, *Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization*, in Lecture Notes in Computer Science, vol. 4666, Springer, 2007, pp. 169–176.

[7] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, Wiley, 2009.

[8] K. Devarajan, *Nonnegative matrix factorization: An analytical and interpretive tool in computational biology*, PLoS Computational Biology, 4 (2008).

[9] I. Dhillon and S. Sra, *Generalized nonnegative matrix approximations with bregman divergences*, in Advances in Neural Information Processing Systems 18, Y. Weiss, B. Schölkopf, and J. Platt, eds., MIT Press, Cambridge, MA, 2006, pp. 283–290.

[10] N. Gillis and F. Glineur, *Nonnegative factorization and the maximum edge biclique problem.* CORE Discussion Paper 2008/64, Universite catholique de Louvain, 2008.

[11] E. F. Gonzalez and Y. Zhang, *Accelerating the Lee-Seung algorithm for non-negative matrix factorization*, tech. report, Tech Report, Department of Computational and Applied Mathematics, Rice University, 2005.

[12] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear gauss-seidel method under convex constraints*, Operations Research Letters, 26 (2000), pp. 127–136.

[13] N.-D. Ho, *Nonnegative Matrix Factorization Algorithms and Applications*, PhD thesis, Univ. Catholique de Louvain, 2008.

[14] J. J. Júdice and F. M. Pires, *A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems*, Computers and Operations Research, 21

(1994), pp. 587–596.

[15] D. Kim, S. Sra, and I. S. Dhillon, *Fast newton-type methods for the least squares nonnegative matrix approximation problem*, in Proceedings of the 2007 SIAM International Conference on Data Mining, 2007.

[16] H. Kim and H. Park, *Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis*, Bioinformatics, 23 (2007), pp. 1495–1502.

[17] ———, *Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 713–730.

[18] J. Kim and H. Park, *Sparse nonnegative matrix factorization for clustering*, tech. report, Georgia Institute of Technology Technical Report GT-CSE-08-01, 2008.

[19] ———, *Toward faster nonnegative matrix factorization: A new algorithm and comparisons*, in Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM), 2008, pp. 353–362.

[20] ———, *Fast active-set-type algorithms for l1-regularized linear regression*, in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, JMLR: W&CP 9:397-404, 2010.

[21] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Society for Industrial and Applied Mathematics, 1995.

[22] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.

[23] ———, *Algorithms for non-negative matrix factorization*, in Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 556–562.

[24] L. Li, G. Lebanon, and H. Park, *Fast algorithm for non-negative matrix factorization with Bregman divergences.* Manuscript, 2011.

[25] S. Z. Li, X. Hou, H. Zhang, and Q. Cheng, *Learning spatially localized, parts-based representation*, in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2001.

[26] C.-J. Lin, *Projected gradient methods for nonnegative matrix factorization*, Neural Computation, 19 (2007), pp. 2756–2779.

[27] P. Paatero, *Least squares formulation of robust non-negative factor analysis*, Chemometrics and Intelligent Laboratory Systems, 37 (1997), pp. 23–35.

[28] P. Paatero and U. Tapper, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126.

[29] V. P. Pauca, J. Piper, and R. J. Plemmons, *Nonnegative matrix factorization for spectral data analysis*, Linear Algebra and Its Applications, 416 (2006), pp. 29–47.

[30] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons, *Text mining using nonnegative matrix factorizations*, in Proceedings of the 2004 SIAM International Conference on Data Mining, 2004.

[31] L. F. Portugal, J. J. Judice, and L. N. Vicente, *A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables*, Mathematics of Computation, 63 (1994), pp. 625–643.

[32] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society. Series B (Methodological), 58 (1996), pp. 267–288.

[33] M. H. Van Benthem and M. R. Keenan, *Fast algorithm for the solution of large-scale nonnegativity-constrained least squares problems*, Journal of Chemometrics, 18 (2004), pp. 441–450.

[34] S. A. Vavasis, *On the complexity of nonnegative matrix factorization*, SIAM Journal on Optimization, 20 (2009), pp. 1364–1377.

[35] W. Xu, X. Liu, and Y. Gong, *Document clustering based on non-negative matrix factorization*, in SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, New York, NY, USA, 2003, ACM Press, pp. 267–273.