Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons

Jingu Kim and Haesun Park
College of Computing, Georgia Institute of Technology
Atlanta, GA 30332, USA
{jingu,hpark}@cc.gatech.edu

Abstract

Nonnegative Matrix Factorization (NMF) is a dimension reduction method that has been widely used for various tasks including text mining, pattern analysis, clustering, and cancer class discovery. The 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 some limitations of active set methods. We introduce ideas to efficiently extend the block principal pivoting method within the context of NMF computation. Our algorithm inherits the convergence theory of the ANLS framework and can easily be extended to other constrained NMF formulations. Comparisons of algorithms using datasets that are from real life applications as well as those artificially generated show that the proposed new algorithm outperforms existing ones in computational speed.

1 Introduction

Nonnegative Matrix Factorization (NMF) [12, 17] 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 where each element has a nonnegative value, and it provides a lower rank approximation formed by factors whose elements are also nonnegative. Due to the nonnegativity, the factors of lower rank approximation give a natural interpretation: each object is explained by an additive linear combination of intrinsic 'parts' of the data [12]. Numerous successes were reported in application areas including text mining [19], text clustering [21], computer vision [14], and cancer class discovery [4, 9].

A mathematical formulation of NMF is given as follows. Given an input matrix $A \in \mathbb{R}^{m\times n}$ where 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 $A \approx WH$. The factors W and H are commonly found by solving the optimization problem:

$$\min_{W,H} f(W,H) = \frac{1}{2}\left\|A - WH\right\|_F^2 \quad (1)$$
$$\text{subject to } \forall ij, W_{ij}, H_{ij} \geq 0.$$

The problem shown in Eqn. (1) is a non-convex optimization with respect to the variables W and H, so one only hopes to find a local minimum.

Many algorithms have been developed for solving Eqn. (1). More than a decade ago, Paatero and Tapper [17] initially suggested an algorithm for NMF (to be precise, *Positive* Matrix Factorization in their terms) based on the alternating nonnegative least squares (ANLS) framework. They used a subroutine for the nonnegativity constrained least squares which was not well optimized for the NMF context, resulting in a very slow algorithm. Lee and Seung popularized NMF in their seminal work [12]. Their multiplicative updating algorithm [13] has been one of the most commonly used for NMF, but several pointed out its poor performance [15, 8, 6] and problem with convergence [7]. A simple algorithm that solves an unconstrained least squares at every iteration was devised [2], but it also suffers from lack of convergence. Recently, interest in the ANLS framework was renewed, and several fast algorithms were developed using this framework [15, 8, 10]. This framework has a convergence property that every limit point of the sequence of solutions in iterations is a stationary point [15].

In this paper, we introduce a new and fast algorithm for NMF using a block principal pivoting method in the ANLS framework. Previous NMF algorithms using the ANLS framework include the active set method [10], the projected gradient method [15], and the projected quasi-Newton method [8]. The names of each method tell how each algorithm solves the nonnegativity constrained least squares subproblem. Projected gradient and projected quasi-Newton methods apply traditional techniques for unconstrained optimization with modifications for nonnegativity constraints. The active set method searches for the optimal active and passive sets by exchanging a variable at each iteration. The block principal pivoting method [20] tries to overcome the limitation of the active set method by exchanging several variables per iteration, with a goal of finding the optimal passive set of variables faster. In this paper, we adopt the block principal pivoting method in the NMF computation. We introduce ideas that improve the block principal pivoting method and then build a new algorithm for NMF.

Experimental comparisons among several NMF algorithms, including the one proposed in this paper, will follow the introduction of the new algorithm. As the fast algorithms by Lin [15], Kim et al. [8], and Kim and Park [10] appeared very recently, no proper comparison among them has yet been completed. Experimental results using commonly used datasets reveal their relative computational efficiency and show that the proposed new algorithm exhibits the best performance for NMF computation.

The rest of this paper is organized as follows. In Section 2, the ANLS framework for NMF and related background are introduced. In Section 3, our new algorithm for NMF is described in detail as well as its extensions. In Section 4, we present the design of experiments that we used to compare several NMF algorithms, and the results and their interpretation are provided in Section 5. We conclude the paper in Section 6 with discussions.

2 Alternating Nonnegative Least Squares Framework for NMF

We describe the alternating nonnegative least squares (ANLS) framework for solving Eqn. (1). The ANLS framework is a simple Expectation Maximization type algorithm where variables are divided into two groups that are updated in turn. The framework is summarized as follows.

1. Initialize $W \in \mathbb{R}^{m\times k}$ with nonnegative elements.

2. Repeat solving the following problems until a convergence criterion is satisfied:

$$\min_{H\geq 0} \|WH - A\|_F^2 \tag{2a}$$

   where W is fixed, and

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

   where H 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 H first and iterate Eqn. (2b) then Eqn. (2a). Note that each subproblem is an instance of the nonnegativity constrained least squares (NNLS) problem. Although the original problem in Eqn. (1) is non-convex, the subproblems in Eqns. (2) are convex problems for which optimal solutions can be found.

It is important to observe that the NNLS problems in Eqns. (2) have a special characteristic. NMF is a dimension reduction algorithm which is applied to high dimensional data. The original dimension is very large, e.g. several thousands, and the reduced dimension is small, e.g. on the order of tens. Therefore, the matrix $W \in \mathbb{R}^{m\times k}$ is very long and thin ($m \gg k$), and the matrix $H^T \in \mathbb{R}^{n\times k}$ is also long and thin ($n \gg k$) depending on the number of data points in A. These observations are critical in designing an efficient algorithm for the subproblems in Eqns. (2), and we will revisit this point in later sections.

For convergence of any NMF algorithm based on the ANLS framework, it is important to find optimal solutions of Eqns. (2) at each iteration. The ANLS framework is a two block coordinate descent algorithm, and a recent result by Grippo and Sciandrone [7] shows 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 its limit point is a stationary point. In a non-convex optimization, most algorithms only guarantee the stationarity of the limit point. In the alternating least squares (ALS) algorithm [2], on the contrary, the subproblems are solved in a rather ad-hoc fashion where an unconstrained least squares solution (without the nonnegativity constraint) is obtained and every negative elements are set to zero. In this case, it is difficult to analyze convergence because the algorithm updates, at each iteration, a solution which is not optimal for the subproblem.

In order to fully describe a NMF algorithm, one has to devise a specific method to solve the subproblems in Eqns. (2). A classic algorithm for the NNLS problem is the active set method by Lawson and Hanson [11]. Active set methods search for the optimal active and passive sets by exchanging a variable between the two sets. Note that if we know the passive (i.e., strictly positive) variables of the solution in advance, then a 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 NNLS problems[1], it is extremely slow when it is used for NMF in a straightforward way. Faster algorithms were recently developed by Bro and de Jong [3] and Van Benthem and Keenan [1], and Kim and Park made them into a NMF algorithm [10].

A major limitation of active set methods is that typically only one variable is exchanged between active and passive sets per iteration, making the algorithm slower when the variable size becomes large. Methods based on iterative optimization schemes such as the projected gradient method due to Lin [15] and the projected quasi-Newton method due to Kim et al. [8] are free of the above limitation. These algorithms are modified from traditional techniques in unconstrained optimization by providing specialized rules to choose step length and projecting the solution to the feasible nonnegative orthant at every iteration.

Block principal pivoting methods try to overcome the limitation of active set methods in a different fashion. We now describe this method in detail.

3 Block Principal Pivoting Algorithm

In this section, we present the block principal pivoting algorithm for NNLS problems. We will first describe the algorithm for the NNLS with a single right-hand side vector in [20] and then introduce methods that improve upon this to handle multiple right-hand sides efficiently.

3.1 Single right-hand side case

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

$$\min_{x\geq 0} \|Cx-b\|_2^2 \tag{3}$$

where $C \in \mathbb{R}^{p\times q}$, $b \in \mathbb{R}^{p\times 1}$, and $x \in \mathbb{R}^{q\times 1}$. The subproblems in Eqns. (2) are decomposed into several independent instances of Eqn. (3) with respect to each right-hand side vector. Thus, an algorithm for Eqn. (3) is a basic building block for an algorithm for Eqns. (2).

The Karush-Kuhn-Tucker optimality condition for Eqn. (3) is written as follows.

$$\begin{aligned} y &= C^TCx - C^Tb && \text{(4a)} \\ y &\geq 0 && \text{(4b)} \\ x &\geq 0 && \text{(4c)} \\ x_iy_i &= 0,\ i=1,\cdots,q && \text{(4d)} \end{aligned}$$

We assume that the matrix C has full column rank. In this case the matrix C^TC is positive definite, and the problem in Eqn. (3) is strictly convex. Then, a solution x that satisfies the conditions in Eqns. (4) is the optimal solution of Eqn. (3).

We devide the index set $\{1,\cdots,q\}$ into two subgroups F and G where $F\cup G=\{1,\cdots,q\}$ and $F\cap G=\phi$. 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 $x_G=0$ and $y_F=0$. By construction, $x=(x_F,x_G)$ and $y=(y_F,y_G)$ always satisfy Eqn. (4d) for any values of x_F and y_G. Now, we compute x_F and y_G using Eqn. (4a) and check whether the computed values of x_F and y_G satisfy Eqns. (4b) and (4c). Computation of x_F and y_G is done as follows.

$$\begin{aligned} x_F &= \arg\min_{x_F} \|C_Fx_F-b\|_2^2 && \text{(5a)} \\ y_G &= C_G^T(C_Fx_F-b) && \text{(5b)} \end{aligned}$$

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

Algorithm 1 Block principal pivoting algorithm for the NNLS with single right-hand side (Eqn. (3))

```
1. Let $F = \phi$, $G = \{1, \cdots, q\}$, $x = 0$, $y = -C^T b$, $p = 3$, $t = q + 1$
2. Compute $x_F$ and $y_G$ by Eqns. (5).
3. Repeat while $(x_F, y_G)$ is infeasible
    (a) If $|H_1 \cup H_2| < t$, set $t = |H_1 \cup H_2|$, $p = 3$ and use $\hat{H}_1 = H_1$ and $\hat{H}_2 = H_2$.
    (b) If $|H_1 \cup H_2| \geq t$ and $p \geq 1$, set $p = p - 1$ and use $\hat{H}_1 = H_1$ and $\hat{H}_2 = H_2$.
    (c) If $|H_1 \cup H_2| \geq t$ and $p = 0$, choose the largest index from $|H_1 \cup H_2|$ and exchange it.
    (d) Update $x_F$ and $y_G$ by Eqns. (5).
```

One can first solve for x_F in Eqn. (5a) and substitute the result into Eqn. (5b). We call the pair (x_F, y_G) a complementary basic solution if it is obtained by Eqns. (5).

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, $x = (x_F, 0)$ is the optimal solution of Eqn. (3), and the algorithm completes. 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. (4b) or Eqn. (4c) does not hold. Formally, we define the following index sets

$$H_1 = \{i \in F : x_i < 0\} \tag{6a}$$
$$H_2 = \{i \in G : y_i < 0\} \tag{6b}$$

and update F and G by the following rules:

$$F = \left(F - \hat{H}_1\right) \cup \hat{H}_2 \tag{7a}$$
$$G = \left(G - \hat{H}_2\right) \cup \hat{H}_1 \tag{7b}$$

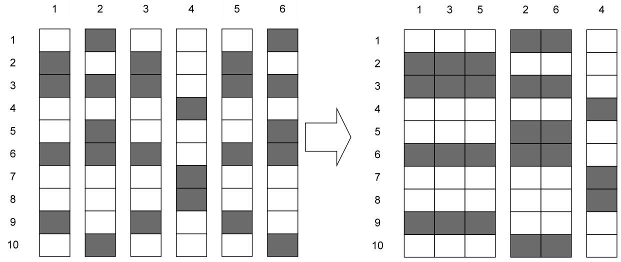
where $\hat{H}_1 \subset H_1$, $\hat{H}_2 \subset H_2$. If $\left|\hat{H}_1 \cup \hat{H}_2\right| > 1$, then the algorithm is called a *block* principal pivoting algorithm. If $\left|\hat{H}_1 \cup \hat{H}_2\right| = 1$, then the algorithm is called a *single* principal pivoting algorithm. The active set algorithm can be understood as an instance of single principal pivoting algorithms. The algorithm repeats this procedure until the number of infeasible variables (i.e., $|H_1 \cup H_2|$) becomes zero.

In order to speed up the search procedure, one usually uses $\hat{H}_1 = H_1$ and $\hat{H}_2 = H_2$ which we call the block exchange rule. The block exchange rule means that we exchange all variables of F and G that do not satisfy Eqns. (4). However, contrary to the active set algorithm where the variable to exchange is carefully selected to reduce the residual, this exchange rule may lead to a cycle and fail to find an optimal solution although it occurs rarely. Therefore, when the exchange rule fails to decrease the number of infeasible variables, we use a backup exchange rule [20] where only one variable is exchanged. As soon as the backup rule reduces the number of infeasible variables, then we return to the block exchange rule. With this modification, the block principal pivoting algorithm terminates in a finite number of iterations [20].

One might connect the two sets, F and G, of the block principal pivoting algorithm to the passive and active sets in the active set algorithm. However, they are not necessarily identical to each other. In the active set algorithm, variable x_i where i is in the passive set is required to satisfy $x_i \geq 0$ while in the block principal pivoting algorithm, variable x_i with $i \in F$ is not required to do so. Therefore, the block principal pivoting algorithm does not need an initial solution with $x \geq 0$ while the active set algorithm does.

The block principal pivoting algorithm for the NNLS problem with single right-hand side is summarized in Alg. 1. The variable p is used as a buffer on the number of the block exchange rules that may be tried. If the block exchange rule increases the number of infeasible variables, then p is reduced by one. Once the value of p becomes zero, we only exchange the infeasible variable with the largest index in $\{1, \cdots, q\}$, which is the backup exchange rule mentioned earlier. We used three as a default value of p which means that we can try the block exchange rule up to three times until it reduces the number of infeasible variables. Numerical experiments show that the algorithm is very efficient for the NNLS [20, 5].

Figure 1: Reordering of right-hand side vectors. Dark cells indicate variables with indices in F which need to be computed by Eqn. (9). By grouping the columns that have a common F set, i.e., columns $\{1,3,5\}$,$\{2,6\}$ and $\{4\}$, we can reduce the computational effort to solve Eqn. (9).



3.2 Multiple right-hand sides case

The subproblems in Eqns. (2) comprise of NNLS problems with multiple right-hand side vectors. Suppose we need to solve the following NNLS problem:

$$\min_{x\geq 0} \|CX - B\|_F^2 \tag{8}$$

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 the single right-hand side algorithm (Alg. 1) for each right-hand side vector $b_1, \cdots, b_r$. However, this is not computationally efficient. Now, we explain how we obtain an efficient algorithm for the multiple right-hand sides case based on ideas from [3] and [1]. Bro and de Jong [3] and Van Benthem and Keenan [1] suggested these ideas for active set methods, and we employ them in the context of block principal pivoting methods here.

In Alg. 1, the major computational burden is with updating x_F and y_G using Eqns. (5). We can solve Eqn. (5a) by a normal equation

$$C_F^T C_F x_F = C_F^T b, \tag{9}$$

and Eqn. (5b) can be rewritten as

$$y_G = C_G^T C_F x_F - C_G^T b. \tag{10}$$

Note that we only need to have $C_F^T C_F$, $C_F^T b$, $C_G^T C_F$, and $C_G^T b$ for solving Eqns. (9) and (10).

The first improvement is based on the observation mentioned in Section 2. For the NNLS problems arising from NMF, the matrix C is typically very long and thin. In this particular case, constructing matrices $C_F^T C_F$, $C_F^T b$, $C_G^T C_F$, and $C_G^T b$ is computationally very expensive. Therefore, our algorithm computes $C^T C$ and $C^T B$ in the beginning and reuses them in later iterations. One can easily see that $C_F^T C_F$, $C_F^T b_j$, $C_G^T C_F$, and $C_G^T b_j,\ j \in \{1, \cdots, r\}$, can be directly retrieved as a submatrix of $C^T C$ and $C^T B$. Because the column size of C is small, storage needed for $C^T C$ and $C^T B$ is not an issue.

The second improvement involves further exploiting common computations. Here we simultaneously run Alg. 1 for many right-hand side vectors. At each iteration, we have the 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. (9) and (10). The idea is to find groups of columns that share the same index sets F_j and G_j. We reorder the columns with respect to these groups and solve Eqn. (9) for the columns in the same group. By doing this, we avoid repeated computations in Cholesky factorization for solving Eqn. (9). Figure 1 illustrates this reordering idea.

We summarize the improved block principal pivoting algorithm for multiple right-hand sides in Alg. 2. The first idea is also applicable to the single right-hand side case, but the impact is more dramatic in the multiple right-hand sides case.

3.3 NMF by block principal pivoting NNLS

In the previous section, we presented the block principal pivoting algorithm for the NNLS with multiple right-hand sides (Alg. 2). We use Alg. 2 to solve the subproblems in Eqns. (2), and then we fully described our new algorithm for NMF. Implementation issues such as a stopping criterion are discussed in Section 4.2.

Algorithm 2 Block principal pivoting algorithm for the NNLS with multiple right-hand sides (Eqn. (8))

```
1. Precompute $C^TC$ and $C^TB$.
2. Let $F(\in \mathbb{R}^{q\times r}) = 0$, $G(\in \mathbb{R}^{q\times r}) = 1$, $X = 0$, $Y = -C^TB$, $P(\in \mathbb{R}^r) = 3$, $T(\in \mathbb{R}^r) = q + 1$
3. Compute $X_F$ and $Y_G$ by Eqns. (5) using column reordering.
4. Repeat while $(X_F, Y_G)$ is infeasible
    (a) For $j$'s where $|H_1(j) \cup H_2(j)| < T(j)$ , set $T(j) = |H_1(j) \cup H_2(j)|$, $P(j) = 3$ and use $\hat{H}_1(j) = H_1(j)$ and $\hat{H}_2(j) = H_2(j)$.
    (b) For $j$'s where $|H_1(j) \cup H_2(j)| \geq T(j)$ and $P(j) \geq 1$, set $P(j) = P(j) - 1$ and use $\hat{H}_1(j) = H_1(j)$ and $\hat{H}_2(j) = H_2(j)$.
    (c) For $j$'s where $|H_1(j) \cup H_2(j)| \geq T(j)$ and $P(j) = 0$, choose the largest index from $|H_1(j) \cup H_2(j)|$ and exchange it.
    (d) Update $X_F$ and $Y_G$ by Eqns. (5) using column reordering.
```

3.4 Extensions

So far, we developed a new algorithm for the original NMF formulation in Eqn. (1), and our results can easily be extended to further constrained formulations. For example, obtaining sparse factors might be of interest for some applications [9]. Sparse NMF [9] is formulated as, when the sparsity is considered for H factor,

$$\min_{W,H} \left\{ \|A - WH\|_F^2 + \eta \|W\|_F^2 + \beta \sum_{j=1}^{n} \|H(:,j)\|_1^2 \right\} \tag{11}$$
$$\text{subject to } \forall ij, W_{ij}, H_{ij} \geq 0.$$

We can solve Eqn. (11), as shown in [9], by solving the following subproblems alternatingly:

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

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, and

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

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

When W and H^T are not necessarily of full column rank, a regularized formulation can be considered [18]:

$$\min_{W,H} \left\{ \|A - WH\|_F^2 + \alpha \|W\|_F^2 + \beta \|H\|_F^2 \right\} \tag{12}$$
$$\text{subject to } \forall ij, W_{ij}, H_{ij} \geq 0.$$

As shown in [10], Eqn. (12) can also be recast into the ANLS framework. We can iterate solving

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

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

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

until convergence.

The proposed new block principal pivoting algorithm is applicable to the problems shown in Eqns. (11) and (12), and therefore, can be used for faster sparse or regularized NMF as well.

4 Comparison - Design and Issues

In this section, we describe the design of experiments and datasets used in our comparison along with a discussion of implementation issues. Due to space limitations, we do not present details of other algorithms but only refer to the papers where they are presented.

4.1 Algorithms

We compare the following algorithms for NMF.

1. (**mult**) Lee and Seung's multiplicative updating algorithm [13]
2. (**als**) Berry et al.'s alternating least squares algorithm [2]
3. (**lsqnonneg**) ANLS with Lawson and Hanson's algorithm [11]
4. (**projnewton**) ANLS with Kim et al.'s projected quasi-Newton algorithm [8]
5. (**projgrad**) ANLS with Lin's projected gradient algorithm [15]
6. (**activeset**) ANLS with Kim and Park's active set algorithm [10]
7. (**blockpivot**) ANLS with block principal pivoting algorithm which is proposed in this paper

We included **mult**, **als**, and **lsqnonneg** for the purpose of complete comparison but paid more detailed attention to algorithms that have not been compared in their entirety previously: **projnewton**, **projgrad**, **activeset**, and **blockpivot**. In all executions that we show in Section 5, all algorithms are provided with the same initial values.

4.2 Stopping criterion

Deciding when to stop a NMF algorithm is based on whether we have reached a local minimum of the objective function $\|A - WH\|_F$. We used the stopping criterion defined in [10] which is based on the Karush-Kuhn-Tucher (KKT) optimality condition for Eqn. (1).

According to the KKT condition, (W, H) is a stationary point of Eqn. (1) if and only if

$$\begin{aligned} W &\geq 0 && \text{(13a)} \\ \partial f(W,H)/\partial W &\geq 0 && \text{(13b)} \\ W.*(\partial f(W,H)/\partial W) &= 0 && \text{(13c)} \\ H &\geq 0 && \text{(13d)} \\ \partial f(W,H)/\partial H &\geq 0 && \text{(13e)} \\ H.*(\partial f(W,H)/\partial H) &= 0. && \text{(13f)} \end{aligned}$$

These conditions can be simplified as

$$\begin{aligned} \min\left(W, \partial f(W,H)/\partial W\right) &= 0 && \text{(14a)} \\ \min\left(H, \partial f(W,H)/\partial H\right) &= 0 && \text{(14b)} \end{aligned}$$

where the minimum is taken component wise [6]. We use a normalized KKT residual as

$$\Delta = \frac{\delta}{\delta_W + \delta_H} \tag{15}$$

where

$$\begin{aligned} \delta = &\sum_{i=1}^{m}\sum_{q=1}^{k}\left|\min(W_{iq}, (\partial f(W,H)/\partial W)_{iq}\right| \\ &+ \sum_{q=1}^{k}\sum_{j=1}^{n}\left|\min(H_{qj}, (\partial f(W,H)/\partial H)_{qj}\right| \end{aligned} \tag{16}$$

$$\delta_W = \#\left(\min(W, (\partial f(W,H)/\partial W) \neq 0\right) \quad (17)$$

$$\delta_H = \#\left(\min(H, (\partial f(W,H)/\partial H) \neq 0\right). \quad (18)$$

Note that the KKT residual is divided by the number of nonzero elements in order to make the residual independent of the sizes of W and H. Using this normalized residual, the convergence criterion is defined as

$$\Delta \leq \epsilon \Delta_0 \quad (19)$$

where Δ_0 is the value of Δ using initial values of W and H and ϵ is a chosen tolerance. We computed Δ_0 using the initial values instead of using the values after the first iteration as done in [10], because the latter is not fair in comparing several algorithms.

4.3 Datasets

We use three datasets for comparisons: synthetic, text, and image. The synthetic dataset is created in the following way. We used $m = 300$ and $n = 200$. For $k = 5, 10, 20, 30, 40, 60$, and 80, we randomly constructed $m \times k$ matrix W and $k \times n$ matrix H with 40% sparsity. Then, we computed $A = WH$ and added Gaussian noise to each element where the standard deviation is 5% of the average magnitude of elements in A. Finally, we normalized matrix A so that the average element-wise magnitude is the same for all k values. Basically, we created small synthetic matrices that have latent sparse nonnegative factors.

For text dataset, the Topic Detection and Tracking 2 (TDT2) text corpus[2] is used. The TDT2 dataset contains news articles from various sources such as NYT, CNN, VOA, etc. 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 20 topics where the number of articles in the topic is greater than 20. The term document matrix is created by TF.IDF indexing and unit-norm normalization [16]. We obtained a 12617 × 1491 term-document matrix.

For image dataset, the Olivetti Research Laboratory (ORL) face image database[3] is used. The database contains 400 face images of 40 different people with 10 images per person. Each face image has 92×112 pixels in 8-bit grey level. We obtained 10304×400 matrix.

5 Comparison of the Experimental Results

In this section, we summarize our experimental results and offer interpretation. We implemented our new block principal pivoting algorithm in MATLAB. For other existing NMF algorithms, we used MATLAB codes presented in [10, 8, 15] after modifying them with our stopping criterion. All experiments were executed on 3.2 GHz Pentium4 Xeon EMT64 machines with Linux OS.

5.1 Experimental results with synthetic datasets

We tested all seven algorithms presented in Section 4.1 on the synthetic datasets. The same initial values were shared in all algorithms, and the average results using 10 different initial values are shown in the Table 1.

As shown in Table 1, **multi** and **als** easily exceeded the maximum number of iterations which was set to be 10,000. These results show that the algorithms have difficulties with convergence. The failure to converge resulted in worse approximations as the residual values show; when $k = 20$, **multi** and **als** gave larger average residuals compared to ANLS type algorithms. Since the number of iterations exceeded the limit and the execution times were among the slowest, we did not include these algorithms in the following experiments.

All ANLS type algorithms appeared to satisfy the convergence criterion within a reasonable number of iterations, giving an empirical confirmation of convergence. The numbers of iterations for ANLS type algorithms were more or less similar to each other except in **projgrad** method. This is because that **projgrad** and **projnewton** are based on iterative optimization schemes. In their subroutines for the NNLS problem, another tolerance value needs to be specified for a stopping criterion, and the tightness of the solution in the subproblem depends upon the tolerance value. On the other hand, **activeset** and **blockpivot** exactly solves the NNLS subproblem at every iteration. The difference might lead to a variation in the number of iterations of their NMF algorithms.

[2] http://projects.ldc.upenn.edu/TDT2/

[3] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Table 1: Experimental results on 300×200 synthetic datasets with latent nonnegative factors with $\epsilon = 10^{-4}$. For each k, all algorithms were executed with the same initial values, and the average results from using 10 different initial values are shown in the table. For the execution time comparison, the shortest execution time is highlighted in **bold** type. For slow algorithms, experiments with larger k values take too much time and are omitted. The residual is computed as $\|A - WH\|_F \, / \, \|A\|_F$.

	k	**multi**	**als**	**lsqnonneg**	**projnewton**	**projgrad**	**activeset**	**blockpivot**
time (sec)	5	35.336	36.697	23.188	5.756	0.976	0.262	**0.252**
	10	47.132	52.325	82.619	13.43	4.157	0.848	**0.786**
	20	72.888	83.232		45.007	9.32	4.41	**4.004**
	30				127.33	62.317	17.252	**14.384**
	40					81.445	22.246	**16.132**
	60					128.76	37.376	**21.368**
	80					276.29	65.566	**30.055**
iterations	5	9784.2	10000	25.6	25.8	30	26.4	26.4
	10	10000	10000	34.8	35.2	45	35.2	35.2
	20	10000	10000		70.8	104	69.8	69.8
	30				166	205.2	166.6	166.6
	40					234.8	118	117.8
	60					157.8	84.2	84.2
	80					131.8	67.2	67.2
residual	5	0.04035	0.04043	0.04035	0.04035	0.04035	0.04035	0.04035
	10	0.04345	0.04379	0.04343	0.04343	0.04344	0.04343	0.04343
	20	0.04603	0.04556		0.04412	0.04414	0.04412	0.04412
	30				0.04313	0.04316	0.04327	0.04327
	40					0.04944	0.04943	0.04944
	60					0.04106	0.04063	0.04063
	80					0.03411	0.03390	0.03390

The **lsqnonneg** algorithm was implemented as a reference point although it is known to be very slow. The execution time of **lsqnonneg** as shown in Table 1 was much longer than in other ANLS algorithms. Among remaining ANLS type algorithms, i.e., **projnewton**, **projgrad**, **activeset**, and **blockpivot**, **projnewton** method was computationally less efficient than others as can be seen from Table 1. Its overall computation time was much longer than those of other ANLS type algorithms while the required number of iterations was similar to others. These results imply that **projnewton** is slower in solving the subproblems in Eqns. (2).

Among all algorithms, **blockpivot** showed the shortest execution time. Note that for small values of k up to 20, the execution time required for **activeset** or **projgrad** was comparable to that of **blockpivot**. However, as k becomes larger, the difference between **blockpivot** and the other two growed to be nontrivial. As these three algorithms show the best efficiency, we focus on comparing these algorithms with large real datasets below.

5.2 Experimental results with text and image datasets

Experimental results with text and image datasets are shown in Tables 2 and 3. As we learned that the three algorithms, **projgrad**, **activeset**, and **blockpivot,** are the most efficient from the previous experiments using smaller synthetic datasets, we focused on comparing the three algorithms.

From Tables 2 and 3, it can be observed that **blockpivot** is often the most efficient algorithm for various values of k. For small values of k, it appeared that **activeset** was slightly faster than **blockpivot**, but the difference was very small. This result agrees with a general understanding that for solving a NNLS problem where the number of variables is small, the active set method is preferred. For larger values of k, **blockpivot** showed much better performance than the other two algorithms. Since NMF or sparse NMF was shown to work well as a clustering method [21, 9] and the value of k is typically small in clustering problems, we recommend **activeset** or **blockpivot** method for a clustering use. Exploratory analysis for text or image database might often use relatively large k values, and our results recommend **blockpivot** in this case. Overall, the experimental results confirm that **blockpivot** is generally superior to the other two algorithms.

Table 2: Experimental results on 12617×1491 text dataset with $\epsilon = 10^{-4}$. For each k, all algorithms were executed with the same initial values, and the average results from using 10 different initial values are shown in the table. The residual is computed as $\|A - WH\|_F \,/\, \|A\|_F$.

	k	**projgrad**	**activeset**	**blockpivot**
time (sec)	5	107.24	**81.476**	82.954
	10	131.12	**87.012**	88.728
	20	161.56	154.1	**144.77**
	30	355.28	314.78	**234.61**
	40	618.1	753.92	**479.49**
	50	1299.6	1333.4	**741.7**
	60	1616.05	2405.76	**1041.78**
iterations	5	66.2	60.6	60.6
	10	51.8	42	42
	20	45.8	44.6	44.6
	30	100.6	67.2	67.2
	40	118	103.2	103.2
	50	120.4	126.4	126.4
	60	154.2	171.4	172.6
residual	5	0.9547	0.9547	0.9547
	10	0.9233	0.9229	0.9229
	20	0.8898	0.8899	0.8899
	30	0.8724	0.8727	0.8727
	40	0.8600	0.8597	0.8597
	50	0.8490	0.8488	0.8488
	60	0.8386	0.8387	0.8387

Table 3: Experimental results on 10304×400 image dataset with $\epsilon = 5 \times 10^{-4}$. For each k, all algorithms were executed with the same initial values, and the average results from using 10 different initial values are shown in the table. The residual is computed as $\|A - WH\|_F \,/\, \|A\|_F$.

	k	**projgrad**	**activeset**	**blockpivot**
time (sec)	16	68.529	**11.751**	11.998
	25	124.05	25.675	**22.305**
	36	109.1	53.528	**35.249**
	49	150.49	115.54	**57.85**
	64	169.7	270.64	**91.035**
	81	249.45	545.94	**146.76**
iterations	16	26.8	16.4	16.4
	25	20.6	15	15
	36	17.6	13.4	13.4
	49	16.2	12.4	12.4
	64	16.6	13.2	13.2
	81	16.8	14.4	14.4
residual	16	0.1905	0.1907	0.1907
	25	0.1757	0.1751	0.1751
	36	0.1630	0.1622	0.1622
	49	0.1524	0.1514	0.1514
	64	0.1429	0.1417	0.1417
	81	0.1343	0.1329	0.1329

Note that the relative efficiency between **activeset** and **projgrad** may be reversed for a larger k. In Table 2, **projgrad** appeared faster than **activeset** for $k \geq 40$, and in Table 3, for $k \geq 49$. In Table 1, however, **activeset** appeared faster than **projgrad** throughout all k values. It seems that their relative efficiency depends on different datasets and different problem sizes although they are both inferior to **blockpivot** in all cases.

Although we explored various values for k (from 5 to 81), it has to be understood that all these values are much smaller than the original dimension, which was 12617 for the text dataset and 10304 for the image dataset. This trend is what we expect from a dimension reduction method, as mentioned in Section 2. We emphasize that the long and thin structure of the NNLS problems arising from NMF is a key feature that enables us to use speed-up techniques explained in Section 3.2 and consequently gives the successful experimental results of **blockpivot**.

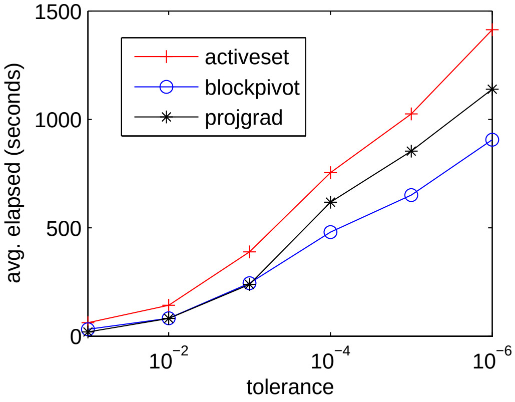
5.3 Execution time and tolerance values

We examined the efficiency of the three algorithms with respect to tolerance values and show the results in Figure 2. Note that **projgrad** was comparable to or faster than **blockpivot** when a loose tolerance was given. When a tighter tolerance was used, however, **blockpivot** was clearly faster than **projgrad**. This result implies that **projgrad** quickly minimizes the objective function in earlier iterations but becomes slower in achieving a good approximation by a tight tolerance.

6 Discussion and Conclusion

In this paper, a new algorithm for computing Nonnegative Matrix Factorization (NMF) based on the alternating nonnegative least squares (ANLS) framework is proposed. The new algorithm is built upon the block principal pivoting algorithm for the nonnegativity constrained least squares (NNLS) problem. We introduced ideas for improvement in efficient handling of the multiple right-hand sides case of NNLS. The newly constructed algorithm inherits the convergence theory of the ANLS framework and can easily be extended to other constrained NMF for-

Figure 2: Execution time with respect to tolerance values on 12617×1491 text dataset. All algorithms were executed with the same initial values, and the average results using 10 different initial values are presented.



mulations such as sparse NMF or regularized NMF. Experimental comparisons with most of the NMF algorithms presented in literature using synthetic, text, and image datesets show that the new algorithm is generally the most efficient method for computing NMF.

A limitation of a NMF algorithm based on active set or block principal pivoting method is that it may break down if the matrix C in Eqn. (3) does not have full column rank. However, the algorithm is expected to behave well in practice as observed in our experiments. The regularization method mentioned in Section 3.4 can be adopted to remedy this problem making these algorithms generally applicable for computations of NMF.

Acknowledgments

The work of authors was supported in part by the National Science Foundation grants CCF-0732318 and CCF-0808863, and the Samsung Scholarship awarded to Jingu Kim. 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. H. V. Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of Chemometrics*, 18:441–450, 2004.

[2] 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(1):155–173, 2007.

[3] R. Bro and S. D. Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11:393–401, 1997.

[4] 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(12):4164–4169, 2004.

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

[6] E. F. Gonzalez and Y. Zhang. Accelerating the lee-seung algorithm for non-negative matrix factorization. Technical report, Tech Report, Department of Computational and Applied Mathematics, Rice University, 2005.

[7] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss-seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[8] 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.

[9] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[10] H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. *SIAM Journal in Matrix Analysis and Applications*, to appear.

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

[12] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[13] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

[14] S. Z. Li, X. Hou, H. Zhang, and Q. Cheng. Learning spatially localized, parts-based representation. In *CVPR '01: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2001.

[15] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[16] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[17] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(1):111–126, 1994.

[18] V. P. Pauca, J. Piper, and R. J. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and Its Applications*, 416(1):29–47, 2006.

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

[20] 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(208):625–643, 1994.

[21] 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*, pages 267–273, New York, NY, USA, 2003. ACM Press.