Annals of Mathematical Sciences and Applications
Volume 7, Number 2, 281–300, 2022

Index search method for solving nonnegative matrix factorization*

Yi-Shin Cheng and Ching-Sung Liu

Nonnegative matrix factorization (NMF) has been widely used for dimensionality reduction in recent years, while playing an important role in many fields such as image processing and data analysis. NMF is a classic non-convex optimization problem, and the alternating nonnegative least squares (ANLS) framework is a popular method for solving the problem. In general, ANLS divides the NMF problem into two convex optimization problems, called nonnegative least squares (NNLS) problems. In this paper, we first introduce an active set method for NNLS, called indexed search method (ISM). Meanwhile, our goal is to propose a robust algorithm that combines ANLS and ISM for solving NMF. Finally, numerical experiments are provided to support the theoretical results.

AMS 2000 subject classifications: Primary 65F15, 65F60.
Keywords and phrases: Nonnegative matrix factorization, index search method, KKT conditions.

1. Introduction

Technology is advancing with each passing day, and the era of big data is coming. Nowadays, there are thousands of data accumulated around us, how to use these data has become a major problem. It is not easy to find latent information in a large amount of data, so dimensionality reduction plays an important role in machine learning and data mining, and nonnegative matrix factorization (NMF) [14, 15] has attracted a lot of attention. The matrices in NMF have nonnegative element and are called nonnegative matrix. In NMF, we will decompose a high-dimensional nonnegative matrix into two low-dimensional nonnegative matrices. After multiplying the two matrices, the original high-dimensional nonnegative matrix can be reduced to a low-dimensional nonnegative matrix. The mathematical formula is as follows:

*C.-S. Liu was supported in part by the National Science and Technology Council in Taiwan.

Figure 1: Use NMF to decompose face images to obtain face features.

$$\min_{W,H\geq 0} \frac{1}{2}\|V - WH\|_F^2, \tag{1}$$

where V $\in \mathbb{R}^{m\times n}$, $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$.

In most data, such as images [1], text [16, 18, 1] and medical data [3, 8], all have nonnegative properties. And NMF can preserve the properties of these data. Due to its nonnegativity, each element in the data is a linear combination of some of the implicit information in the data. It can be seen that the factors include the implicit information in the data and the way of combination. Figure 1 shows one of the factors in a graphical way after performing NMF on the images of 2429 faces. It can be clearly seen that the figure presents the outline of the face. Hence NMF also has the function of feature extraction. Compared with other decompositions, such as Singular Value Decomposition (SVD), which can provide higher accuracy matrix decomposition, NMF can provide higher sparsity, because it has far more zeros than other decompositions with negative numbers. Although some people use the projection method to make nonnegative SVD [17], and has lower accuracy. However, when the matrix size is large, it will not only take more time, but also have the disadvantage of insufficient memory.

There are many kinds of algorithms for NMF. Paatero and Tapper [15] initially proposed the alternating nonnegative least squares (ANLS) [5, 13] framework for NMF. At first they created an slow algorithm, but then Lee and Seung developed new algorithms and popularized NMF. Their multiplicative update algorithm (MULT) [12] is well known, and the original W and H are multiplied by an update term (2) by repeatedly using multiplication until the stopping criterion is satisfied. The algorithm is fast at each iteration because it does not solve a linear system, but because of this,

the final number of iterations is usually very large. Although a convergence proof for the algorithm has been proposed in the past, the proof was later overturned, so it lacks convergence. MULT has bad performance when there is a zero element in the multiple update term, the denominator will be 0 and cause error. In addition, there is also an algorithm that directly solves the unconstrained least squares at each step. After obtaining the solution, the negative number is directly projected to 0 to achieve nonnegativity. Although this method is very intuitive, it is ineffective. But this idea is good and worth keeping and improving.

$$\begin{cases} W_{ia} = & W_{ia}\dfrac{(VH^T)_{ia}}{(WHH^T)_{ia}}, \forall i, a, \\ H_{bj} = & H_{bj}\dfrac{(W^TV)_{bj}}{(W^TWH)_{bj}}, \forall b, j. \end{cases} \tag{2}$$

There are two other more mainstream methods, projected gradient method (PG) [13] and block principal pivoting method (BP) [10]. PG is to use the gradient method to solve NNLS and set the negative part to 0, which is called projection in this method. The gradient method is to select the negative gradient in the subproblem as the direction, and search for the step size to move forward in the selected direction. The most important part of PG is to select the step size. If the step size is too large, it may skip the optimal solution, resulting in infinite loops and failure to converge; if the step size is too small, the convergence speed may be very slow. Therefore, PG will spend a long time in selecting a good step size, and how to choose a good step size is the most challenging part of this algorithm.

BP is different from PG, regardless of the selection of step size and direction, BP solves linear systems. To satisfy the constraints, there must be only positive numbers and 0 in the optimal solution. If the correct positive element index can be found, then solve the least squares problem for these indices (with no constraints), and fill in 0 for the other indices, the optimal solution can be obtained. Due to dual feasible (9c) and the complementary property (9d), the dual problem also has similar properties, when selecting the positive element indices, if the KKT criterion is not satisfied, delete the negative element index in $\mathbf{x}$, and then add the negative element index in $\mathbf{u}$, recalculated until all of the KKT conditions are satisfied. This method may enter an infinite loop (the deleted indices are added back in the next step), so the algorithm has a setting that if an infinite loop occurs, it will be changed to eliminate the indices one by one until the number of negative element indices decreases.

In the ANLS framework, the NMF problem is transformed into two nonnegativity constrained least squares subproblem. The methods described above are all algorithms for solving subproblems. The algorithms that can be used to solve subproblems in the ANLS framework also have the active set method [9], the projected quasi-Newton method [7] and a new algorithm, Index Search Method [11], is introduced in this paper. Different from the gradient descent methods, we use the method of finding the correct indices to solve the subproblems. Unlike gradient descent, we can get a more accurate solution.

The rest of this paper is organized as follows. Section 2 provides preliminaries and introduces the ANLS framework of NMF and the related background, and shows the methods are used to solve the subproblems in NMF using the ANLS framework. Section 3 introduce the ISM algorithm and improve it for the case of multiple right hand-side. Except for nonnegative constraints, this section will generalizes ISM for box constraints. Section 4 introduce two experiments used to compare several NMF algorithms and provide the results and their interpretation. Finally, we conclude the paper in Section 5 with discussions.

2. Preliminaries

Throughout the paper, we use capital letters to denote matrices and lowercase (bold) letters to denote scalars (vectors). A real matrix $A = [A_{ij}] \in \mathbb{R}^{n\times k}$ is called nonnegative (positive) if $A_{ij} \geq 0$ ($A_{ij} > 0$). The superscript T denotes the transpose of vector or matrix. For each j, we use $\mathbf{v}(j)$ to represent the j-th element of a vector $\mathbf{v}$ and $A(:, j)$ to represent the j-th column of a matrix A. In addition, we use typewriter typestyle for any subset $\mathtt{I} \subseteq \{1, ..., m\}$, let $\mathtt{I} = \{i_1, ..., i_\ell\}$ with $i_1 < ... < i_\ell$, $|\mathtt{I}|$ denotes the number of elements in $\mathtt{I}$, i.e., $|\mathtt{I}| = \ell$. For a matrix $A \in \mathbb{R}^{n\times k}$ with $\min\{n, k\} \geq i_l$, A(:,I) (or A(I,:)) is an n×$|\mathtt{I}|$ (or an $|\mathtt{I}| \times k$) submatrix of A formed by columns (or rows) $i_1, ..., i_\ell$. Denote $\mathtt{I}^c = \{j_1, ..., j_{m-\ell}\}$ with $j_1 \leq j_{m-\ell}$ the complement of $\mathtt{I}$. Denote a permutation matrix

$$P_\mathtt{I} = [\mathbf{e}_{i_1}, ..., \mathbf{e}_{i_\ell}, \mathbf{e}_{j_1}, ..., \mathbf{e}_{j_{m-\ell}}] \in \mathbb{R}^{m\times m},$$

where $\mathbf{e}_k$ is the j-th column of $m \times m$ identity matrix. Then for each $\mathbf{v} \in \mathbb{R}^m$, we have $P_\mathtt{I}^T \mathbf{v} = [\mathbf{v}(\mathtt{I})^T, \mathbf{v}(\mathtt{I}^c)^T]^T$ and $P_\mathtt{I}^T P_\mathtt{I} = I_m$.

An optimization problem in standard form is as follow:

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m \end{aligned} \tag{3}$$

with variable $\mathbf{x} \in \mathbb{R}^n$. And the *Lagrangian* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with the problem (3) is defined as

$$L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{u}(i) g_i(\mathbf{x})$$

in [2], where the vector $\mathbf{u} \in \mathbb{R}^m$ is called the *dual variables* or *Lagrange multiplier vectors* associated with the problem (3).

Assume that the function $f, g_1, ..., g_m$ are differentiable, and let $\mathbf{x}_*$ and $\mathbf{u}_*$ be any primal and dual optimal points with zero duality gap. Since $\mathbf{x}_*$ minimizes $L(\mathbf{x}, \mathbf{u}_*)$ over $\mathbf{x}$, it follows that its gradient must vanish at $\mathbf{x}_*$, i.e.

$$\partial_{\mathbf{x}} f(\mathbf{x}_*) + \sum_{i=1}^{m} \mathbf{u}_*(i) \partial_{\mathbf{x}} g_i(\mathbf{x}_*) = 0.$$

Thus

$$\begin{aligned} g_i(\mathbf{x}_*) &\leq 0, \quad i = 1, ..., m, \\ \mathbf{u}_*(i) &\geq 0, \quad i = 1, ..., m, \\ \mathbf{u}_*(i) g_i(\mathbf{x}_*) &= 0, \quad i = 1, ..., m, \\ \partial_{\mathbf{x}} f(\mathbf{x}_*) + \sum\nolimits_{i=1}^{m} \mathbf{u}_*(i) \partial_{\mathbf{x}} g_i(\mathbf{x}_*) &= 0, \end{aligned} \tag{4}$$

which are called the *Karush-Kuhn-Tucker* (KKT) conditions [2].

To summarize, for *any* optimization problem with differentiable objective and constraint functions for which strong duality obtains, any pair of primal and dual optimal points must satisfy the KKT conditions (4).

2.1. Alternating nonnegative least squares framework for nonnegative matrix factorization

This section detail the Alternating Nonnegative Least Squares framework used to solve NMF. Alternating nonnegative least squares (ANLS) framework is widely used to solve NMF. The ANLS framework divide the problem into two blocks and solve the NNLS alternately to get the optimal solution. The framework is summarized as follows.

1. Initialize a nonnegative matrix $H_0 \in \mathbb{R}^{r \times n}$.
2. Solve the following problems repeatedly until the convergence criteria is met:

$$W_{k+1} = \arg\min_{W \geq 0} \|H_k^T W^T - V^T\|_F^2, \tag{5}$$

where H_k is fixed, and

$$H_{k+1} = \arg\min_{H \geq 0} \|W_{k+1}H - V\|_F^2, \tag{6}$$

where W_{k+1} is fixed.

This framework can swap the order, initialize W first, and reverse the order in the second step. Although the two orders are mathematically correct, the order of W and H when V is a sparse matrix will be discussed in later section.

The subproblems in NMF are nonnegative least squares. Although the original problem (1) is non-convex, after dividing the original problem into two subproblems, the subproblems are convex, so the optimal solution can be found in the subproblem. In addition, $W \in \mathbb{R}^{m\times r}$ and $H^T \in \mathbb{R}^{n\times r}$ are slender matrices with respect to V since $r \ll \min(m, n)$. These observations play an important role in improving the algorithm later.

The ANLS framework is a two block coordinate descent algorithm, shown in the results of Grippo and Sciandrone [5], any limit point of the sequence of optimal solutions of two block subproblems is a stationary point. With the guarantee of the above, the convergent of any NMF algorithm based on the ANLS framework, finding the optimal solution to the subproblem is the most important. If the subproblem has good convergence properties and the optimal solution can be obtained at each iteration, then the NMF problem can obtain the solution.

3. Index search method algorithm for NNLS

This section introduce the index search method (ISM) algorithm for the NNLS problem. First, describe the NNLS algorithm with a single right-hand vector. And then, the improved methods will be introduced to effectively handle multiple right-hand side method, since the subproblems (5) and (6) in the NMF of the ANLS framework need to solve multiple NNLS problems.

3.1. Single right-hand side case

The single right-hand side is NNLS problem, and it is formulated as

$$\min_{\mathbf{x} \geq 0} \|A\mathbf{x} - \mathbf{b}\|_2^2, \tag{7}$$

where $A \in \mathbb{R}^{p\times q}, \mathbf{b} \in \mathbb{R}^{p\times 1}$, and $\mathbf{x} \in \mathbb{R}^{q\times 1}$. The subproblems in (5) and (6) can be regarded as several independent instances decomposed into (7).

Therefore, the algorithm of (7) is the basic building block of the algorithms of (5) and (6). Before introducing the algorithm, we will first deduce the KKT conditions of problem (7) in detail to facilitate the subsequent introduction.

Assume A has full column rank. Since the problem in (7) is convex optimization problem, it has a unique optimal solution. The NNLS problem (7) can also be regarded as a quadratic optimization problem with linear inequality constraints:

$$\min_{\mathbf{x}\geq 0} f(\mathbf{x}) = \min_{\mathbf{x}\geq 0}(\frac{1}{2}\mathbf{x}^T A^T A\mathbf{x} - \mathbf{x}^T A^T \mathbf{b}). \tag{8}$$

This function is also a convex optimization problem. To form the Lagrangian we introduce multipliers u_i for the q inequality constraints (i.e., $\mathbf{x} \in \mathbb{R}^q$ and $\mathbf{x} \geq 0$) and let $\mathbf{u} \in \mathbb{R}^q$ with $\mathbf{u} \geq 0$ and $\mathbf{u}(i) = u_i$. The Lagrangian is

$$L(\mathbf{x},\mathbf{u}) \equiv f(\mathbf{x}) - \mathbf{u}^T\mathbf{x} = \frac{1}{2}\mathbf{x}^T A^T A\mathbf{x} - \mathbf{x}^T A^T \mathbf{b} - \mathbf{u}^T\mathbf{x},$$

and we can compute

$$\partial_{\mathbf{x}} L(\mathbf{x},\mathbf{u}) = A^T A\mathbf{x} - A^T\mathbf{b} - \mathbf{u}.$$

Finally, the KKT conditions of (8) are

$$\mathbf{u} = A^T A\mathbf{x} - A^T\mathbf{b}, \tag{9a}$$

$$\mathbf{x} \geq 0, \tag{9b}$$

$$\mathbf{u} \geq 0, \tag{9c}$$

$$\mathbf{x}(i)\mathbf{u}(i) = 0, i = 1, 2, ..., q. \tag{9d}$$

That is, if there are $\mathbf{x}$ and $\mathbf{u}$ satisfy the KKT conditions if and only if $\mathbf{x}$ and $\mathbf{u}$ are primal and dual optimal, and we denote $\mathbf{x}_*$ and $\mathbf{u}_*$ as primal and dual optimal.

To classify, ISM is more similar to BP, which also needs to solve linear systems. The initial idea is derived from the KKT condition. Since the optimal solution of NNLS must satisfy (9), all elements in $\mathbf{x}_*$ must be non-negative numbers, and only positive numbers provide information. So we assume that if we can find the indices of the positive numbers in $\mathbf{x}$, we only have to solve the linear system for the indices. How to find the positive index in the optimal solution is an important part of ISM.

Assuming that the index set $\mathtt{I}_k$ is the set of all positive indices of $\mathbf{x}_k$, the purpose is to find the $\mathtt{I}_*$, where

$$\mathtt{I}_* = \{i|\mathbf{x}_*(i) > 0\}.$$

Besides primal feasible (9b), KKT conditions also have dual feasible (9c) and complementary properties (9d). Let's define a set that has indices in $\mathbf{x}$ that satisfy the primal feasible as

$$\mathtt{J} = \{i \in \mathbb{N} | \mathbf{x}(i) > 0\},$$

and the set that has indices in $\mathbf{u}$ that do not satisfy the dual feasible as

$$\mathtt{N} = \{i \in \mathbb{N} | \mathbf{u}(i) < 0\}.$$

If any element in $\mathbf{u}$ is negative, it means that $\mathbf{x}$ is not the optimal solution. Hence, if $\mathtt{N}$ is non-empty, select those indices into $\mathtt{I}$, increase the indices position where $\mathbf{x}$ needs to be solved. Therefore, in the process of searching $\mathtt{I}_*$, the indices with positive elements in $\mathbf{x}$ and the indices with negative elements in the corresponding $\mathbf{u}$ are selected into $\mathtt{I}$. In other words, every iteration before we find $\mathtt{I}_*$, we should update $\mathtt{I}$ by the following equation

$$\mathtt{I} = \mathtt{J} \cup \mathtt{N}.$$

After determining how to update $\mathtt{I}$, we can use the index set to solve for the corresponding position, then check whether the current solution is the optimal.

Since $\mathtt{I}$ is the index set of the indices in $\mathbf{x}$ which elements are positive, so $\mathbf{u}(\mathtt{I})$ must be zero by complementary property (9a). Hence, to keep the complementary property (9a) and stationary (9d), the equation for solving $\mathbf{y}$ is as follows:

$$A_{\mathtt{I}}^T A_{\mathtt{I}} \mathbf{y} - A_{\mathtt{I}}^T \mathbf{b} = 0, \tag{10}$$

where $A_{\mathtt{I}} = A(:, \mathtt{I})$. Since we assume that the matrix A has full column rank, then the matrix $A^T A$ is positive definite, and (10) is strictly convex. If it is found that the element in $\mathbf{y}$ has any negative number, remove those indices from $\mathtt{I}$, and solve (10) again; otherwise, set $\mathbf{x}(\mathtt{I}) = \mathbf{y}$, and $\mathbf{x}(\mathtt{I}^c) = 0$, let $\mathbf{x}$ satisfies the primal feasible. Then compute the corresponding $\mathbf{u}$ by (9d), and check whether $\mathbf{u}$ satisfies dual feasible, if there are still negative elements in $\mathbf{u}$, use the pair ($\mathbf{x}$,$\mathbf{u}$) to find new $\mathtt{I}$. Otherwise, it means that this pair of solutions satisfies all KKT conditions. Then $\mathbf{x}_* = \mathbf{x}$, and $\mathbf{u}_* = \mathbf{u}$. The idea of Algorithm 1 is shown in Figure 2, and the flow chart in Figure 3.

Theorem 3.1 ([11]). *Assume that* $\{\boldsymbol{x}_k\}$ *is the sequence generated by Algorithm 1. Then* $\{\boldsymbol{x}_k\}$ *converges to the optimal* $\boldsymbol{x}_*$ *of NNLS (7).*

Algorithm 1 Index search method (Single)

1. Compute $\mathbf{r}_0 = -\mathbf{b}$, $\mathbf{u}_0 = A^T\mathbf{r}_0$, $Q = A^TA$ and $\mathbf{c} = A^T\mathbf{b}$.
   Given $\mathbf{x}_0 = 0$ and $\mathsf{tol} > 0$. Set $\mathbf{y} = -\mathbf{1}$.
2. For $k = 0, 1, 2, ...$
3. $\mathtt{J}_k = \{i \in \mathbb{N} | \mathbf{x}_k(i) > 0\}$, $\mathtt{N}_k = \{i \in \mathbb{N} | \mathbf{u}_k(i) < 0\}$ and $\mathtt{I}_k = \mathtt{J}_k \cup \mathtt{N}_k$.
4. While $\mathbf{y} \not> 0$
5. Solve the linear system $Q(\mathtt{I}, \mathtt{I})\mathbf{y} = \mathbf{c}(\mathtt{I})$.
6. Find the index set $\mathtt{P} = \{i \in \mathbb{N} | \mathbf{y}(i) > 0\}$.
7. Set $\mathtt{I} = \mathtt{I}(\mathtt{P})$.
8. EndWhile
9. Set $\mathtt{J} = \mathtt{I}$ and $\mathbf{x}_{k+1} = P_{\mathtt{J}} \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}$.
10. Compute $\mathbf{r}_{k+1} = A\mathbf{x}_{k+1} - \mathbf{b}$.
11. $\mathbf{u}_{k+1} = A^T\mathbf{r}_{k+1}$.
12. EndFor
13. **until** $|\min(\mathbf{u}_{k+1})| \leq \mathsf{tol}$. (seek dual feasibility)

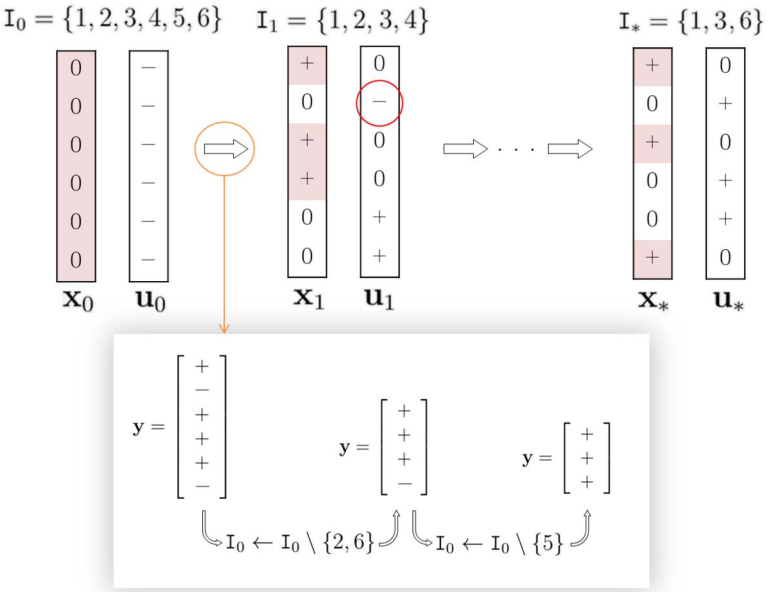

Figure 2: The idea of Algorithm 1.

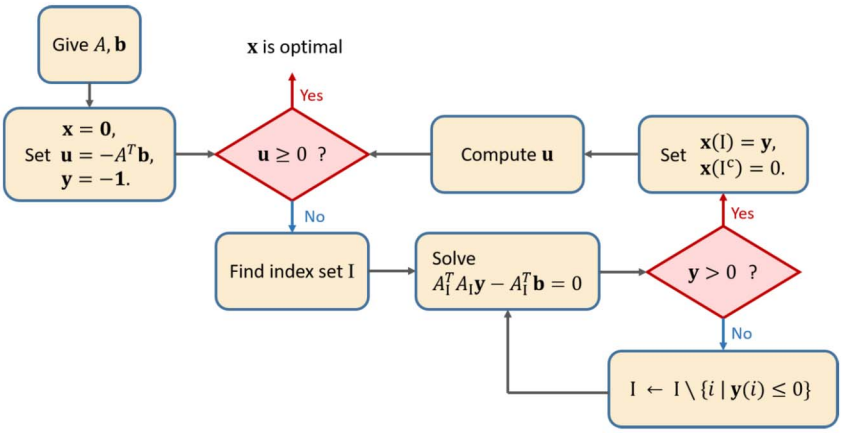

Figure 3: The flow chart of Algorithm 1.

3.2. Multiple right-hand sides case

The subproblems (5) and (6) can be viewed as NNLS problems with multiple right-hand side vectors. First we assume that the following NNLS problem needs to be solved:

$$\min_{X \geq 0} \|AX - B\|_F^2,$$

where $A \in \mathbb{R}^{p \times q}, B \in \mathbb{R}^{p \times s}$, and $X \in \mathbb{R}^{q \times s}$. The above problem can be simply regarded as performing a single right-hand algorithm (Algorithm 1) on each right-hand vector $B_1, ..., B_s$, but this method is computationally inefficient. Now, we will introduce the efficient methods for this case, and the improvement methods from [10] have two main parts.

The first idea of improvement is the result observed from Section 3. For subproblems in NMF, the matrix A of the NNLS problem which is W or H^T in subproblems (5) and (6) will be very thin and long. In this case, it would be computationally expensive to construct A^TA and A^TB, and the submatrix of those matrix must be constructed repeatedly in each iteration of the algorithm to solve the linear system, which would make the algorithm very inefficient. Therefore, our algorithm precomputes $Q = A^TA$ and $C = A^TB$ at the beginning, then saves them and reuses them in later iterations. It can be seen that the 6th line in Algorithm 1 and the 7th line in Algorithm 3 both use this method. Although both sides can improve the efficiency, the effect is more significant in the case of multiple right-side vectors. Because A is very thin and long, the size of A^TA and A^TB will be small, so the storage space of the two matrices is not an issue.

Table 1: Improve result (the matrix size is 5000×1000)

Linear System					
	Multiple		Single		
r	#	Total Time	#	Total Time	Pre-Compute
5	1	0.116	1000	0.250	0.189
10	1	0.099	1000	0.255	0.169
20	6	0.127	1005	0.438	0.266
40	576	0.262	1923	1.566	0.787
80	2015	0.293	3014	3.195	1.600
100	2231	0.358	3230	4.401	2.230

The most time-consuming part of the ISM algorithm is to solve the linear system, so we hope that by reducing the number of linear systems that need to be solved, the efficiency of the algorithm can be improved. When the NNLS problem is changed from a single right-side vector to multiple right-side vectors, it needs to be solved one by one, the number of linear systems to be solved is greatly increased, and the efficiency is greatly reduced. In the beginning, we form a sparse matrix with r matrices (Q) on the diagonal, and transform a multiplicative linear system of vectors into a linear system with the variables as matrices. We hope this approach can reduce execution time and improve efficiency. Although the goal has been achieved, the efficiency is still not good enough.

In each iteration, we solve the linear system according to a different index set `I`, so if we can find vectors with the same index set `I`, we can collect those vectors together and solve the linear system once. This avoids the problem of repeating computations for Q in the Cholesky decomposition. As mentioned in Section 3, A will be a thin and long matrix, and X will be a wide and flat matrix (i.e. $q \ll s$), because it will be H or W^T which in the subproblem of NMF. In this special case, the value of combinations C_q^s will not be very large. Since there are not many choices, it may happen that different vectors have the same index set `I` among the s vectors, which can reduce the number of linear systems to be solved. As shown in the Table 1, when q is smaller, the efficiency will be higher; if q is larger, the worst case will only change back to one-by-one operation. But r is usually much smaller than $\min(m, n)$ in NMF ($q \ll \min(p, s)$ in NNLS problem), so this problem is less likely to occur. Therefore, by reordering the vectors in C, the vectors with the same index set `I` are formed into a group, and the linear system can be solved sequentially, which can improve the efficiency. Figure 4 is the concept of rearranging vectors.

Finally, in order to avoid recomputing vectors that already satisfy the stopping criterion, the vectors that do not satisfy the stopping criterion will

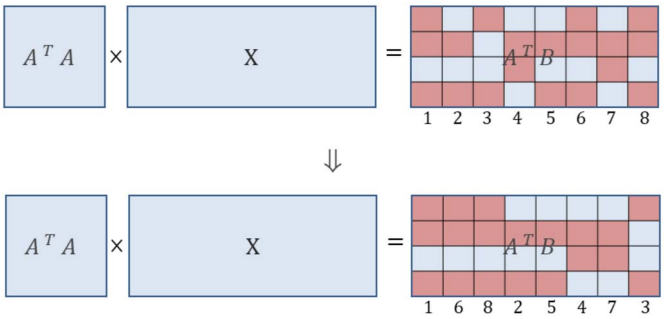

Figure 4: The idea of improving multiple right-hand sides case.

be picked out before solving the linear system. Even if the number of linear systems needed to be solved is reduced, Repeatedly calculating a vector that is already an optimal solution will only reduce efficiency. Therefore, computing only vectors that do not satisfy the stopping criterion is also a major factor in reducing computation time. The last improvement will be shown in Algorithm 2, and the complete algorithm after improved will be presented at Algorithm 3.

The Algorithm 2 is referring from [10] and line 2-6 are the matlab code. The function *sortrows* can sort rows of a matrix. First output is the sorted matrix, and second output will returns an index vector **Inx** which describes the order of the sorted rows. After line 2, the columns have same elements will be grouped together. Then we should know how many groups we have and what elements are in the columns inside those groups. Hence, we use *diff* for matrix to find the row differences and *any* to mark columns with any non-zero elements as TRUE. Then obtain the index of the first column of each group by function *find*, since it can find indices of nonzero elements. Finally, which columns (cols) and the indices (var) need to be solved in group

Algorithm 2 Y =Reorder$(Q, C, \mathtt{I})$

1. Given $Q \in \mathbb{R}^{q\times q}, C \in \mathbb{R}^{q\times s}$, and index set $\mathtt{I} \in \mathbb{R}^{q\times s}$.
2. Let $[\ \tilde{\mathtt{I}}, \mathbf{Inx}\ ] = \text{sortrows}(\mathtt{I}^T)$.
3. Set $G = \text{any}(\text{diff}(\tilde{\mathtt{I}})^T)$ and $\mathbf{g} = [\ 0\ \text{find}(G)\ s\ ]$.
4. For $i = 1, 2, ..., \text{length}(\mathbf{g}) - 1$
5. cols $= \mathbf{Inx}(\mathbf{g}(i) + 1 : \mathbf{g}(i + 1))$.
6. var $= \mathtt{I}(:, \text{cols}(1))$.
7. Solve the linear system Q(var,var)Y(var,cols) $= C$(var,cols).
8. EndFor

Algorithm 3 Index search method (Multiple)

1. Compute $R_0 = -B$, $U_0 = A^T R_0$, $Q = A^T A$ and $C = A^T B$.
   Given $X_0 = 0$ and $\mathsf{tol} > 0$. Set $Y = -1$.
2. For $k = 0, 1, 2, ...$
3. $\mathtt{J}_k = \{i \in \mathbb{N} | X_k(i) > 0\}$, $\mathtt{N}_k = \{i \in \mathbb{N} | U_k(i) < 0\}$ and $\mathtt{I}_k = \mathtt{J}_k \cup \mathtt{N}_k$.
4. While $\mathbf{Y} \not> 0$
5. $Cols = \{i \in \mathbb{N} |$ the $i\text{-}th$ column which contains nonnegative element$\}$.
6. $Y(:, Cols) = \text{Reorder}(Q, C(:, Cols), \mathtt{I}(:, Cols))$.
7. Find the index set $\mathtt{P} = \{i \in \mathbb{N} | Y(i) > 0\}$.
8. Set $\mathtt{I} = \mathtt{I}(\mathtt{P})$.
9. EndWhile
10. Set $\mathtt{J} = \mathtt{I}$ and $X = P_{\mathtt{J}} \begin{bmatrix} \mathrm{Y} \\ 0 \end{bmatrix}$.
11. Compute $R_{k+1} = AX_{k+1} - B$.
12. $U_{k+1} = A^T R_{k+1}$.
13. EndFor
14. **until** $|\min(U_{k+1})| \leq \mathsf{tol}$. (seek dual feasibility)

i can be picked out. So we can solve the linear system group by group, then get output Y.

4. Numerical experiments

We compare ISM with projected gradient Method (PG) [13], block principle pivoting (BP) [10], multiplicative update algorithm (MULT) [12] and alternating least squares (ALS) [1] by using the same stop criterion for NMF. The stopping criterion for NMF is based on the objective function. We have two stopping criterions, designed for different experiments. It is mainly divided into two types: strict and loose, which are introduced in detail below. Strict stopping criterion is based on KKT conditions, and the following residual is deduced in [10] as

$$\triangle = \frac{\delta}{\delta_W + \delta_H},$$

where

$$\delta = \sum_{i=1}^{m} \sum_{\ell=1}^{r} |\min(W_{i\ell}, (\partial f(W,H)/\partial W)_{i\ell})| + \sum_{\ell=1}^{r} \sum_{j=1}^{n} |\min(H_{\ell j}, (\partial f(W,H)/\partial H)_{\ell j})|,$$

$$\delta_W = \#(\min(W, \partial f(W,H)/\partial W) \neq 0),$$
$$\delta_H = \#(\min(H, \partial f(W,H)/\partial H) \neq 0).$$

The purpose of removing non-zero elements is to make the size of W and H independent of the residual, in other words, to normalize the residual. The stopping criterion will use this residual and be defined as

$$\triangle \leq \epsilon \triangle_0,$$

where $\triangle_0$ is the value of $\triangle$ by using initial values and ϵ is a chosen tolerance. We set ϵ as 10^{-4} in the numerical experiments. To achieve fairness among multiple algorithms, we choose initial values instead of the value after the first iteration to calculate $\triangle_0$.

Loose stopping criterion is designed by relative error. Loose stopping criterion is used in image applications, especially image restoration. We consider this result to be sufficient for the image when the relative error is small enough. The color in the image is divided into 255, the value is from small to large, and the color is from dark to light. Adjacent numerical values will be very similar in color and cannot be distinguished by the human eye. In addition, the current image pixels are very large, rather than waiting a long time to obtain accuracy, it is much more efficient to obtain similar results in a shorter time. So for the image we set this stop criterion, defined as

$$\triangle_{k-1} - \triangle_k \leq \epsilon \triangle_0,$$

where

$$\triangle_k = \frac{\|V - W_k H_k\|_F}{\|V\|_F}.$$

$\triangle_0$ has the same setting as strict stop criterion by using initial values and ϵ is a chosen tolerance. We set ϵ as 10^{-4} in the numerical experiments. It is to achieve fairness in comparison with many algorithms.

4.1. Experimental results with synthetic datasets

In this subsection, three datasets are used as our test samples, including synthetic images, images, and RGB images. The synthetic image generation method is the first numerical experiment in [10]. First, for $r = 5, 10, 20, 30, 40, 60$ and 80, randomly generate W and H with a sparsity of 40%. Second, compute $V = WH$ and add Gaussian noise with a standard deviation of 5%. Then normalize, so that all the V generated by r can have

Table 2: Experimental results on synthetic datasets

	r	ISM	BP	PG	MULT	ALS
iterations	5	11.4	15.5	13.5	966.8	>1000
	10	17	19.9	24	>1000	>1000
	20	20.7	24.9	22.6	>1000	>1000
	30	26.4	27.6	33.2		
	40	31.6	33.8	40.2		
	60	47.4	52.4	61.5		
	80	72.7	86.3	150.1		
residual	5	0.04845	0.04845	0.04845	0.04845	0.04847
	10	0.04792	0.04792	0.04792	0.04795	0.04809
	20	0.04651	0.04651	0.04653	0.04664	0.04761
	30	0.04580	0.04580	0.04581		
	40	0.04446	0.04446	0.04465		
	60	0.04233	0.04233	0.04236		
	80	0.03961	0.03961	0.03963		
time	5	0.04215	0.04123	0.02095	0.59147	0.96277
	10	0.14345	0.12306	0.05384	1.21621	1.71411
	20	0.35128	0.22358	0.10089	2.11936	2.97298
	30	0.58829	0.31406	0.21464		
	40	0.96627	0.46409	0.40370		
	60	2.36516	1.13260	2.20056		
	80	5.70527	2.62258	7.15772		

the same mean element-wise magnitude. Finally, a sparse matrix V with implicit information can be obtained.

To be fair, all algorithms start with the same initial value, and 10 experiments yield 10 different initial values to prevent special cases. Since it is a numerical experiment, strict stopping criterion is used, and finally the results of 10 experiments for each algorithm will be averaged and presented in Table 2. As shown in Table 2, in this experiment, the number of iterations of MULT and ALS is much larger than other algorithms. When r is small, they can still have about the same relative error. But when r increases, the relative errors of MULT and ALS cannot be more accurate. Therefore, the comparison of the two algorithms is not performed after r exceeds 20. In contrast, the remaining three algorithms are relatively stable. But as r increases, the steps of PG also start to increase slowly, although not as steep as MULT and ALS. This is something that did not happen to ISM and BP. This also highlights the stability of their number of iterations with respect to r.

From the residual, it can be found that PG is in a small weak position, because PG does not solve the linear system, and if the stopping criterion is

not adjusted, it will not have a good residual. But under the same criterion, ISM and BP can maintain the best residuals, which is also a big advantage of this type of method. In addition, it can be seen that the number of iteration steps of ISM is the least. From the small number of steps required and the same residual, it can be seen that the residual decline rate of ISM is faster than that of BP, so we infer that the way of ISM to find the index set `I` is better than that of BP.

4.2. Experimental results with image datasets

For the image, we perform image compression on a grayscale image with a size of 1164×1406 pixels and observe the results to compare the pros and cons of each algorithm. This experiment performed image compression with $r = 50, 100$ and 200 respectively, and the compression results are presented in Figure 6 as the image. From Figure 6, it can be clearly seen that ALS has obvious distortion problems. The restored picture can no longer display the color correctly, and even some exposure situations, that is to say, the restored matrix is too far away from the original matrix.

In addition, the performance of PG is extremely poor when $r = 100$ and 200. As we know, the accuracy of PG is of a certain level, so it is guessed that the poor performance should be related to the speed of the residual drop. It can be seen from Figure 5 that our speculation is not wrong. Due to the slow decline of the PG residual, the stop criterion is reached early, resulting in poor results. Although the speed is faster when using PG, there is a need to adjust ϵ for different situations. Under the same criterion, other algorithms will not have this problem. Finally, although MULT has roughly

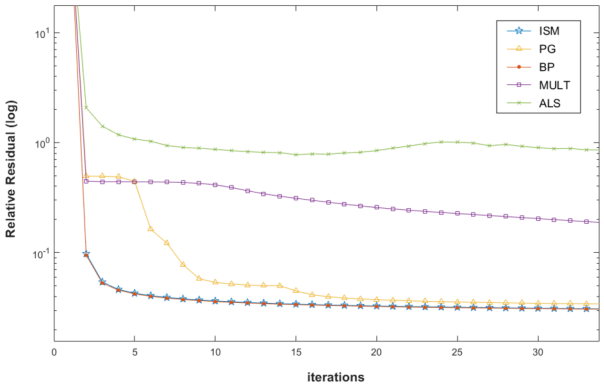

Figure 5: The residual comparison of all algorithms.

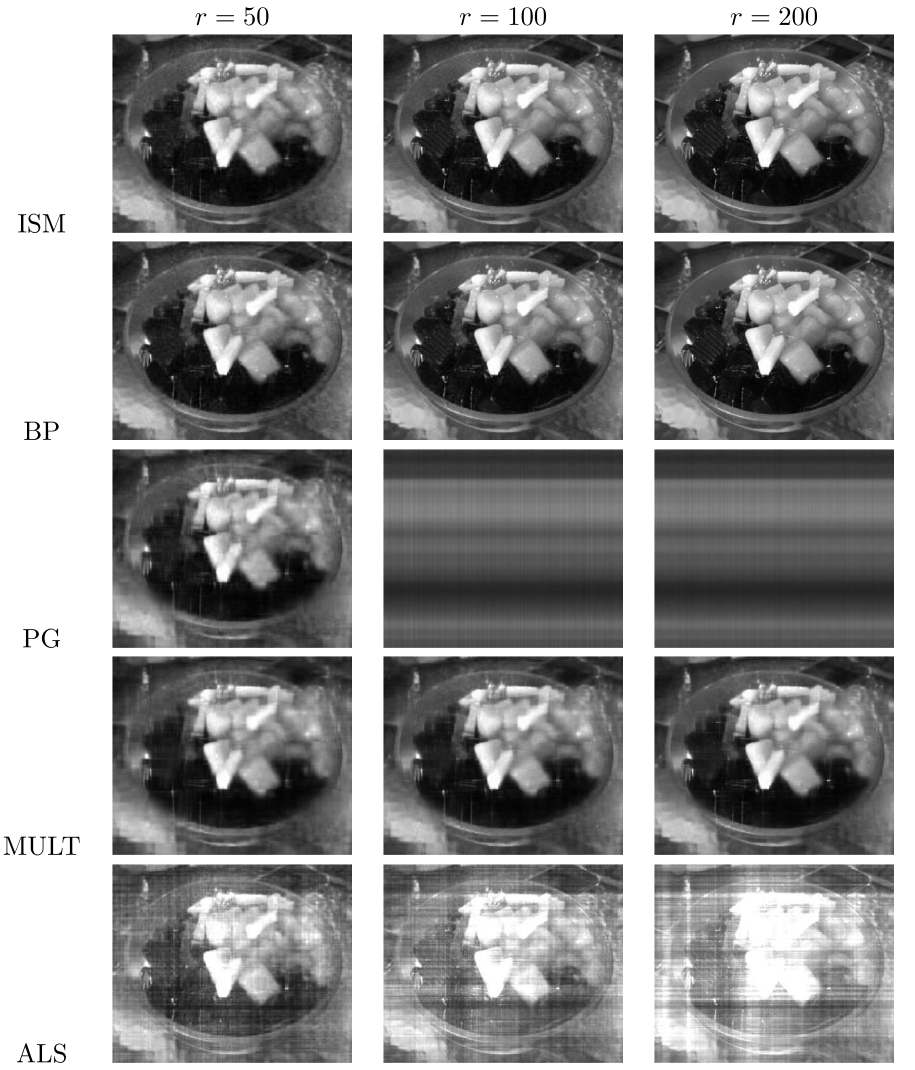

Figure 6: The result of all algorithms for experiment 2.

restored images, the results are still not satisfactory compared to ISM and BP. Therefore, the following experiment will exclude ALS and MULT.

5. Conclusion

In this paper, the ISM method is introduced to solve the NNLS problem. We apply it to the subproblems of NMF, use the method of reordering the

right side to improve the efficiency, and finally we compare it with other algorithms. It can be seen from the experimental results that BP and ISM can obtain the smallest error, and the number of outer iteration steps does not increase sharply with the increase of r in NMF. Although PG does not perform as well as them in these two results, good image results can also be obtained by adjusting the stopping criterion. In addition, when r is small, the time spent by PG is very small. Therefore, if the problem is a case where r is small, it is recommended to use PG as the solution to the NMF subproblem. If r is large and high accuracy is desired, BP can be used as the solution to the NMF subproblem. Finally, it can be seen from experiment 2 that ISM has good stability for sparse matrices. Therefore, if the matrix to be decomposed has high sparsity or is relatively uniform, ISM can be used as the solution method of the NMF subproblem to obtain a more stable NMF. The above are the results obtained by numerical experiments, and we can continue to discuss related issues in the future.

Acknowledgements

We would like to thank Yi-Shin Cheng for her master's thesis. This work is supported by the National Science and Technology Council in Taiwan.

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(1): 155–173, 2007. MR2409971

[2] S. Boyd and L. Vandenberghe. Convex Optimization. U.K., Cambridge: Cambridge Univ. Press, 215–243, 2004. MR2061575

[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(12): 4164–4169, 2004.

[4] F. E. Curtis, Z. Han, and D. P. Robinson. A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization. Computational Optimization and Applications, 60(2): 311–341, 2015. MR3316681

[5] 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. MR1746833

[6] P. Hungerländer and F. Rendl. A feasible active set method for strictly convex quadratic problems with simple bounds. SIAM J. Optim., 25(3): 1633–1659, 2015. MR3384841

[7] 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, 343–354, 2007. MR2407162

[8] 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. MR2421467

[9] H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl., 30(2): 713–730, 2008. MR2421467

[10] J. Kim and H. Park. Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons. 2008 Eighth IEEE International Conference on Data Mining, 353–362, 2008.

[11] Y.-C. Kuo and C.-S. Liu. Index Search Method for solving the Nonnegative Least Squares Problems.

[12] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, Advances in Neural Information Processing Systems 13, 556–562, 2001.

[13] C.-J. Lin. Projected Gradient Methods for Non-negative Matrix Factorization. Neural Computation, 19(10): 2756–2779, 2007. MR2348161

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

[15] 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. MR4051191

[16] 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. MR2388467

[17] G.-J. Song and M. K. Ng. Nonnegative Low Rank Matrix Approximation for Nonnegative Matrices. Appl. Math. Lett. 105: 106300, 2020. MR4072844

[18] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. ACM SIGIR Conference on Information Retrieval, 267–273, 2003.

Yi-Shin Cheng
Department of Applied Mathematics
National University of Kaohsiung
Kaohsiung 811
Taiwan
E-mail address: s93177@gmail.com

Ching-Sung Liu
Department of Applied Mathematics
National University of Kaohsiung
Kaohsiung 811
Taiwan
E-mail address: chingsungliu@nuk.edu.tw

Received August 23, 2022