Efficient Rank-one Residue Approximation Method for Graph Regularized Non-negative Matrix Factorization

Qing Liao and Qian Zhang

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{qnature,qianzh}@cse.ust.hk

Abstract. Nonnegative matrix factorization (NMF) aims to decompose a given data matrix X into the product of two lower-rank nonnegative factor matrices UV^T. Graph regularized NMF (GNMF) is a recently proposed NMF method that preserves the geometric structure of X during such decomposition. Although GNMF has been widely used in computer vision and data mining, its multiplicative update rule (MUR) based solver suffers from both slow convergence and non-stationarity problems. In this paper, we propose a new efficient GNMF solver called rank-one residue approximation (RRA). Different from MUR, which updates both factor matrices (U and V) as a whole in each iteration round, RRA updates each of their columns by approximating the residue matrix by their outer product. Since each column of both factor matrices is updated optimally in an analytic formulation, RRA is theoretical and empirically proven to converge rapidly to a stationary point. Moreover, since RRA needs neither extra computational cost nor parametric tuning, it enjoys a similar simplicity to MUR but performs much faster. Experimental results on real-world datasets show that RRA is much more efficient than MUR for GNMF. To confirm the stationarity of the solution obtained by RRA, we conduct clustering experiments on real-world image datasets by comparing with the representative solvers such as MUR and NeNMF for GNMF. The experimental results confirm the effectiveness of RRA.

Keywords: Nonnegative matrix factorization, Manifold regularization, Rank-one residue iteration, Block coordinate descent.

1 Introduction

Given n data points arranged in a nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$ and m stands for the dimension of the data, nonnegative matrix factorization (NMF) decomposes X into the product of two lower-rank nonnegative factor matrices, that is, UV^T, where $U \in \mathbb{R}_+^{m\times r}$ and $V \in \mathbb{R}_+^{n\times r}$ signify the basis of the low-dimensional space and the coefficient, respectively. Although NMF performs well in several tasks, it completely ignores the property where many datasets, for example, human faces or hand-written digits, reside in a manifold that lies in a low-dimensional space. Recently, Cai et al. [3] proposed graph regularized NMF

(GNMF) to address this problem. GNMF assumes that the neighborhoods of one data point in high-dimensional space should be as close as possible to the images of that point in low-dimensional space. Since GNMF preserves the geometric structure of the data set, any label information can be propagated along the surface of the manifold to its neighbourhoods. Such an advantage greatly enhances the clustering performance of NMF and makes GNMF a powerful tool in data mining [2][18].

Recently, many GNMF variants have been proposed for various applications. Zhang et al. [18] proposed a topology preserving NMF (TPNMF) for face recognition. TPNMF considers the manifold structure in face datasets and preserves the local topology while face images are transformed to a subspace. For the same purpose, Gu et al. [6] proposed neighborhood preserving NMF (NPNMF). NPNMF represents each data point by a linear combination of its neighbours in high-dimensional space and keeps such relationships in low-dimensional space with the same combination coefficient. Yang et al. [17] developed non-negative graph embedding (NGE) to integrate both intrinsic graph and penalty graph in NMF in the spirit of marginal fisher analysis [16]. Guan et al. [7] proposed a manifold regularized discriminative NMF (MD-NMF) for classification tasks. MD-NMF preserves both local geometry and label information of data points simultaneously by expecting data points in the same class close to be each other and data points in different classes far from each other. Shen and Si [13] proposed an NMF on multiple manifolds method (MM-NMF) to model the intrinsic geometrical structure of data on multiple manifolds. MM-NMF assumes the data points are drawn from multiple manifolds, if one data point can be reconstructed by several neighbourhoods on the same manifold in high-dimensional space, it can also be reconstructed in a similar way in low-dimensional space.

Although GNMF and its variants perform well in many fields, the multiplicative update rules (MUR) based algorithm suffers two drawbacks: 1) MUR converges slowly because it is intrinsically a first-order gradient descent method, and 2) MUR does not guarantee convergence to any stationary point [10]. These two deficiencies seriously prohibit GNMF from practical applications. To remedy these problems, Guan et al. [8] has recently proposed a NeNMF method for optimizing GNMF. NeNMF applies Nesterovs's method to alternatively update each factor matrix. Since Nesterovs's method updates each factor matrix in a second-order convergence rate, it converges rapidly for optimizing GNMF. However, NeNMF needs a stop condition to check when to stop Nesterovs's method, and it is non-trivial to determine such tolerance on many datasets.

In this paper, we propose an efficient rank-one residue approximation (RRA) method for GNMF and its variants. RRA decomposes the reconstruction UV^T into a summation of r rank-one matrices, i.e., $X \approx \sum_{i=1}^{r} U_{\cdot i} V_{\cdot i}^T$. In contrast to MUR which recursively updates the whole factor matrix, RRA recursively updates each column of each factor matrix with the remaining variables fixed, that is, $U_k V_k^T \approx X - \sum_{l \neq k}^{r} U_l V_l^T$ for each $1 \leq k \leq r$. It is obvious that each column of U can be updated in an analytic formulation based on non-negative least squares, but it is difficult to update columns of V due to the incorporated manifold regularization term. In this paper, we show that each column of V can also be updated in an analytic formulation. Since the objective function is continuously differentiable on a Cartesian product of $2r$ closed convex sets, RRA is proved to converge to a stationary point. However, an inverse of the Hessian matrix is needed in updating each column of V and such matrix inverse operator costs too much computational time. To overcome such deficiency, we introduce the Sherman-Morrison-Woodbury (SMW) formula to approximate the inverse of Hessian matrix. Since the approximation can be efficiently calculated in advance, the SMW-based formula greatly reduces the time cost of RRA. Experimental results on real-world datasets show the efficiency of RRA compared with representative GNMF solvers. To evaluate the effectiveness of RRA for GNMF, we conduct clustering experiments on two popular image datasets including COIL-20 [12] and CMU PIE [14], the experimental results show that RRA is effective.

The rest of the paper is organized as follows. Section 2 briefly reviews the graph regularized NMF (GNMF) method and its state-of-the-art solvers. Section 3 presents the RRA method for GNMF. Section 4 evaluate the efficiency and effectiveness and gives the experimental results, while Section 5 summaries this paper.

2 Related Works

Nonnegative matrix factorization (NMF) is a popular dimension reduction method which has been widely used in pattern recognition and data mining. Given a dataset $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}_+^{m\times n}$, where each column of X presents an data point, NMF aims to find two lower-rank nonnegative matrices $U \in \mathbb{R}_+^{m\times r}$ and $V \in \mathbb{R}_+^{n\times r}$, where $r \le \min\{m, n\}$ is the reduced dimensionality, such that their product UV^T approximates the original matrix X:

$$X = UV^T + E \tag{1}$$

where E denotes the residual error. Assuming the entries in E to be I.I.D. Gaussian distributed, we get the objective function of NMF:

$$\min_{U\ge 0, V\ge 0} \frac{1}{2}\|X - UV^T\|_F^2 \tag{2}$$

where $\|\cdot\|_F$ signifies the Frobenius norm. The smaller the cost function, the better the approximation of UV^T.

Since NMF does not consider the intrinsic geometrical structure of dataset, it does not always perform well in some real-world datasets, for example, face images and hand-written digits. To remedy this problem, Cai et al. [3] proposed graph regularized NMF (GNMF), which considers the geometrical structure of the dataset in NMF. The basic assumption is that data points reside on the surface of a manifold that lies in a low-dimensional space, that is, if two data points are close enough in high-dimensional space they are still close in low-dimensional space. To this end, GNMF constructs an adjacent graph G on a scatter of data points to represent the local geometric structure. In graph G, each node associates an data point and an edge is established between two nodes once one node belongs to the k nearest neighbourhoods of another. Based on G, we can build an adjacent matrix W as follows:

$$W_{ij} = \begin{cases} 1, & x_j \in N_k(x_i) \mid x_i \in N_k(x_j) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $N_k(x_i)$ denotes the k nearest neighbourhoods of x_i. We are now ready to preserve the geometrical structure of X in the low-dimensional space, the objective is to minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\|v_i - v_j\|_2^2 W_{ij} = tr(V^T L V) \tag{4}$$

where $L = D - W$ is the Laplacian matrix of G, where D is a diagonal matrix and $D_{jj} = \sum_l W_{jl}$, and $tr(\cdot)$ signifies the trace operator over a symmetric matrix. Combing (2) and (4) together, we arrive at the objective of GNMF:

$$\min_{U \geq 0, V \geq 0} f(U,V) = \frac{1}{2}\|X - UV^T\|_F^2 + \frac{\beta}{2} tr(V^T L V) \tag{5}$$

where $\beta > 0$ is a trade-off parameter over the manifold regularization term.

Although $f(U,V)$ is jointly non-convex w.r.t. both U and V, it is convex w.r.t. either U or V. Therefore, we can apply block coordinate descent method [1] to solve (5). Cai et al. [3] proposed a multiplicative update rule (MUR) to recursively update matrices U and V, respectively. At the $t-th$ iteration round, U_{t+1} and V_{t+1} are updated as follows:

$$U_{t+1} = U_t \circ \frac{XV_t}{U_t V_t^T V_t} \tag{6}$$

and

$$V_{t+1} = V_t \circ \frac{X^T U_{t+1} + \beta W V_t}{V_t U_{t+1}^T U_{t+1} + \beta D V_t} \tag{7}$$

where $\circ$ signifies the element-wise multiplication.

Although MUR is proved to reduce the objective function $f(U,V)$, it converges slowly in terms of number of iteration because it is intrinsically a first-order gradient descent method. In addition, MUR suffers from non-stationarity problems like NMF [10].

Recently, to remedy the problem of MUR, Guan et al. [8] proposed a NeNMF algorithm to update both factor matrices by solving the following two subproblems

$$\min_{U \geq 0} \frac{1}{2}\|X - UV^T\|_F^2 \tag{8}$$

and

$$\min_{V \geq 0} \frac{1}{2}\|X - UV^T\|_F^2 \tag{9}$$

Since NeNMF solves both (8) and (9) with Nesterovs's method [8], it converges rapidly because Nesterovs's method is intrinsically a second-order gradient descent method. However, NeNMF needs a tolerance to check the stopping of the Nesterovs's method. If the tolerance is set too small, NeNMF costs too much extra iterations for solving (8) and (9). If the tolerance is set too large, NeNMF gets a low-quality solution for either (8) or (9). Therefore, it is not easy to use NeNMF in practical applications because it is non-trivial to determine a suitable tolerance on many datasets.

3 Rank-one Residue Approximation for GNMF

In this section, we proposed an efficient rank-one residue approximation (RRA) algorithm to overcome the deficiencies of both MUR and NeNMF by recursively updating columns of U and V with an analytic formulation.

The main idea is inspired by the well-known rank-one residue iteration (RRI, [9]) method and hierarchical alternating least squares (HALS, [5]) method for NMF. In contrast to MUR and NeNMF which alternately updates the whole U and V, RRA recursively updates columns of them with the remaining variables fixed. For the $k-th$ column of U and V, the sub-problems are

$$\min_{U_{\cdot k}\geq 0} \frac{1}{2}\|R_k - U_{\cdot k}V_{\cdot k}^T\|_F^2 \tag{10}$$

and

$$\min_{V_{\cdot k}\geq 0} \frac{1}{2}\|R_k - U_{\cdot k}V_{\cdot k}^T\|_F^2 + \frac{\beta}{2}V_{\cdot k}^T L V_{\cdot k} \tag{11}$$

where R_k denotes the residue of X after eliminating the $k-th$ column of U and V, i.e., $R_k = X - \sum_{l\neq k} U_{\cdot l}V_{\cdot l}^T$. The formula (11) is derived from the following equation: $tr(V^T L V) = \sum_{k=1}^{r} V_{\cdot k}^T L V_{\cdot k}$.

The problem (10) should be solved in two cases, that is, $V_{\cdot k} = 0$ and $V_{\cdot k} \neq 0$. If $V_{\cdot k} = 0$, eq. (10) has an infinite number of solutions. Therefore, the $k-th$ column of both U and V should be taken off in the remaining computation. If $V_{\cdot k} \neq 0$, according to [3], the sub-problem (10) has a closed-form solution

$$U_{\cdot k} = \frac{\prod_+(R_k V_{\cdot k})}{\|V_{\cdot k}\|_2^2} \tag{12}$$

where $\prod_+(x) = \max(0, x)$ is an element-wise projection that shrinks negative entries of x to zero. Similar to (10), the problem (11) should be considered in two cases, that is, $U_{\cdot k} = 0$ and $U_{\cdot k} \neq 0$. If $U_{\cdot k} = 0$, the $k-th$ column of both U and V does not take part in the remaining computation and should be taken off. If $U_{\cdot k} \neq 0$, below we show how to solve (11) in an analytic formulation though it is not as direct as (12).

We solved the constrained optimization (11) by using the Lagrangian multiplier method [1]. The Lagrangian function of (11) is

$$\mathcal{L} = \frac{1}{2}\|R_k - U_{\cdot k}V_{\cdot k}^T\|_F^2 + \frac{\beta}{2}V_{\cdot k}^T L V_{\cdot k} - \langle V_{\cdot k}, \lambda\rangle \tag{13}$$

where λ is the Lagrangian multiplier for the constraint $V_{\cdot k} \geq 0$. Based on the K.K.T. conditions, the solution of (11) satisfies

$$\begin{cases} V_{\cdot k} \geq 0, \lambda \geq 0, \\ \frac{\partial \mathcal{L}}{\partial V_{\cdot k}} = -R_k^T U_{\cdot k} + (\|U_{\cdot k}\|_2^2 I + \beta L) V_{\cdot k} - \lambda = 0 \\ \lambda \circ V_{\cdot k} = 0 \end{cases} \tag{14}$$

where $I \in \mathbb{R}^{n \times n}$ is an identity matrix. With simple algebra, based on (14), we update columns of V as follows:

$$V_{\cdot k} = \prod_{+}((\|U_{\cdot k}\|_2^2 I + \beta L)^{-1} R_k^T U_{\cdot k}) \tag{15}$$

by updating columns of U and V alternatively with (12) and (15), respectively, until convergence, RRA solves GNMF. The following **Proposition 1** shows that alternating (12) and (15) reaches a stationary point. The proof is similar to [9], we only include it here for completeness.

Proposition 1. *Every limited point generated by alternating (12) and (15) is a stationary point.*

Proof. From (10) and (11), we know that the feasible sets of $U_{\cdot k}$ and $V_{\cdot k}$ are $\Omega_k^U \subset \mathbb{R}_+^m$ and $\Omega_k^V \subset \mathbb{R}_+^n$. According to [11], since X is bounded, we can set an upper bound for Ω_k^U andΩ_k^V and thus can consider them as closed convex sets.

Therefore, the GNMF problem can be written as a bound-constrained optimization problem

$$\min_{[U,V] \in \Omega} \frac{1}{2}\|X - \sum_{k=1}^{r} U_{\cdot k} V_{\cdot k}^T\|_F^2 + \frac{\beta}{2} \sum_{k=1}^{r} V_{\cdot k}^T L V_{\cdot k} \tag{16}$$

where $\Omega = \prod_{k=1}^r \Omega_k^U \times \prod_{k=1}^r \Omega_k^V$ is a Cartesian product of closed convex sets. Since the objective function of (16) is continuously differentiable over Ω and RRA updates the $k-t$h column of $U$and V with the optimal solutions of (12) and (15), according to **Proposition 2.7.1** in [1], every limit point generated by alternating (12) and (15) is a stationary point.

For completeness, we must consider cases when either $U_{\cdot k}$ or $V_{\cdot k}$ is zero. In these cases, eq. (12) and (15) do not give unique solutions, and **Proposition 2.7.1** in [1] cannot be applied. As mentioned above, such columns should be taken off without changing the value of the objective function of (16). Therefore, these columns do not destroy the theoretic analysis. It completes the proof.

Since L is a positive semi-definite matrix, the matrix inverse operator $(\|U_{\cdot k}\|_2^2 I + \beta L)^{-1}$ in (15) is well-defined if $U_{\cdot k} \neq 0$. However, the matrix inverse operator is inefficient because its time cost is $O(n^3)$. Fortunately, the matrix $\|U_{\cdot k}\|_2^2 I + \beta L$ has a nice property, that is, it is composed of a symmetric positive semi-definite matrix and a diagonal matrix. This property motivates us to apply the well-known Sherman-Morrison-Woodbury (SMW, [15]) formula to approximate the matrix inverse operator efficiently. That is why our method is called rank-one

Algorithm 1. Rank-one Residue Approximation for GNMF

```
Input: $X \in \mathbb{R}_+^{m\times n}, L \in \mathbb{R}^{n\times n}, 1 \le r \le \min(m,n), \beta$
Output: $U \in \mathbb{R}_+^{m\times r}, V \in \mathbb{R}_+^{n\times r}$
 1: Initialize: $U^1 \in R_+^{m\times r}, V^1 \in R_+^{n\times r}, t = 1$
 2: Calculate: $L = \Theta\Sigma\Theta^T \approx \tilde{\Theta}\tilde{\Sigma}\tilde{\Theta}^T, R^1 = X - U^1{V^1}^T$
 3: repeat
 4:    for $k = 1 \dots r$ do
 5:       Calculate: $R_k^t = R^t + U_{\cdot k}^t {V_{\cdot k}^t}^T$
 6:       Update: $U_{\cdot k}^{t+1} = \prod_+(R_k^t V_{\cdot k}^t))/\|V_{\cdot k}^t\|_2^2)$
 7:       Calculate: $A_k^t \approx (\|U_{\cdot k}^{t+1}\|_2^2 I + \beta L)^{-1}$
 8:       Update: $V_{\cdot k}^{t+1} = \prod_+(A_k^t {R_k^t}^T U_{\cdot k}^{t+1})$
 9:       Update: $R^t = R_k^t - U_{\cdot k}^{t+1} {V_{\cdot k}^{t+1}}^T$
10:    end for
11:    Update: $R^{t+1} = R^t$
12:    Update: $t \leftarrow t + 1$
13: until The Stopping Condition (18) is Satisfied.
14: $U = U^t, V = V^t$
```

residue approximation (RRA). In particular, we can approximate the matrix inverse in (15) as

$$\begin{aligned}(\|U_{\cdot k}\|_2^2 I + \beta L)^{-1} &= \beta^{-1}(\frac{\|U_{\cdot k}\|_2^2}{\beta} I + \Theta\Sigma\Theta_T)^{-1} \\ &\approx \beta^{-1}(\frac{\|U_{\cdot k}\|_2^2}{\beta} I + \tilde{\Theta}\tilde{\Sigma}\tilde{\Theta}^T)^{-1} \\ = (\frac{1}{\|U_{\cdot k}\|_2^2)} I - \frac{\beta}{\|U_{\cdot k}\|_2^4)} &\tilde{\Theta}(\tilde{\Sigma}^{-1} + \frac{\beta}{\|U_{\cdot k}\|_2^2})^{-1}\tilde{\Theta}^T) \end{aligned} \tag{17}$$

where $L = \Theta\Sigma\Theta^T$ is a SVD of L, and $\tilde{\Theta}\tilde{\Sigma}\tilde{\Theta}^T$ is its approximation by taking the first p most important components. The value p is usually determined by keeping 95% of the energy, i.e., $\sum_{i=1}^p \sigma_i^2 / \sum_{i=1}^n \sigma_i^2 \le 95\%$, where σ_i^2 is the $i - th$ largest singular value. The following section will discuss how to choose the percentage of energy kept. The main time cost of (17) is spent on the calculation of the second term, whose time complexity is $O(n^2 p)$ because the inverse operator is performed over a diagonal matrix. Since $p \ll n$, the SMW formula greatly reduces the time cost of (16) from $O(n^3 + mn)$ to $O(n^2 p + mn)$.

By recursively updating columns of U and V with (12) and (15), respectively, we can solve GNMF efficiently without tuning extra parameters. The total procedure is summarized in **Algorithm 1**, where the stopping condition is given as follows:

$$|f(U^t, V^t) - f(U^{t+1}, V^{t+1})| \le \varepsilon |f(U^1, V^1) - f(U^2, V^2)| \tag{18}$$

where ε is the tolerance, for example, $\varepsilon = 10^{-3}$.

The proposed RRA algorithm avoids parameter tuning thus it is more flexible and convenient in practical applications. In **Algorithm 1**, since the SVD of L

can be calculated beforehand and the residue R_k^t can be updated recursively as lines 5 and 9 in $O(mn)$ time, the time complexity of RRA is mainly spent on line 7 because the matrix-vector multiplication in lines 6 and 8 cost $O(mn)$ time. In summary, the time complexity of one iteration round of RRA is $O(mnr + prn^2)$. According to [8], the time complexity of one iteration round of NeNMF is $O(mnr + mr^2 + nr^2) + K \times O(mr^2 + nr^2)$, where K is the number of iterations performed by Nesterovs's method. An unsuitable tolerance leads to a rather large K for NeNMF and pulls down its efficiency. RRA overcomes such deficiency and performs more efficiently than NeNMF without any parameter tuning. Although the time complexity of one iteration round of MUR is $O(mnr + rn^2)$ [3], RRA costs less time in total than MUR because it converges in far less iteration rounds.

4 Experiments

In this section, we evaluate the efficiency of our RRA method by comparing it with state-of-the-art GNMF solvers including MUR and NeNMF in terms of CPU seconds. In addition, we evaluate the clustering performance of the RRA based GNMF on popular image datasets, such as, COIL-20 [12] and CMU PIE [14] to confirm its effectiveness.

4.1 Preliminaries

We followed [3] to evaluate RRA on two popular image datasets including COIL-20 [12] and CMU PIE [14]. The COIL-20[1] image library contains images of 20 objects viewed from different angles. Totally 72 images were taken for each object and each image was cropped to 32×32 pixels and rescaled to an 1024-dimensional long vector. The CMU PIE[2] face image database contains face images of 68 individuals. There are totally 42 facial images for each individual taken under different lighting and illumination conditions. Similarly, each image was cropped to 32×32 pixels and rescaled to a 1024-dimensional long vector. In summary, the COIL-20 is composed of a 1024×1440 matrix and CMU PIE is composed of a 1024×2856 matrix.

4.2 Efficiency Evaluation

We evaluated the efficiency of RRA by comparing its CPU seconds with those spent by both MUR and NeNMF on whole COIL-20 and CMU PIE datasets. For GNMF, we set the number of neighborhoods to $k = 5$, the trade-off parameter $\beta = 1$, and $r = 10, 50, 100, 200$ to study the scalability of RRA. To keep the fairness of comparison, all GNMF solvers start from an identical point and stop when they reach an identical objective value. Then the CPU seconds it costs are compared for the purpose of evaluation.

[1] `http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php`
[2] `http://www.ri.cmu.edu/projects/project_418.html`

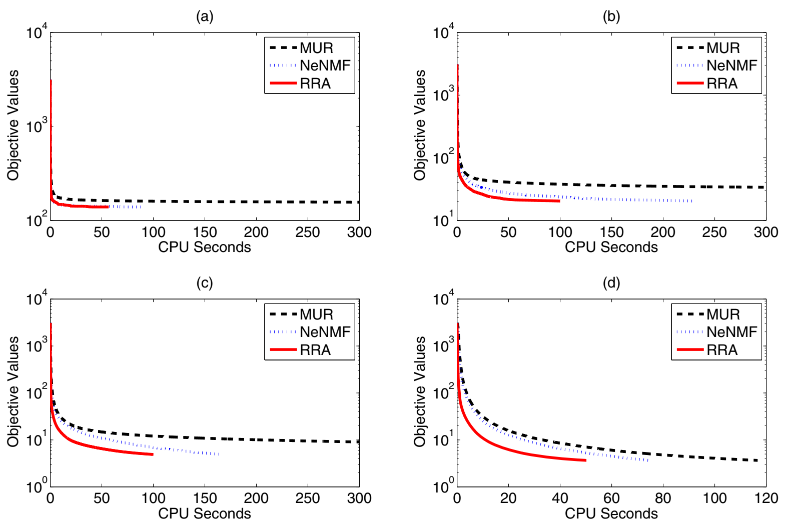

Fig. 1. Objective values versus CPU seconds of MUR, NeNMF, and RRA on COIL-20 dataset. (a) r=20, (b) r=50, (c) r=100, (d) r=200

Figure 1 compares the objective values versus CPU seconds of MUR, NeNMF and RRA on COIL-20 dataset. It shows that RRA performs much more rapidly than MUR and NeNMF because it reaches lower objective values in the same number of CPU seconds. In other words, to get a similar solution RRA costs far less CPU seconds. When the reduced dimensionality $r = 20$, NeNMF performs comparably with RRA because its time complexity $O(mnr + mr^2 + nr^2) + K \times O(mr^2 + nr^2 + rn^2)$ is dominated by $O(mnr) + K \times O(rn^2)$ which is comparable with the time complexity of RRA, that is, $O(mnr + prn^2)$. From subfigures (a) to (d), we can see that RRA always performs better and better than MUR, but it performs closely to NeNMF when the reduced dimensionality is 200.

Figure 2 compares the objective values versus CPU seconds of MUR, NeNMF and RRA on the CMU PIE dataset. From Figure 2, we have the same observation as Figure 1. It confirms that RRA is much more efficient than MUR and NeNMF. To study the speedup rate of RRA versus MUR and NeNMF, we repeated the experiments on the PIE dataset with r varying from 10 to 100. MUR, NeNMF, and RRA start from the identical initial point and stop when the same objective value is reached. Then, we calculated the speedup rate as the ratio of time costs of MUR (or NeNMF) dividing those of RRA. Such trial was repeated ten times with different randomly generated initial point.

Figure 3 gives the speedup rates of RRA versus NeNMF and MUR. It shows that RRA is much faster than both MUR especially when the reduced dimensionality is 10. RRA is also faster than NeNMF when the reduced dimensionality is 10, and costs comparable CPU time in other cases. This observation shows that RRA is much efficient when the reduced dimensionality is relatively small.

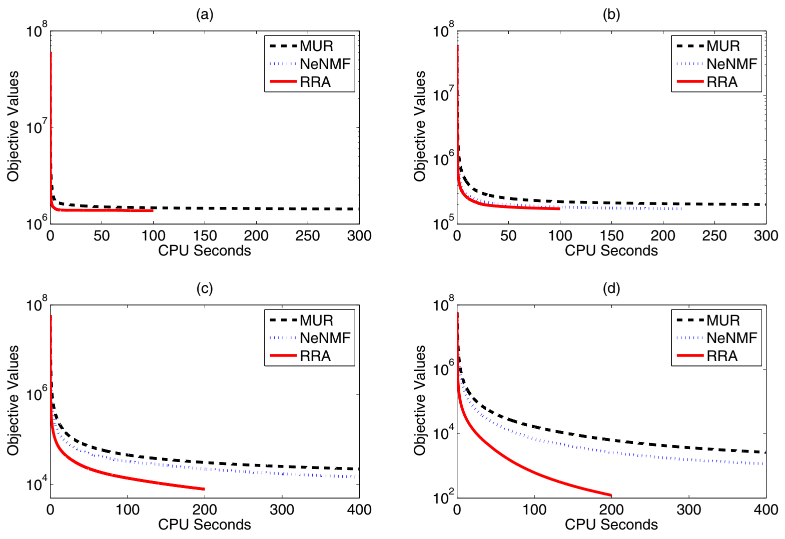

Fig. 2. Objective values versus CPU seconds of MUR, NeNMF, and RRA on PIE dataset. (a) r=20, (b) r=50, (c) r=100, (d) r=200.

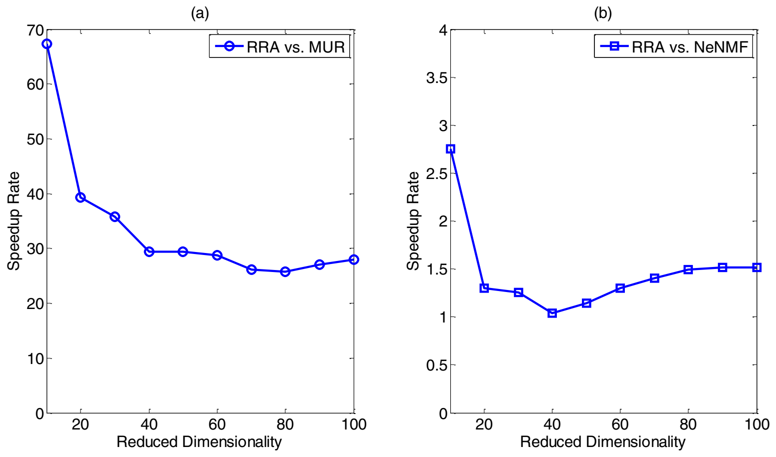

Fig. 3. Speedup rate versus reduced dimensionality: (a) RRA versus MUR and (b) RRA versus NeNMF

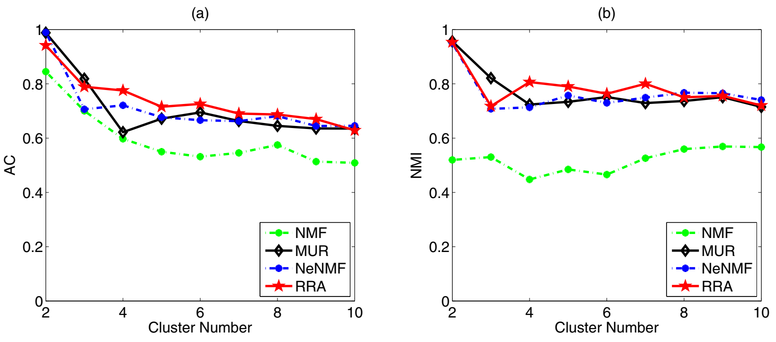

Fig. 4. AC and NMI of GNMF by MUR, NeNMF, and RRA on COIL-20 dataset. NMF is used as a baseline.

4.3 Image Clustering

Based on **Proposition 1**, RRA converges to a stationary point of GNMF while MUR does not. To evaluate the stationary level of RRA, we compared the clustering performance of GNMF solved by RRA and MUR, respectively, on both COIL-20 and CMU PIE datasets in terms of accuracy (AC) and normalized mutual information (NMI). The details of AC and NMI can be found in [3]. To keep the fairness of comparison, all GNMF solvers start from an identical initial point and stop when the same stopping condition (18) is satisfied. In this experiment, we randomly selected $r = 2, \ldots, 10$ individuals from both datasets and the selected images are clustered by using GNMF based on different solvers, that is, RRA, NeNMF, and MUR. Such trails were repeated ten times and the average AC and NMI are used to compare their performance. The standard NMF is also compared as a baseline.

Figure 4 compares the averaged AC and NMI by MUR, NeNMF, and RRA based GNMF on the COIL-20 dataset. It shows that the RRA slightly outperforms MUR and NeNMF in terms of AC and NMI. There are two reasons for this observation: 1) RRA gets a stationary point (see **Proposition 1**) which better approximates the data points and thus might perform better in clustering, and 2) RRA approximates the graph Laplacian L in (4) with the largest eigenvectors (see (17)), according to the spectral graph theory [4], these eigenvectors associate with most smooth functions over graph G, that is, RRA eliminates some outlier functions on the graph and thus RRA propagates the geometrical information better than the original GNMF method. For this reason, RRA clusters the data points better.

Figure 5 compares the averaged AC and NMI by MUR, NeNMF, and RRA based GNMF on the PIE dataset. Figure 5 confirms our analysis Figure 3. In summary, RRA is effective for optimizing GNMF.

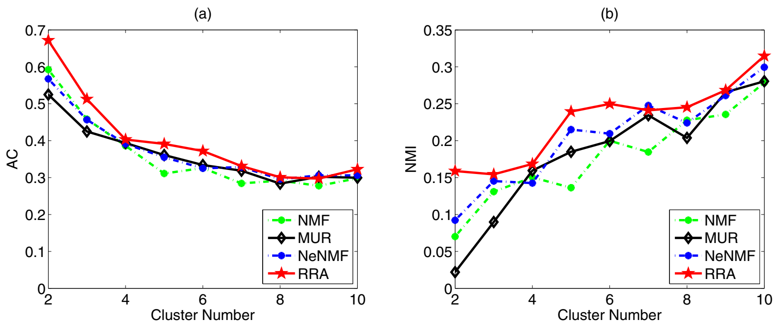

Fig. 5. AC and NMI of GNMF by MUR, NeNMF, and RRA on PIE dataset. NMF is used as a baseline.

Table 1. CPU seconds versus the percentage of energy kept for RRA on the PIE dataset

the percentage of energy kept	65%	75%	85%	95%	100%
CPU Seconds	199.18	184.59	183.92	177.76	209.47

4.4 Parameter Selection

In RRA, the percentage of energy kept in (17) controls the approximation quality of the SMW formula to the Hessian inverse as well as the time complexity because one iteration round of RRA costs $O(mnr + prn^2)$ time. The more energy kept, the larger the value p. It implies the higher the approximation quality and the less iteration rounds RRA spent. However, a large p means that each iteration round consumes much CPU time. So the percentage of energy kept must be carefully selected to balance the two facts. To study this point, we test RRA on PIE dataset when setting the reduced dimensionality to 10 and varying the percentage of energy kept from 65% to 95% with a step-size 10%. Table 1 compares the CPU time spent. To keep the fairness of comparison, all trials of RRA start from identical initial point and stop when same objective values are reached.

Table 1 shows that the time cost decreases with increasing of the percentage of energy kept. That is because the approximation quality is improved and iteration number is reduced in this case. However, the percentage of energy kept cannot be chosen too large such as 100% because that will increase the time complexity of each iteration round. In our experiment, 95% is a good choice.

5 Conclusion

In this paper, we proposed a novel efficient rank-one residue approximation (RRA) solver for graph regularized non-negative matrix factorization (GNMF). Unlike the existing GNMF solvers which recursively update each factor matrix as a whole, RRA recursively updates each column of both factor matrices in an analytic formulation. Although RRA needs a time-consuming matrix inverse operator, it can be approximated by using the Sherman-Morrison-Woodbury formula. Since the objective function of GNMF is continuously differentiable over a Cartesian product of several closed convex sets and RRA finds the optimal solution for each column of both factor matrices, RRA theoretically converges to a stationary point of GNMF. Experimental results and real-world image datasets confirm both the efficiency and the effectiveness of RRA.

References

1. Bertsekas, D.P.: Nonlinear programming. Athena Scientific (1999)
2. Cai, D., He, X., Han, J., Huang, T.S.: Graph regularized nonnegative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(8), 1548–1560 (2011)
3. Cai, D., He, X., Wu, X., Han, J.: Non-negative matrix factorization on manifold. In: Eighth IEEE International Conference on Data Mining (ICDM 2008), pp. 63–72. IEEE (2008)
4. Chung, F.R.: Spectral graph theory. In: CBMS Regional Conference Series in Mathematics, vol. 92 (1997)
5. Cichocki, A., Zdunek, R., Amari, S.-I.: Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization. In: Davies, M.E., James, C.J., Abdallah, S.A., Plumbley, M.D. (eds.) ICA 2007. LNCS, vol. 4666, pp. 169–176. Springer, Heidelberg (2007)
6. Gu, Q., Zhou, J.: Neighborhood preserving nonnegative matrix factorization. In: Proc. 20th British Machine Vision Conf. (2009)
7. Guan, N., Tao, D., Luo, Z., Yuan, B.: Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. IEEE Transactions on Image Processing 20(7), 2030–2048 (2011)
8. Guan, N., Tao, D., Luo, Z., Yuan, B.: Nenmf: an optimal gradient method for nonnegative matrix factorization. IEEE Transactions on Signal Processing 60(6), 2882–2898 (2012)
9. Ho, N.D., Van Dooren, P., Blondel, V.D.: Descent methods for nonnegative matrix factorization. In: Numerical Linear Algebra in Signals, Systems and Control, pp. 251–293. Springer (2011)
10. Lin, C.J.: On the convergence of multiplicative update algorithms for nonnegative matrix factorization. IEEE Transactions on Neural Networks 18(6), 1589–1596 (2007)
11. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. Neural Computation 19(10), 2756–2779 (2007)
12. Nene, S.A., Nayar, S.K., Murase, H.: Columbia object image library (coil-20). Technical Report CUS-005-96, Columbia Univ. USA (1996)
13. Shen, B., Si, L.: Nonnegative matrix factorization clustering on multiple manifolds. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, pp. 575–580 (2010)
14. Sim, T., Baker, S., Bsat, M.: The cmu pose, illumination, and expression database. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(12), 1615–1618 (2003)

15. Woodbury, M.A.: Inverting modified matrices. Memorandum Report 42, 106 (1950)
16. Xu, D., Yan, S., Tao, D., Lin, S., Zhang, H.-J.: Marginal fisher analysis and its variants for human gait recognition and content-based image retrieval. IEEE Transactions on Image Processing 16(11), 2811–2821 (2007)
17. Yang, J., Yang, S., Fu, Y., Li, X., Huang, T.: Non-negative graph embedding. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8. IEEE (2008)
18. Zhang, T., Fang, B., Tang, Y.Y., He, G., Wen, J.: Topology preserving non-negative matrix factorization for face recognition. IEEE Transactions on Image Processing 17(4), 574–584 (2008)