Towards Evolutionary Nonnegative Matrix Factorization

AAAI Press
Association for the Advancement of Artificial Intelligence
445 Burgess Drive
Menlo Park, California 94025

Abstract

Nonnegative Matrix Factorization (*NMF*) techniques has aroused considerable interests from the field of artificial intelligence in recent years because of its good interpretability and computational efficiency. However, in many real world applications, the data features usually evolve over time smoothly. In this case, it would be very expensive in both computation and storage to rerun the whole *NMF*procedure after each time when the data feature changing. In this paper, we propose *Evolutionary Nonnegative Matrix Factorization* (*eNMF*), which aims to incrementally update the factorized matrices in a computation and space efficient manner with the variation of the data matrix. We devise such evolutionary procedure for both asymmetric and symmetric *NMF*. Finally we conduct experiments on several real world data sets to demonstrate the efficacy and efficiency of *eNMF*.

Introductions

The recent years have witnessed a surge of interests on *Nonnegative Matrix Factorization* (*NMF*) from the artificial intelligence field (Lee and Seung 1999)(Lee and Seung 2001)(Lin 2007)(Kim and Park 2008). Differernt from traditional spectral decomposition methods such as *Principal Component Analysis* (PCA) and *Singular Value Decomposition* (SVD), *NMF* (1) is usually additive, which results in a better interpretation ability; (2) does not require the factorized latent spaces to be orthogonal, which allows more flexibility to adapt the representation to the data set. *NMF* has successfully been used in many real world applications, such as information retrieval (Shahnaz et al. 2006), environmental study (Anttila et al. 1995), computer vision (Guillamet, Bressan, and Vitrià 2001) and computational social/network science (Wang et al. 2010).

Formally, what *NMF* does is to factorize a nonnegative data matrix into the product of two (low-rank) nonnegative latent matrices. As *NMF* requires both factorized matrices to be nonnegative, this will generally lead to sparse, part-based representation of the original data set, which is semantically much more meaningful compared to traditional factorization/basis learning methods. Due to the empirical and theoretical success of *NMF*, people have been working on a lot of *NMF* extensions in the last decade to fit in more application scenarios. Some representative algorithms include nonnegative sparse coding (Eggert and Korner 2004), semi and convex *NMF* (Ding, Li, and Jordan 2010), and orthogonal tri-*NMF* (Ding et al. 2006).

Many algorithms have been proposed to solve *NMF*, such as multiplicative updates (Lee and Seung 2001), active set (Kim and Park 2008) and projected gradient (Lin 2007). However, all these algorithms require to hold the whole data matrix in main memory in the entire *NMF* process, which is quite inefficient in terms of storage cost when the data matrix large (either in data size or the feature dimensionality). To solve this problem, several researchers proposed memory efficient online implementations for NMF in recent years (Cao et al. 2007)(Wang, Li, and König 2011). Rather than processing all data points in a batch mode, these approaches process the data point one at a time in a streaming fashion. Thus they only require the memory to hold one data point through the whole procedure.

In this paper, we consider the problem of *NMF* in another scenario where the data features are evolving over time[1]. A straightforward solution is to rerun the whole *NMF* procedure at each time stamp when the data feature change. However, this poses several challenges in terms of *space cost*, *computational time* as well as *privacy*. Let $\mathbf{X}$ and $\widetilde{\mathbf{X}} = \mathbf{X} + \Delta\mathbf{X}$ be the old and new data feature matrices respectively. In many real applications, $\Delta\mathbf{X}$ is usually very sparse while $\widetilde{\mathbf{X}}$ is not[2]. It therefore is not efficient in terms of space cost to re-run *NMF* since we need to store the whole data feature matrix $\widetilde{\mathbf{X}}$. It is also not efficient in computation since it requires some matrix-matrix multiplication between $\widetilde{\mathbf{X}}$ and the two factorized matrices. What is more, this strategy becomes infeasible for those privacy-sensitive applications where the whole data feature matrix $\widetilde{\mathbf{X}}$ might not be available at a given time stamp. For instance, Facebook's[3]

[1]The difference between this setting and online learning is that in online learning, the data points are processed one by one, i.e., the elements in the data matrix are changed one column at a time. However, in our scenario, we allow any elements in the data matrix to change from time to time.

[2]Even if $\widetilde{\mathbf{X}}$ is also sparse, it is usually much denser compared with the $\Delta\mathbf{X}$ matrix. See table 1 for some examples.

[3]http://www.facebook.com/

privacy policy prohibits the user to keep the downloaded data longer than 24 hours. So, if a data analyst wants to track the community structure on the daily-base, s/he would only have the access to the data feature within a 24 hour window.

For evolutionary data, one common assumption is that the data features evolve *smoothly* over time (Chi et al. 2007), i.e., the norm of the difference between the data feature matrices at two consecutive time stamps is very small. Based on this assumption, we develop a novel *Evolutionary Nonnegative Matrix Factorization* (*eNMF*) algorithm in this paper, where we assume that the factorized matrices also evolve smoothly over time. Instead of minimizing a new similar objective on the evolved feature matrix, eNMF minimizes an upper bound of the objective, and we devise an efficient *projected gradient* method to solve the problem. Finally we conduct experiments on several real world data sets to demonstrate the efficacy and efficiency of eNMF.

It is worthwhile to highlights several aspects of eNMF.

- *eNMF is space efficient*. *eNMF* only needs to hold the difference matrix, which is usually much sparser due to the smoothness assumption.
- *eNMF is computationally efficient*. One major computational cost in *NMF* the matrix-matrix multiplications. Our *eNMF* achieves computational savings by using a much sparser matrix in such matrix-matrix multiplications.
- *eNMF is privacy-friendly*. *eNMF* does not need to know the exact data feature matrix. It only requires the factorized matrices at the initial time stamp and the difference data feature matrix. This is particularly useful for those privacy-sensitive applications, e.g., the data feature is only available for a short time window.
- *eNMF can be applied to different types of data*. We developed two instantiations for both traditional asymmetric *NMF*(where the feature matrix is rectangular) and symmetric *NMF*(where the feature matrix is symmetric square, e.g., data similarity matrix).

The rest of this paper is organized as follows. Section 2 introduces the problem formulation and algorithm details. The experimental results are presented in Section 3, followed by the conclusions in Section 4.

The Algorithm

In this section we will introduce our eNMF algorithm in detail. First we will introduce the basic notations that will be used throughout this paper and problem formulation.

Problem Formulation

Suppose we have a nonnegative matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, and we want to factorize it into the product of two nonnegative matrices $\mathbf{F} \in \mathbb{R}^{n\times k}$ and $\mathbf{G} \in \mathbb{R}^{d\times k}$ (usually $k \ll \min(d, n)$) under some loss. In this paper we will concentrate on the Frobenius norm loss as it is one of the most popular loss forms, the algorithms under other loss such as Kullback-Leighbler divergence (Lee and Seung 2001), β-divergence (Févotte and Idier 2010) and Bregman divergence (Dhillon and Sra 2005) can be derived similarly. The optimization problem we need to solve is

$$\min_{\mathbf{F}\geqslant 0,\mathbf{G}\geqslant 0} \left\|\mathbf{X}-\mathbf{F}\mathbf{G}^\top\right\|_F^2 \tag{1}$$

where $\|\mathbf{A}\|_F^2 = tr\left(\mathbf{A}^\top\mathbf{A}\right)$ is the square of the matrix Frobenius norm. This problem can be solved via multiplicative updates (Lee and Seung 2001), active set method (Kim and Park 2008) or projected gradient (Lin 2007) method.

Now suppose there is a small on variation on $\mathbf{X}$ so that $\mathbf{X}$ becomes

$$\widetilde{\mathbf{X}} = \mathbf{X} + \Delta\mathbf{X} \tag{2}$$

and $\widetilde{\mathbf{X}} \in \mathbb{R}^{n\times d}$ is also nonnegative. Our goal is to factorize $\widetilde{\mathbf{X}}$ into the product of two nonnegative matrices $\widetilde{\mathbf{F}} \in \mathbb{R}^{n\times k}$ and $\widetilde{\mathbf{G}} \in \mathbb{R}^{d\times k}$, then we need to solve the following optimization problem

$$\min_{\widetilde{\mathbf{F}}\geqslant 0,\widetilde{\mathbf{G}}\geqslant 0} \left\|\widetilde{\mathbf{X}}-\widetilde{\mathbf{F}}\widetilde{\mathbf{G}}^\top\right\|_F^2 \tag{3}$$

We assume $\|\Delta\mathbf{X}\|_F^2$ is very small, and $\widetilde{\mathbf{F}}$, $\widetilde{\mathbf{G}}$ can be represented as

$$\widetilde{\mathbf{F}} = \mathbf{F} + \Delta\mathbf{F} \tag{4}$$

$$\widetilde{\mathbf{G}} = \mathbf{G} + \Delta\mathbf{G} \tag{5}$$

Bringing Eq.(4) and Eq.(5) into problem (3), we can get that

$$\begin{aligned}\left\|\widetilde{\mathbf{X}}-\widetilde{\mathbf{F}}\widetilde{\mathbf{G}}^\top\right\|_F^2 &= \left\|\mathbf{X}+\Delta\mathbf{X}-(\mathbf{F}+\Delta\mathbf{F})(\mathbf{G}+\Delta\mathbf{G})^\top\right\|_F^2 \\ &= \left\|\mathbf{X}+\Delta\mathbf{X}-\mathbf{F}\mathbf{G}^\top-\Delta\mathbf{F}\mathbf{G}^\top-\mathbf{F}\Delta\mathbf{G}^\top-\Delta\mathbf{F}\Delta\mathbf{G}^\top\right\|_F^2 \end{aligned} \tag{6}$$

and the constraint here is that

$$\widetilde{\mathbf{F}} = \mathbf{F} + \Delta\mathbf{F} \geqslant 0 \tag{7}$$

$$\widetilde{\mathbf{G}} = \mathbf{G} + \Delta\mathbf{G} \geqslant 0 \tag{8}$$

For matrix Frobenius norm, we have the following *triangle inequality*

$$\begin{aligned}&\left\|\mathbf{X}+\Delta\mathbf{X}-\mathbf{F}\mathbf{G}^\top-\Delta\mathbf{F}\mathbf{G}^\top-\mathbf{F}\Delta\mathbf{G}^\top-\Delta\mathbf{F}\Delta\mathbf{G}^\top\right\|_F \\ &\leqslant \left\|\mathbf{X}-\mathbf{F}\mathbf{G}^\top\right\|_F + \left\|\Delta\mathbf{X}-\Delta\mathbf{F}\mathbf{G}^\top-\mathbf{F}\Delta\mathbf{G}^\top-\Delta\mathbf{F}\Delta\mathbf{G}^\top\right\|_F\end{aligned}$$

In our evolutionary setting, we already got the optimal $\mathbf{F}$ and $\mathbf{G}$ by solving problem (1), thus $\|\mathbf{X}-\mathbf{F}\mathbf{G}^\top\|_F$ is already minimized. In order to minimize the objective of problem (3), we propose to solve the following optimization problem

$$\begin{aligned}\min_{\Delta\mathbf{F},\Delta\mathbf{G}} \quad & \left\|\Delta\mathbf{X}-\Delta\mathbf{F}\mathbf{G}^\top-\mathbf{F}\Delta\mathbf{G}^\top-\Delta\mathbf{F}\Delta\mathbf{G}^\top\right\|_F^2 \\ s.t. \quad & \mathbf{F}+\Delta\mathbf{F}\geqslant 0,\ \mathbf{G}+\Delta\mathbf{G}\geqslant 0\end{aligned} \tag{9}$$

This is an optimization problem with box constraints, and we propose to apply *Projected Gradient* (PG) (Lin 2007) method to solve it.

Projected Gradient

In this section we will introduce how to make use of PG to solve problem (9). For notational convenience, we introduce a box projection operator $P_{\mathbf{B}}[\mathbf{A}]$ as

$$\left(P_{\mathbf{B}}[\mathbf{A}]\right)_{ij} = \begin{cases} A_{ij} & \text{if } A_{ij} \geqslant B_{ij} \\ B_{ij} & \text{otherwise}\end{cases} \tag{10}$$

Algorithm 1 Projected Gradient

Require: $0 < \beta < 1, 0 < \sigma < 1$. Initialization $\mathbf{A}^{(0)}$.
Ensure: $\mathbf{A}^{(0)} \geqslant \mathbf{B}$
  for $k = 1, 2, \cdots$ **do**
    $\mathbf{A}^{(k)} = P_{\mathbf{B}}\left[\mathbf{A}^{(k-1)} - \alpha_k \nabla f\left(\mathbf{A}^{(k-1)}\right)\right]$
    where $\alpha_k = \beta^{t_k}$, and t_k is the first nonnegative integer for which

$$f\left(\mathbf{A}^{(k)}\right) - f\left(\mathbf{A}^{(k-1)}\right) \leqslant \sigma \left\langle \nabla f\left(\mathbf{A}^{(k-1)}\right), \left(\mathbf{A}^{(k)} - \mathbf{A}^{k-1}\right) \right\rangle \quad (12)$$

  end for

Then the PG method for solving the problem

$$\min_{\mathbf{A} \geqslant \mathbf{B}} f(\mathbf{A}) \quad (11)$$

can be presented in Algorithm 1, where $\langle \cdot, \cdot \rangle$ is the sum of elementwise multiplication, and the rule for determining the step size in Algorithm 1 is usually referred to as the *Armijo rule* (Bertsekas 1999).

Now let us return to problem (9). If we denote the objective of the above problem by

$$\mathcal{J} = \left\| \Delta\mathbf{X} - \Delta\mathbf{F}\mathbf{G}^\top - \mathbf{F}\Delta\mathbf{G}^\top - \Delta\mathbf{F}\Delta\mathbf{G}^\top \right\|_F^2 \quad (13)$$

Then the gradient of $\mathcal{J}$ with respect to $\Delta\mathbf{F}$ and $\Delta\mathbf{G}$ are

$$\frac{\partial \mathcal{J}}{\partial \Delta\mathbf{F}} = -2\left(\Delta\mathbf{X} - \Delta\mathbf{F}\mathbf{G}^\top - \mathbf{F}\Delta\mathbf{G}^\top - \Delta\mathbf{F}\Delta\mathbf{G}^\top\right)(\mathbf{G} + \Delta\mathbf{G}) \quad (14)$$

$$\frac{\partial \mathcal{J}}{\partial \Delta\mathbf{G}} = -2\left(\Delta\mathbf{X} - \Delta\mathbf{F}\mathbf{G}^\top - \mathbf{F}\Delta\mathbf{G}^\top - \Delta\mathbf{F}\Delta\mathbf{G}^\top\right)^\top(\mathbf{F} + \Delta\mathbf{F}) \quad (15)$$

We can observe that there are two variables, $\Delta\mathbf{F}$ and $\Delta\mathbf{G}$, in problem (9). It is not easy to solve for $\Delta\mathbf{F}$ and $\Delta\mathbf{G}$ simultaneously. However, if we fix one variable, then the problem is convex with respect to the other. Therefore it is natural to adopt the *block coordinate descent* scheme (Bertsekas 1999), which is an alternating optimization strategy, to solve it. At each round of the iteration, we fix one variable and solve the other (via PG), until some stopping criterion is satisfied. As the objective is lower bounded by zero and after each round its value will decrease, the algorithm is guaranteed to converge. The basic algorithm sketch is summarized in Algorithm 2.

Algorithm 2 eNMF

Require: Initialization $\Delta\mathbf{F}^{(0)}$, $\Delta\mathbf{G}^{(0)}$.
Ensure: $\Delta\mathbf{F}^{(0)} \geqslant 0$, $\Delta\mathbf{G}^{(0)} \geqslant 0$
  for $t = 1, 2, \cdots$ **do**
    Fix $\Delta\mathbf{F} = \Delta\mathbf{F}^{(t-1)}$, update $\Delta\mathbf{G}^{(t)}$ using PG
    Fix $\Delta\mathbf{G} = \Delta\mathbf{G}^{(t)}$, update $\Delta\mathbf{F}^{(t)}$ using PG
  end for

Complexity Analysis

For Algorithm 2, we need to hold $\mathbf{F}$, $\mathbf{G}$ and $\Delta\mathbf{F}$, $\Delta\mathbf{G}$ in the main memory, thus the total storage complexity is $O(2k(n+d))$. Actually in our experiments, we usually find that the obtained $\Delta\mathbf{F}$ or $\Delta\mathbf{G}$ is sparse, therefore the storage cost can be further reduced by only storing the nonzero elements.

For computational complexity, as both eNMF and NMF need to evaluate the function objective value (in Armijo rule) when applying PG, the main difference would lie in the evaluation of the function gradient. Suppose that we have m and $\hat{m}$ non-zero elements in the matrices $\mathbf{X} + \Delta\mathbf{X}$ and $\Delta\mathbf{X}$ respectively, then the time cost for eq. (14) and Eq. (15) is $O(\hat{m}k) + O(nk^2) + O(dk^2)$. In contrast, the time complexity for computing the gradient for the original NMF is $O(mk) + O(nk^2) + O(dk^2)$. In many real applications, the matrix $\Delta\mathbf{X}$ is usually much more sparser than the matrix $\mathbf{X} + \Delta\mathbf{X}$ (i.e., $\hat{m} \ll m$). Moreover, since $k \ll n, l < m$, $O(mk)$ is dominant term of the time complexity for the original NMF. Therefore, we would expect that the proposed algorithm is much more efficient in computation compared with the original NMF.

Evolutionary Symmetric NMF

Another interesting scenario is *Symmetric NMF* (Wang et al. 2010), where we have a symmetric square nonnegative feature matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ (e.g., the connectivity matrix of an undirected graph). The goal is to factorize it into the product of a nonnegative matrix $\mathbf{G} \in \mathbb{R}^{n \times k}$ (usually $k \ll n$) and its transpose by solving the following optimization problem

$$\min_{\mathbf{G} \geqslant 0} \left\| \mathbf{S} - \mathbf{G}\mathbf{G}^\top \right\|_F^2 \quad (16)$$

Wang et al.(2010) derived an multiplicative update approach to solve problem (16). Actually, as problem (16) is also a minimization problem with box constraint, we can also apply PG to solve it. Specifically, if we denote the objective of the above problem as

$$\mathcal{J}_S = \left\| \mathbf{S} - \mathbf{G}\mathbf{G}^\top \right\|_F^2 \quad (17)$$

then we can also solve it by PG using Algorithm 1 with $\mathbf{A} = \mathbf{G}$, $f(\mathbf{A}) = \mathcal{J}_S$, $\mathbf{B} = \mathbf{O}$ ($\mathbf{O} \in \mathbb{R}^{n \times k}$ is an all-zero matrix), and the gradient

$$\nabla f(\mathbf{A}) = \frac{\partial \mathcal{J}_S}{\partial \mathbf{G}} = -4\left(\mathbf{S} - \mathbf{G}\mathbf{G}^\top\right)\mathbf{G} \quad (18)$$

In the evolutionary setting, $\mathbf{S}$ is changed to $\widetilde{\mathbf{S}} = \mathbf{S} + \Delta\mathbf{S}$ with a small $\|\Delta\mathbf{S}\|_F^2$. Then we want to factorize $\widetilde{\mathbf{S}}$ by solving the following optimization problem

$$\min_{\widetilde{\mathbf{G}} \geqslant 0} \left\| \widetilde{\mathbf{S}} - \widetilde{\mathbf{G}}\widetilde{\mathbf{G}}^\top \right\|_F^2 \quad (19)$$

We assume $\widetilde{\mathbf{G}}$ takes the following form

$$\widetilde{\mathbf{G}} = \mathbf{G} + \Delta\mathbf{G} \quad (20)$$

with a small $\|\Delta\mathbf{G}\|_F^2$. Bringing Eq.(20) into the objective of problem (19), we obtain

$$\begin{aligned} \left\| \widetilde{\mathbf{S}} - \widetilde{\mathbf{G}}\widetilde{\mathbf{G}}^\top \right\|_F &= \left\| \mathbf{S} + \Delta\mathbf{S} - (\mathbf{G} + \Delta\mathbf{G})(\mathbf{G} + \Delta\mathbf{G})^\top \right\|_F \\ &\leqslant \left\| \mathbf{S} - \mathbf{G}\mathbf{G}^\top \right\|_F + \left\| \Delta\mathbf{S} - \mathbf{G}\Delta\mathbf{G}^\top - \Delta\mathbf{G}\mathbf{G}^\top - \Delta\mathbf{G}\Delta\mathbf{G}^\top \right\|_F \end{aligned} \quad (21)$$

Similar as in the asymmetric case, we also minimize the upper bound instead of the original objective in problem (19). As $\left\|\mathbf{S}-\mathbf{G}\mathbf{G}^\top\right\|_F^2$ already minimized, we solve the following optimization problem instead for evolutionary SNMF

$$\begin{array}{ll}\min_{\Delta\mathbf{G}} & \left\|\Delta\mathbf{S}-\mathbf{G}\Delta\mathbf{G}^\top-\Delta\mathbf{G}\mathbf{G}^\top-\Delta\mathbf{G}\Delta\mathbf{G}^\top\right\|_F^2 \\ s.t. & \mathbf{G}+\Delta\mathbf{G}\geqslant 0\end{array} \quad (22)$$

This problem is still a minimization problem with box constraints, which can be solved by PG. We denote the objective of the above problem by

$$\mathcal{J}_S^e(\Delta\mathbf{G}) = \left\|\Delta\mathbf{S}-\mathbf{G}\Delta\mathbf{G}^\top-\Delta\mathbf{G}\mathbf{G}^\top-\Delta\mathbf{G}\Delta\mathbf{G}^\top\right\|_F^2 \quad (23)$$

Then problem (22) can be solved using PG in Algorithm 1 with $\mathbf{A}=\Delta\mathbf{G}$, $f(\mathbf{A})=\mathcal{J}_S^e$, $\mathbf{B}=-\mathbf{G}$, and the gradient

$$\frac{\partial\mathcal{J}_S^e}{\partial\Delta\mathbf{G}} = -4\left(\Delta\mathbf{S}-\mathbf{G}\Delta\mathbf{G}^\top-\Delta\mathbf{G}\mathbf{G}^\top-\Delta\mathbf{G}\Delta\mathbf{G}^\top\right)(\mathbf{G}+\Delta\mathbf{G})$$

We summarize the procedure of eSNMF in Algorithm 3.

Algorithm 3 eSNMF

Require: $0<\beta<1, 0<\sigma<1$. Initialization $\Delta\mathbf{G}^{(0)}$.
Ensure: $\Delta\mathbf{G}^{(0)}\geqslant-\mathbf{G}$
for $k=1,2,\cdots$ **do**
$\Delta\mathbf{G}^{(k)}=P_{-\mathbf{G}}\left[\Delta\mathbf{G}^{(k-1)}-\alpha_k\nabla\mathcal{J}_S^e\left(\Delta\mathbf{G}^{(k-1)}\right)\right]$
where $\alpha_k=\beta^{t_k}$, and t_k is the first nonnegative integer for which

$$\mathcal{J}_S^e\left(\Delta\mathbf{G}^{(k)}\right)-\mathcal{J}_S^e\left(\Delta\mathbf{G}^{(k-1)}\right)\leqslant\sigma\left\langle\nabla\mathcal{J}_S^e\left(\Delta\mathbf{G}^{(k-1)}\right),\left(\Delta\mathbf{G}^{(k)}-\Delta\mathbf{G}^{(k-1)}\right)\right\rangle$$

end for

Suppose that we have m and $\hat{m}$ non-zero elements in the matrices $\mathbf{S}+\Delta\mathbf{S}$ and $\Delta\mathbf{S}$ respectively, then the time cost for eq. (24) is $O(\hat{m}k)+O(nk^2)$. In contrast, the time complexity for computing the gradient for the original NMF is $O(mk)+O(nk^2)$. In many real applications, the matrix $\Delta\mathbf{S}$ is usually much more sparser than the matrix $\mathbf{S}+\Delta\mathbf{S}$ (i.e., $\hat{m}\ll m$). Moreover, since $k\ll n, l<m$, $O(mk)$ dominates the time complexity for the original SNMF. Thus we would expect that the proposed algorithm is computationally more efficient compared with the original SNMF.

Experiments

We conduct experimental results to evaluate the proposed algorithms from the following three aspects:

1 *Convergency.* How does the overall reconstruction error change wrt the iteration steps?

2 *Effectiveness.* How effective are the proposed algorithms, compared with the original NMF and SNMF, respectively.

3 *Speed.* How fast are the proposed algorithms?

The data set we used for evaluation is from DBLP[4]. We construct time-evolving matrices using the publication records from one of the following four conferences: *AAAI, KDD, SIGIR, NIPS*. For each conference, we first construct the author-paper and the co-authorship snapshot matrices from each of its publication years. For the author-paper snapshot matrices, they are asymmetric where each rows are the authors and columns are the papers. If a given author wrote a paper, the corresponding element in the matrix is 1 and 0 otherwise. We aggregate the first 6 snapshot matrices as the initial $\mathbf{X}$ matrix, and treat each of the remaining snapshot matrices as the $\Delta\mathbf{X}$ matrix in Algorithm 2. We denote these four asymmetric time-evolving matrices as *AAAI-AP, KDD-AP, SIGIR-AP, NIPS-AP* respectively, which are summarized in Table 1. Each of these four asymmetric time-evolving matrices typically contains a few thousands of rows and columns ($n\times d$), and a few thousands of non-zeros elements (m) in $\mathbf{X}$, a few (T) $\Delta\mathbf{X}$ matrices, and a few hundreds of non-zero elements ($\hat{m}$) in the $\Delta\mathbf{X}$ matrix on average.

For the co-authorship snapshot matrices, they are symmetric where each row/column corresponds to an author and edge weights are the number of the co-authored papers. We also aggregate the first 6 snapshot matrices as the initial $\mathbf{S}$ matrix, and treat each of the remaining snapshot matrices as the $\Delta\mathbf{S}$ matrix in Algorithm 3. We denote these four symmetric time-evolving matrices as *AAAI-AA, KDD-AA, SIGIR-AA, NIPS-AA* respectively, which are summarized in Table 1. For each of these four symmetric time-evolving matrices, it typically contains a few thousands of rows/columns ($n\times n$), and a few thousand, or a few tens of thousands of non-zeros elements (m) in the initial $\mathbf{S}$ matrix, a few (T) $\Delta\mathbf{S}$ matrices, and a few thousands of non-zero elements ($\hat{m}$) in the $\Delta\mathbf{S}$ matrix on average.

Table 1: Summary of the data sets

Name	$\mathbf{n}\times\mathbf{n}$ $(\mathbf{n}\times\mathbf{d})$	$\mathbf{m}$	$\mathbf{T}$	$\hat{\mathbf{m}}$
AAAI-AP	3,659 × 2,651	5,762	9	265
KDD-AP	1,974 × 1,118	3,202	7	256
SIGIR-AP	2,489 × 1,867	4,584	22	124
NIPS-AP	3,417 × 2,927	7,111	13	355
AAAI-AA	3,659 × 3,659	10,849	9	5,059
KDD-AA	1,974 × 1,974	3,717	7	5,639
SIGIR-AA	2,489 × 2,489	3,957	22	6,336
NIPS-AA	3,417 × 3,417	6,063	13	6,860

Convergence. In both *eNMF* and *eSNMF*, instead of minimizing the true reconstruction error directly, we try to minimize its upper bound. Here, we test how the true reconstruction error change wrt the iteration steps. Figures 1-2 show the results on *NIPS-AP* and *NIPS-AA* for one time stamp, respectively. We compared our algorithms with the original *NMF* and *SNMF* respectively. From the figures, it can be seen that for both *eNMF* and *eSNMF*, the overall reconstruction error decreases quickly and reaches an steady state wrt the iteration steps, suggesting that our algorithms indeed converge fast. It is worth pointing out that the final reconstruction error of *eNMF* is very close to that of the original *NMF*. We have similar observation for *eSNMF* and *SNMF*.

Effectiveness Comparison. Here, we evaluate the effectiveness of the proposed *eNMF* and *eSNMF* in terms of

[4] http://www.informatik.uni-trier.de/~ley/db/

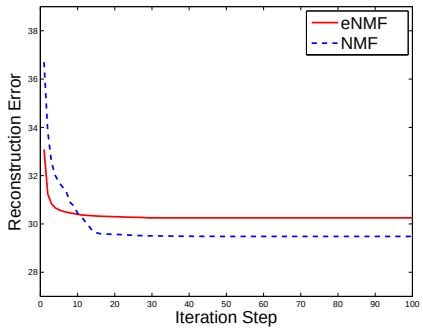


Figure 1: *The reconstruction error vs. iteration steps on NIPS-AA data. The proposed* eNMF *converges quickly, leading to a similar reconstruction error as* NMF *after convergence.*

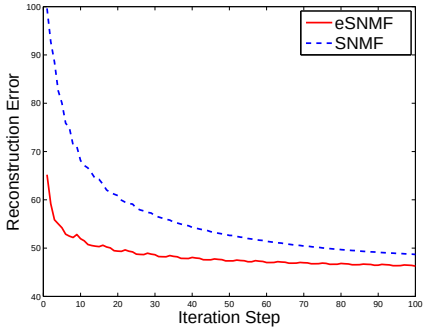


Figure 2: *The reconstruction error vs. iteration steps on NIPS-AA data. The proposed* eSNMF *converges quickly, leading to a similar reconstruction error as* SNMF *after convergence.*

the final reconstruction error (i.e., the reconstruction error after the algorithms converge). For each time stamp of a given data set, we run *eNMF*(or *eSNMF* for the symmetric matrix) and *NMF* (or *SNMF* for the symmetric matrix) until convergence, and then compare the reconstruction errors. The results are summarized in figures 3-4. It can be seen that in terms of the final reconstruction error, our algorithms are very close to the original methods (*NMF* and *SNMF*). In most cases, the final reconstruction error of the proposed *eNMF* and *eSNMF* is within the range of $1 \pm 10\%$ of that by the original *NMF* and *SNMF* respectively. For those exceptions (e.g., *KDD-AP, KDD-AA, SIGIR-AA*), our algorithms actually lead to *smaller* reconstruction error. On the other hand, our proposed methods do not require the original $\mathbf{X}$ and $\mathbf{S}$ matrices, which in turn leads to some nice properties for the applications (e.g., space and computation efficient, privacy-friendly, etc).

Speed Comparison. We also compared the speed between our algorithms and the original *NMF* and *SNMF*. Figures 5-6 show the average wall-clock time on each data set. The results are consistent with the complexity analysis in Section . In most cases, our *eNMF* and *eSNMF* are faster than the original *NMF* and *SNMF* respectively. The only exception is the *NIPS-AA* data set, where the proposed *eSNMF* is slightly slower than the original *SNMF*. This is because for this data set, we have more non-zero elements in the $\Delta\mathbf{S}$ matrix ($\hat{m} = 6,860$) than that of the original the $\mathbf{S}$ matrix ($m = 6,063$) on average. Compared with *eNMF* and *eSNMF*, we can see that the speed saving is more significant in *eNMF*. This is because the $\Delta\mathbf{X}$ matrix is much more sparser than the $\Delta\mathbf{S}$ matrix, - on average, there are a few *hundred* of non-zero elements in $\Delta\mathbf{X}$, and a few *thousand* of non-zero elements in $\Delta\mathbf{S}$.

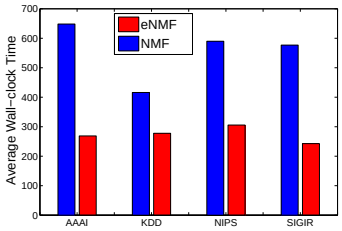


Figure 5: *Speed comparison of* eNMF *and* NMF. *Our* eNMF *is much faster than* NMF

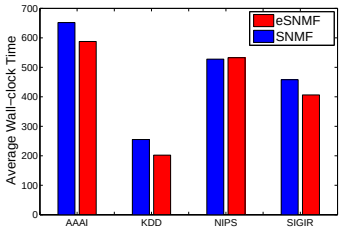


Figure 6: *Speed comparison of* eSNMF *and* SNMF. *Our* eSNMF *is faster than or similar to* SNMF

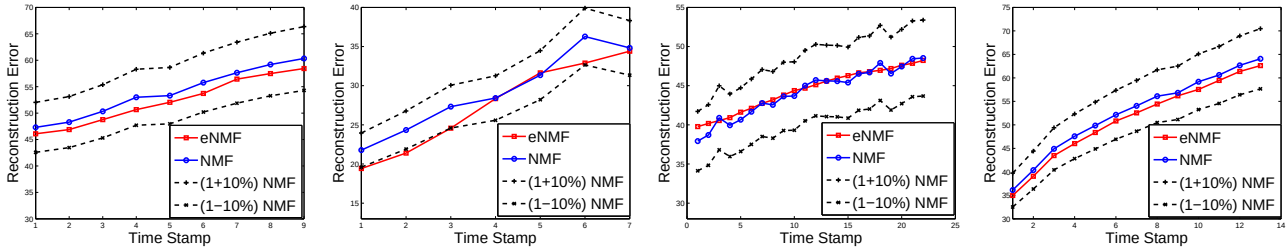
Conclusion

We present a novel *evolutionary Nonnegative Matrix Factorization* (*eNMF*) strategy to efficient perform NMF in the scenario where the data features are evolving over time. Our method is both storage and computational efficient as well as privacy friendly. The experimental results on real world time evolving networks are presented to demonstrate the effectiveness of our proposed methods.

References

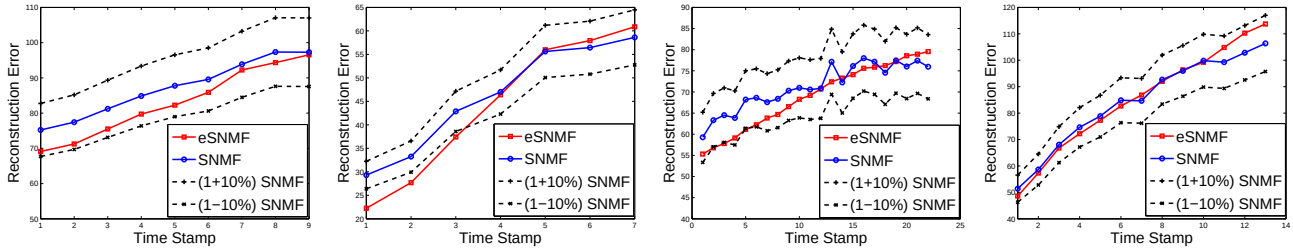
Anttila, P.; Paatero, P.; Tapper, U.; and Järvinen, O. 1995. Source identification of bulk wet deposition in finland by positive matrix factorization. *Atmospheric Environment* 29(14):1705–1718.

Bertsekas, D. P. 1999. *Nonlinear Programming*. Athena Scientific, 2nd edition.



(a) AAAI-AP (b) KDD-AP (c) SIGIR-AP (d) NIPS-AP

Figure 3: *Comparison for asymmetric matrices. The x-axis is time stamp (each corresponds to a publication year.), and y-axis is the final reconstruction error. The reconstruction error of the proposed* eNMF *is very close to that of the original* NMF.



(a) AAAI-AA (b) KDD-AA (c) SIGIR-AA (d) NIPS-AA

Figure 4: *Comparison for symmetric matrices. The x-axis is time stamp (each corresponds to a publication year.), and y-axis is the final reconstruction error. The reconstruction error of the proposed* eSNMF *is very close to that of the original* SNMF.

Cao, B.; Shen, D.; tao Sun, J.; Wang, X.; Yang, Q.; and Chen, Z. 2007. Detect and track latent factors with online nonnegative matrix factorization. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2689–2694.

Chi, Y.; Song, X.; Zhou, D.; Hino, K.; and Tseng, B. L. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD '07: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 153–162.

Dhillon, I. S., and Sra, S. 2005. Generalized nonnegative matrix approximations with Bregman divergences. In *Advances in Neural Information Proc. Systems*, 283–290.

Ding, C.; Li, T.; Peng, W.; and Park, H. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 126–135.

Ding, C.; Li, T.; and Jordan, M. I. 2010. Convex and Semi-Nonnegative Matrix Factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(1):45–55.

Eggert, J., and Korner, E. 2004. Sparse coding and nmf. In *Proceedings of IEEE International Joint Conference on Neural Networks*, volume 4, 2529–2533.

Févotte, C., and Idier, J. 2010. Algorithms for nonnegative matrix factorization with the beta-divergence. *CoRR* abs/1010.1763.

Guillamet, D.; Bressan, M.; and Vitrià, J. 2001. A weighted non-negative matrix factorization for local representations. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 942–947.

Kim, H., and Park, H. 2008. Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method. *SIAM Journal on Matrix Analysis and Applications* 30(2):713–730.

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

Lee, D. D., and Seung, H. S. 2001. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*, 556–562. MIT Press.

Lin, C.-J. 2007. Projected Gradient Methods for Nonnegative Matrix Factorization. *Neural Comp.* 19(10):2756–2779.

Shahnaz, F.; Berry, M. W.; Pauca; and Plemmons, R. J. 2006. Document clustering using nonnegative matrix factorization. *Information Processing & Management* 42(2):373–386.

Wang, F.; Li, T.; Wang, X.; Zhu, S.; and Ding, C. 2010. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*.

Wang, F.; Li, P.; and König, C. 2011. Efficient document clustering via online nonnegative matrix factorization. In *Proceedings of the 11th SIAM Conference on Data Mining*.