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). Differen 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. 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 $X$ and $	ilde{X} = X + \Delta X$ be the old and new data feature matrices respectively. In many real applications, $\Delta X$ is usually very sparse while $\tilde{X}$ is not. It therefore is not efficient in terms of space cost to re-run NMF since we need to store the whole data feature matrix $\tilde{X}$. It is also not efficient in computation since it requires some matrix-matrix multiplication between $\tilde{X}$ and the two factorized matrices. What is more, this strategy becomes infeasible for those privacy-sensitive applications where the whole data feature matrix $\tilde{X}$ might not be available at a given time stamp. For instance, Facebook’s

1The 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.

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

3http://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 $X \in \mathbb{R}^{n \times d}$, and we want to factorize it into the product of two nonnegative matrices $F \in \mathbb{R}^{n \times k}$ and $G \in \mathbb{R}^{d \times k}$ (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-Leibghler divergence (Lee and Seung 2001), $\beta$-divergence (Févotte and Idier 2010) and Bregman divergence (Dhillon and Sra 2005) can be derived similarly. The optimization problem we need to solve is

$$\min_{F \geq 0, G \geq 0} \|X - FG^\top\|_F^2$$

where $\|A\|_F^2 = tr (A^\top A)$ 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 $X$ so that $X$ becomes

$$\tilde{X} = X + \Delta X$$

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

$$\min_{\tilde{F} \geq 0, \tilde{G} \geq 0} \|\tilde{X} - \tilde{F}\tilde{G}^\top\|_F^2$$

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

$$\tilde{F} = F + \Delta F$$

$$\tilde{G} = G + \Delta G$$

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

$$\|\tilde{X} - \tilde{F}\tilde{G}^\top\|_F^2 = \|X + \Delta X - (F + \Delta F) (G + \Delta G)^\top\|_F^2$$

$$\|X + \Delta X - FG^\top - \Delta FG^\top - F\Delta G^\top - \Delta FG^\top\|_F^2$$

and the constraint here is that

$$F = F + \Delta F \geq 0$$

$$G = G + \Delta G \geq 0$$

For matrix Frobenius norm, we have the following triangle inequality

$$\|X + \Delta X - FG^\top - \Delta FG^\top - F\Delta G^\top - \Delta FG^\top\|_F \leq \|X - FG^\top\|_F + \|\Delta X - \Delta FG^\top - F\Delta G^\top - \Delta FG^\top\|_F$$

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

$$\min_{\Delta F, \Delta G} \|\Delta X - \Delta FG^\top - F\Delta G^\top - \Delta FG^\top\|_F^2$$

s.t.

$$F + \Delta F \geq 0, \ G + \Delta G \geq 0$$

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 $P_B[A]$ as

$$(P_B[A])_{ij} = \begin{cases} A_{ij} & \text{if } A_{ij} \geq B_{ij} \\ B_{ij} & \text{otherwise} \end{cases}$$
Then the gradient of the above problem by elementwise multiplication, and the rule for determining the
(Bertsekas 1999). The PG method for solving the problem
Algorithm 1

Require: $0 < \beta < 1$, $0 < \sigma < 1$. Initialization $A^{(0)}$.
Ensure: $A^{(0)} \succeq B$

for $k = 1, 2, \cdots$ do
\[
A^{(k)} = P_B \left[ A^{(k-1)} - \alpha_k \nabla f \left( A^{(k-1)} \right) \right]
\]
where $\alpha_k = \beta t_k$, and $t_k$ is the first nonnegative integer for which
\[
f \left( A^{(k)} \right) - f \left( A^{(k-1)} \right) \leq \sigma \left\langle \nabla f \left( A^{(k-1)} \right), A^{(k)} - A^{(k-1)} \right\rangle
\]
end for

Then the PG method for solving the problem
\[
\min_{A \succeq B} f(A)
\]
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
\[
J = \| \Delta X - \Delta F G^T - F \Delta G^T - \Delta F \Delta G^T \|^2_F
\]
Then the gradient of $J$ with respect to $\Delta F$ and $\Delta G$ are
\[
\begin{align*}
\frac{\partial J}{\partial \Delta F} &= -2 \left( \Delta X - \Delta F G^T - F \Delta G^T - \Delta F \Delta G^T \right) (G + \Delta G) \quad (14) \\
\frac{\partial J}{\partial \Delta G} &= -2 \left( \Delta X - \Delta F G^T - F \Delta G^T - \Delta F \Delta G^T \right) (F + \Delta F) \quad (15)
\end{align*}
\]

We can observe that there are two variables, $\Delta F$ and $\Delta G$, in problem (9). It is not easy to solve for $\Delta F$ and $\Delta G$
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 $\Delta F^{(0)}$, $\Delta G^{(0)}$.
Ensure: $\Delta F^{(t)} \succeq 0$, $\Delta G^{(t)} \succeq 0$

for $t = 1, 2, \cdots$ do

Fix $\Delta F = \Delta F^{(t-1)}$, update $\Delta G^{(t)}$ using PG
Fix $\Delta G = \Delta G^{(t)}$, update $\Delta F^{(t)}$ using PG
end for

**Complexity Analysis**

For Algorithm 2, we need to hold $F$, $G$ and $\Delta F$, $\Delta G$ in the
main memory, thus the total storage complexity is $O(2k(n + d))$. Actually in our experiments, we usually find that the
obtained $\Delta F$ or $\Delta G$ 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 $\tilde{m}$ non-zero elements in the matrices $X + \Delta X$ and $\Delta X$ respectively, then the time cost for eq. (14) and Eq. (15) is $O(n \tilde{m} k) + O(n k^2) + O(d k^2)$. In contrast, the time complexity for computing the gradient for the original NMF is $O(nk) + O(n k^2) + O(d k^2)$. In many real applications, the matrix $\Delta X$ is usually much more sparser than the matrix $X + \Delta X$ (i.e., $\tilde{m} \ll m$). Moreover, since $k \ll n$, $\tilde{m}$ 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 $S \in \mathbb{R}^{n \times n}$ (e.g., the connectivity matrix of an
undirected graph). The goal is to factorize it into the product of a nonnegative matrix $G \in \mathbb{R}^{n \times k}$ (usually $k \ll n$) and its
transpose by solving the following optimization problem
\[
\min_{G \succeq 0} \left\| S - GG^T \right\|^2_F
\]
Wang et al.(2010) derived a 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
\[
J_S = \left\| S - GG^T \right\|^2_F
\]
then we can also solve it by PG using Algorithm 1 with $A = G$, $f(A) = J_S$, $B = O$ ($O \in \mathbb{R}^{n \times k}$ is an all-zero matrix), and the gradient
\[
\nabla f(A) = \frac{\partial J_S}{\partial G} = -2 \left( S - GG^T \right) G
\]
In the evolutionary setting, $S$ is changed to $\tilde{S} = S + \Delta S$ with a small $\| \Delta S \|^2_F$. Then we want to factorize $\tilde{S}$ by solving the following optimization problem
\[
\min_{G \succeq 0} \left\| \tilde{S} - \tilde{G} \tilde{G}^T \right\|^2_F
\]
We assume $\tilde{G}$ takes the following form
\[
\tilde{G} = G + \Delta G
\]
with a small $\| \Delta G \|^2_F$. Bringing Eq.(20) into the objective of problem (19), we obtain
\[
\left\| \tilde{S} - \tilde{G} \tilde{G}^T \right\|^2_F = \left\| S + \Delta S - (G + \Delta G)(G + \Delta G)^T \right\|^2_F
\]
\[
\leq \left\| S - GG^T \right\|^2_F + \left\| \Delta S - G \Delta G^T - \Delta G G^T - \Delta G \Delta G^T \right\|^2_F
\]
Similar as in the asymmetric case, we also minimize the upper bound instead of the original objective in problem (19). As $\|S - GG^T\|_F^2$ already minimized, we solve the following optimization problem instead for evolutionary SNMF

$$\min_{\Delta G} \left\| \Delta S - GG^T - \Delta GG^T - \Delta G\Delta G^T \right\|_F^2$$

subject to $G + \Delta G \succeq 0$ (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

$$\mathcal{J}_S(\Delta G) = \left\| \Delta S - GG^T - \Delta GG^T - \Delta G\Delta G^T \right\|_F^2$$

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

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

We summarize the procedure of eSNMF in Algorithm 3.

Algorithm 3 eSNMF

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

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

Experiments

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

1. Convergence. 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. 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 $X$ matrix, and treat each of the remaining snapshot matrices as the $\Delta X$ matrix in Algorithm 2. We denote these four asymmetric time-evolving matrices as $\text{AAAI-AP}$, $\text{KDD-AP}$, $\text{SIGIR-AP}$, $\text{NIPS-AP}$ 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-zero elements ($m$) in $X$, a few ($T$) $\Delta X$ matrices, and a few hundreds of non-zero elements ($\bar{m}$) in the $\Delta S$ 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 $S$ matrix, and treat each of the remaining snapshot matrices as the $\Delta S$ matrix in Algorithm 3. We denote these four symmetric time-evolving matrices as $\text{AAAI-AA}$, $\text{KDD-AA}$, $\text{SIGIR-AA}$, $\text{NIPS-AA}$ 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 $S$ matrix, a few ($T$) $\Delta S$ matrices, and a few thousands of non-zero elements ($\bar{m}$) in the $\Delta S$ matrix on average.

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 $\text{NIPS-AP}$ and $\text{NIPS-AA}$ 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

Table 1: Summary of the data sets

<table>
<thead>
<tr>
<th>Name</th>
<th>$n \times n$</th>
<th>$(n \times d)$</th>
<th>$m$</th>
<th>$T$</th>
<th>$\bar{m}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAI-AP</td>
<td>3,659 × 2,651</td>
<td>5,762</td>
<td>9</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>KDD-AP</td>
<td>1,974 × 1,118</td>
<td>3,202</td>
<td>7</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>SIGIR-AP</td>
<td>2,489 × 1,867</td>
<td>4,584</td>
<td>22</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>NIPS-AP</td>
<td>3,417 × 2,927</td>
<td>7,111</td>
<td>13</td>
<td>355</td>
<td></td>
</tr>
<tr>
<td>AAAI-AA</td>
<td>3,659 × 3,659</td>
<td>10,849</td>
<td>9</td>
<td>5,059</td>
<td></td>
</tr>
<tr>
<td>KDD-AA</td>
<td>1,974 × 1,974</td>
<td>3,717</td>
<td>7</td>
<td>3,639</td>
<td></td>
</tr>
<tr>
<td>SIGIR-AA</td>
<td>2,489 × 2,489</td>
<td>3,957</td>
<td>22</td>
<td>6,336</td>
<td></td>
</tr>
<tr>
<td>NIPS-AA</td>
<td>3,417 × 3,417</td>
<td>6,060</td>
<td>13</td>
<td>6,860</td>
<td></td>
</tr>
</tbody>
</table>
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 $\Delta S$ matrix ($\hat{m} = 6,860$) than that of the original the $S$ 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 $\Delta X$ matrix is much more sparse than the $\Delta S$ matrix, - on average, there are a few hundred of non-zero elements in $\Delta X$, and a few thousand of non-zero elements in $\Delta S$.

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 efficiently 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 large scale networks are presented to demonstrate the effectiveness of our proposed methods.

References


the final reconstruction error. The reconstruction error of the proposed eNMF is very close to that of the original NMF.

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 eSNMF is very close to that of the original SNMF.


