Enforced Sparse Non-Negative Matrix Factorization

Brendan Gavin*†, Vijay Gadepally†, Jeremy Kepner†

*University of Massachusetts, Amherst
†MIT Lincoln Laboratory

January 23, 2016

***Abstract*—Non-negative matrix factorization (NMF) is a dimensionality reduction algorithm for data that can be represented as an undirected bipartite graph. It has become a common method for generating topic models of text data because it is known to produce good results, despite its relative simplicity of implementation and ease of computation. One challenge with applying the NMF to large datasets is that intermediate matrix products often become dense, thus stressing the memory and compute elements of the underlying system. In this article, we investigate a simple but powerful modification of the alternating least squares method of determining the NMF of a sparse matrix that enforces the generation of sparse intermediate and output matrices. This method enables the application of NMF to large datasets through improved memory and compute performance. Further, we demonstrate, empirically, that this method of enforcing sparsity in the NMF either preserves or improves both the accuracy of the resulting topic model and the convergence rate of the underlying algorithm.**

I. Introduction

A common analyst challenge is searching through large quantities of text documents to find interesting pieces of information. With limited resources, analysts often employ automated text-mining tools that highlight common terms or topics. The machine learning and natural language processing communities often refer to this as topic modeling. Topic modeling is a vast field in which there have been many fundamental contributions. For example, latent dirichlet allocation (LDA) [1] uses Bayesian networks to model how a mixture of topics constitutes a document. Other common methods for topic modeling include the following: latent semantic analysis (LSA) [1], probabilistic latent semantic analysis (PLSA) [2], and term frequency-inverse document frequency (TF-IDF) [3] analysis. More recently, non-negative matrix factorization (NMF) [4]–[7] is used as a technique for document classification and topic modeling. The NMF has also been used for graph clustering [8], graph embedding [9] and graph regularization [10]. As we described in [11], the NMF is quite amenable to computation via the GraphBLAS kernels.

A set of definitions for terms and variables used throughout the article are given below:

Definitions

NMF:	Non-negative matrix factorization, i.e., $A = UV^T$
A:	Term/document biadjancency/data matrix
U:	Term/topic biadjacency/data matrix
V:	Document/topic biadjacency/data matrix
Topic:	A cluster of related documents or terms. Columns of U are term topics and columns of V are document topics.
ALS:	Alternating least squares; an algorithm for finding the NMF.
Residual:	Relative norm of the difference between U matrices at subsequent iterations of ALS.
Error:	Relative norm of the difference between data matrix A and its NMF approximation UV^T
NNZ:	Number of nonzeros

The NMF operates on data sets that have a natural interpretation as undirected bipartite graphs. For text processing, collections of text documents are often stored in a "Bag of Words" data matrix A, where each element a_{ij} is a count of how many times term i occurs in document j. The data matrix A is then the biadjacency matrix of a bipartite graph, where one partition of nodes represents the terms present in the documents in the collection, and the other partition of nodes represents the documents in the collection.

NMF approximately factorizes the data matrix A into a product of two other matrices.

$$\begin{gathered} A \approx UV^T \\ A \in \mathbb{R}^{n\times m},\ U \in \mathbb{R}^{n\times k},\ V \in \mathbb{R}^{m\times k} \\ A \geq 0,\ U \geq 0,\ V \geq 0 \end{gathered} \tag{1.1}$$

The rank k of the new matrices is often chosen heuristically to be much smaller than the rank of the original data matrix, thereby reducing the dimensionality of the data. However, while information is lost in this process, the expectation is that the most important relationships in

Distribution A: Public Release. This work is sponsored by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, recommendations, and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

the data will be retained.

Importantly, the factor matrices U and V are calculated such that their entries are non-negative. This allows the resulting matrix factorization to be naturally intepreted as another undirected, bipartite graph whose biadjacency matrix is given by

$$A_{nmf} = \begin{bmatrix} U \\ V \end{bmatrix} \tag{1.2}$$

Figure 1 illustrates the difference between the graph corresponding to the initial data A and the graph corresponding to A_{nmf}.

If the original edge weights a_{ij} are interpreted as as a count of the number of paths from node i to node j in the original graph, then the matrix factorization $a_{ij} = \sum_n u_{in} v_{jn}$ calculates the number of paths from node i to node j as being the sum of the number of paths from node i to node j when passing through a new set of intermediate nodes that are indexed by n. In the context of text analysis and topic modeling, these intermediate nodes are referred to as "topics". If the number of topics is chosen

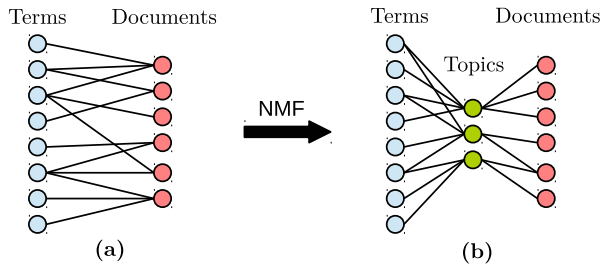

Figure 1. Illustration of the action of non-negative matrix factorization on a "Bag of Words" text data set. NMF takes as input the original data A **(a)** and produces as output a new data set A_{nmf} **(b)** that has new set of intermediate nodes (i.e. "topics") that indicate clusters of related terms and related documents.

to be small enough, then there can be many fewer edges in the new graph given by A_{nmf} than in the original graph given by A.

The intermediate topic nodes are interpreted to define clusters of nodes in the original graph. When the NMF is used in text analysis, terms and documents that are connected to the same topic node are interpreted as being a part of the same cluster of related ideas. Further, the edge weights of the connections to that topic are interpreted as being proportional to the probability of membership in the corresponding cluster. Text analysis is performed by finding the NMF of the original data set and then examining the edge connections with the largest weights for each topic node in order to determine which terms and which documents are most closely associated with each other.

The NMF is found by solving a minimization problem that makes the factorization UV^T as close to the original data matrix as possible, thereby preserving the number of shortest paths a_{ij} between the term nodes and the document nodes in the corresponding graph:

$$\min_{U,V} ||A - UV^T||, \quad \text{s.t. } U \geq 0,\ V \geq 0 \tag{1.3}$$

There are a variety of ways to approach this problem as described in [12]. Perhaps the most common method is to use the multiplicative update rules of Lee and Seung [6]. The benefit of these update rules is that they are simple to implement and that analytical results can be established about the convergence properties of updates. Other methods include gradient descent algorithms (of which the multiplicative update rules of [6] are an example) and the alternating non-negativitity constrained least squares (ANLS) method. In ANLS, alternating least squares problems are solved by using optimization methods to enforce the non-negativity constraint [13] in Equation (1.3). Many of these algorithms have analytical results regarding their convergence properties and have proven to be effective in practice; however, they tend to be slow to converge.

II. Calculating NMF via Alternating Least Squares

In the method we describe in this article, we have chosen to solve Equation (1.3) by using the conventional alternating least squares (ALS) algorithm combined with a projection step, as described in [12]. In ALS, we hold one of the matrices U or V constant, and then solve for the other by using linear least squares. By repeating this process many times, alternating back and forth between solving for U and solving for V, we hope to converge to a good approximation of the NMF.

The projection step enforces the non-negativity constraint of Equation (1.3) by setting all the negative entries of U and V to zero at each iteration, rather than by using constrained optimization methods as ANLS does. Because this is a projection onto the space of non-negative solutions, it is sometimes called projected alternating least squares and is described in Algorithm 1. While there are no analytic results regarding the convergence properties of the ALS algorithm, in practice, the projected ALS algorithm consistently produces good results.

We prefer to use projected ALS because it is the fastest of the available methods for finding the NMF, and it can be implemented by using only basic linear algebra operations (specifically matrix-vector multiplication). This makes it ideal for systems that are designed to perform fast matrix operations in order to support specifications for graph algorithms such as the GraphBLAS [11], [14], [15].

Projected ALS does, however, have the drawback that,

Algorithm 1 Projected Alternating Least Squares

Input: data matrix $A \in \mathbb{R}^{n \times m}$, initial guess $U_0 \in \mathbb{R}^{n \times k}$
Output: Approximation $UV^T \approx A$, $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{m \times k}$

START: Set $U = U_0$

Do until convergence:

1. Find V using $V = A^T U (U^T U)^{-1}$, and set negative entries of V to zero.
2. Find U using $U = AV(V^T V)^{-1}$, and set negative entries of U to zero.

end do

Output U, V

END

as stated in Algorithm 1, it does not preserve sparsity in the matrix factorization that it produces. In most natural data sets the original data matrix A tends to be very sparse, whereas the U and V matrices in the resulting NMF are not necessarily sparse. Figure 2 describes a few such examples. This can result in excessive memory use and computation time when using sparse matrix storage formats, posing a bottleneck for performing calculations on very large data sets.

Reuters-21578

Matrix	Sparsity
A	99.65%
U	61.0%
V	61.0%
UV^T	4.15%

Wikipedia

Matrix	Sparsity
A	99.6%
U	45.0%
V	41.0%
UV^T	11.0%

Figure 2. Tables showing the change in sparsity from the original data matrix A to the NMF approximation of it, UV^T, using two different data sources. Sparsity is measured as the fraction of a matrix's entries that are exactly equal to zero.

In the following Section, we describe our approach for modifying Algorithm 1 so it can be used to produce sparse intermediate and output matrices, allowing it to take advantage of the performance benefits of sparse matrix storage formats. In Section IV, we apply the resulting enforced sparsity NMF algorithm to several example datasets. We further show that the modified algorithm converges at least as well as Algorithm 1 and that it produces NMF topics that are empirically and qualitatively as accurate as those produced by the unmodified ALS algorithm. We also demonstrate a drawback of producing NMF matrices that are extremely sparse, and in Section V we discuss methods that can be used to alleviate this problem. We conclude in Section VI.

III. Sparcity Enforced ALS NMF

There have been many studies that have looked at producing sparse matrices through the NMF [13], [16]–[19]. Some studies use NMF to produce sparse matrices by adding sparsity constraints to the minimization problem (Equation 1.3), and others do so by adding terms to the minimization problem that penalize having larger numbers of nonzeros in the factorization. Some popular methods for ensuring sparsity include using Hoyer's sparsity measure [17] or the L_1 norm of U or V [13] as either constraints or penalty terms in the minimization problem (Equation 1.3). These approaches work well, but they require the use of algorithms for NMF that are slower than the projected ALS algorithm.

Our goal is to maintain good performance in finding the NMF, and we take the least computationally–expensive approach for maintaining sparsity: at each iteration of ALS, in order to ensure that either of the U or V matrices has exactly t nonzero entries in it, we set all the entries in that matrix to zero except for the t largest ones. The resulting modification of Algorithm 1 is shown in Algorithm 2.

Algorithm 2 Enforced Sparsity ALS

Input: matrix $A \in \mathbb{R}^{n \times m}$, initial guess $U_0 \in \mathbb{R}^{n \times k}$, maximum NNZ(U) t_u and maximum NNZ(V) t_v
Output: Approximation $UV^T \approx A$, $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{m \times k}$

START: Set $U = U_0$

Do until convergence:

1. Find V using $V = A^T U (U^T U)^{-1}$, and set negative entries of V to zero.
2. Sort nonzero entries of V, keep only t_v largest nonzeros in V
3. Find U using $U = AV(V^T V)^{-1}$, and set negative entries of U to zero.
4. Sort nonzero entries of U, keep only t_u largest nonzeros in U.

end do

Output U, V

END

Herein, this algorithm is referred to as the enforced sparsity ALS, because we are producing sparse matrices by enforcing sparsity at each iteration of the ALS algorithm. In this method, we keep the t largest entries of a given matrix by finding the magnitude of the t^{th} largest entry and then setting all the entries with magnitudes lower than that of the t^{th} largest entry to zero. This method requires slightly more computation than a simpler method

of enforcing sparsity, which is to set all entries of a matrix that fall below an arbitrary threshold to zero; however, it has the benefit of allowing us to consistently set exactly the amount of sparsity that we want, regardless of the scaling of the matrices that we are working with.

This method of enforcing sparsity also synergizes well with sparse matrix storage formats: it requires only the use of list sorting algorithms, which operate naturally on one-dimensional lists of numbers, which is how sparse matrices are stored. While relatively simple, the Algorithm of 2 qualitatively works well.

We claim that, in practice, Algorithm 2 does indeed converge, and that it consistently produces results of equal quality to those of Algorithm 1, despite its heuristic method of preserving sparsity. In Section IV we demonstrate convergence by examining several empirical measures of convergence, using a few example data sets.

IV. Sparcity Enforced ALS NMF Results

We illustrate the results of using Algorithm 2 by applying it to data matrices that are derived from several different real datasets of text documents. For each of these datasets, we produce a term-document data matrix A, where each column represents a single document, each row represents a single term, and each entry a_{ij} is the number of times that term i occurs in document j. We discard very common terms from each document by using a stop word list, and we also discard terms that appear only once in a particular dataset. We divide each row of the data matrix by the number of nonzero entries in that row in order to prevent our results from being biased by commonly used terms. All of the matrices that we use to produce our results are stored in the MATLAB sparse matrix storage format.

In each example case, we produce an NMF with a number of topics that seem appropriate for illustrating the effect that we are discussing. In practical applications, the choice of the number of topics (k) depends entirely on the data set and on the user's needs, and can range from being very small (i.e. 3) to being very large (hundreds or thousands).

A. Convergence Behavior: First, we examine the effect of sparsity enforcement on the convergence of ALS by examining the relative error and the relative residual of the NMF UV^T at each iteration of ALS in Algorithm 2.

The relative error is given by

$$E = ||A - UV^T||/||A|| \tag{4.4}$$

and measures the difference between our original matrix A and the factorization of UV^T and then normalized by the norm of A. As the rank of U and V is increased, the error is reduced. Since there is no way of knowing *a priori* what the lowest possible error is for a given rank, this is generally an unreliable measure of convergence.

The relative residual, given by

$$R = ||U_i - U_{i-1}||/||U_i||, \tag{4.5}$$

measures the difference between our current solution for U and our solution for U at the previous iteration. The relative residual indicates the amount our solution changes with each iteration and a low value will indicate close proximity to a stable point in the iterations. The residual is thus considered to be a more natural measure of convergence.

Empirically, as measured by the relative residual, we find that the sparse NMF using Algorithm 2 converges as fast as, or faster than the dense NMF using Algorithm 1. As an example, Figures 3 and 4 describe results of using projected ALS, with and without sparsity enforcement, to find a five-topic NMF of a data matrix derived from the Reuters-21578 dataset (provided by [20]) using 1,985 documents with 6,424 different terms. Figure 3 shows the relative residual and the relative approximation error at each iteration of Algorithm 2.

In Figure 3, when the NMF is generated using the enforced sparsity ALS, the term/topic matrix U is forced by Algorithm 2 to have only 55 nonzeros (in order to maintain sparsity); in the dense case, U is allowed by Algorithm 1 to have any number of nonzeros. In this example, the experiment with a sparse U converges quicker than the fully dense version (as measured by the relative residual), and finishes with a higher relative L_2 error.

NMF With and Without Sparsity Enforcement: Convergence

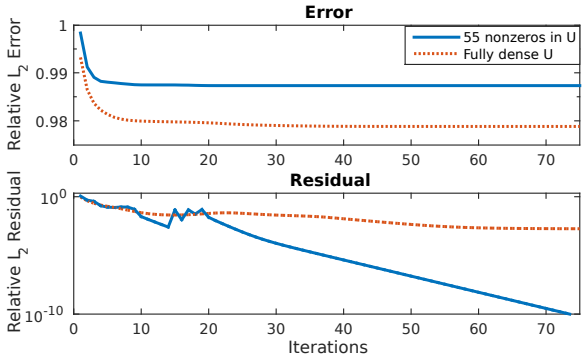

Figure 3. Example NMF with and without sparsity enforcement. The plots show the relative error and relative residual at each ALS iteration when using sparsity enforcement on the term/topic matrix U and when allowing it to be fully dense.

Figure 4 shows the five terms with the largest magnitudes for each topic in the NMF that was generated for making Figure 3. Sparse NMF produces topic terms that are qualitatively as coherent as those produced by the dense NMF, although the topics produced by each are somewhat different.

Sparsity Enforced U Matrix (55 nonzeros for 5 topics):

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
miles	risk	coffee	repurchase	yen
load	contracts	quotas	motors	firms
factor	paper	ico	class	plaza
revenue	proposals	crop	spending	currencies
passenger	futures	colombia	buyback	movements

Fully Dense U Matrix:

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
miles	paper	coffee	iran	senate
load	risk	crop	crude	baker
factor	proposals	quotas	opec	legislation
revenue	england	ico	iraq	vote
passenger	yen	producer	iranian	surplus

Figure 4. Example NMF with and without sparsity enforcement. The tables show the five terms with the largest magnitudes for each resulting topic.

The results described in Figures 3 and 4 are representative of the empirical behavior of Algorithm 2 in producing solutions that converge and provide good results.

The distribution of nonzeros amongst the column vectors of U and V change depending on which matrices are being made sparse. When we force one or both of U and V to be sparse (particularly when making them very sparse), the nonzeros in either matrix tend to end up unevenly distributed among the matrix's columns. As a result, some topics will have more terms or documents allocated to them, and other topics will have fewer.

For example, allowing 50 nonzero entries in the term/topic matrix U of a five-topic NMF typically does not result in having 10 nonzero entries in each column of that matrix. Instead, some columns will have very many nonzeros, and other columns will have very few. The table in Figure 5 shows an example of the topic terms that are produced in this situation. The dataset used to produce this table is derived from the first 12,439 pages of the monthly Wikipedia database dump, with a total of 143,462 unique terms after stop words are filtered out.

Nonzeros Distributed Unevenly from Sparsity Enforcement

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
government	league	electrons	album	jewish
party		electron	band	jews
war		atoms	albums	judaism
elections		hydrogen		israel
president		isotopes		hebrew

Figure 5. Top five terms in each of five topics produced by NMF as applied to Wikipedia data, with term/topic matrix U forced to have only 50 nonzeros. Nonzero entries are distributed unevenly amongst the columns of U as a result.

The skew of the distribution of nonzeros in the column vectors of U or V is most severe when both the U and the V matrices are made very sparse. In this case, this skew may result in all of the nonzeros being concentrated in a single column in each matrix.

The skew in the number of nonzeros per column of the U and V matrices may indicate that topics with fewer nonzeros are less significant in the original data source than topics with more nonzeros. In that sense this effect can be informative. However, it prevents the user from being able to examine the actual content of a given topic, and in that sense it can constitute a problem, depending on one's goal in using NMF. In Section V, we discuss two methods for addressing this problem.

B. Clustering Accuracy: An empirical measure of document clustering accuracy can be used to examine the effect of sparsity enforcement. This way we can validate the assertion that our method of sparsity enforcement produces output of the same quality as dense NMF.

To do so, we use a corpus of documents that consists of the abstracts of papers from five PubMed academic journals: BioMed Central (BMC) Bioinformatics, BMC Genetics, BMC Medical Education, BMC Neurology, and BMC Psychiatry. The resulting corpus, after stop words are removed, consists of 20,112 unique terms and 7,510 documents.

We can devise a measure of the accuracy of an NMF topic model for this dataset by considering each journal as defining an empirical topic by assuming that an academic journal is a cluster of related documents. We believe that it is a reasonable assumption to say that a clustering algorithm, when applied to the abstracts of papers from academic journals, produces accurate clusters if it groups abstracts from the same journals (provided that the subject matter of each journal is sufficiently different from the subject matters of the others). Our choice of particular journals was made, in part, because they cover different and distinct topics.

We measure the accuracy of each topic by counting the number of pairs of documents that belong to a topic and that are from the same journal, and then by dividing that number by the total number of possible pairs of documents. For each topic in the NMF of this dataset, we consider a document as "belonging" to a topic if its corresponding entry in the V matrix is nonzero. We consider a topic from the NMF to be perfectly accurate if all the documents that belong to that topic are from the same journal, in which case the number of same-journal document pairs is equal to the number of all possible pairs. On the other hand, a topic has the lowest possible accuracy if the documents that belong to it are uniformly distributed among the journals in the dataset. In that case, many of the pairs of documents belong to different journals, and the ratio of the number of same-journal pairs to the number of total possible pairs will be small.

We use the following expression for this measure of the accuracy of an NMF document topic:

$$Acc = \frac{\sum_{i=1}^{n_D-1}\sum_{k=i+1}^{n_D} Jnl(i,k)}{\beta-\alpha} - \frac{\alpha}{\beta-\alpha} \tag{4.6}$$

Where α and β are given by

$$\alpha = \lfloor n_D/n_J \rfloor \left(\frac{n_J(\lfloor n_D/n_J \rfloor - 1)}{2} + n_D (\text{mod } n_J) \right)$$
$$\beta = \frac{n_D(n_D-1)}{2} \tag{4.7}$$

The parameter n_D is the number of documents that belong to the topic whose accuracy we are measuring (i.e., the number of nonzero entries in that column of the matrix V), n_J is the number of journals that was used to make our dataset, and $Jnl(i,k)$ returns 1 if document i comes from the same journal as document k and 0 otherwise. The value of α is the number of same-journal pairs of documents when the documents belonging to a topic are uniformly distributed amongst all of the journals in the dataset; this will be the case when that topic's accuracy is the lowest. The value of β is the maximum possible number of pairs of documents.

The values α and β scale and offset the measure (4.6) so that it is equal to 1 when all of the documents in a topic come from the same journal, and it is equal to 0 when they are perfectly uniformly distributed amongst the journals in the corpus. For cases in which a topic has only one or zero documents belonging to it, we set $Acc = 1$.

Figure 6 shows the results of applying the measure in Equation (4.6) to the NMF of the PubMed journal dataset. For these results, we perform 50 iterations of Algorithm 2 in order to find a five-topic NMF (that is the largest number of correct topics that the NMF should find for five journals). As before, we calculate the NMF when enforcing various levels of sparsity for either the U matrix, the V matrix, or both the U and V matrices.

We find that the accuracy is, in general, higher for sparser U and V matrices, with the accuracy being the lowest for the fully dense conventional NMF. This result is not surprising since we consider a document as "belonging" to a topic if it has any nonzero value in the corresponding entry of the V matrix. Further, we don't take into account the fact that many of the entries for each document in a given topic have small magnitudes, indicating that they probably do not belong to that topic despite the fact that their value is nonzero.

This measure is still useful, however, in allowing us to compare the accuracy of dense NMF to the accuracy of the proposed enforced sparse NMF. We can measure the accuracy of a topic from dense NMF by defining some threshold value below which we would consider the entries

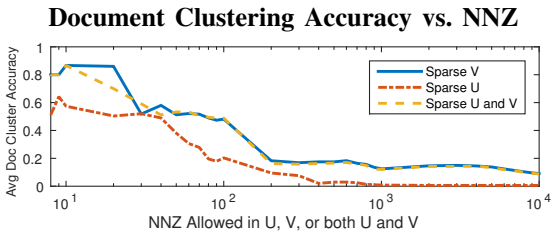

Figure 6. Average document clustering accuracy versus number of nonzeros when using sparsity enforcement for finding the NMF of a data matrix derived from the abstracts of papers from PubMed journals. Accuracy for each individual topic is measured by using equation (4.6), and averaged over each of the topics in a 5 topic NMF. We show the results when enforcing sparsity for just the U matrix, just the V matrix, and both the U and V matrices.

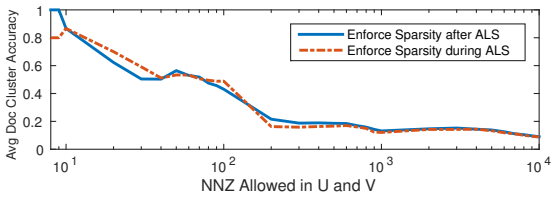

Figure 7. Average document clustering accuracy, as measured by using Equation (4.6), versus the number of nonzeros in the U and V matrices. The plot curve labeled "Enforce Sparsity during ALS" shows the accuracy of the NMF produced by using Algorithm 2, and the plot curve labeled "Enforce Sparsity after ALS" shows the accuracy of the NMF produced by using Algorithm 1, where we have made the final NMF matrices sparse by enforcing sparsity only after the NMF has been calculated. By this measure, Algorithm 2 typically produces NMF document clusters that are at least as accurate as those produced by Algorithm 1.

of V to be zero, and then applying the accuracy measure in Equation (4.6) to the newly sparse V. To use Equation (4.6) with dense matrices, we enforce sparsity after completing ALS iterations when finding the NMF, instead of enforcing sparsity during each ALS iteration. Figure 7 shows the result of measuring the document clustering accuracy for the NMF of the PubMed dataset when we enforce sparsity during each iteration of ALS, as is done in Algorithm 2, and when we enforce sparsity only after we have finished all our ALS iterations using Algorithm 1.

The accuracy of the document clustering is approximately the same, regardless of whether we enforce sparsity during ALS or after finishing ALS. This result is encouraging because it suggests that we are producing equally good topics by enforcing sparsity at each iteration, but it raises an additional question: if we can make our final NMF just as sparse, and just as accurate, by enforcing sparsity only once after we have already calculated the NMF, why do it at each iteration? The reason is because we want to reduce the maximum memory footprint during the entire runtime

of the NMF algorithm, not just just the memory footprint of the final ouput.

C. Memory footprint: The reason for enforcing sparsity at each iteration is that we would like to keep our matrices as sparse as possible at all times when computing the NMF, in order to reduce the memory footprint of both the final result matrices and the intermediate matrices that are generated while calculating the NMF. By enforcing sparsity at each iteration, we can reduce our memory footprint considerably throughout the calculation.

Exactly how much we are able to reduce the memory footprint throughout our calculation depends on the level of sparsity enforced, and on the level of sparsity in our initial guess. Figure 8 shows the maximum number of nonzeros in U and V combined when we enforce sparsity for both matrices during the calculation of the NMF for the PubMed dataset. We show the results for various levels of sparsity in the initial guess U_0. Unsurprisingly, the maximum number of nonzeros that we need to store during our calculations is determined by the sparsity of the initial guess when we are forcing U and V to have very small numbers of nonzeros, and it is determined by the level of sparsity that we enforce in U and V when they are allowed to be more dense than the initial guess.

This example demonstrates the memory benefits of using sparsity enforcement at each iteration: we can reduce the amount of memory that we need to use to store U and V by more than an order of magnitude.

Maximum NNZ Stored When Calculating Sparse NMF

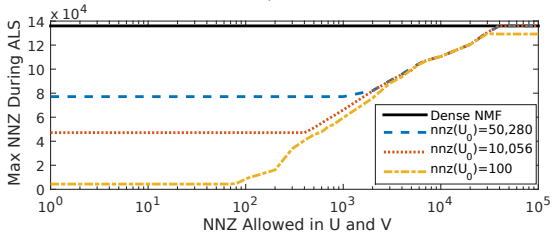

Figure 8. Plot curves showing the maximum number of nonzeros that need to be stored for the U and V matrices combined when using Algorithm 2 to calculate the NMF of the PubMed dataset. Results are shown for several initial guesses with varying numbers of nonzeros. Depending on the sparsity of the initial guess, the maxmimum memory footprint can be reduced from the dense case by over an order of magnitude.

V. The Sequential NMF

In Section IV-A we showed that, when using Algorithm 2, we may produce an NMF that has unevenly distributed terms and documents among its topics. This problem is typically only as severe as it is in the results in Figure 5 when forcing our matrices to be very sparse but, in our opinion, the fact that Algorithm 2 can be used to produce matrices of such extreme sparsity is one of its benefits, and so we have considered two ways of alleviating this problem.

One particularly straightforward method to ensure even distribution of nonzeros among columns of our matrices is to enforce sparsity for each column individually rather than for the matrix as a whole. This procedure results in convergence and produces good topic models, but at the cost of reduced performance. This performance reduction occurs because, for most sparse matrix storage formats, addressing the entries in specific columns of a matrix is slower than addressing the entries of that matrix irrespective of which column they belong to. This additional time to address the contents results in reduced performance.

Another way to ensure an even distribution of nonzeros among the columns of our matrices is to find the NMF topics sequentially by converging one topic at a time. We can do this by considering the NMF using block matrices

$$A \approx UV^T = [U_1\ U_2][V_1\ V_2]^T = U_1V_1^T + U_2V_2^T \quad (5.8)$$

where U_1 and V_1 are matrices whose column vectors consist of previously converged NMF topics, and U_2 and V_2 are single column vectors that represent the new topics that we seek to find. We can derive the update rules for finding U_2 and V_2 by rewriting the original minimization problem using the matrix blocks:

$$\min ||A - UV^T||_2^2 = \min ||A - U_1V_1^T - U_2V_2^T||_2^2 \quad (5.9)$$

The solution for one of U_2 or V_2, while holding the other constant, is then a modified least squares solution:

$$V_2 = (A^TU_2 - V_1U_1^TU_2)(U_2^TU_2)^{-1} \quad (5.10)$$

$$U_2 = (AV_2 - U_1V_1^TV_2)(V_2^TV_2)^{-1} \quad (5.11)$$

We can find a k-topic NMF using these update rules by doing projected ALS k times; we store each new topic that we generate as additional columns of the matrices U_1 and V_1 and then find the next topic by doing projected ALS again using Equations (5.10) and (5.11). We call this procedure sequential ALS because we are finding our NMF as a sequence of individual topics rather than as a block of topics. The full algorithm for sequential ALS is given in Algorithm 3.

A similar procedure was previously proposed in [21], where the author uses it as part of a method for generating a high-rank NMF in order to reconstruct missing data. The article in [21] does not use any means of ensuring sparsity in the resulting NMF, however, which would be necessary for the proposed algorithm to be used successfully on realistically large data sets. Sequential NMF is also similar to the Hierachical Alternating Least Squares (HALS) procedure [22], which was developed in order to

Algorithm 3 Sequential ALS NMF

Input: data matrix $A \in \mathbb{R}^{n \times m}$, initial guess $U_0 \in \mathbb{R}^{n \times k_2}$, maximum NNZ(U) t_u and maximum NNZ(V) t_v, topics per block k_2, number of blocks η (total number of topics $k = \eta \times k_2$)
Output: Approximation $UV^T \approx A$, $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{m \times k}$

START: Find $U_1 \in \mathbb{R}^{n \times k_2}$ and $V_1 \in \mathbb{R}^{m \times k_2}$ such that $U_1 V_1^T \approx A$ using Algorithm 2, using U_0 as initial guess.

For $i = 2$ **to** η

Set $U_2 = U_0$

Do until convergence:

1. Find V_2 using Equation (5.10), and set negative entries of V_2 to zero.
2. Sort nonzero entries of V_2, keep only t_v largest nonzeros in V_2
3. Find U_2 using Equation (5.11), and set negative entries of U_2 to zero.
4. Sort nonzero entries of U_2, keep only t_u largest nonzeros in U_2

end do

Append the column vectors of U_2 and V_2 to the matrices U_1 and V_1 respectively, increasing their rank by k_2: $U_1 = [U_1\ U_2]$, $V_1 = [V_1\ V_2]$.

end for

Output $U = U_1$, $V = V_1$
END

improve convergence by iterating on columns of U and V individually.

Figure 9 shows the results of enforcing sparsity column by column and the results of using sequential NMF for the same Wikipedia data matrix that was used to produce the table in Figure 5. Both methods produce an even distribution of nonzero entries among the columns of our matrices. Column-wise sparsity enforcement yields good topic terms, and sequential ALS yields good topic terms with the exception of topic 4. This behavior is consistent with what we have observed for the sequential ALS algorithm. While the algorithm often produces good topic terms, it is less robust than the typical projected ALS is. Figure 10 shows the accuracy of the NMF document topics when we use sequential ALS and column by column sparsity enforcement on the PubMed dataset, as measured by using Equation (4.6). By this measure, both methods produce document clusters that are approximately as accurate as the document clusters produced when we use the unmodified Algorithm 2.

In order to compare the performance of sequential NMF and column-wise sparsity enforcement, Figure 11

Sparsity Enforcement with Even Nonzero Distribution

Enforce Sparsity by Column

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
government	proteins	electrons	album	jewish
party	protein	electron	band	jews
war	cells	atoms	music	judaism
president	cell	atom	albums	hebrew
election	dna	hydrogen	songs	torah

Enforce Sparsity with Sequential ALS

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
government	city	album	film	game
war	population	band	church	games
party	airport	music	empire	players
military	census	albums	country	team
soviet	county	songs	united	league

Figure 9. Top five terms for each of five topics from Wikipedia data. This time we enforce sparsity for each column individually or by using sequential topic generation, with a limit of 10 nonzero entries per topic (for a total of 50 nonzero entries in the U matrix). Compare with the table in Figure 5; here our topic terms are evenly distributed.

Document Clustering Accuracy with Sequential and Column-wise Topic Sparsity

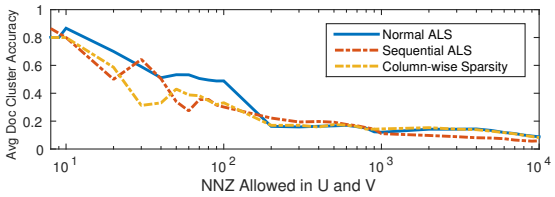

Figure 10. Mean document clustering accuracy as measured by Equation (4.6) when calculating a five-topic NMF of the PubMed dataset by using either Algorithm 3 or Algorithm 2 with sparsity enforced for each column of U and V individually.

shows the time required for 100 ALS iterations for finding a five-topic NMF for the PubMed journal dataset, using normal projected ALS NMF with sparsity enforcement, projected ALS NMF with column-wise sparsity enforcement, and sequential ALS. Enforcing sparsity for each column individually takes longer than doing so for the entire matrix at once, as we would expect when using sparse matrix formats. Sequential ALS is quite a bit faster than the other two methods. This result is not surprising, as sequential ALS does not require the use of a matrix inverse when U_2 and V_2 of Equations (5.10) and (5.11) have rank 1, as they do here; in that case, the inverse amounts to floating point division. Despite indications that sequential ALS provides less coherent term topics than the regular ALS NMF, this improvement in runtime suggests that it bears further investigation.

VI. Discussion and Conclusion

In this article, we described a method to enforce sparsity in the computation of the NMF of a potentially large dataset. By setting the level of sparsity in our matrices

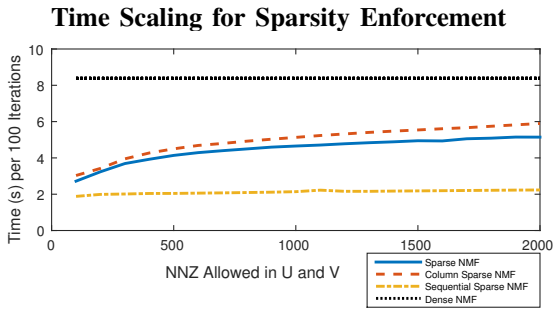

Figure 11. Time required for 100 ALS iterations when finding a 5 topic NMF for the PubMed dataset. Results are shown using sparsity enforcement for the whole U and V matrices at once, for each column of U and V individually, and for columns of U and V generated sequentially. For the normal and column-wise NMF results, 100 projected ALS iterations are performed. For the Sequential ALS NMF, 20 iterations are performed for each of 5 topics that are generated, for a total of 100 ALS iterations.

that we desire at each iteration of alternating least squares, we can substantially reduce the memory resources that are required to compute NMF without sacrificing additional compute resources. In spite of the simplicity of the proposed approach, our experiments indicate that the algorithm converges at least as quickly and reliably as regular, dense projected ALS and that the NMF topic models that it produces are similarly accurate.

Using the methods described in this article, we can produce an NMF with matrices of fairly extreme sparsity, relatively inexpensively. In doing so we find that we produce NMF topic models with very unevenly distributed terms and documents among the topic clusters. We can produce topic models with perfectly evenly distributed topic terms and documents by enforcing sparsity for each topic individually. If we do this directly by enforcing sparsity for each column of U and V individually, then we find that we produce good term topics at the cost of sacrificing performance resulting from the slowdown caused by accessing sparse matrix formats column by column. If we do this by converging each topic sequentially, we produce quality term topics less reliably but with a considerable improvement in speed. Improving these techniques so that we do not have to sacrifice computation time or topic quality is a subject of continuing research.

We believe that these methods of sparse NMF are amenable to implementation using the GraphBLAS kernels. The dense ALS algorithm uses the GraphBLAS kernels of SpRef/SpAsgn, SpGEMM, Scale, SpEWiseX and Reduce kernels. To enforce sparsity, a user defined functions that sets an element to zero if it is greater than a particular threshold applied either matrix wide for Algorithm 2 or column-by-column for Algorithm 3 on intermediate products can ensure memory and computational efficiency.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[2] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1999, pp. 50–57.

[3] S. Gerard and J. M. Michael, "Introduction to Modern Information Retrieval," 1983.

[4] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[5] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[6] D. Lee and S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, 2001, pp. 556–562.

[7] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[8] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 8, pp. 1548–1560, 2011.

[9] J. Yang, S. Yang, Y. Fu, X. Li, and T. Huang, "Non-negative graph embedding," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[10] J.-Y. Wang, I. Almasri, and X. Gao, "Adaptive graph regularized nonnegative matrix factorization via feature selection," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 963–966.

[11] V. Gadepally, J. Bolewski, D. Hook, D. Hutchison, B. Miller, and J. Kepner, "Graphulo: Linear algebra graph kernels for NoSQL databases," in *IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW),*. IEEE, 2015.

[12] M. Berry, M. Browne, A. Langville, P. Pauca, and R. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational statistics & data analysis*, vol. 52, no. 1, pp. 155–173, 2007.

[13] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[14] J. Kepner and V. Gadepally, "Adjacency matrices, incidence matrices, database schemas, and associative arrays," in *International Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*. IEEE, 2014.

[15] "The graph blas forum," http://istc-bigdata.org/GraphBlas/.

[16] J. Eggert and E. Körner, "Sparse Coding and NMF," in *IEEE International Joint Conference on Neural Networks*, vol. 4. IEEE, 2004, pp. 2529–2533.

[17] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *The Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[18] R. Peharz and F. Pernkopf, "Sparse nonnegative matrix

factorization with l0-constraints," *Neurocomputing*, vol. 80, pp. 38–46, 2012.

[19] P. O. Hoyer, "Non-negative Sparse Coding," in *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*. IEEE, 2002, pp. 557–565.

[20] "Reuters-21578 text categorization test collection," http://www.daviddlewis.com/resources/testcollections/reuters21578/.

[21] M. D. Gupta, "Additive non-negative matrix factorization for missing data," *arXiv preprint arXiv:1007.0380*, 2010.

[22] A. Cichocki and P. Anh-Huy, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 3, pp. 708–721, 2009.