Parallel Non Negative Matrix Factorization for Document Clustering

Khushboo Kanjani, khush@cs.tamu.edu, Texas A & M University

07 May 2007

Abstract

Non-negative matrix factorization has been used as an effective approach for document clustering lately. One advantage of this method is that clustering results can be directly concluded from the factor matrices. This project gives parallel implementation of three algorithms for Non-negative matrix factorization. Experiments of these parallel algorithms for large datasets shows good speedup for each of these methods.

1 Introduction

Document Clustering is the process of collecting similar documents into clusters. Many document clustering techniques have been developed because of applications in information retrieval, web navigation, pattern recognition and organization of huge volumes of text documents. These techniques can be broadly divided into categories :

- **Hierarchical** - The algorithms find successive clusters using the existing clusters in top-down(divisive) or bottom-up(agglomerative) manner.
- **Partitioning** (Flat Clustering) - This approach divides the documents into disjoint clusters. The various methods in this category are : K-Means Clustering, probabilistic clustering using the Naive Bayes or Gaussian model, latent semantic indexing (LSI), spectral clustering, Non-negative Matrix Factorization(NMF).

Some extensions of NMF have been presented by adding constraints like orthogonality and sparsity on the factor matrices. These constraints improve the performance of the method.

The organization of the paper is as follows. Section 2 discusses the related work done in document clustering and the algorithms for NMF. Section 3 describes the three algorithms for non-negative matrix factorization and in Section 4 the database and the preprocessing done to create to the document/word-weight matrix is discussed. Section 5 gives a parallel version of the NMF algorithm explains its implementation using openmp. Section 6 gives a brief discussion of the source code and how to compile and run it. Section 7 discusses the accuracies of the document clustering results and the speed-up with the parallel code.The conclusions are in Section 8.

2 Related Work

In [7],Lee and Seung introduced the use of non-negative matrix factorization for learning parts of objects. In [8], they give two algorithms for NMF considering the objective functions- euclidean distance and Kullback-Leibler divergence. Wei Xu et. al were the first ones to use NMF for document clustering in [14]. Yang et al. [15] extended this work by adding the sparsity constraints because sparseness is one of the important characters of huge data in semantic space. In a recent work by Ding et al. [5], an algorithm for bi-orthogonal 3-factor NMF is presented which provides a strong capability of simultaneously clustering rows and columns. Algorithms for low-rank approximate non-negative matrix factorization are presented in [2]. Email surveillance [2, 1] is another significant application of NMF.

In a survey paper on document clustering [12] published in 2000, the main approaches for document clustering discussed are agglomerative hierarchical clustering and K-means and its variants[6].In a recent paper [11] a MinMax-Cut [4] based algorithm for document clustering is presented.

A parallel algorithm which combines the k-means and the Principal Component Analysis approach is presented in [13]. To the best of my knowledge, there is only one parallel algorithm for NMF presented in [10]. The parallel version I propose in this work is different from the one presented in [10].

3 Algorithm

NMF algorithms find non-negative factors of a given non-negative matrix. The aim is to minimize the objective function:

$$J = \|V - WH\| \tag{1}$$

where $V \in \Re_+^{p \times n}, W \in \Re_+^{p \times d}, H \in \Re_+^{d \times n}, \|A\|$ is the squared sum of all the elements in the matrix A. **p, n, d** are the number of words , documents and clusters respectively. This objective function is non-increasing for the iterative update rules mentioned in Algorithm 1, 2 and 3. The solution to this objective function J is not unique. If **W** and **H** are a solution to J then **WD** and **inv(D)H** are also solutions to J for any positive diagonal matrix **D**. So various constraints are added to the matrices W and H so that the solution is unique.
Algorithm 1 adds a sum constraint on the columns of W.

$$W_{ij} = W_{ij} \sum_k \frac{V_{ik}}{(WH)_{ik}} H_{jk} \qquad W_{ij} = \frac{W_{ij}}{\sum_k W_{kj}}$$

$$H_{ij} = H_{ij} \sum_k W_{ki} \frac{V_{kj}}{(WH)_{kj}}$$

Algorithm 1: Iterative update rules for W and H from the paper [7]

Algorithm 2 adds a constraint that the euclidean distance of the column vector in **W** is one.

$$W_{ij} = W_{ij} \frac{(VH^T)_{ij}}{(WHH^T)_{ij}}$$

$$H_{ij} = H_{ij} \frac{(W^T V)_{ij}}{(W^T WH)_{ij}}$$

$$H_{ij} = H_{ij} \sqrt{\sum_1 H_{ij}^2} \qquad W_{ij} = \frac{W_{ij}}{\sqrt{\sum_1 H_{ij}^2}}$$

Algorithm 2: Iterative update rules for W and H from the paper [14]

The update rule for H in Algorithm 3 adds an orthogonality constraint $\mathbf{H^T H = I}$. It has been proved in [3] that this constraint makes NMF equivalent to k-means clustering.

$$W_{ij} = W_{ij} \frac{(VH^T)_{ij}}{(WHH^T)_{ij}}$$

$$H_{ij} = H_{ij} \sqrt{\frac{(V^T W)_{ij}}{(H^T H V^T W)_{ij}}}$$

Algorithm 3: Iterative update rules for W and H with orthogonality constraint for H from the paper [5]

In all these algorithms, document d_j is assigned to cluster k if $H_{kj} = max_i H_{ij}$. An intuition on why W and H group similar documents into clusters is that each element w_{ij} of W represents the degree to which word t_i belongs to cluster j, while each element h_{ij} of matrix H indicates to which degree document j belongs to cluster i.

4 Database

The database I use for experiments is the 20 newsgroups database which contains 19997 documents evenly divided into 20 Usenet newsgroups. Some of these documents are small one two line email replies and so do not have enough meaningful words to categorize it into a cluster. The tf-idf code removes such documents which do not contain any of the top 1000 words selected by mutual information gain.

4.1 PreProcessing

We use the rainbow library [9] for preprocessing the dataset. Stop words which do not contribute to the semantics of the document(like a,the,he she and similar pronouns) and html tags are removed from each file. Stemming is done to combine words with same semantics but different forms/tense. For example (sleeping will be truncated to sleep.) The rainbow command used for this is:
rainbow -d model -h –use-stemming –index 20_newsgroups/
The top 1000 words based on mutual information gain are used for our experiments. The vocabulary is reduced to top 1000 words using the following rainbow command:
rainbow -d model –prune-vocab-by-infogain=1000.
Then the document/word-count matrix is created by using the print-matrix option.
rainbow -d model –print-matrix=aiw

4.2 Creating the input matrix V

The word count of the words in each document is not effective way to represent a document vector for clustering. There are many reasons for this:

- The length of all documents is not uniform. So a lengthy document will have higher word counts.
- A word frequently present in all documents will not be useful for clustering and so should not be used.

So we use the tf-idf(term frequency- inverse document frequency) weight as the weight for each word in a document. The document vector is constructed by the tf-idf weights of the top 1000 words in the dataset.

$$tf_i = \frac{n_i}{\sum N_k} \tag{2}$$

- the numerator is the number of occurrences of the considered word in the document.
- the denominator is the sum of the occurrences of all 1000 used words in the document.

$$idf_i = log(\frac{|D|}{|d : d \in t_i|}) \tag{3}$$

The inverse document frequency(idf) is a measure of the general importance of a word in the dataset.

- the numerator is the number of the documents in the dataset.
- the denominator is the number of documents where the word i appears

$$tfidf_i = tf_i * idf_i \tag{4}$$

A high value in *tfidf* is reached by a word with high term frequency (in the given document) and a low document frequency of the word in the whole collection of documents. The *idf* term is 0 for words which are present in all documents and so it tends to filter out common terms.

5 Parallel Version

I use openmp for parallelizing the code for NMF. The code gives good speedup and efficiency when d¡¡min(p,n) which is normally the case in the application of document clustering. The number of words and documents in the data set is much higher than the possible number of categories.

I assign to each process a start and end index for p and n. So each process is assigned a block of rows or columns of the whole matrix for all the computations like matrix multiplication, matrix transpose etc. A smart placement of the single and barrier directives in the code makes it efficient. The algorithm reads the input matrix V from a file in the start and writes the clustering results to a file in the end. This reading from and writing to file part is not parallelized.

6 Source Code

The file names and their use:

1. **common.cpp** contains all the matrix related functions like initialize, multiply, transpose, read from file and write to file.
2. **tf-idf.cpp** takes the input as the word count matrix and creates the document/tf-idf weight matrix. The output of this code is used by the nmf algorithms.
3. **parallel-nmf.cpp** is the parallel implementation of algorithm 1.
4. **parallel-nmf-standard.cpp** is the parallel implementation of algorithm 2.
5. **parallel-nmf-orth.cpp** is the parallel implementation of algorithm 3.
6. **category.sh** is a shell script to make the document/category matrix.
7. **getac.cpp** is a C++ code to calculate the accuracy of the clustering results. It uses the output of the NMF code and the matrix created by rainbow.

Command for compiling:
% **xlC_r -qsmp=omp -o nmf parallel-nmf.cpp**
To run the program:
% **./nmf p n d m np InputFile ResultFile**
The program takes 7 arguments:
p: number of words
n: number of documents
d: numbers of clusters
m: number of iterations
np: number of processors
InputFile: name of the file which stores the input V matrix
ResultFile: name of the file where the result should be saved.

7 Performance Evaluation

The three algorithms are tested for two datasets:

1. **rec-data:** This dataset consists of documents from categories rec.autos, rec.motorcycles, rec.sport.baseball and rec.sport.hockey. The total number documents after processing are 3966. Number of clusters(d) is 4. The top 1000 words based on mutual information gain are used.

2. **talk-data:**This dataset consists of documents from categories talk.politics.guns, talk.politics.mideast and talk.politics.misc. Total number documents after processing are 2939. Number of clusters(d) is 3. The top 500 words based on mutual information gain are used.

7.1 Accuracy of Document clustering

The performance of the clustering algorithm is evaluated by calculating the accuracy[14] as defined in Eq.5:

$$AC = \frac{\sum_{i=1}^{n} \delta(\alpha_i, map(l_i))}{n} \tag{5}$$

The accuracy was higher for rec-data as compared to talk-data because the categories in rec-data were more independent than in talk-data. The accuracy of Algorithm 3 is low because the clustering results of simple k-means were used to initialize W and H and for such large sparse matrices,

the final clustering result was biased by the initial data of H. For algorithm 1 and 2, W and H were initialized with random values.

Table 1: Accuracy of the three algorithms for the two data-sets.

	$rec-data$	$talk-data$
Algorithm 1	0.85	0.79
Algorithm 2	0.81	0.74
Algorithm 3	0.76	0.72

7.2 Parallel speedup

The testing was done on the IBM p690 NCSA machine. The code was run interactively for 1, 2 and 4 processors and jobs were submitted to the queue for 8 and 16 processors.The execution times for the three algorithms are in Table 1 for rec-data and Table 2 for talk-data. T,S,E correspond to Time, SpeedUp and Efficiency. All the testing results are for 50 iterations of the code. The corresponding plots are in Figure 1 and Figure 2 respectively. It is observed that efficiency is more than 90% for upto 8 processors for all the methods.

Table 2: Execution time for the three algorithms for the rec-data.

NP	$T1$	$S1$	$E1$	$T2$	$S2$	$E2$	$T3$	$S3$	$E3$
1	109.5	-	-	17.5	-	-	18.4	-	-
2	59.71	1.83	91.6	8.9	1.96	98.3	9.77	1.89	94.8
4	28.18	3.88	97.1	4.51	3.91	98.1	6.21	3.67	93.9
8	14.65	7.47	93.4	2.66	6.57	82.2	2.51	7.37	92.1
16	9.62	11.38	71.1	1.84	9.51	59.4	1.76	10.45	65.3

8 Conclusion

This work presents parallel implementations of the various algorithms for NMF. A good amount of speedup was observed in all the methods. Also the recent algorithms 2 and 3 take much lesser time as compared to algorithm 1 but the accuracy is higher of Algorithm 1.

Table 3: Execution time for the three algorithms for the talk-data.

NP	$T1$	$T2$	$T3$
1	23.06	4.95	6.4
2	11.73	2.57	3.5
4	6.168	1.41	1.6
8	3.01	0.88	0.7
16	2.29	0.57	0.4

ACKNOWLEDGMENTS I am grateful to Dr. Vivek Sarin for his guidance. The material he taught in class was very helpful for the project. I thank Chris Ding and Wei Xu for the electronic discussions to clarify the initialization of the factor matrices and the check for convergence.

References

[1] Michael W. Berry and Murray Browne. Email surveillance using non-negative matrix factorization. *Computational and Mathematical Organization Theory*, pages 249–264, 2005.

[2] Michael W. Berry, Murray Browne, Amy Langville, Paul Pauca, and Robert Plemmons. Algorithms and applications for approximate non-negative matrix factorization. *Computational Statistics and Data Analysis*, 2006.

[3] C.Ding, X. He, and H.D.Simmon. On the equivalence of nonnegative matrix factorization and spectral clustering. *Proceedings of the SIAM Data Mining Conference*, 2005.

[4] C.Ding, X.He, H.Zha, M. Gu, and H.Simon. A min-max cut algorithm for graph partitioning data clustering. *Proceedings IEEE International Conference on Data Mining*, pages 107–114, 2001.

[5] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix tri-factorizations for clustering. *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 126–134, 2006.

[6] V. Faber. Clustering and the continuous k-means algorithm. *Los Alamos Science*, pages 138–144, 1994.

[7] Daniel D. Lee and H.Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature, 401*, pages 788–791, 1999.

[8] Daniel D. Lee and H.Sebastian Seung. Algorithms for non-negative matrix factorization. *NIPS 13*, 2001.

[9] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling,text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[10] Stefan A. Robila and Lukasz G.Maciak. A parallel unmixing algorithm for hyperspectral images. Technical report, Center for Imaging and Optics, Montclair State University, 2006.

[11] S.Oliveira and S.C.Seok. Matrix-based algorithms for document clustering. *International Workshop on Combinatorial Scientific Computing (CSC05)*, 2005.

[12] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, 2000.

[13] Shuting Xu and Jun Zhang. A parallel hybrid web document clustering algorithm and its performance study. *The Journal of Supercomputing*, 30:117–131, 2004.

[14] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. *Proceedings of ACM SIGIR*, pages 267–273, 2003.

[15] C.F. Yang, Mao Ye, and Jing Zhao. Document clustering based on nonnegative sparse matrix factorization. *International Conference on advances in Natural Computation*, pages 557–563, 2005.

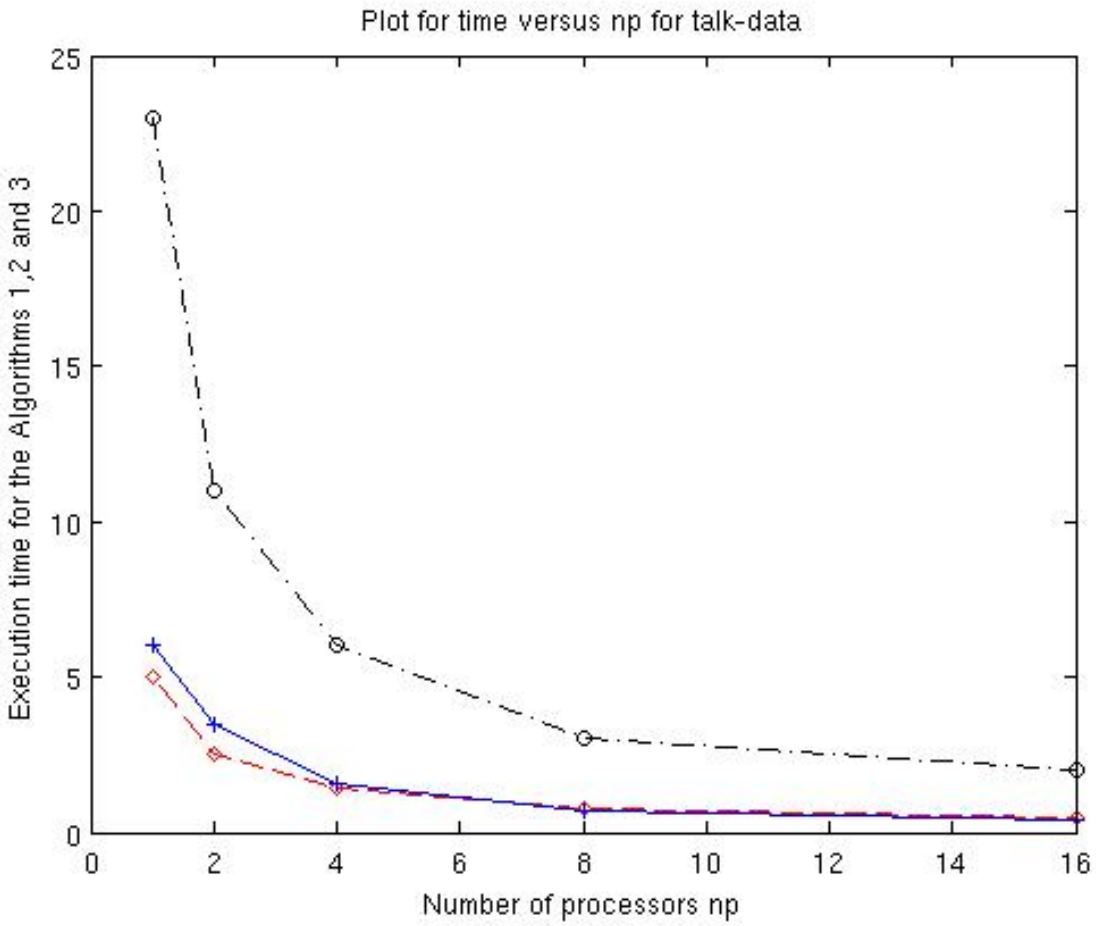

Figure 1: Time vs NP for rec-data

———	Algorithm 1
- - - - - - - - -	Algorithm 2
———	Algorithm 3

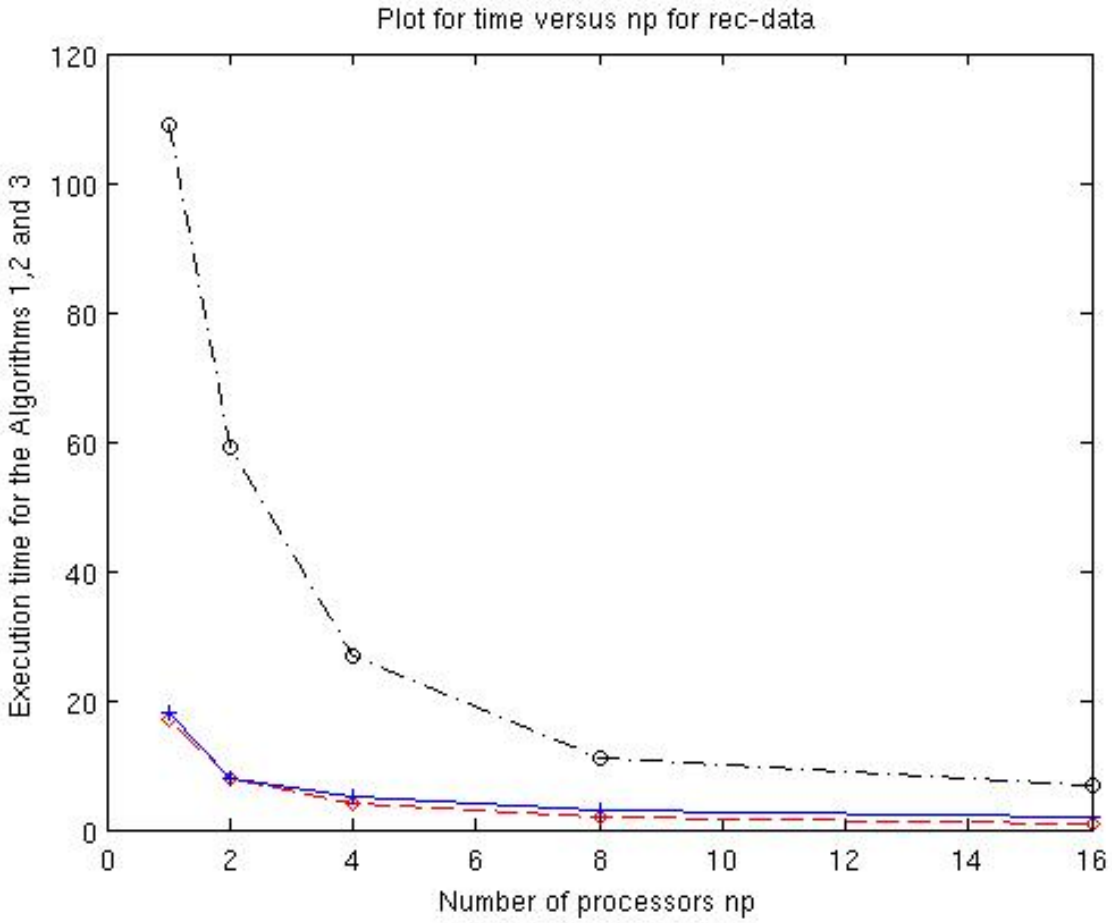

Figure 2: Time vs. NP for talk-data