Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-26

Three-Way Data Recommendation Method and Application

Linyao Tang, Ruifang Liu

***Abstract*—Nonnegative Tensor Factorization has previously been used in many multi-way data analyses. We use NTF model to do personalized paper recommendation. For recommendation, we analyze four different multiplicative algorithms for NTF based on different decomposition models and different optimization functions. On one hand, one part of algorithms use CP decomposition, the other part use Tucker decomposition. On the other, half of algorithms minimize the least squares error while the others minimize the Kullback-Leibler divergence. Further, we also compare recommendation performance with different rank NTFs. From our experiments, nonnegative Tucker decomposition based on KL divergence has the better result, and to some extent, lower rank NTF can get most of information from dataset.**

***Keywords*—personalized recommendation, sparsity, tensor factorization, multiplicative update**

I. Introduction

With latent semantic analysis (LSA[1]) applications in the field of information retrieval, matrix factorization are frequently used today in variety fields. Especially in the recommended system[2], e.g., participants using collaborative filtering method won the first place [3].

Gradually transformations become nonnegative, data interpretability become stronger, and nonnegative matrix factorization (NMF) can well represent the local characteristics of things, so it is successfully applied to psychometric, chemometrics, image analysis, graph analysis and signal processing. The most popular approach of NMF algorithm is proposed by Lee et al.[4], which uses multiplicative update.

However, 2D matrix sometimes cannot represent data more than three-mode well. E.g., recommending papers according keywords experiment in our study, if we only use papers and their keywords' relation, this will lose other latent but important information such paper's authors, venue and so on. Therefore, if we directly transfer high dimension structural data into low data reluctantly, it will lose a lot of structure. Data mining in higher dimension becomes a trend.

Linyao Tang
Beijing University of Posts and Telecommunications
Beijing, 100876, China

Ruifang Liu
Beijing University of Posts and Telecommunications
Beijing, 100876, China

Nowadays, machine learning method based on tensor factorization has been widely studied and applied. For recommendation, we cannot know exactly all values of tensor structural data, so sparse tensor factorization is particularly important in our study.

We make use of authors information in CiteSeer dataset besides title and citation. So for a given title or keywords, we can recommend papers for a particular author, it is more appropriate for recommendation because different authors are good in different areas. We have more reasons to believe that authors would like to read these papers about their areas.

In this paper, we use sparse tensor factorization to do personalized paper recommendation. There are many factorization tools such as the MATLAB Tensor Toolbox by Badar et at[5]. However, this toolbox not only cannot handle large-scale sparse tensors, but also cannot select flexibly optimization function and decomposition method.

Because of the sparsity of the tensor data, it decomposition can certainly be optimized. We are inspired by the algorithm in [6], where we take advantage of alternative sigma symbols. It can greatly reduce Tucker decomposition's computation complexity on sorted sequential data. we compare the different factorization methods and different optimization functions on three-mode data recommendation by doing experiments. In addition, we compare efficiencies under different rank decompositions.

In the remaining part, we survey related works in section II. In section III, we introduce our main models with some details on parameters computations, and analyze the iterative complexity of training algorithm. In section IV, we apply our methods on personalize recommendation. Finally, we give conclusions in section VI.

II. Related Work

Lee et al.[4] propose nonnegative matrix factorization, i.e., variables are nonnegative in model. This decomposition is widely useful as it results' easy interpretable representations. For another thing, In [7], Donoho D et al. demonstrated the feasibility of large data compression and completion, and matrix completion theory is also a new information collection method [8].

Kim et al. [9] expand the NMF to in higher dimension, i.e., nonnegative tensor factorization (NTF), and give proofs of convergence. After that, a hot wave of high dimension nonnegative tensor factorizations are rising. Mørup et al. [10] proposed algorithm for sparse NTF. It guarantee to find the global minima by adding to a part of parameters' constraints.

There are many experiment compare NTF with NMF. E.g., T.hazan et al. [11] use the NTF to exact image's local parts feature, and compare with using nonnegative matrix factorization on a matrix of vector images. The results show that treating these images as a three-order tensor is better than 2D matrices.

B.W.Bader et al.[12] put email data as a 3-dimenstion tensor when studying Enron Email data. Chi et al. [6] applied NTF to analyze the blogosphere and personalized recommendation. Phan et al.[13] apply tensor decompositions to feature extraction and classification in high dimensional data. Nickel et al. [14] solved relational learning task with NTF. More application about tensor factorization can be seen in [15].

In tensor completion, Liu ji et al. [16] proposed low-rank tensor completion(LRTC) to estimate miss value in incomplete tensor.

III. Model

A. *Background*

A brief introduction of mathematical notations is presented, which will be used in later sections. Tensor is a promotion of 1D vector and 2D matrix, which can represent high dimensional information directly. In this paper, scalars by lowercase letter including Latin characters (e.g., a,b,λ), vectors by lowercase(e.g., $\vec{p},\vec{q}$), matrices by capital letter (e.g., X,Y), and tensors by calligraphic letters(e.g., $\mathcal{A},\mathcal{B}$). Additional, we reserve i,j,k,l,m,n,p,q,r to indicate the indices of matrices and tensors, O,H,I to indicate all-zeros matrix, all-ones matrix and identity matrix respectively, $\mathcal{O},\mathcal{H},\mathcal{I}$ to indicate all-zeros tensor, all-ones tensor and identity tensor respectively. Other important notations about tensor are in Table. I.

TABLE I. NOTATION DESCRIPTION

Notation	Description
$\lVert\mathcal{A}\rVert$	$(\sum_{ijk}\mathcal{A}_{ijk}^{2})^{1/2}$
$\mathcal{A}+\mathcal{B}$	$(\cdot)_{ijk}=\mathcal{A}_{ijk}+\mathcal{B}_{ijk}$
$\mathcal{A}-\mathcal{B}$	$(\cdot)_{ijk}=\mathcal{A}_{ijk}-\mathcal{B}_{ijk}$
$\mathcal{A}.*\mathcal{B}$	$(\cdot)_{ijk}=\mathcal{A}_{ijk}*\mathcal{B}_{ijk}$
$\mathcal{A}./\mathcal{B}$	$(\cdot)_{ijk}=\mathcal{A}_{ijk}/\mathcal{B}_{ijk}$
$\langle\mathcal{A},\mathcal{B}\rangle$	$\sum_{ijk}\mathcal{A}_{ijk}*\mathcal{B}_{ijk}$
$LS(\mathcal{A},\mathcal{B})$	$\lVert\mathcal{A}-\mathcal{B}\rVert^{2}$
$KL(\mathcal{A}\parallel\mathcal{B})$	$\sum_{ijk}{}_{ijk}\log\frac{\mathcal{A}_{ijk}}{\mathcal{B}_{ijk}}-\sum_{ijk}\mathcal{A}_{ijk}+\sum_{ijk}\mathcal{B}_{ijk}$

In (L,M,N) rank tensor decomposition, 3-dimension tensor can be represented by a smaller core tensor and n related factor matrices. We use $X_{I\times L},Y_{J\times M},Z_{K\times N}$ to denote the n matrices and $\mathcal{G}_{L\times M\times N}$ to denote the core tensor, X_{il},Y_{jm},Z_{kn} to denote matrices' elements and $\mathcal{G}_{lmn}$ to denote core tensor's elements. We call n-rank decomposition when $L=M=N=n$. Two common decomposition methods are CP decomposition (CANDECOMP/PARAFAC) [17][18] and Tucker decomposition [19]. CP decomposition split tensor into n factor matrices, where is denoted by (1)

$$\mathcal{A}\approx[\![X,Y,Z]\!] \tag{1}$$

Which can be written in an element-wise form as (2)

$$\mathcal{A}_{ijk}=\sum_{n}X_{in}*Y_{jn}*Z_{kn} \tag{2}$$

Tucker decomposition split tensor into n factor matrices, where is donated by (3)

$$\mathcal{A}\approx[\![\mathcal{G};X,Y,Z]\!]\text{ , }\mathcal{G}\text{ is a core tensor.} \tag{3}$$

Which can be written in an element-wise form as (4)

$$\mathcal{A}_{ijk}=\sum_{lmn}\mathcal{G}_{lmn}*X_{il}*Y_{jm}*Z_{kn} \tag{4}$$

CP decomposition can be seen as a special case from Tucker decomposition, when $L=M=N$ and core tensor $\mathcal{G}$ is (5)

$$\mathcal{G}_{lmn}=\begin{cases}1 & l=m=n\\ 0 & else\end{cases} \tag{5}$$

That is $\mathcal{G}=\mathcal{I}$.

Finally, we define some symbols to denote few certain computation, which are in Table II.

TABLE II. SYMBOL DEFINITION

Symbol	Definition
$\langle\mathcal{A},[\![\bullet;X,Y,Z]\!]\rangle$	$(\cdot)_{lmn}=\sum_{ijk}\mathcal{A}_{ijk}*X_{il}*Y_{jm}*Z_{kn}$
$\langle\mathcal{A},[\![\mathcal{G};\bullet,Y,Z]\!]\rangle$ [a]	$(\cdot)_{il}=\sum_{jk,mn}\mathcal{A}_{ijk}*\mathcal{G}_{lmn}*Y_{jm}*Z_{kn}$
$\langle\mathcal{A},[\![\bullet,Y,Z]\!]\rangle$ [a]	$(\cdot)_{in}=\sum_{jk}\mathcal{A}_{ijk}*Y_{jn}*Z_{kn}$

a. Same as *Y* and *Z*

B. *Nonnegative Tensor Factorization*

We formally consider algorithms for solving an nonnegative tensor factorization (NTF). NTF can be transformed as follows:

Give a nonnegative tensor $\mathcal{A}$, find a nonnegative tensor $\mathcal{G}$ and nonnegative matrices X,Y,Z such as $\mathcal{A}\approx[\![X,Y,Z]\!]$ or $\mathcal{A}\approx[\![\mathcal{G};X,Y,Z]\!]$. The above are CP decomposition and Tucker decomposition.

1) Cost functions

To find a proper approximate for tensor factorization. We need to define cost functions that quantify the cost of approximation. The most common measure are least squares(LS) minimization based on Gaussian noise model and Kullback-Leibler(KL) divergence minimization based on Poisson noise.

Least squares minimization use the Euclidean distance between $\mathcal{A}$ and $\mathcal{B}$, i.e., $LS(\mathcal{A},\mathcal{B})$, and KL-divergence minimization use the KL-divergence between $\mathcal{A}$ and $\mathcal{B}$, i.e., $KL(\mathcal{A}\parallel\mathcal{B})$.

So we consider two alternative loss functions of NTF as optimization problems. That loss function based on LS is (6)

$$\min_{\mathrm{G},X,Y,Z} LS(\mathrm{A},[\mathrm{G};X,Y,Z]) \quad \text{or} \quad \min_{X,Y,Z} LS(\mathrm{A},[X,Y,Z]) \qquad (6)$$
$$s.t. \quad \mathrm{G},X,Y,Z \geq 0 \qquad s.t. \quad X,Y,Z \geq 0$$

And based on KL is (7)

$$\min_{\mathrm{G},X,Y,Z} KL(\mathrm{A} \,\|\, [\mathrm{G};X,Y,Z]) \quad \text{or} \quad \min_{X,Y,Z} KL(\mathrm{A} \,\|\, [X,Y,Z]) \qquad (7)$$
$$s.t. \quad \mathrm{G},X,Y,Z \geq 0 \qquad s.t. \quad X,Y,Z \geq 0$$

2) Update rules

We solve the problems by multiplicative update based on gradient descent. E.g., for Tucker decomposition's LS loss function problem as (8)

$$\min_{\mathrm{G},X,Y,Z} LS(\mathrm{A},[\mathrm{G};X,Y,Z]) \qquad (8)$$
$$s.t. \quad \mathrm{G},X,Y,Z \geq 0$$

Above A is known, where will not change with iteration, and value of G,X,Y,Z are associated with the result of each iteration.

First, we write the basic formula from gradient descent as (9-13)

$$\mathrm{A} \approx \mathrm{B} = [\mathrm{G};X,Y,Z] \qquad (9)$$

$$\mathrm{G}_{lmn} \leftarrow \mathrm{G}_{lmn} + \eta_{lmn}\left[\langle \mathrm{A},[\bullet;X,Y,Z]\rangle_{lmn} - \langle \mathrm{B},[\bullet;X,Y,Z]\rangle_{lmn}\right] \qquad (10)$$

$$X_{il} \leftarrow X_{il} + \eta_{il}\left[\langle \mathrm{A},[\mathrm{G};\bullet,Y,Z]\rangle_{il} - \langle \mathrm{B},[\mathrm{G};\bullet,Y,Z]\rangle_{il}\right] \qquad (11)$$

$$Y_{jm} \leftarrow Y_{jk} + \eta_{jm}\left[\langle \mathrm{A},[\mathrm{G};X,\bullet,Z]\rangle_{jm} - \langle \mathrm{B},[\mathrm{G};X,\bullet,Z]\rangle_{jm}\right] \qquad (12)$$

$$Z_{kn} \leftarrow Z_{kn} + \eta_{kn}\left[\langle \mathrm{A},[\mathrm{G};X,Y,\bullet]\rangle_{kn} - \langle \mathrm{B},[\mathrm{G};X,Y,\bullet]\rangle_{kn}\right] \qquad (13)$$

Then, we rescale the η and set as(14-17)

$$\eta_{lmn} = \mathrm{G}_{lmn} * \langle \mathrm{A},[\bullet;X,Y,Z]\rangle_{lmn} / \langle \mathrm{B},[\bullet;X,Y,Z]\rangle_{lmn} \qquad (14)$$

$$\eta_{il} = X_{il} * \langle \mathrm{A},[\mathrm{G};\bullet,Y,Z]\rangle_{il} / \langle \mathrm{B},[\mathrm{G};\bullet,Y,Z]\rangle_{il} \qquad (15)$$

$$\eta_{jm} = Y_{jm} * \langle \mathrm{A},[\mathrm{G};X,\bullet,Z]\rangle_{jm} / \langle \mathrm{B},[\mathrm{G};X,\bullet,Z]\rangle_{jm} \qquad (16)$$

$$\eta_{kn} = Z_{kn} * \langle \mathrm{A},[\mathrm{G};X,Y,\bullet]\rangle_{kn} / \langle \mathrm{B},[\mathrm{G};X,Y,\bullet]\rangle_{kn} \qquad (17)$$

So, we get the multiplicative update rules of Tucker decomposition's LS loss function problem on (18-21)

$$\mathrm{G} \leftarrow \langle \mathrm{A},[\bullet;X,Y,Z]\rangle ./ \langle \mathrm{B},[\bullet;X,Y,Z]\rangle .*\mathrm{G} \qquad (18)$$

$$X \leftarrow \langle \mathrm{A},[\mathrm{G};\bullet,Y,Z]\rangle ./ \langle \mathrm{B},[\mathrm{G};\bullet,Y,Z]\rangle .*X \qquad (19)$$

$$Y \leftarrow \langle \mathrm{A},[\mathrm{G};X,\bullet,Z]\rangle ./ \langle \mathrm{B},[\mathrm{G};X,\bullet,Z]\rangle .*Y \qquad (20)$$

$$Z \leftarrow \langle \mathrm{A},[\mathrm{G};X,Y,\bullet]\rangle ./ \langle \mathrm{B},[\mathrm{G};X,Y,\bullet]\rangle .*Z \qquad (21)$$

The similar as Tucker decomposition's KL loss function problem on(22-26)

$$\mathrm{P} = \mathrm{A} ./ \mathrm{B} \qquad (22)$$

$$\mathrm{G} \leftarrow \langle \mathrm{P},[\bullet;X,Y,Z]\rangle ./ \langle \mathrm{H},[\bullet;X,Y,Z]\rangle .*\mathrm{G} \qquad (23)$$

$$X \leftarrow \langle \mathrm{P},[\mathrm{G};\bullet,Y,Z]\rangle ./ \langle \mathrm{H},[\mathrm{G};\bullet,Y,Z]\rangle .*X \qquad (24)$$

$$Y \leftarrow \langle \mathrm{P},[\mathrm{G};X,\bullet,Z]\rangle ./ \langle \mathrm{H},[\mathrm{G};X,\bullet,Z]\rangle .*Y \qquad (25)$$

$$Z \leftarrow \langle \mathrm{P},[\mathrm{G};X,Y,\bullet]\rangle ./ \langle \mathrm{H},[\mathrm{G};X,Y,\bullet]\rangle .*Z \qquad (26)$$

Optimization based on CP decomposition will get similar update rules, just to make $\mathrm{G} = \mathrm{I}$ and not to updates G.

C. Implementation Details and Complexity Analysis

1) Implementation details

The update equations on the above section involve tensor A 's I,J,K dimensions and matrices X,Y,Z 's L,M,N (N) dimensions. The most commonly solution need six(four) cycles complexity at least in Tucker(CP) decomposition. It will take long time on updating parameters, especially in Tucker decomposition. However, we notice the given tensor A is a very sparse tensor, because many values of A are zeros in NTF model. For this reason, on the one hand, we can store only non-zero elements of A . On the other hand, we can greatly reduce Tucker decomposition's computation complexity by making use of alternative sigma symbols and sorted sparse data, e.g., these formula can been write as (27-28).

$$\langle \mathrm{A},[\bullet;X,Y,Z]\rangle_{lmn} = \sum_i X_{il} \sum_j Y_{jm} \sum_k \mathrm{A}_{ijk} Z_{kn} \qquad (27)$$

Which used in updating G of Tucker decomposition. If A 's known elements are access in sequential way and ordered by $\langle i,j,k\rangle$, the above formulas can be calculated at a lower complexity.

Algorithm 1 using pseudo-code illustrates how to calculate $\langle \mathrm{A},[\bullet;X,Y,Z]\rangle$ on these idea as in Figure 1.

Algorithm 1 Calculate $\langle \mathrm{A},[\bullet;X,Y,Z]\rangle$

```
1   for $n \leftarrow 1$ to $N$
2       $D(i,j) \leftarrow 0$
3       for $\mathrm{A}_{ijk}$ to length[ $\mathrm{A}_{ijk}$ ]
4           $D(i,j) \xleftarrow{+} \mathrm{A}_{ijk} * Z_{kn}$
5       for $m \leftarrow 1$ to $M$
6           $d_i \leftarrow 0$
7           for $D(i,j)$ to length[ $D(i,j)$ ]
8               $d_i \xleftarrow{+} D(i,j) * Y_{jm}$
9           for $l \leftarrow 1$ to $L$
10              $\langle \mathrm{A},[\bullet;X,Y,Z]\rangle_{lmn} \leftarrow 0$
11              for $d_i$ to $I$
12                  $\langle \mathrm{A},[\bullet;X,Y,Z]\rangle_{lmn} \xleftarrow{+} d_i * X_{il}$
```

Figure 1. Algorithm of calculate $\langle \mathrm{A},[\bullet;X,Y,Z]\rangle$ in updating G

By sorting A 's known elements in the format of $\langle i,j,k\rangle$, $\langle j,i,k\rangle$ and $\langle k,i,j\rangle$, computing X,Y and Z can be conducted in a similar fashion.

2) Complexity analysis

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-26

In the following discussion, it is assumed that there are $num(i,j,k)$ distinct (i,j,k) pairs in the dataset, $num(i,j)$, $num(i,k)$ and $num(j,k)$ distinct (i,j) pairs, (i,k) pairs and (j,k) pairs in the dataset. Further on, we define $v1=\max\{I,J,K\}$, $v2=\max\{num(i,j),num(i,k),num(j,k)\}$, $v3=num(i,j,k)$, and $m3=LMN$, $m2=\max\{MN,LN,LM\}$, $m1=\max\{L,M,N\}$. Obviously, $v1<v2<v3$, $m1<m2<m3$.

Importantly, We only need to store non-zero elements of A in some sequential way. Because of core tensor is a diagonal tensor, CP decomposition's $\langle \mathrm{A},[\bullet,Y,Z]\rangle$ can be solved in $O(v3*N)$, and $\langle \mathrm{B},[\bullet,Y,Z]\rangle$ can be solved in $O(v1*N^2)$ if we write $\langle \mathrm{B},[\bullet,Y,Z]\rangle_{in}$ as (28)

$$\langle \mathrm{B},[\bullet,Y,Z]\rangle_{in}=\sum_r\left\{X_{ir}*\sum_j Y_{jn}Y_{jr}*\sum_k Z_{kn}Z_{kr}\right\} \tag{28}$$

For Tucker decomposition, it can be easily seen that in algorithm 1, we can simplify the $\langle \mathrm{A},[\bullet;X,Y,Z]\rangle$'s complexity of $O(v1*m3+v2*m2+v3*m1)$ by separating the unrelated sigma symbols. And $\langle \mathrm{B},[\bullet;X,Y,Z]\rangle$ can be solved in $O(v1*m2+m3^2)$ if we open up some space to store temporary results and write $\langle \mathrm{B},[\bullet;X,Y,Z]\rangle_{lmn}$ as (29)

$$\langle \mathrm{B},[\bullet;X,Y,Z]\rangle_{lmn}=\sum_{pqr}\left\{\mathrm{G}_{pqr}*\sum_i X_{ip}X_{il}*\sum_j Y_{jq}Y_{jm}*\sum_k Z_{kr}Z_{kn}\right\} \tag{29}$$

Sorting Sequence of length $v3$ can be done in $O(v3)$ complexity, so if we consider the number of factors L,M,N as constants that always much less than I,J,K, the complexity of these nonnegative tensor factorization are $O(v3)$.

IV. Experiment

A. *Preliminary*

We evaluate our models on personalized recommendation. We execute the experiments on the CiteSeer dataset[a]. The title, authors, references of each article is extracted and the title are split into many keywords. Next, we clear the data and remove high document frequency words from keywords. Then, 101 highest frequency words are picked, which is used to scale the dataset. Finally, we get 68403 authors, 101 keywords, 62397 papers , and 638780 data records, with 0.00148‰ sparsity. For a certain person(author), given a keyword, recommend relevant references.

In our train and test data, we use binary value to represent the occur times, where (i,j,k)'s value is 1 if author i, keyword j and paper k's combination exists in dataset and 0 otherwise. And doing low-rank nonnegative tensor factorization on dataset, so as to predict unknown values. Rank r for CP decomposition means $N=r$, and rank r for Tucker decomposition means $L=M=N=r$.

a. http://citeseerx.ist.psu.edu

We measured average normalized discounted cumulative gain(nDCG) as the indices to the results, which is often used to measure effectiveness of web search engine algorithms or related applications' ranking quality. Specifically, we use $nDCG@50$ to measure the results, where discounted cumulative gain(DCG) is defined as in (30)

$$DCG@50=\sum_{i=1}^{50}\frac{2^{rel(i)}-1}{\log_2(1+i)} \tag{30}$$

$rel(i)$ is relevance score in our study, denoted as the times of i's occurs. In our study, we use binary value where $rel(i)=1$ if item i exists in the test data and 0 otherwise, where we only consider top 50 (top 0.08%) personalized recommendation called $DCG@50$. For the reason that different queries has different records in test data (query in our study can be seen as a pair of author and keyword), DCG should be normalized across queries, we use $nDCG@50$ to measure the final quality. nDCG is comes from DCG, as in (31)

$$nDCG@50=\frac{DCG@50}{IDCG@50} \tag{31}$$

Where $IDCG@50$ means the maximum possible $DCG@50$ in test data.

So, the average $nDCG@50$ is the $nDCG@50$ scores averages over all different queries in test data.

B. *Experiment Result*

For the performance and rationality of our experiment, we only recommend papers for these pair(author, keyword) not occur in train data. Based on the principle above, data are randomly split into training data and testing data with an 4:1 ratio.

We compare the performance of rank-10 nonnegative tensor factorization between different models. Figure 2 shows the performance of these models.

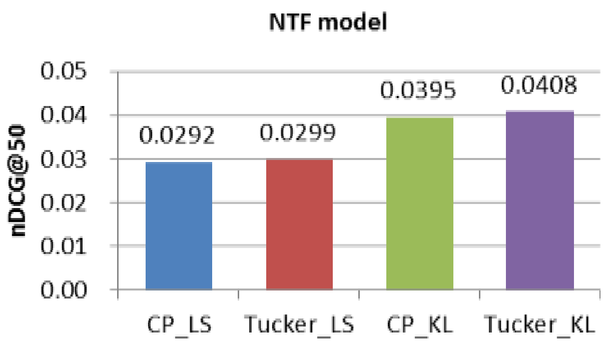

Figure 2. Performance of NTF on Citeseer dataset

Results show that KL-divergence is more appropriate than using LS distance in this paper recommendation. Tucker decomposition's performance get a little better results than CP decomposition in our personalized recommendation experiments, but not improve much compared to CP decomposition. If we consider the

Proc. of the Third Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-077-4 doi: 10.15224/ 978-1-63248-077-4-26

performance and time complexity, the nonnegative CP factorization model based on is a compromise choice.

In addition, the higher rank decomposition, the better results we get. We compare performance on different ranks CP decompositions, shows in Figure 3. We can analyze almost all of the available information on rank-40. The performance on rank-50 CP_KL decomposition increases by 15.3% than on rank-20, but the using rank number is 2.5 times of rank-20. Rank-20 decomposition is better than rank-50 on efficiency.

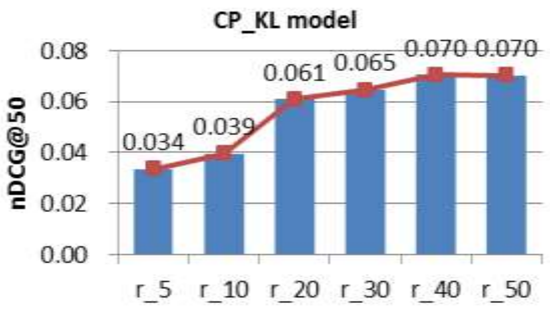

Figure 3. Performance of defferent rank CP_KL decompositions

V. Conclusion and Future Works

A. Conclusion

In this paper, we apply nonnegative tensor factorization to personalized recommendation, and compare factorization on different models and different optimization functions. Model using KL-divergence loss function get better results in recommendation, especially in nonnegative Tucker decomposition. From nonnegative CP_KL decompositions between different ranks, it can be seen some lower rank decomposition can obtain the considerable performance with higher rank decomposition. So in recommendation based on tensor factorization, we should select a proper rank that can get most of useful information, instead of the higher the better efficiency.

B. Future works

In our NTF, updating parameters still cost lots of time on large-scale tensors, although we optimize the algorithm about sparse tensor. Executing tensor factorization on GPU with parallel algorithm[20] is one of our task in the future.

The result matrices of tensor factorization, can be applied to other areas, such cluster and relation learning. Explore these matrices' application is our another target in the future. In addition, we will study high dimension factorization on small tensor, to open a new idea on multi-mode data.

References

[1] Deerwester S. Indexing by latent semantic anlysis[J]. Journal of Am.soc.of Info, 1990, 41(6):391--407.

[2] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009 (8): 30-37.

[3] Koren Y. The bellkor solution to the netflix grand prize[J]. Netflix prize documentation, 2009, 81.

[4] Lee D D, Seung H S. Algorithms for non-negative matrix factorization[C]//Advances in neural information processing systems. 2001: 556-562.

[5] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors, SIAM Journal on Scientific Computing 30(1):205-231, December 2007.

[6] Chi Y, Zhu S, Hino K, et al. iOLAP: a framework for analyzing the internet, social networks, and other networked data[J]. Multimedia, IEEE Transactions on, 2009, 11(3): 372-382.

[7] Donoho D L. Compressed sensing[J]. Information Theory, IEEE Transactions on, 2006, 52(4): 1289-1306.

[8] Candès E J, Tao T. The power of convex relaxation: Near-optimal matrix completion[J]. Information Theory, IEEE Transactions on, 2010, 56(5): 2053-2080.

[9] Kim Y D, Choi S. Nonnegative tucker decomposition[C]//Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE, 2007: 1-8.

[10] Mørup M, Hansen L K, Arnfred S M. Algorithms for sparse nonnegative Tucker decompositions[J]. Neural computation, 2008, 20(8): 2112-2131.

[11] Hazan T, Polak S, Shashua A. Sparse image coding using a 3D non-negative tensor factorization[C]//Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. IEEE, 2005, 1: 50-57.

[12] Bader B W, Berry M W, Browne M. Discussion tracking in Enron email using PARAFAC[M]//Survey of Text Mining II. Springer London, 2008: 147-163.

[13] Phan A H, Cichocki A. Tensor decompositions for feature extraction and classification of high dimensional datasets[J]. Nonlinear theory and its applications, IEICE, 2010, 1(1): 37-68.

[14] Nickel M, Tresp V, Kriegel H P. A three-way model for collective learning on multi-relational data[C]//Proceedings of the 28th international conference on machine learning (ICML-11). 2011: 809-816.

[15] Kolda T G, Bader B W. Tensor decompositions and applications[J]. SIAM review, 2009, 51(3): 455-500.

[16] Liu J, Musialski P, Wonka P, et al. Tensor completion for estimating missing values in visual data[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, 35(1): 208-220.

[17] Carroll J D, Chang J J. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition[J]. Psychometrika, 1970, 35(3): 283-319.

[18] Harshman R A. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis[J]. 1970.

[19] Tucker L R. Some mathematical notes on three-mode factor analysis[J]. Psychometrika, 1966, 31(3): 279-311.

[20] Kirk D. NVIDIA CUDA software and GPU parallel computing architecture[C]//ISMM. 2007, 7: 103-104.

About Author (s):

Linyao Tang received the bachelor degree in communication engineering from Beijing University of Posts and Telecommunications, Beijing, in 2014.

She is currently a master majored in information and communication engineering in BUPT.

Ruifang Liu received the Ph.D. degree in electronic circuit and system from Beijing Univ. of Posts and Telecomm. in 2006. She is an associate professor at BUPT. Her research interests include information retrieval, data mining and machine learning.