Recommending in Social Tagging Systems based on Kernelized Multiway Analysis*

Alexandros Nanopoulos and Artus Krohn-Grimberghe

Institute of Computer Science, Information Systems and Machine Learning Lab
University of Hildesheim, Germany
{nanopoulos,artus}@ismll.de

Summary. Along with the new opportunities introduced by Web 2.0 and collaborative tagging systems, several challenges have to be addressed too, notably the problem of information overload. Recommender systems are among the most successful approaches for increasing the level of relevant content over the "noise". Traditional recommender systems fail to address the requirements presented in collaborative tagging systems. This paper considers the problem of item recommendation in collaborative tagging systems. It proposes to model data from collaborative tagging systems with 3-mode tensors, in order to capture the 3-way correlations between users, tags, and items. By applying multi-way analysis, latent semantic correlations are revealed, which help to improve the quality of recommendations. Nevertheless, high-order tensors tend be sparse, a fact that hinders the application of multi-way analysis. To address this problem, we propose the application of kernel-based methods, which act as smoothing functions against sparsity. Experimental comparison, using data from a real collaborative tagging system (Bibsonomy), indicates the superiority of the proposed method against the non kernel-based method and also against other baseline methods.

Key words: Social tagging, Recommender systems, Multi-way analysis, Kernel functions.

1 Introduction

Social tagging (a.k.a. collaborative tagging) is a process by which users assign labels in the form of keywords to a set of resources with a purpose to share, discover and recover them. Discovery enables users to find new content of their interest, that is shared by other users. Nowadays, collaborative tagging

* The authors gratefully acknowledge the partial co-funding of their work through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006. For your inquiries please contact info@mymediaproject.org.

services proliferate on the Web. Some notable examples are Flickr, Delicious, or "My Web 2.0". Collaborative tagging is also very popular for multimedia data, e.g., Last.fm and YouTube.

Along with the new opportunities introduced by Web 2.0 and collaborative tagging systems, several challenges have to be addressed too, notably the problem of information overload. One of the most successful approaches for increasing the level of relevant content over the "noise" relies on recommender systems. Tags can be considered as indicators of interests and preferences. For this reason, tags are a valuable source for recommendation.

Traditional recommender systems fail to address the requirements presented in collaborative tagging systems, because they usually operate over 2-way data arrays, ignoring the 3-way (users, items, tags) aspect of information that is present in collaborative tagging systems. To address the new requirements, recent research,e.g., [4], has started to examine novel approaches for developing recommender systems in the context of collaborative tagging systems. Most of these approaches concern the recommendation of tags, in order to create a "collabulary" (collaborated vocabulary). However, the central purpose of Web 2.0 and collaborative tagging systems is to facilitate users in discovering items of interests (e.g., documents, products, songs, video, etc.). Recently, Tso et al. [6] examined the problem of item recommendation in collaborative tagging systems. However, in this work the 3-way-correlation between users, items, and tags is treated as a repeated 2-way-problem.

An effective algorithm for recommending items in collaborative tagging systems models all 3-way correlations between users, items, and tags. This is attained by modelling the data as 3-dimensional matrixes, which are called *3-order tensors*. This approach allows to reveal latent semantics, by performing multi-way analysis based on Tucker decomposition [1]. Nevertheless, tensors that model social tagging data tend to be highly sparse, because users provide limited numbers of tags. Sparsity can significantly hinder the application of multi-way analysis. To address sparsity, in this paper we propose the use of kernel functions that smooth the data and allow for effective application of multi-way analysis. The superiority of the kernel-based method is supported with experimental results on data from a real-world social tagging system (Bibsonomy).

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 reviews the multi-way analysis methods that are employed in the proposed approach. The proposed method for providing recommendations is described in Section 4, whereas Section 5 elaborates on the use of kernels as smoothing functions. Experimental results are given in Section 6. Finally, Section 7 concludes this paper.

2 Related work

The problem of recommending tags has attracted significant attention the previous years. Several such algorithms detect conceptual structures in folk-

sonomies similarly to the hyperlink structures detected by search engines. The FolkRank algorithm [4] exploits folksonomies by applying the Personalized PageRank algorithm (a modification of global PageRank) to identify important tags to suggest.

Regarding methods that model data from folksonomies with tensors, Xu et al. [7] proposed a method that recommends tags by using ternary analysis. Symeonidis et al. [5] used HOSVD for recommending tags. The two aforementioned methods, and all methods summarized in the previous paragraph, focus on how to manage folksonomies for the problem of tag recommendation, whereas the method proposed in this paper is focused on the problem of item recommendation. The two problems are different, because tag recommendation aims at creating a converged vocabulary for tags that will be commonly used and, hopefully, alleviate the main problems in folksomies, like polysemy and synonymy. Conversely, item recommendation is about helping users to find relevant items (e.g., documents, products, songs, video, etc.). Therefore, both problems are interesting and solutions to them can act complementary.

As mentioned, the problem of tag-aware item recommendations has recently started to attract attention. A generic, state-of-the-art item recommendation algorithm is the Tag-aware Fusion, proposed by Tso et al. [6]. They propose a generic method that allows tags to be incorporated to traditional, 2-way recommender algorithms, by reducing the 3-way correlations to three 2-way correlations and then applying a fusion method to re-associate these correlations. As will be shown by experimental results, this decomposition breaks the original 3-way structure of the folksonomy and reduces the effectiveness of recommendation.

3 Tensors and Tucker Decomposition

This section provides a concise introduction to the topic of tensors and their decomposition. A *tensor* is a multi-dimensional matrix. A N-order tensor $\mathcal{A}$ is denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \cdots I_N}$, with elements $a_{i_1,\ldots,i_N}$. In this paper, for the purposes of the proposed approach, only 3-order tensors are used. In the following, tensors are denoted by calligraphic uppercase letters (e.g., $\mathcal{A}$, $\mathcal{B}$), matrices by uppercase letters (e.g., A, B), and scalars by lowercase letters (e.g., a, b).

Tucker decomposition for tensor [3] generalizes SVD to multi-dimensional matrices. To compute Tucker decomposition on a 3-order tensor $\mathcal{A}$, we need the definition of the following three *matrix unfoldings*:

$$A_1 \in R^{I_1 \times I_2 I_3}, \qquad A_2 \in R^{I_2 \times I_1 I_3}, \qquad A_3 \in R^{I_1 I_2 \times I_3}$$

Each A_n, $1 \leq n \leq 3$, is called the n-mode matrix unfolding of $\mathcal{A}$ and is computed by arranging the corresponding fibers of $\mathcal{A}$ as columns of A_n. Next, the n-mode product of an N-order tensor $\mathcal{A} \in R^{I_1 \times \cdots \times I_N}$ by a matrix $U \in R^{J_n \times I_n}$ is defined, which is denoted as $\mathcal{A} \times_n U$. The result of the n-mode product is an $(I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N)$-tensor, the entries of which are defined as follows:

$$(\mathcal{A} \times_n U)_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{j_n i_n} \quad (1)$$

Since the focus is on 3-order tensors, $n \in \{1, 2, 3\}$, only 1-mode, 2-mode, and 3-mode products are being used.

By extending SVD, the Tucker decomposition of 3-order tensor $\mathcal{A}$ can be written as follows [3]:

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} \quad (2)$$

where $U^{(1)}$, $U^{(2)}$, $U^{(3)}$ contain the orthonormal vectors (called the 1-mode, 2-mode and 3-mode singular vectors, respectively) spanning the column space of the A_1, A_2, A_3 matrix unfoldings. S is the core tensor and has the property of all orthogonality. Figure 1 illustrates the Tucker decomposition.

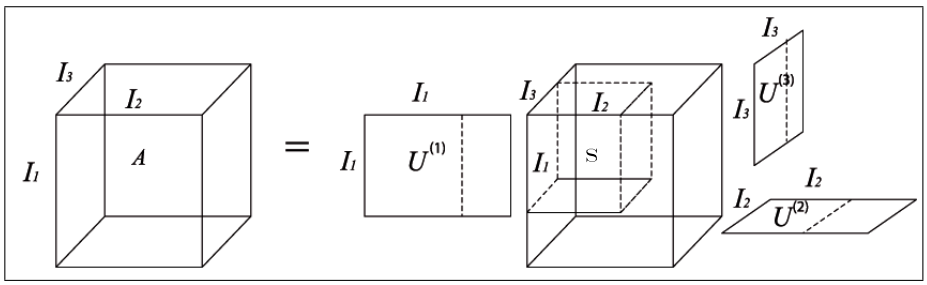

Fig. 1. Visualization of Tucker decomposition.

4 Recommendation based on Tucker decomposition

This section elaborates on how Tucker decomposition is applied on tensors and on how the recommendation of items is performed according to the detected latent associations. This procedure is summarized with the following sequence of steps.

1. The initial construction of tensor $\mathcal{A}$. From the usage data triplets (user, tag, item), an initial 3-order tensor $\mathcal{A} \in R^{u \times t \times i}$ is constructed, where u, t, i are the numbers of users, tags and items, respectively. Each tensor element measures the preference of a (user u, tag t) pair on an item i.

2. Matrix unfoldings of tensor $\mathcal{A}$ As described, a tensor $\mathcal{A}$ can be matricized i.e., generate matrix representations in which all the column (row) vectors are stacked one after the other. In the proposed approach, the initial tensor $\mathcal{A}$ is matricized in all three modes. Thus, after the unfolding of tensor $\mathcal{A}$ for all three modes, 3 new matrices A_1, A_2, A_3, are created as follows:

$$A_1 \in R^{I_u \times I_t I_i}, \qquad A_2 \in R^{I_t \times I_u I_i}, \qquad A_3 \in R^{I_u I_t \times I_i}$$

3. Application of SVD on each mode SVD is applied to the three matrix unfoldings A_n, $1 \leq n \leq 3$:

$$A_n = U^{(n)} \cdot \Sigma^{(n)} \cdot {V^{(n)}}^T \tag{3}$$

4. Computing the low-rank approximations In matrix dimensionality reduction, low-rank approximations are used to filter out the small singular values that introduce "noise". Thus, SVD is truncated to the first c higher singular values and the corresponding singular vectors. The resulting matrix is denoted as rank-c approximation and SVD is optimal in the sense that it computes the rank-c approximation with the minimum *Frobenious norm*.

In the case of tensor dimensionality reduction, a rank-c_1, c_2, c_3 approximation tensor has to be computed, where c_i is the number of dimensions maintained for i-mode. To compute the rank-c_1, c_2, c_3 approximation, c_i singular values and the corresponding left singular vectors from $U^{(i)}$ have to be retained, when applying SVD on the unfolded matrix A_i of i-mode. The selection of c_1, c_2, c_3 determines the final dimensionality of the core tensor $\mathcal{S}$. Since each of the three diagonal singular matrices $S^{(1)}$, $S^{(2)}$, and $S^{(3)}$ are calculated by applying SVD on matrices A_1, A_2 and A_3 respectively, a different c_i value is used for each matrix $U^{(i)}$ $(1 \leq i \leq 3)$. This results to $(U^{(i)}_{c_i})$ matrixes, which denote the c_i-dimensionally reduced $U^{(i)}$ matrix $(1 \leq i \leq 3)$.

5. The core tensor $\mathcal{S}$ construction The core tensor $\mathcal{S}$ governs the interactions among users, items, and tags. Since the dimensions of $U^{(1)}$, $U^{(2)}$ have been selected, and $U^{(3)}$ matrices, the proposed method proceeds to the construction of the core tensor $\mathcal{S}$, as follows:

$$\mathcal{S} = \mathcal{A} \times_1 {U_{c_1}}^{(1)^T} \times_2 {U_{c_2}}^{(2)^T} \times_3 {U_{c_3}}^{(3)^T} \tag{4}$$

where $\mathcal{A}$ is the initial tensor and ${U_{c_n}}^{(n)^T}$ is the transpose of the c_n-dimensionally reduced $U^{(n)}$ matrix, $1 \leq n \leq 3$.

6. The tensor $\hat{\mathcal{A}}$ construction Finally, tensor $\hat{\mathcal{A}}$ is reconstructed by the product of the core tensor $\mathcal{S}$ and the mode products of the three matrices $U^{(1)}$, $U^{(2)}$ and $U^{(3)}$ as follows:

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 {U_{c_1}}^{(1)} \times_2 {U_{c_2}}^{(2)} \times_3 {U_{c_3}}^{(3)}, \tag{5}$$

where $\mathcal{S}$ is the $c_1 \times c_2 \times c_3$ reduced core tensor and ${U_{c_n}}^{(n)}$ is the c_n-dimensionally reduced $U^{(n)}$ matrix, $1 \leq n \leq 3$.

7. The generation of item recommendations The reconstructed tensor $\hat{\mathcal{A}}$ measures the associations among the users, tags and items, so that the elements of $\hat{\mathcal{A}}$ represent quadruplets of the form $\{u, t, i, p\}$, where p is the likeliness that user u will tag item i with tag t. Therefore, items can be recommended to u according to their weights associated with $\{u, t\}$ pair. Specifically, if it is required to recommend to a user u N items that are related to a tag t, then the N items with the highest p values among all $\{u, t, i, p\}$ quadruplets are recommended.

5 Smoothing with Kernel functions

In Section 4 we described that Tucker decomposition applies SVD on the three matrix unfoldings A_n that results to the three matrixes $U^{(n)}$, $1 \leq n \leq 3$, which contain the orthonormal vectors (left singular vectors) for each mode. As already mentioned, sparsity is a severe problem in 3-dimensional data and it can affect the outcome of SVD. To address this problem, instead of SVD we can apply Kernel-SVD [2] in the three unfolded matrices. Kernel-SVD is the application of SVD in the Kernel-defined feature space.

For each unfolding A_i ($1 \leq i \leq 3$) we have to non-linearly map its contents to a higher dimensional space using a mapping function ϕ. Therefore, from each A_i matrix we can derive an F_i matrix, where each element a_{xy} of A_i is mapped to the corresponding element f_{xy} of F_i, i.e., $f_{xy} = \phi(a_{xy})$. Next, we can apply SVD and decompose each F_i as follows:

$$F_i = U^{(i)} S^{(i)} (V^{(i)})^T \tag{6}$$

The resulting $U^{(i)}$ matrixes are then used to construct the core tensor, that is, the procedure continues as described in Section 4. Nevertheless, to avoid the explicit computation of F_i, all computations must be done in the form of inner products. In particular, as we are interested to compute only the matrixes with the left-singular vectors, for each mode i we can define a matrix B_i as follows:

$$B_i = F_i F_i^T \tag{7}$$

As B_i is computed using inner products from F_i, we can substitute the computation of inner products with the results of a kernel function. This technique is called the "kernel trick" [2] and avoids the explicit (and expensive) computation of F_i. As each $U^{(i)}$ and $V^{(i)}$ are orthogonal and each $S^{(i)}$ is diagonal, it easily follows from Equations 6 and 7 that:

$$B_i = (U^{(i)} S^{(i)} (V^{(1)})^T)(U^{(i)} S^{(i)} (V^{(i)})^T)^T = U^{(i)} (S^{(i)})^2 (U^{(i)})^T \tag{8}$$

Therefore, each required $U^{(i)}$ matrix can be computed by diagonalizing each B_i matrix (which is square) and taking its eigen-vectors. Regarding the kernel function, in our experiments we use two widely used functions: (a) the Gaussian kernel $K(x, y) = exp(-\frac{||x-y||^2}{c})$, where c parameter is called the *width* of the kernel, and (b) the Polynomial kernel $K(x, y) = (x \cdot y + 1)^d$ (d parameter is called the *degree* of the Polynomial).

6 Experimental results

In this section we experimentally compared four methods: (i) "Kernel", which combines kernel functions with tucker decomposition of tensors, (ii) "Tensor", which uses only Tucker decomposition without kernel functions, (iii) "Epsilon", which simply adds a small ϵ value to every zero element in the tensor

(we used $\epsilon = 0.001$), and (iv) "Popular", which recommends to a user the most frequently tagged items that has not been already tagged by him in the training data. Comparison between Kernel and Tensor helps to understand the usefulness of kernel functions, whereas Epsilon and Popular act as simple baseline methods. The three first methods were implemented using the Tensor Toolbox (csmr.ca.sandia.gov/ tgkolda/TensorToolbox) by setting $c_n = 70\%$, $1 \leq n \leq 3$.

We used real data from the Bibsonomy (www.bibsonomy.org), a social bookmarking system for web resources. The original data was a snapshot taken on April 30, 2007, with 1,037 users, 28,648 items, and 86,563 tags. To remove noise, after applying the 5-core preprocessing (each user, item, and tag must occur in at least 5 posts), 9,763 triples were maintained with 116 users, 361 items, 412 tags. We performed Random Sub-sampling by keeping a fraction of triples for testing (building the tensor) and the rest for training. Each presented results is the average of 30 repetitions (standard deviation is also reported). For each user u - tag t combination in the test data, all methods tried to predict the items tagged by u as t in the test data. We selected recall as the performance measure. Sparsity in the data is characterized by the ratio between the size of the test data to the size of original data, which are given as fractions compared to the size of the original data (e.g., test/training: 0.2/0.8 means that the test and training data was 0.2 and 0.8, respectively, of the original data size, measured in the number of triplets). All measurements are given versus the number of recommended items.

Comparing the results in Figure 2a and b, we observe that, for the Polynomial kernel ($d = 2$), Kernel is more efficient than Tensor when the ratio between test/training data sizes increases, i.e., when sparsity increases in the results of Figure 2b compared to lower sparsity in the results of Figure 2a. Please notice that the differences in Figure 2b are much higher than one standard deviation (depicted with error bars). The same conclusion holds for the Gaussian kernel in Figure 2c. However, the Polynomial kernel is sensitive to the degree d. Figure 2d present results for $d = 4$. The lower performance of Kernel is due to overfitting that incurs due to high d. We observed that the Gaussian kernel is less sensitive to its width c (result not presented here due to lack of space).

7 Conclusions

We proposed the enhancement of recommending items based on Tucker decomposition with the use of Kernel functions that smooth the tensor data and address the sparsity. Our experimental results with real data presented the superiority of the proposed method over current standard approaches.

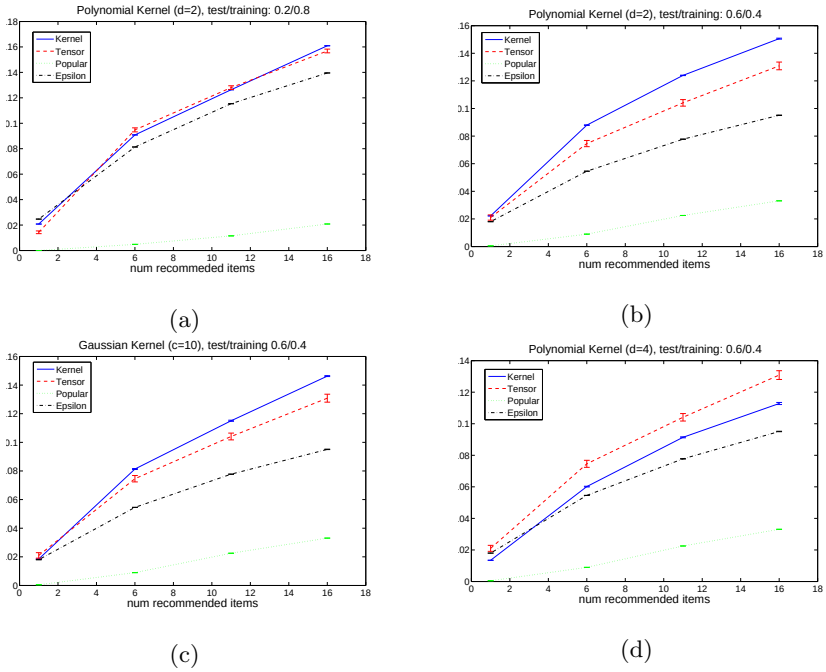

Fig. 2. Recall vs. number of recommended items: (a)-(c) Polynomial kernel, (d) Gaussian kernel.

References

1. E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, (to appear), 2008.
2. N. Cristianini and J. Shawe-Taylor. Kernel methods for pattern analysis. *Cambridge University Press*, 2004.
3. L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
4. A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *The Semantic Web: Research and Applications*, pages 411–426, 2006.
5. P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *2nd ACM Conference in Recommender Systems (RecSys'2008)*, pages 43–50, 2008.
6. K. Tso-Sutter, B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. *Proc. SAC Conf.*, 2008.
7. Y. Xu, L. Zhang, and W. Liu. Cubic analysis of social bookmarking for personalized recommendation. In *Frontiers of WWW Research and Development - APWeb 2006*, pages 733–738, 2006.