Augmented Matrix Factorization with Explicit Labels for Recommender Systems

ABSTRACT

Both industry and academia have shown emerging interests in recommender systems by using latent factor models. In practice, items are associated with different labels (i.e. genre or category) to reflect its intrinsic properties. However, most conventional matrix factorization approaches attempt to infer the unseen user preference by the observed behaviors but ignore explicit label information associated with items. In this paper, we introduce a novel learning approach termed *Augmented Matrix Factorization* (AMF) to combine item label information with original user-item relationship in a seamless way. Specifically, the proposed AMF augments matrix tri-factorization with an explicit label space, by incorporating a novel structured sparsity and probabilistic simplex constraints. Moreover, the proposed AMF learns label probabilities from data, showing how items are associated with their assigned labels. We provide an efficient block coordinate descent algorithm for solving the AMF. The experimental results show that AMF can significantly outperform baseline methods and produce interesting and meaningful label probabilities.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Experimentation

Keywords

Matrix Factorization, Explicit Label, Label Probability.

1. INTRODUCTION

Matrix factorization has been very successful in recommender systems. Since its introduction in 2007 Netflix contest [2], it has soon shown to be outperforming other existing methods that are memory-based. From mobile applications [17], to restaurant recommendations [8], matrix factorization based methods have been applied to a wide range of application.

Classical matrix factorization decomposes the original user-item matrix into sub-matrices and derives latent factors. The limitation of standard matrix factorization is that it is hard to explicitly add information about items or users outside user-item matrix. For example, in product recommendation on E-commerce domain, we know certain products belong to the same category. In movie recommendation, we know certain movies have the same actors, and certain restaurants have the same cuisine.

Various work has tried to extend matrix factorization with item information or user information. Existing work is focusing on adding item content information as an additional step besides matrix factorization [10]. Among various content information, one frequent type is the multi-label. For example, a product can be both toys and gifts, and a restaurant can have both Vietnamese and Thai cuisine. The most of existing CF approaches are typically insufficient to integrate these label information with model learning in a seamless and efficient way. On one hand, it is hard to specify a predefined label space in matrix factorization algorithms, which typically seek a latent label space. On the other hand, the label space may be very large (e.g., consider the space spanned by all actors in the IMDB database), and thus directly modeling label space in the factorization may lead to computational difficulties.

To address these challenges, in this paper we propose the Augmented Matrix Factorization with Explicit Labels (AMF), which simultaneously learns a latent space between user and a label space, and a mapping connecting the label space to the item space. This factored representation allows us to explicitly model the label-item relationship. We design *structured sparsity* on the label-item mapping matrix to encode the prior information about the labels assigned to the item. It largely reduces the degree of freedom in solving the mapping and makes the computation feasible. Moreover, in order to explore how items are associated with the labels, we also impose a probabilistic simplex constraint on the mapping. The aforementioned contributions distinguish us from the traditional tri-factorization (e.g., [4]) and label-based factorization (e.g., [13]).

The key contributions of this paper are the following:

- We proposed novel tri-factorization approach for recommendation that explicitly leverages the available multi-label information of the items.
- We designed a novel structured sparsity on the label-item factor, which learns a useful probability interpretation for each item related to all labels.

Our experimental results on a TV watch data show that the proposed AMF approach outperforms existing approaches, which demonstrates the effectiveness of the proposed method and the use of label information. Moreover, from our learned models, we discover meaningful probability assignments associated with different labels for our items.

[*] Work done while author was at Samsung Research America.

WOODSTOCK '97 El Paso, Texas USA

2. AUGMENTED MATRIX FACTORIZATION WITH EXPLICIT LABELS

2.1 Learning with Explicit Labels

In the traditionally factorization-based approaches, the recommendation score for a user i on an unrated item j is approximated by the inner product of the user profile row vector u^i from the user profile matrix $U \in \mathbb{R}^{m \times r}$, and the item profile column vector $\hat{v}_j$ from the item profile matrix $\hat{V}$, where m is the number of users and r is the dimension of latent factors. Let s_{ij} be the prediction score, then $\bar{s}_{ij} = u^i \hat{v}_j$.

Given the label information, we explicit model the effects of the label profiles and assume the item profiles are linked to user profiles via the provided label space. Specifically, we assume that users' interests are directly related to these labels, and their interests in items are established by aggregating the label profiles that associate to the items. We introduce the label profile matrix $H \in \mathbb{R}^{r \times g}$ in which the column vector h_k denotes a label profile for the label k, where g is the size of the label space (*i.e.* if we label movies by genres, then g corresponds to the number of unique genres). Let v_{ji} be the weight measuring how an item i associates to the label j. The proposed AMF model assume that $\hat{v}_j = \sum_{k \in \mathcal{A}_j} h_k v_{kj}$, where $\mathcal{A}_j$ is the set containing the side information of label assignments for item j, i.e., $k \in \mathcal{A}_j$ means the label k has been assigned to the item j. We can also vectorize v_{kj} into a label-item matrix $V \in \mathbb{R}^{g \times n}$, whose column v_j represents the weights of labels to the program j (v_{kj} is zero if the item does not belong to the label). To this end, the recommendation score is given by the following:

$$\bar{s}_{ui} = u^i \sum\nolimits_{j \in \mathcal{A}_i} h_j v_{ji} = u^i H v_j. \tag{1}$$

Note that in some applications, it is preferred to introduce user bias and item bias in the recommendation model. Our propose formulation and algorithm can be easily extended to incorporate the bias terms. In the following discussion, with loss of generality we present the model without bias to focus our core innovation.

We denote the partial observed user-item matrix as $X \in \mathbb{R}^{m \times n}$. Let Ω to be the set of indices of X such that $(i, j) \in \Omega$ are the entries contain observed values. Moreover, denote $\mathcal{P}_\Omega(\cdot)$ as a projector taking an input matrix from $\mathbb{R}^{m \times n}$ such that if $(i, j) \in \Omega$ then the (i, j) entry of $\mathcal{P}_\Omega(X)$ is X_{ij} and 0 otherwise. To this end, the objective of minimizing the squared error can be represented by the following matrix form:

$$\min\nolimits_{U,H,V} \quad \|\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(UHV)\|_F^2. \tag{2}$$

Figure 1 provides an overall illustration of the proposed factorization framework.

Since that usually each item is only associated with a small set of given labels, the many locations in V must be zeros. In order to guide the factorization to satisfy this sparse structure, we constraint the solution space of V by imposing a constraint $\mathcal{C}$ defined as

$$\mathcal{C} = \{V : \mathcal{P}_{\Lambda^c}(V) = 0\}.$$

Here the set Λ is a set indices that V can be nonzero (correspond to the knowledge in the set $\{\mathcal{A}_i\}$ in equation 1). In another word, if a item i is labeled with two genre information as j and k, we then enforce V_{ji} and V_{ki} entry nonzero and zero otherwise for other entries in the i^{th} column of V. This is not only avoid massive computation and storage expense, but also able to eliminate uncorrelated information to bias user preference prediction.

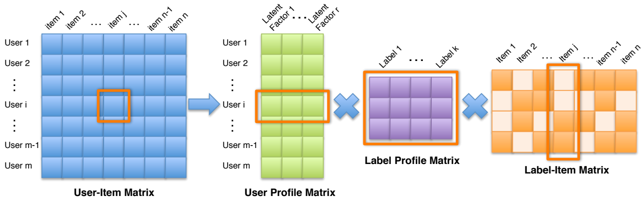

Figure 1: The illustration of the proposed AMF framework.

Furthermore, in order to identify how each label contributes the users' tastes and how much one item relates to the given labels, we impose a probabilistic interpretation on each column of V as well, by another constraint given by:

$$\mathcal{S} = \{V : V_i \in \Delta^+, \forall i\}, \tag{3}$$

where $\Delta^+ = \{x : \sum_i x_i = 1, x_i \geq 0\}$ is the *probabilistic simplex constraint*. This constraint requires that all elements to be nonnegative and also that all nonzero elements in each column sum to 1. Since each column corresponds to an item, and enforcing two constraints $\mathcal{C}$ and $\mathcal{S}$ jointly give a probabilistic interpretation showing how items are associated with their assigned labels. To this end, we formally formulate the AMF model as:

$$\begin{aligned} \min_{S, U \geq 0, H \geq 0, V} & \|S - UHV\|_F^2 + \lambda \left(\|U\|_F^2 + \|H\|_F^2\right) \\ \text{s.t. } & \mathcal{P}_\Omega(S) = \mathcal{P}_\Omega(X), V \in \mathcal{C} \cap \mathcal{S}, \end{aligned} \tag{4}$$

where we use an intermediate matrix S to simplify computation and we also add Frobenius norm on U and V to prevent from overfitting. Note that the nonnegative constraint on both U and H are application specific. However, due to the fact that most feedback types are non-negative, the constraint is usually appreciated.

2.2 Block Coordinate Descent Algorithm

In equation (4), matrix U, H and V are coupled together that leads to a non-convex optimization objective. We propose to use the block coordinate descent (BCD) to iteratively optimize with respect to one block while fixing other blocks. It is not hard to verify that each sub-problem is a convex optimization problem, which guarantees the objective to reach a local optimal solution.

Solving U: The update of U is given by the following problem:

$$U^+ = \text{argmin}_{U \geq 0} \|S - UHV\|_F^2 + \lambda \|U\|_F^2, \tag{5}$$

which can be solved efficiently using the projected gradient [1].

Solving H: The update of H is given by the following problem:

$$H^+ = \text{argmin}_{H \geq 0} \|S - UHV\|_F^2 + \lambda \|H\|_F^2, \tag{6}$$

which can also be solved using similar strategies as U.

Solving V: To solve V, we obtain the following objective:

$$V^+ = \text{argmin}_V \|S - UHV\|_F^2 \quad \text{s.t. } V \in \mathcal{C} \cap \mathcal{S}, \tag{7}$$

which can also be solved using projected gradient, and the key is to solve the proximal operator [1] associated with the problem in (7):

$$\min\nolimits_V \|V - \hat{V}\|_F^2 \quad \text{s.t. } V \in \mathcal{C} \cap \mathcal{S}, \tag{8}$$

where $\hat{V}$ is given by a gradient step [1]. It is not hard to observe that $\mathcal{C}$ and $\mathcal{S}$ are both convex sets, and thus in equation (8) we need to solve an Euclidean projection onto the intersect of two convex sets, which is highly non-trivial. In general the projection into two convex sets can be solved iteratively using proximal splitting methods [5]. However, we show that the problem in equation (8) admits a closed-form solution and thus it can be efficiently computed.

The simplex constraint has coupled for each column of V. For each column we solve the following Euclidean projection problem:

$$\min\nolimits_v \|v - \hat{v}\|_F^2 \;\text{ s.t. } \mathcal{P}_{\Lambda_i^c}(v) = 0, \Sigma_i v_i = 1, v_i \geq 0, \forall i, \tag{9}$$

where Λ_i is the constraint operated on the i^{th} column. And c indicates the complement of a given set. We notice that for $v_i = 0$ for $i \in \Lambda_i^c$. Therefore we only need to solve the variables where $i \notin \Lambda_i^c$. Define $\hat{t} = \hat{v}[i \notin \Lambda_i^c]$, the problem then reduced to a standard probabilistic simplex projection:

$$\min_t \|t - \hat{t}\|_F^2 \quad \text{s.t. } \Sigma_i t_i = 1, t_i \geq 0, \forall i. \tag{10}$$

If we sort t the elements in a descending order, then it admits an analytical solution [16, 7, 11]:

$$t^* = t_i - \tau, \quad \text{where } \tau = \left(\Sigma_{i=1}^{\rho} w_i - \lambda\right)/\rho \tag{11}$$

and $\rho = \max\{j : w_j > (\sum_{i=1}^{j} w_i - \lambda)/j\}$. Thus the projection step in equation (8) can be solved efficiently.

Solving S: The problem of solving S^+ is a is given by:

$$S^+ = \text{argmin}_S \|S - UHV\|_F^2 \quad \text{s.t. } \mathcal{P}_\Omega(S) = \mathcal{P}_\Omega(X). \tag{12}$$

It is straightforward to see that the problem admits a closed-form solution: $S^+ = \mathcal{P}_\Omega(X) + \mathcal{P}_{\Omega^c}(UHV)$, which means we "restore" the values at the observed locations at the reconstruction from the factors UHV.

We notice that the update of S can be rewritten as the following form: $S^+ = \mathcal{P}_\Omega(X - UHV) + UHV$. Therefore, the matrix S will never be stored as a whole piece in the memory, instead as a combination of a sparse matrix in additional with three low-dimensional matrices. This special structure leads to linear complexity with respect to m, n when computing function values and gradients, and thus provides the capability in handling large-scale data.

3. EXPERIMENTS

3.1 TV Watch Data

To demonstrate the effectiveness of the proposed approach, we evaluate it on task of TV show recommendation using our TV watch data. Our TV watch data captures different types of events of a smart TV (e.g., channels/TV shows that the users are watching) has been widely used to model user's preference and deliver personalized program recommendations [3, 9]. In this study we restrict our focus on the users that have watched more than 20 TV show view activities and programs that have been watched by at least 1000 users within a week, resulting in 201,253 users and 2,738 TV shows. In our data we have in total 3,505,074 watch records. The watch behavior of the users and statistic of TV shows are shown in Figure 2.

We process the raw TV watch data and compute the average daily watch time for each TV show and for each user, from which we can construct a user by TV show matrix of the corresponding size. For each TV show in the study, we are provided with associated genre information from the ROVI program guide, which consists of a set of 12 first-level genres and in total 390 sub-genres. The genre information has been shown to be informative in modeling different groups of users [3, 13] and in our experiments, we use the sub-genres to form the label space and construct the label-item assignments to be encoded in Λ.

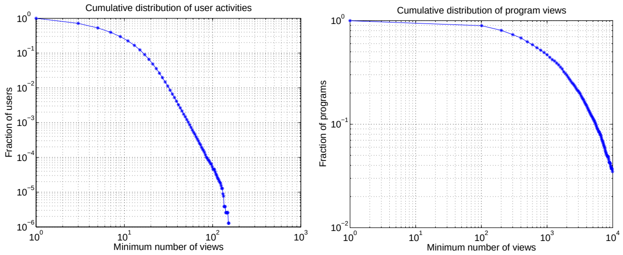

Figure 2: Cumulative distribution of user activities and TV Show popularity.

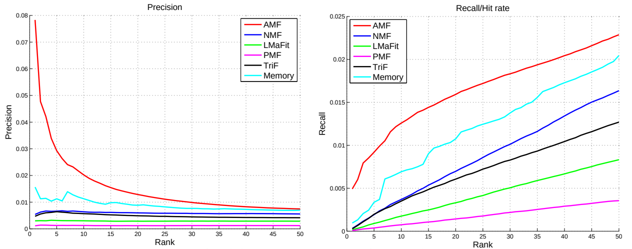

Figure 3: Precision (left) and recall (right) at top N prediction for different N with 10 latent factors.

3.2 Experimental Settings

We evaluate our algorithm in two settings. The first setting is the top N recommendation, in which we recommend items to users and evaluate the algorithms by how many relevant items retrieved in the top N items recommended ranked by the predicted value s_{ij}. In this setting, we use a binary representation: whenever a user i watches a TV show j, we set $s_{ij} = 1$ and 0 otherwise. We also performed top N recommendation directly using watch time, and we find that the model trained by binary values is consistently better than watch time. In the second setting we evaluate the prediction in terms of matrix reconstruction error, and we directly predict the watch time.

We compare the following methods in our experiment, which includes: **LMaFit** [18]: A fast low-rank matrix completion algorithm for completing matrices. **PMF** [14]: The probabilistic matrix factorization, which generalizes the latent factor method to a probabilistic framework. **NMF** [12]: The non-negative matrix factorization, which performs a low rank matrix approximation on the incomplete matrix. **TriF** [4]: Collaborative filtering using orthogonal nonnegative matrix tri-factorization, which generalizes the ONMTF [6] to a CF framework and combines both memory-based and model based model to overcome the sparsity and computational issues. **AMF**: The proposed method in equation (4). **Item-Item** [15]: An item-based memory recommendation model. We use the cosine similarity to compute neighborhood similarity. prediction $\hat{s}_{i,j}$ is computed by retrieving the most items and perform weighted average to compute the prediction.

In the experiments we randomly select 10% training data as a validation set, and the model parameters except for the number of latent factors are tuned on the validation set. We fix the same number of latent factors for different approaches.

3.3 Performance Comparison

To evaluate the competing methods, we use the leave-k-out precision and recall framework. In the leave-k-out scheme, we have k relevant items to be predicted for each user (the left out TV shows). Given the top-N recommendation list, for user i we compute the number of items that hit the relevant items h_u^N, and the hit rate is given by h_u^N/k. The definition of top N precision and recall is given as follows:

$$\text{Precision(N)} = \Sigma_{u=1}^m h_u^N/(mN), \text{Recall(N)} = \Sigma_{u=1}^m h_u^N/(km)$$

where the modified recall function is also the average hit rate over users for top N prediction. We repeat the random leave-k-out process for 10 iterations and report the average precision and recall. In our experiment k is set to 20, and we note that for different k, the performance patterns are very similar to each other.

Figure 3 gives the precision (the upper figure) and recall (the lower figure) curves for the competing methods on our data for 10 latent factors. We see that the proposed AMF method is the best performer. Especially, for top N prediction given $N \leq 50$, both precision and recall of the proposed approach significantly out-

Method	k=1	k=5	k=10	k= 20	k = 30
AMF	**0.5080**	**0.5328**	**0.5651**	**0.6330**	0.7016
Item	0.9402	0.8948	0.8388	0.7605	0.7098
LMaFit	0.9174	0.8864	0.8242	0.7148	**0.6777**
NMF	1.1640	1.1794	1.1912	1.1882	1.1884
PMF	1.1095	1.0991	1.0766	0.8756	0.6941
TriF	1.2380	1.5320	1.7204	2.1069	2.5735

Table 1: Performance in terms of RMSE. Lower value indicates smaller error and thus better performance.

performs other methods. The item-based memory method is the runner-up method which outperforms all other factorization-based methods. This agrees with the findings in many industrial recommender systems in that the memory-based approaches are typically amongst top performers (e.g., [17]). We see similar performance patterns for different latent factors.

We also use root mean squared error (RMSE) to evaluate the performance from the perspective of matrix completion. The experimental results of the competing methods are reported in Table 1. We observe that the proposed method can also significantly outperform other methods.

3.4 Label Probability

In this section, we inspect the item-label matrix V and present some interesting results obtained by AMF on our data. Recall that during the learning process we enforce a probability interpretation on the matrix V, and thus the summation of the weights of the assigned genres for each TV show equals to 1. The interpretation here is how much the TV show relates to the assigned genres. In Table 2 we show some TV shows and their label probability.

In this *Pizza Cuz* show, the hosts travel across the US, visiting the top pizza shops in the country. Thus, the show is assigned to two genres: Food and Travel. However, due to the main theme of food, the show may attract more viewers who are interested in food than those in travel. In our model, this preference is successfully learned: the probability assigned to Food is larger than Travel. We also see the similar probability assignment to the show *Diners, Drive-Ins and Dives*, which also involves travel and dining. Another example is the is the show *Chasing Classic Cars*, which has genres Hobbies & Crafts and Auto Info. We observe that in the learned model, the former genre has been given much higher than the latter. This infers that although the program itself is about auto, it attracts more viewers that have crafting hobbies than general auto lovers.

We also observe that AMF has the capability of identifying noisy genres. The program *Behind the Music*, for example, has been assigned three genres: Documentary, Profile and R & B. The program documents musical artists and groups who are interviewed and profiled, followed by discussion on the road to success and setbacks on it. Assigning the genre R & B might because that there are R & B artists/groups interviewed in this episode such as *Boyz II Men*. In this sense, the music genre R & B may not be a good genre because there are many other artists that belong to music genres other than R & B. The genre Profile is very close to Documentary in terms of their semantic meanings, and however the latter genre is more popular among the TV shows. In Table 2 we see that the genre Documentary dominates other two genres, which is consistent with our analysis. Similar in the program *Law & Order*, we see that Crime Drama dominates the other genre.

TV Show	Genre	Probability
Pizza Cuz	Food	0.6434
	Travel	0.3566
Diners, Drive-Ins and Dives	Food	0.6376
	Travel	0.3624
Chasing Classic Cars	Hobbies & Crafts	0.8633
	Auto Info	0.1367
Behind the Music	Documentary	0.9870
	Profile	0.0069
	R & B	0.0061
Law & Order: Criminal Intent	Crime Drama	0.9885
	Spin-Off	0.0115
Family Guy	Animated Comedy	0.7993
	Sitcom	0.2007

Table 2: Genre probability learnt from the TV watch data by the proposed AMF.

4. CONCLUSION

In this paper, we propose a novel augmented matrix factorization (AMF) method for personalized recommendation. The proposed method addresses problems of the "preference gap" in the recommender system using implicit feedback by leveraging label information. Moreover, the proposed method can discover the label probability which describes how much items are associated to their labels. In order to explicitly model the label space with provided labels, we impose structured sparsity on the label-item mapping matrix and constraint the columns of the matrix to a probabilistic simplex. Our experimental results on a TV watch data show that the proposed AMF approach delivers promising results and outperforms many existing approaches.

5. REFERENCES

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. on Im. Sci.*, 2(1):183–202, 2009.

[2] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Exp.*, 9(2):75–79, 2007.

[3] J. A. Chaney, M. Gartrell, Hofman, J. Guiver, N. Koenigstein, P. Kohli, and U. Paquet. Mining large-scale tv group viewing patterns for group recommendation. Technical report, Microsoft tech. rep., MSR-TR-2013-114, 2013.

[4] G. Chen, F. Wang, and C. Zhang. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. *Info. Proc. & Manag.*, 45(3):368–379, 2009.

[5] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. 2011.

[6] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*, pages 126–135, 2006.

[7] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *ICML*, pages 272–279. ACM, 2008.

[8] H. Feng and X. Qian. Recommendation via user's personality and social contextual. In *CIKM*, pages 1521–1524, 2013.

[9] E. Kim, S. Pyo, E. Park, and M. Kim. An automatic recommendation scheme of tv program contents for (ip) tv personalization. *Broadca., IEEE Trans. on*, 57(3):674–684, 2011.

[10] N. Koenigstein and U. Paquet. Xbox movies recommendations: variational bayes matrix factorization with embedded feature selection. In *RecSys*, pages 129–136, 2013.

[11] A. Kyrillidis, S. Becker, V. Cevher, and C. Koch. Sparse projections onto the simplex. In *ICML*, pages 235–243, 2013.

[12] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[13] M. G. Manzato. Discovering latent factors from movies genres for enhanced recommendation. In *RecSys*, pages 249–252, 2012.

[14] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[16] S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *JMLR*, 7:1567–1599, 2006.

[17] K. Shi and K. Ali. Getjar mobile application recommendations with very sparse datasets. In *KDD*, pages 204–212, 2012.

[18] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Math. Prog. Comp.*, 4(4):333–361, 2012.