A Hierarchical Bayesian Model for Personalized Apparel Recommendation

Abstract

Re.

1 Introduction

Real-world organizations in business domains operate in a multi-item, multi-level environment. Items and their corresponding information collected by these organizations often reflect a hierarchical structure. For examples, products in retail stores are usually stored in hierarchical inventories. News on web pages are created and placed hierarchically in most web sites. These hierarchical structures and the data within them provide large amount of information when building effective recommendation systems. Especially in e-commerce domain, all products are displayed in a site-wide hierarchical catalog and how to build an accurate recommendation engine on top of it becomes one of the keys to majority companies' business success nowadays.

However, how to utilize the rich information behind hierarchical structures to make personalized and accurate product recommendations still remains challenging due to the unique characteristics of hierarchical structures and the modeling trade-offs arising from them. Briefly, most well-established recommendation algorithms cannot naturally take hierarchical structures as additional inputs and flatting the hierarchy usually don't work well. It will not only blow up the entire feature space but introduce nosies when training the recommendation models. On the other hand, completely ignoring the hierarchies will lead to recommendation inaccuracies. The most common way to alleviate this problem is to feed every piece of data from the hierarchy into a complex deep neural network and hope the network itself can figure out a way to use the hierarchical knowledge. However, such approaches usually behave more like black boxes and cannot provide any interpretation on their intermediate results and outcomes.

In this work, we propose and develop a hierarchical Bayesian, a.k.a., *HBayes*, modeling framework that is able to flexibly capture various relations between items in hierarchical structures from different recommendation scenarios. More specifically, xxxx. Moreover, we develop a variational inference algorithm for learning parameters of HBayes. To illustrate the modeling power of the proposed HBayes approach, we introduce HBayes by using a real-world apparel recommendation problem as an example. However, our approach is not only limited on apparel recommendation and at the end, we show its flexibility and effectiveness on a music recommendation problem as well.

Overall this paper makes the following contributions:

- It presents a generalized hierarchical Bayesian learning framework to learn models from rich data with hierarchies.
- It provides a variational inference algorithm that can learn the model parameters with few iterations.
- We evaluate our HBayes and its benefits comprehensively in tasks of apparel recommendation on a real-world data set.
- We test our framework in different recommendation scenarios to show its generalization and applicability.

The remainder of the paper is organized as follows: Section 2 provides a review of existing recommendation algorithms and their extensions in hierarchal learning settings. Section 3 introduces the notations and our generalized HBayes learning framework and its variational inference algorithm. In Section 4, we conduct experiments in a real-world e-commerce data set to show the effectiveness of the our proposed recommendation algorithm in different aspects. In addition, we test our model on a music recommendation data set to illustrate the generalization and extended ability of HBayes. We summarize our work and outline potential future extensions in Section 5.

2 Related Work

With the explosive growth of digital information over the Internet, the recommender system is considered to be one of the most effective approaches to overcome such information overload. Traditional recommendation algorithms such as item-based approaches including collaborative filtering [22, 23] and matrix factorization [21] recommend an item by learning the interaction between users and items, while content-based approaches including [15, 14, 26] compare the auxiliary information of both user and item to recommend similar items. Further, by looking into time variant nature of recommender systems, there are time-aware recommendation approaches [9, 24] that take *time* as another input dimension besides item and user, which aims at explicitly modeling user interest over time slices to combine with various classic recommender algorithms. [12] develops a collaborative filtering type of approach with predictions from static average value combining with a dynamic changing factor. [25] propose a user-tag-specific temporal interest model to track user interest over time by maximizing the time weighted data likelihood. For better personalization, [10] propose an item-based recommender system combining with each user's social network information. Despite multiple branches of research regarding recommender system, none of these approaches take into account of hierarchical information existing in products for better structural understanding, which leaves their recommendation results sub-optimal.

Recently, there are a group of work using Bayesian methods for recommendations. [20] combines the Bayesian analysis and matrix factorization together and by learning users implicit feedbacks (click & purchase) to directly optimize the ranking results in the sense of AUC metrics. [1] and [27] takes user preference consistency into account and develops a variational Bayesian personalized ranking model for better music recommendation. However, none of these aforementioned approaches take the product rich structure into account while learning the Bayesian model, therefore the product recommendation still remains imprecise.

In this work, our method focuses on building a generic hierarchical Bayesian learning scheme that takes account of both the user preferences and product hidden structural properties. It is able to recommend products that align to users' long term tastes. Meanwhile the learned latent variables can be well interpreted and explained, which helps us well analyze and understand users interests and preference.

3 Methodology(Jerry)

4 Experiment

In this section, we conduct several experiments on two data sets: (1) the productional e-commerce data set; (2) the publicly available music data set. We compare our proposed method HBayes against several other *state-of-the-art* recommendation methods which are briefly described as follows:

KNN [13]:

SVD++ [17, 11]: combines the collaborative filtering approach and latent factor approach, so to provide a more accurate neighboring based recommendation result. In this paper, we adopt SVD++ as another baseline.

CoClustering [8]: another collaborative filtering flavored recommendation approach based on weighted co-clustering improvements that simultaneously cluster users and items. We adopt CoClustering as another baseline.

Fatorization Machine (FM) [18]: combines support vector machines (SVM) with factorization models which takes the advantage of SVM meanwhile overcomes the feature sparsity issues. In this paper, we adopt LibFM implementation mentioned in [19] as another baseline.

LambdaMART [3]: is the boosted tree version of LambdaRank [7], which is based on RankNet [2]. LambdaMART proves to be a very successful approach for ranking as well as recommendation. We include LambdaMART as another baseline.

4.1 Evaluation Metrics

Throughout the experiment section, we compare HBayes against other baselines on the testing held-out data set under the 5-fold cross-validation setting. For each fold, after fitting the model on the training set, we form the predictive rating for the test set, and generate the top M samples from each method for recommendation. Regarding with metrics, we adopt precision and recall as well as F1-score for measuring the product retrieval quality and normalized discounted information gain (NDCG) for measuring the recommendation's ranking quality. More specifically, precision, recall, and F1-score are defined as follows:

$$\begin{aligned}
\text{precision}_M &= \frac{\text{\# of products clicked in top } M}{M} \\
\text{recall}_M &= \frac{\text{\# of products clicked in top } M}{\text{\# of products the user clicked}} \\
\text{F1-score@}M &= 2\frac{\text{precision}_M \cdot \text{recall}_M}{\text{precision}_M + \text{recall}_M}
\end{aligned}$$

discounted cumulative gain (DCG) measures the ranking quality based on the result list product positions, defined as:

$$\mathrm{DCG}_M \quad = \quad \sum_{i=1}^{M} \frac{r_i}{\log_2(i+1)} = r_1 + \sum_{i=2}^{M} \frac{r_i}{\log_2(i+1)}$$

sorting all potential products to produce the maximum possible DCG through position M, also called Ideal DCG (IDCG). For each user, we can define NDCG as follows:

$$\mathrm{NDCG}_M \quad = \quad \frac{\mathrm{DCG}_M}{\mathrm{IDCG}_M}$$

4.2 Recommendation on E-commerce Data

The first data set is collected from a large e-commerce company. In this dataset, each sample represents a particular product which is recorded by various features including: category, title, and other properties, etc. Meanwhile, the users' click and purchase history are also recorded. Throughout the experiment, positive labels indicate that certain products in recommendation are clicked by the user, whereas negative samples indicate that products in recommendation are skipped by the user. By data cleaning and preprocessing for merging duplicated histories, removing users with too few historical samples, the final data set ends up of 895 users, $81,223$ products, $5,535$ brands with $380,595$ uniquely observed user-item pairs. In average, each user has 425 products records, ranging from 105 to $2,048$, and 61.2% of the users have fewer than 425 product clicking records. For each product, we encode each popularity and category features to a separate 20 dimension vector; title and product property feature to a separate 50 dimension vector. The total dimension for each product sample ends up of 140.

4.2.1 Feature Analysis

The e-commerce data are composed by four types of features: (1) product popularity; (2) product category; (3) product title; (4) product properties and we briefly explain their physical meanings and how we preprocess the data as follows:

Product Popularity (POP): product popularity is a measure of the prevalence of a product in the dataset. In general, consumers have a preference for a particular product during a period of time. This phenomenon is pretty common in apparel product. For a particular product i, the popularity is computed as follows: $\mathrm{POP}_i := \frac{n_{x_i}}{\mathcal{N}_{\mathbf{x}}}$, where n_{x_i} are the number of orders or the contribution of gross merchandise volume (GMV) for product i, and $\mathcal{N}_{\mathbf{x}} = \sum_{\forall x_i} n_{x_i}$ is the summation of n_{x_i} for all such products in the dataset.

Product Category (CID): user actions towards a single item such as click, purchase, etc. are typically sparse event, while more patterns could be recognized if we cluster different items into different groups based on their functionalities and utilities. In most e-commerce websites, such a categorical hierarchy exists. We encode each product's category into a high dimensional vector via one-hot encoding and adjust the feature weights by the popularity of such category.

Product Title (TITLE): product titles are created by vendors and they are typically in the forms of natural languages indicating the product functionality and utility. Examples could be like 'INMAN short sleeve round neck triple color block stripe T-shirt 2017'. We preprocess the product titles by generating the sentence vector embedding based on [5]. The main idea is to average the word weights in the title sentence based on the inverse document frequency (IDF) value of each individual word involved.

Product Property Features (PROP): other product metadata features are also provided in this dataset. For example, product 'color' feature takes value of "black", "white", "red", etc, and product 'size' feature takes value of "S", "M", "L", "XL", etc. Similar as category features, each product property is first encoded as a binary vector $\boldsymbol{x}_i \in \{0,1\}^{|N|}$, where N denotes the set of all possible corresponding product values. Then we encode all property binary vectors into a fixed length (50 throughout the experiment) vector.

On one hand by utilizing more types of product features HBayes in general reaches better results in terms of precision-recall metrics. We report PR-AUC in table (1) to prove our argument. On the other hand, more features typically needs much more training time. Figure (1) reports the change in likelihood comparing against training time (minutes) for each feature combination case [1]. It is shown that by taking one POP feature set, model takes less than 5 minutes to converge while taking feature combination POP+CID+TITLE+PROP leaves us more than 6 hours in training.

Features	PR AUC
POP	0.0406
POP+CID	0.0414
POP+CID+TITLE	0.0489
POP+CID+TITLE+PROP	0.0491

Table 1: Model performance under different feature combinations in terms of PR AUC

[1]The experiment is conducted via the same linux qual core 2.8 GHz Intel Core i7 macbook with 16 Giggabytes of memory

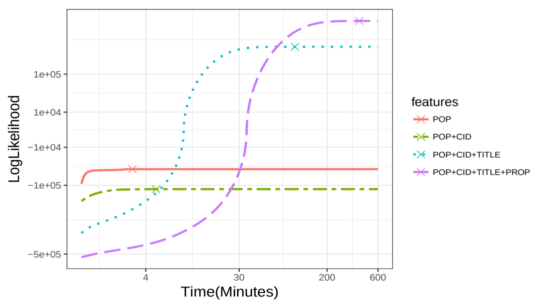

Figure 1: Training time under different feature combinations on e-commerce recommendations

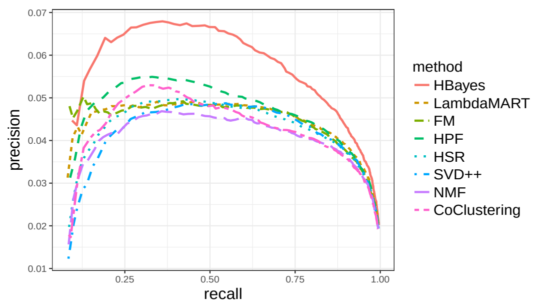

Figure 2: PR curves on e-commerce recommendations

4.2.2 Performance Comparison

We first report the model performance regarding precision-recall for HBayes and other baselines in Figure (2). As shown when each method recommend fewer number of products, HBayes, FM, shares similar performance, with the increment of recommended items, HBayes becomes more efficient, in the sense of picking more number of products that users take interest in. In terms of overall PR-AUC, HBayes is able to reach 0.053, while beating the second best LambdaMART of 0.041 PR-AUC by 29.3% percent. In terms of F1-score, we are the best cross out different number of products recommended by beating against the second best LambdaMART by 29% in average.

Regarding with ranking quality, HBayes is also superior against other baselines through out different items

Method	**F1-score@5**	**F1-score@10**	**F1-score@25**	**F1-score@50**
CoClustering	0.0640	0.0804	0.0896	0.0809
NMF	0.0565	0.0702	0.0832	0.0809
SVD++	0.0467	0.0689	0.0875	0.0865
HSR	0.0592	0.0755	0.0889	0.0850
HPF	0.0741	0.0874	0.0960	0.0876
FM	0.0720	0.0756	0.0874	0.0880
LambdaMART	0.0667	0.0782	0.0879	0.0874
HBayes	**0.0885**	**0.1065**	**0.1168**	**0.1002**

Table 2: F1-score on e-commerce recommendations

Method	**NDCG@5**	**NDCG@10**	**NDCG@25**	**NDCG@50**
CoClustering	0.1288	0.1637	0.2365	0.3050
NMF	0.1249	0.0156	0.2272	0.3020
SVD++	0.1138	0.1487	0.2287	0.3073
HSR	0.1266	0.1603	0.2354	0.3107
HPF	0.1412	0.1757	0.2503	0.3229
FM	0.1363	0.1592	0.2291	0.3117
LambdaMART	0.1287	0.1585	0.2304	0.3123
HBayes	**0.1557**	**0.1974**	**0.2871**	**0.3590**

Table 3: NDCG on e-commerce recommendations

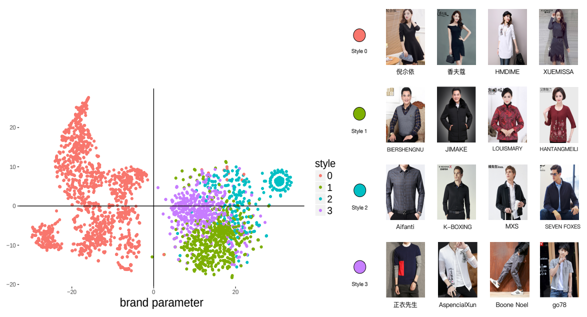

Figure 3: tSNE of four latent apparel style clusters

recommended. Specially HBayes beats the second best CoClustering at $M = 5, 10, 25$ by 21.0% in average and LambdaMART at $M = 50$ by 15.0%.

4.2.3 Model Learning Analysis

Like mentioned in Sec.3, HBayes learns the latent style clusters, and group different brands of products based on their different hidden style representations. Figure (3) shows the tSNE[16] representation of different style clusters learned by HBayes and we randomly pick four samples within each clusters and present their product images at the right subfigure. As shown, cluster one which takes a great proportion of apparel products are seemingly about young stylish female garment. This is making sense since a majority of apparel customers are young females while most products are focusing on this group of customer audience. The second cluster seems about senior customers who are in elder age. Interestingly, the third cluster and the fourth cluster which are closed tied together are both about young male customers. However, the third cluster seems more focusing on office business men style garment while the fourth cluster is more focusing on asian street styles. This indicates us that our hierarchical Bayesian model does learn meaningful intrinsic garment styles from apperal products with customer behavioral data.

4.3 Recommendation on Last.fm Music Data

The second data set is collected from Last.fm dataset [4] and Free Music Archive (FMA) [6]. Last.fm is a publicly available dataset contains the whole listening habits (till May, 5th 2009) for $1,000$ users. FMA is an open and easily accessible dataset providing 917 GiB and 343 days of Creative Commons-licensed audio from $106,574$ tracks, $16,341$ artists and $14,854$ albums, arranged in a hierarchical taxonomy of 161 genres. It also provides full-length and high quality audios with precomputed features. In our experiment, tracks in Last.fm dataset were further intersected with FMA dataset for better feature generation. The resulting dataset contains 500 users, $16,328$ tracks and 36 genres.

4.3.1 Performance Comparison

We conduct the same set of experiments as we do for e-commerce data and report different precision-recall performances in Figure (4).

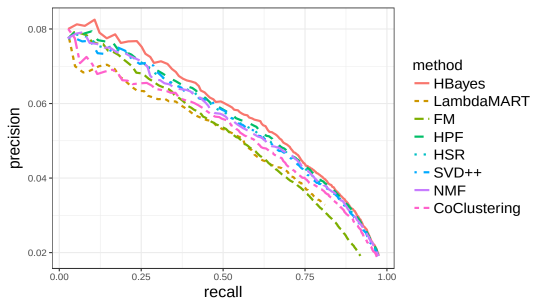

Figure 4: PR curves on Last.fm recommendations

Method	F1-score@5	F1-score@10	F1-score@25	F1-score@50
CoClustering	0.0831	0.0986	0.1049	0.0915
NMF	0.0913	0.1088	0.1075	0.0941
SVD++	0.0903	0.1084	0.1084	0.0946
HSR	0.0916	0.1089	0.1083	0.949
FPF	0.0931	0.1097	0.1092	0.961
FM	0.0863	0.1041	0.1059	0.942
LambdaMART	0.0622	0.0788	0.0995	0.0987
HBayes	**0.0962**	**0.1159**	**0.1107**	**0.0985**

Table 4: F1-score on Last.fm recommendations

Method	NDCG@5	NDCG@10	NDCG@25	NDCG@50
CoClustering	0.2415	0.2314	0.2289	0.2349
NMF	0.2556	0.2494	0.2368	0.2431
SVD++	0.2493	0.2478	0.2381	0.2439
HSR	0.2544	0.2495	0.2384	0.2448
HPF	0.2584	0.2513	0.2405	0.2474
FM	0.2527	0.2453	0.2284	0.2333
LambdaMART	0.2372	0.2337	0.2272	0.2218
HBayes	**0.2655**	**0.2620**	**0.2455**	**0.2537**

Table 5: NDCG on Last.fm recommendations

For PR-AUC, HBayes (0.055) again beats against the second best, which is LambdaMART (0.043) by 27.9%. In terms of F1-score, HBayes in average beats against LambdaMART by 14.0%. Regarding with the ranking quality, HBayes beats against the second best, which is NMF for $M = 5, 10, 25$ by 4.20% in average and SVD++ for $M = 50$ by 4.02%.

5 Conclusion(Chris)

References

[1] Ben-Elazar, S., Lavee, G., Koenigstein, N., Barkan, O., Berezin, H., Paquet, U., and Zaccai, T. Groove radio: A bayesian hierarchical model for personalized playlist generation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2017), WSDM '17, ACM, pp. 445–453.

[2] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning* (2005), ACM, pp. 89–96.

[3] Burges, C. J. From ranknet to lambdarank to lambdamart: An overview. *Learning 11*, 23-581 (2010), 81.

[4] Celma, O. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[5] De Boom, C., Van Canneyt, S., Demeester, T., and Dhoedt, B. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters 80* (2016), 150–156.

[6] Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X. Fma: A dataset for music analysis. In *18th International Society for Music Information Retrieval Conference* (2017).

[7] Donmez, P., Svore, K. M., and Burges, C. J. On the local optimality of lambdarank. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (2009), ACM, pp. 460–467.

[8] George, T., and Merugu, S. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE international conference on* (2005), IEEE, pp. 4–pp.

[9] Gultekin, S., and Paisley, J. A collaborative kalman filter for time-evolving dyadic processes. In *Proceedings of the 2014 IEEE International Conference on Data Mining* (Washington, DC, USA, 2014), ICDM '14, IEEE Computer Society, pp. 140–149.

[10] GUY, I., ZWERDLING, N., CARMEL, D., RONEN, I., UZIEL, E., YOGEV, S., AND OFEK-KOIFMAN, S. Personalized recommendation of social software items based on social relations. In *Proceedings of the Third ACM Conference on Recommender Systems* (New York, NY, USA, 2009), RecSys '09, ACM, pp. 53–60.

[11] KOREN, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 426–434.

[12] KOREN, Y. Collaborative filtering with temporal dynamics. *Commun. ACM 53*, 4 (Apr. 2010), 89–97.

[13] KOREN, Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD) 4*, 1 (2010), 1.

[14] LIU, Y., MIAO, J., ZHANG, M., MA, S., AND RU, L. How do users describe their information need: Query recommendation based on snippet click model. *Expert Systems with Applications 38*, 11 (2011), 13847 – 13856.

[15] LOPS, P., DE GEMMIS, M., AND SEMERARO, G. *Content-based Recommender Systems: State of the Art and Trends*. 2011, p. 73.

[16] MAATEN, L. V. D., AND HINTON, G. Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[17] MNIH, A., AND SALAKHUTDINOV, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems* (2008), pp. 1257–1264.

[18] RENDLE, S. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on* (2010), IEEE, pp. 995–1000.

[19] RENDLE, S. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST) 3*, 3 (2012), 57.

[20] RENDLE, S., FREUDENTHALER, C., GANTNER, Z., AND SCHMIDT-THIEME, L. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* (2009), AUAI Press, pp. 452–461.

[21] RENDLE, S., FREUDENTHALER, C., AND SCHMIDT-THIEME, L. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 811–820.

[22] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (New York, NY, USA, 2001), WWW '01, ACM, pp. 285–295.

[23] SU, X., AND KHOSHGOFTAAR, T. M. A survey of collaborative filtering techniques. *Adv. in Artif. Intell. 2009* (Jan. 2009), 4:2–4:2.

[24] TANG, J., HU, X., AND LIU, H. Social recommendation: a review. *Social Network Analysis and Mining 3*, 4 (1 2013), 1113–1133.

[25] YIN, D., HONG, L., XUE, Z., AND DAVISON, B. D. Temporal dynamics of user interests in tagging systems. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (2011), AAAI'11, AAAI Press, pp. 1279–1285.

[26] YUAN, Q., CONG, G., ZHAO, K., MA, Z., AND SUN, A. Who, where, when, and what: A nonparametric bayesian approach to context-aware recommendation and search for twitter users. *ACM Trans. Inf. Syst. 33*, 1 (Feb. 2015), 2:1–2:33.

[27] ZHANG, Y., AND KOREN, J. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (2007), ACM, pp. 47–54.