mygraygray.9

A Bayesian Hierarchical Model for Personalized Apparel Recommendation

Abstract

Abstract.

1 Introduction(Jerry)

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, daily products in retail stores are usually stored in hierarchical inventories. News on web pages are created and placed hierarchically in most web sites. Such hierarchical structures and the data within themselves 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 key to the 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 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 into a complex deep neural network and hope the network itself can figure out the way to use the hierarchical knowledge. However, such approaches usually behave more like black boxes and cannot provide any interpretation on their outcomes.

In this work, we propose and develop an Bayesian modeling framework

In this work, we solve this problem by bayesian.

Our contribution.

Paper oganization.

2 Related Work(Chris)

With the explosive growth of digital information over the internet, the recommender system is considered to be an effective approach to overcome such information overload. Traditional recommendation algorithms such as item-based approaches including collaborative filtering [**?**, **?**] and matrix factorization [**?**] recommend an item by learning the interaction between users and items, while content-based approaches including [**?**, **?**, **?**] compare the auxiliary information of both user and item, so to recommend the similar items. Further, by looking into time variant nature of recommender systems, there are time-aware recommendation approaches [**?**, **?**] 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 classlic recommender algorithms. [**?**] develops a collaborative filtering type of approach with predictions from static average value combining with a dynamic changing factor. [**?**] propose a user-tag-specific temporal interest model to track user interst over time by maximizing the time weighted data likelihood. For better personalization, [**?**] 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. [**?**] 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. [?] and [?] 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 take account of both the user preferences and product hidden structural properties, which recommend products that align to users' long term tastes. Meanwhile by interpreting the modeling results, we can have better understanding on users' likes and dislikes, so to target certain group of users based on their social behavior more accurate and efficient, which is another very important topic in the advertising business.

3 Methodology(Jerry)

4 Experiment(Chris)

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 [?]:

SVD & SVD++ [?, ?]:

CoClustering [?]:

Fatorization Machine (FM) [Rendle(2012)]:

LambdaMART [?]: LambdaMART is the boosted tree version of LambdaRank [?], which is based on RankNet [?]. 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 for measuring the product retrieval quality and normalized discounted information gain (NDCG) for measuring the recommendation's ranking quality. More specifically, precision and recall are defined as follows:

$$\begin{aligned} precision_M &= \frac{\#of products clicked in top M}{M} \\ recall_M &= \frac{\#of products clicked in top M}{\#of products the user clicked} \end{aligned}$$

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

$$DCG_M = \sum_{i=1}^{M} \frac{r_i}{\log_2(i+1)} = r_1 + \sum_{i=2}^{M} \frac{r_i}{\log_2(i+1)}$$

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:

$$NDCG_M = \frac{DCG_M}{IDCG_M}$$

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

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 $\mathcal{N}_{\mathbf{x}=\sum_{\forall \mathbf{x_i}} \mathbf{n_{x_i}}}$ is the summation of n_{x_i} for all such products in the dataset.

Product Category (CID): many user actions are related to some items. To alleviate the sparsity issue, we represent each item by its category. Actually, All items in our dataset are mapped to a category hierarchy. Thus, given a behavior sequence, we obtain a set of product category IDs and could encode these into the one-hot vector. In order to reduce the influence of popular categories, the feature weight is adjusted by the term frequency-inverse document frequency (TFIDF) equation. Here, we treat each user as a document and each category as a term.

Product Title (TITLE): product titles are created by sellers which are 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 [**?**]. The main idea is to average the word weights in the title sentence based on the IDF value of each individual word involved.

Product property Features (PROP): Metadata features are provided in our dataset. For example, we will label 'Color' as "black", "white" and "red", etc, or label 'Size' as "L", "M" and "X", etc. Each property is first encoded as a binary vector $x_i \in \{0,1\}^{|N|}$, where N denotes the set of possible tags. Then, we encode these binary vectors into a fixed length(50 in our experiment) vector. We can see the property features vector is generally sparse since each product is only associated with a small number of feature.

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 (4.2.1) reports the change in likelihood comparing against training time (minutes) for each feature combination case. 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.

[width=]fig/Lik$_t ime$

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

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

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

[width=]fig/pre-recall

Figure 2: PR curves on e-commerce recommendations

4.2.2 Performance Comparison

4.2.3 Model Learning Analysis

4.3 Recommendation on Last.fm Music Data:

the second data set is collected from Last.fm dataset [**?**] and Free Music Archive (FMA) [**?**]. 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 audio, precomputed features. In our experiment, tracks in Last.fm dataset were further intersected with FMA dataset for better feature generation. Therefore, the resulting dataset contains 500 users, $16,328$ tracks and 36 genres.

4.3.1 Performance Comparison

5 Conclusion(Chris)

References

[Rendle(2012)] Rendle, Steffen. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

0.7

Method	**F1-score@5**	**F1-score@10**	**F1-sco**
mygray CoClustering	0.0640	0.0804	0.0
KNN	0.0591	0.0679	0.0
mygray NMF	0.0565	0.0702	0.0
SVD++	0.0467	0.0689	0.0
mygray FM	0.0720	0.0756	0.0
LambdaMART	0.0667	0.0782	0.0
mygray HBayes	**0.0885**	**0.1065**	**0.1**

Table 2: F1-score on e-commerce recommendations

0.7

Method	NDCG@5	NDCG@10	NDCG@25	NDCG@50
mygray CoClustering	0.1288	0.1637	0.2365	0.3050
KNN	0.1268	0.1540	0.2136	0.2868
mygray NMF	0.1249	0.0702	0.1567	0.3020
SVD++	0.1138	0.0689	0.1487	0.3073
mygray FM	0.1363	0.0756	0.1592	0.3117
LambdaMART	0.1287	0.1585	0.2304	0.3123
mygray HBayes	**0.1557**	**0.1974**	**0.2871**	**0.3590**

Table 3: NDCG on e-commerce recommendations

[width=0.32height=3cm]fig/brand$_t sne[width = 0.25height = 4cm]fig/style - brand$

Figure 3: tSNE representation for apparel style clusters