Hierarchical Bayesian Personalized Recommendation: A Case Study and Beyond

Author
Company
Address
City, State, Country zip code
email

ABSTRACT

Items in modern recommender systems are often organized in hierarchical structures. These hierarchical structures and the data within them provide valuable information for building personalized recommendation systems. In this paper, we propose a general hierarchical Bayesian learning framework, i.e., *HBayes*, to learning both the structures and associated latent factors. Furthermore, we develop a variational inference algorithm that is able to learn model parameters with fast empirical convergence rate. The proposed HBayes is evaluated on two real-world datasets from different domains. The results demonstrate the benefits of our approach on item recommendation tasks, and show that it can outperform the state-of-the-art models in terms of both F1 measurement and normalized discounted cumulative gain.

KEYWORDS

Hierarchical Bayesian; Recommendation; Personalization

ACM Reference format:
Author. 2016. Hierarchical Bayesian Personalized Recommendation: A Case Study and Beyond. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 8 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

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 hierarchical structures usually doesn'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. They are difficult to debug 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. By introducing latent variables, all hierarchical structures are encoded as conditionaly independences in HBayes graphical models. 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. As an illustration, we consider apparel styles, product brands and apparel items and form them into a three level hierarchy. We add additional latent variables as the apparel style membership variables to capture the diverse and hidden style properties of each brand. Furthermore, we include user specific features into HBayes and extend HBayes into the superivsed learning setting where users feedback actions such as clicks and conversions are incorporated. Please note that our HBayes framework 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

Conference'17, Washington, DC, USA
2016. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

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

2.1 Recommender Systems

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 learn the interaction between users and items and recommend items to users who share similar historical behaviors: collaborative filtering [28, 30] and matrix factorization [26] are both effective approaches under this category. Content-based approaches including [19, 20, 35] take use of the auxiliary information of both users and items in the content spaces so to recommend similar items. Furthermore, by looking into users session information and analyzing such visiting patterns, there are another branch of recomender system methodologies which are session-based [13, 31]. They take *time* as another input dimension besides items and users, which aims at explicitly modeling the user interest over time slices to combine with various classic recommender algorithms. [18] develops a collaborative filtering type of approach with predictions from static average value combining with a dynamic changing factor. [34] proposes a user-tag-specific temporal interest model to track the user interest over time by maximizing the time weighted data likelihood. For better personalization, [14] proposes an item-based recommender system combining with each user's social network information.

2.2 Bayesian Approaches for Recommendation

Recently, there are works using Bayesian inferencing for recommender systems. [25] combines the Bayesian inference and matrix factorization together for learning users implicit feedbacks (click & purchase) so to directly optimize the ranking results in the sense of AUC metrics. [1] and [36] takes user preference consistency into account and develops a variational Bayesian personalized ranking model for better music recommendation. However, none of these approaches leverage the entity structural information when learning the Bayesian models. Given the fact that hierarchical structures widely exist in several recommendation practices such like e-commerce recommender systems (Fig.**??** shows a typical e-commerce structure); social-network recommender systems; music recommender systems; etc. fails to build in such structural infomation typically results in the those Bayesian approaches not as effcient as they supposed to.

2.3 Hierarchical Recommender Systems

Hierarchical information for recommender systems, like mentioned, is a natural yet powerful entity structure that encode human knowledge by means of tree based dependency contraints between items and the corresponding hyper-parameters for recommender systems. Recommender system hierarchies could be explicit or implicit, and there are quite a few approaches take use of such information in order to recommend items to users who have explicitly visited other hierarchically related items or shown preferences to items that are hierarchically belonging to the same sub-categories. In social tagging systems, Shepitsen et.al. [29] relies on the hierarchies generated by user-taggings, a.k.a folksonomies to build a better personalized recommender system. In e-commerce, Wang et.al. [33] introduced a hierarchical matrix factorization approach that exploits the intrinsic hierarchical information to alleviate cold start and data sparsity problems. Despite the fact that these hierarchical recommender systems have received some sucess, there are still some challenges such like: 1. how to design the hierarchical structure and learn such structure efficiently if it is not completely explicit; 2. how to better understand the implicit hierarchical topologies discovered by recommendation approaches and ... all remain unsolved issues.

2.4 Hierarchical Bayesian Inference

In our work, we focus on building a generic recommender system that combines the hierarchical structural information as well as the Bayesian inference to better tackle the recommendation problem. Bayesian learning scheme 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 grants us the capability to analyze and understand users interests and preference.

3 THE HBAYES FRAMEWORK

In this work, we develop our generalized hierarchical Bayesian modeling framework that is able to capture the hierarchical structural relations and latent relations in the real-world recommendation scenarios. In our framework, we incorporate both user specific features and user actions into model learning so that more personalized recommendation results can be achieved. Furthermore, we present a variational inference algorithm for the HBayes framework to provide fast learning convergence.

In the following, we will describe how HBayes works by using apparel recommendation as an illustration scenario. Using a real-world example helps explain the hierarchy structural relations and latent variable conditional independence assumptions implied in HBayes. Please note that even we explain HBayes in apparel recommendation, the framework itself can be generalized into other recommendation scenarios with little modification.

3.1 Generative Process

In the real-world scenario, each item or product has to come with a brand and a brand may have more than one items in the hierarchical structures. Therefore, we denote each event t as a 4-tuple (Item, Brand, User, IsClick), i.e, $(\mathbf{X}_t, b_t, u_t, y_t)$. $\mathbf{X}_t$ represents the item features associated with event t and y_t is the binary label that indicates whether user u_t has clicked $\mathbf{X}_t$ or not. b_t is the brand of item $\mathbf{X}_t$.

Furthermore, we expand the hierarchy by a hidden factor, i.e., "style". Products from each brand b_t tends to exhibit different styles

or tastes, which are unknown but exist. In this paper, brands are represented as random mixtures over latent styles, where each style is characterized by a distribution over all the items. Let S, B, U and N be the total number of styles, brands, users and events.

The generative process of HBayes can be described as follows:

Step 1. Draw a multivariate Gaussian prior for each style j, i.e, $\mathbf{S}_j \sim \mathcal{N}(\mathbf{w}, \delta_s^{-1}\mathbf{I})$ where $j \in \{1, \cdots, S\}$.

Step 2. Draw a multivariate Gaussian prior for each user k, i.e, $\mathbf{U}_k \sim \mathcal{N}(\mathbf{0}, \delta_u^{-1}\mathbf{I})$ where $k \in \{1, \cdots, U\}$.

Step 3. Draw a style proportion distribution $\boldsymbol{\theta}$ for each brand i, $\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\gamma})$ where $i \in \{1, \cdots, B\}$.

Step 4. For each brand i:

Step 4.1 Draw style assignment z_i for brand i, $z_i \sim \text{Mult}(\boldsymbol{\theta})$.

Step 4.2 Draw $\mathbf{B}_i \sim \mathcal{N}(\mathbf{S}_{z_i}, \delta_b)$.

Step 5. For each event t, draw y_t from Bernoulli distribution where the probability p is defined as $p(y_t|\mathbf{x}_t, \mathbf{B}_{b_t}, \mathbf{U}_{u_t})$.

where δ_s, δ_u and δ_b are the scalar precision parameters and $\mathbf{w}$ is the prior mean of $\mathbf{S}_j$.

In this work, in order to have the higher modeling flexibility and capacity, we also treat each distribution's parameter as a random variable and define hyper-priors over them. More specifically, We draw the prior mean $\mathbf{w}$ from $\mathcal{N}(\mathbf{0}, \delta_w^{-1}\mathbf{I})$. For δ_w, δ_s, δ_u and δ_b, we define Gamma priors over them i.e., $p(\delta_*) = \mathcal{G}(\alpha, \beta)$, where $\delta_* \in \{\delta_w, \delta_s, \delta_u, \delta_b\}$.

The family of probability distributions corresponding to this generative process is depicted as a graphical model in Figure 1.

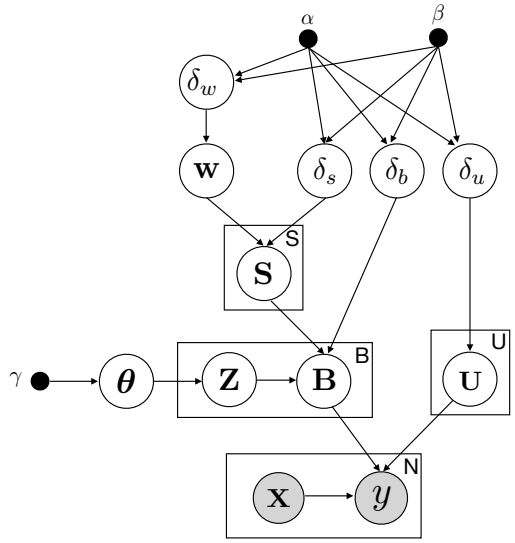

Figure 1: A graphical model representation of HBayes.

3.2 Probability Priors & Models

In this work, we model the probability of a single event t given $(\mathbf{X}_t, b_t, u_t, y_t)$ as

$$p\left(y_t|\boldsymbol{X}_t, \boldsymbol{B}_{b_t}, \boldsymbol{U}_{u_t}\right) = \left[\sigma\left(h_t\right)\right]^{y_t} \cdot \left[1 - \sigma\left(h_t\right)\right]^{1-y_t} \tag{1}$$

where $\sigma(\cdot)$ is a logistic function, i.e. $\sigma(x) = (1 + e^{-x})^{-1}$. $h_t \stackrel{\text{def}}{=} \boldsymbol{X}_t^T(\boldsymbol{B}_{b_t} + \boldsymbol{U}_{u_t})$. $\mathbf{X}^T$ is the vector transpose of $\mathbf{X}$. $\boldsymbol{U}_{u_t}$ represents user specific information encoded in HBayes for user u_t and $\boldsymbol{B}_{b_t}$ denotes the brand b_t's specific information.

As mentioned in Step 3 of the generative process of HBayes, each brand i's style proportion distribution $\boldsymbol{\theta}$ follows a Dirichlet distribution: $p(\boldsymbol{\theta}) \sim Dir(\boldsymbol{\gamma})$, which is defined as follows:

$$Dir(\boldsymbol{\theta}|\boldsymbol{\gamma}) = \frac{\Gamma(\sum_{j=1}^{S}\theta_j)}{\prod_{j=1}^{S}\Gamma(\theta_j)}\prod_{j=1}^{S}\theta_j^{\gamma_j - 1}$$

where $\boldsymbol{\gamma}$ is the S-dimensional Dirichlet hyper-parameter. We initialize γ_j by $\frac{1}{S}$.

Furthermore, in Step 4 of the generative process of HBayes, a brand is modeled as a random mixture over latent styles. Hence, we model the brand parameters by a mixture of multivariate Gaussian distribution defined as follows:

$$p(\boldsymbol{B}_i|\boldsymbol{z}_i, \boldsymbol{S}_{z_i}, \delta_b) = \prod_{j}^{S} p(\boldsymbol{B}_i|\boldsymbol{S}_j, \delta_b)^{\mathbb{I}(\boldsymbol{z}_i=1)} = \prod_{j}^{S} \mathcal{N}(\boldsymbol{B}_i; \boldsymbol{S}_j, \delta_b)^{\mathbb{I}(\boldsymbol{z}_i=1)}$$

where $\mathbb{I}(\xi)$ is an indicator function that $\mathbb{I}(\xi) = 1$ if the statement ξ is true; $\mathbb{I}(\xi) = 0$ otherwise.

Therefore, the log joint likelihood of the dataset $\mathcal{D}$, latent variable $\boldsymbol{Z}$ and the parameter Θ by given hyper-parameters $\mathcal{H} = \{\gamma, \alpha, \beta\}$ could be written as follows:

$$\begin{aligned}
&\log\left(p(\mathcal{D}, \boldsymbol{Z}, \Theta|\mathcal{H})\right) \\
&= \sum_{t=1}^{N}\log p(y_t|\boldsymbol{X}_t, \boldsymbol{B}_{b_t}, \boldsymbol{U}_{u_t}) + \sum_{i=1}^{B}\log p(\boldsymbol{B}_i|\boldsymbol{z}_i, \boldsymbol{S}_{z_i}, \delta_b) \\
&+ \sum_{i=1}^{B}\log p(\boldsymbol{z}_i|\boldsymbol{\theta}) + \sum_{j=1}^{S}\log p(\boldsymbol{S}_j|\boldsymbol{w}, \delta_s) + \log p(\boldsymbol{\theta}|\boldsymbol{\gamma}) \\
&+ \sum_{k}^{U}\log p(\boldsymbol{U}_k|\delta_u) + \log p(\boldsymbol{w}|\delta_w) \\
&+ \log p(\delta_u) + \log p(\delta_b) + \log p(\delta_w) + \log p(\delta_s)
\end{aligned} \tag{2}$$

We use Θ to denote all model parameters:

$$\Theta \stackrel{\text{def}}{=} \left\{\{\boldsymbol{U}_k\}, \{\boldsymbol{B}_i\}, \{\boldsymbol{S}_j\}, \boldsymbol{w}, \boldsymbol{\theta}, \delta_u, \delta_b, \delta_s, \delta_w\right\},$$

where $k \in \{1, \cdots, U\}$, $i \in \{1, \cdots, B\}$, $j \in \{1, \cdots, S\}$.

3.3 Optimization

Since both Z and Θ defined by the HBayes are unobserved, we cannot learn our HBayes directly. Instead, we infer the expectations of these latent variables and compute the expected log likelihood of the log joint probability with respect to the latent variables distribution. In the following, we omit the explicit conditioning on $\mathcal{H}$ for notational brevity.

$$\mathcal{Q} = \int_{\Theta}\sum_{\boldsymbol{Z}} p(\boldsymbol{Z}, \Theta|\mathcal{D})\log\left(p(\mathcal{D}, \boldsymbol{Z}, \Theta)\right)d\Theta \tag{3}$$

From the Bayes rule, we can see that the posteriors distribution of $\mathbf{Z}$ and Θ can be represented by $p(Z, \Theta|\mathcal{D}) = \frac{p(\mathcal{D}, Z, \Theta)}{p(\mathcal{D})}$. However, this above distribution is intractable to compute in general [9]. To tackle this problem, a wide variety of approximate inference algorithms are developed, such as Laplace approximation [27], variational approximation [2], and Markov Chain Monte Carlo (MCMC) approach [3], etc.

In this work, we choose to solve this problem by using variational Bayes approximation [2]. More specifically, we approximate the original posterior distribution $p(Z, \Theta|\mathcal{D})$ with a tractable distribution $q(Z, \Theta)$ such that instead of maximizing the Q function defined in eq.(3), we maximize the variational fee energy defined as

$$Q'(q) = \int_{\Theta} \sum_{Z} q(Z, \Theta) \log \frac{p(\mathcal{D}, Z, \Theta)}{q(Z, \Theta)} d\Theta \tag{4}$$

which is also equal to minimize the KL divergence of $p(Z, \Theta|\mathcal{D})$ and $q(Z, \Theta)$.

Here we choose to apply *Mean Field* approximation technique to approximate $p(Z, \Theta|\mathcal{D})$, where we assume independence among all different variables ($\mathbf{Z}$ and Θ) and define $q(Z, \Theta)$ as follows:

$$\begin{aligned} q(Z, \Theta) =& q(\mathbf{Z}) \cdot \prod_{k=1}^{K} q(\mathbf{U}_k) \cdot \prod_{j=1}^{S} q(\mathbf{S}_j) \cdot \prod_{i=1}^{B} q(\mathbf{B}_i) \\ & \cdot q(\mathbf{w}) \cdot q(\boldsymbol{\theta}) \cdot q(\delta_u) \cdot q(\delta_b) \cdot q(\delta_s) \cdot q(\delta_w) \end{aligned} \tag{5}$$

where q denotes different distribution function for notation brevity.

3.3.1 Sigmoid Approximation. The Gaussian priors from our log joint probability (see eq.(2)) are not conjugate to the data likelihood due to the fact that our events are modeled by a sigmoid function (see eq.(1)). In order to conduct tractable inference on $Q'(q)$, we apply a variational lower bound approximation on eq.(1) that has the "squared exponential" form. Therefore, they are conjugate to the Gaussian priors.

$$\sigma(h_t) \geq \sigma(\xi_t) \exp\left\{\frac{1}{2}(h_t - \xi_t) - \lambda_t(h_t^2 - \xi_t^2)\right\} \tag{6}$$

where $\lambda_t \stackrel{\text{def}}{=} \frac{1}{2\xi_t}[\sigma(\xi_t) - \frac{1}{2}]$ and ξ_t is a variational parameter. This lower bound is derived using the convex inequality. The similar problem was discussed in [15, 16].

Therefore, each event likelihood can be expressed as follows:

$$\begin{aligned} & \left[\sigma(h_t)\right]^{y_t} \cdot \left[1 - \sigma(h_t)\right]^{1-y_t} = \exp\left(y_t h_t\right)\sigma(-h_t) \\ \geq & \sigma(\xi_t) \exp\left(y_t h_t - \frac{1}{2}(h_t + \xi_t) - \lambda_t(h_t^2 - \xi_t^2)\right) \end{aligned} \tag{7}$$

By using the sigmoid approximation in eq.(7), our variational fee energy $Q'(q)$ (eq.(4)) can be bounded as:

$$Q'(q) \geq Q'_{\xi}(q) = \int_{\Theta} \sum_{Z} q(Z, \Theta) \log \frac{p_{\xi}(\mathcal{D}, Z, \Theta)}{q(Z, \Theta)} d\Theta \tag{8}$$

In the following, we will maximize the lower bound of the variational free energy $Q'_{\xi}(q)$ for parameter estimation.

3.3.2 Parameter Estimation. We develop an Expectation-Maximization (EM) algorithm for HBayes parameter estimation where in the E-step, we compute the expectation of the hidden variables $\mathbf{Z}$ and in the M-step, we try to find Θ that maximizes lower bound of the variational free energy $Q'_{\xi}(q)$ (eq.(8)). In the M-step, we use coordinate ascent variational inference (CAVI) [2] to optimize $Q'_{\xi}(q)$. CAVI iteratively optimizes each factor of the mean field variational distribution, while holding the others fixed.

E-step

M-step

3.3.3 Summary. The parameter estimation method for the HBayes is summarized by Algorithm 1.

Algorithm 1 Parameter Estimation in HBayes

1: **INPUT**:
2: Hyper-parameters $\mathcal{H}$: $\mathcal{H} = \{\alpha, \beta, \boldsymbol{\gamma}\}$
3: Data samples $\mathcal{D}$: $(\mathbf{X}_t, b_t, u_t, y_t)$, $t = 1, \cdots N$
4: **procedure** Learning HBayes
5: **repeat**
6: E-step: compute expectation of $\mathbf{Z}$ by eq.().
7: M-step: estimate $\{\boldsymbol{U}_k\}, \{\boldsymbol{B}_i\}, \{\boldsymbol{S}_j\}, \boldsymbol{w}, \boldsymbol{\theta}, \delta_u, \delta_b, \delta_s, \delta_w$ by eq.() - eq.().
8: **until** Convergence
9: **return** Θ

3.4 Prediction

In the recommender system, the task is to generate top K products list for each user. So, given the user u^*, it's natural to expose top M products based on the probability of positive outcome. For the m^{th} item, the probability is calculated as:

$$\begin{aligned} \hat{y}_m = p(y_m = 1|\boldsymbol{x}_m, \mathcal{D}, \mathcal{H}) &\approx \int \sigma(h_m) q(Z, \Theta) \mathrm{d}\Theta \\ &= \int \sigma(h_m) \mathcal{N}(h_m|\mu_m, \sigma_m^2) \mathrm{d}h_m \approx \sigma\left(\frac{\mu_m}{\sqrt{1 + \pi\sigma_m^2/8}}\right) \end{aligned}$$

where h_m is a random variable with Gaussian distribution:

$$h_m \stackrel{\text{def}}{=} \boldsymbol{x}_m^T(\boldsymbol{w}_{b_m}^{(b)} + \boldsymbol{w}_{u^*}^{(u)}) \sim \mathcal{N}(h_m; \mu_m, \sigma_m^2)$$

$$\mu_m \stackrel{\text{def}}{=} \mathbb{E}\left[\boldsymbol{x}_m^T(\boldsymbol{w}_{b_m}^{(b)} + \boldsymbol{w}_{u^*}^{(u)})\right]$$

$$\sigma_m^2 \stackrel{\text{def}}{=} \mathbb{E}\left[\left(\boldsymbol{x}_m^T(\boldsymbol{w}_{b_m}^{(b)} + \boldsymbol{w}_{u^*}^{(u)}) - \mu_m\right)^2\right]$$

4 EXPERIMENT

In this section, we conduct several experiments on two data sets: (1) the real-world e-commerce apparel data set; (2) the publicly available music data set. For both data sets, we compare HBayes against several other *state-of-the-art* recommendation methods which are briefly mentioned as follows:

HSR [32]: is an item-based recommendation approach that employs a special non-negative matrix factorization for exploring the

implicit hierarchical structure of users and items so the user preference towards certain products are better understood. We adopt HSR as one baseline.

HPF [12]: generates a hierarchical Poisson factorization model for better modeling users' rating towards certain items based upon each one's latent preferences. Unlike proposed HBayes, HPF does not leverage the content item feature for constructing the hierarchical structure. We adopt HPF as another baseline.

SVD++ [17, 22]: combines the collaborative filtering approach and latent factor approach, so to provide a more accurate neighboring based recommendation result. We adopt SVD++ as another baseline.

CoClustering [11]: another collaborative filtering type of approach based on weighted co-clustering improvements that simultaneously cluster users and items. We adopt CoClustering as another baseline.

Factorization Machine (FM) [23, 24]: 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 [24] specifically as another baseline.

LambdaMART [5]: is the boosted tree version of LambdaRank [10], which is based on RankNet [4]. 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 experiments, we compare HBayes against other baselines on the testing held-out data set under the 5-folds cross-validation settings. For each fold, after fitting the model on the training set, we rank on the test set by each model, and generate the top M samples with maximal ranking scores for recommendation. Regarding metrics, we adopt precisions and recalls as well as F1-scores for evaluating the product retrieval quality and normalized discounted information gain (NDCG) for the recommendation's ranking quality. More specifically, precisions, recalls, and F1-scores are defined as follows:

$$\begin{aligned}
\text{Precision@K} &= \frac{\text{\# of products clicked in top K}}{\text{K}} \\
\text{Recall@K} &= \frac{\text{\# of products clicked in top K}}{\text{\# of products the user clicked}} \\
\text{F1-score@K} &= 2 \cdot \frac{\text{Precision@K} \cdot \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}}
\end{aligned}$$

discounted cumulative gain (DCG) measures the ranking quality based on the item positions from the resulting list, which is defined as follows:

$$\text{DCG@K} = \sum_{i=1}^{K} \frac{r_i}{\log_2(i+1)} = r_1 + \sum_{i=2}^{K} \frac{r_i}{\log_2(i+1)}$$

re-rank all potential products to produce the ideal maximal possible DCG through out first K items, and its corresponding DCG score is called the ideal discounted cumulative gain (IDCG). By using both DCG and IDCG, the normalized discounted cumulative gain (NDCG) could be defined as follows:

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}$$

4.2 Recommendation on Apparel Data

The first apparel data set is collected from a large e-commerce company. In this dataset, each sample represents a particular apparel product which is recorded by various features including: categories, titles, and other properties, etc. Meanwhile, the user click information are also recorded and translated into data labels. Throughout the experiment, positive labels indicate that certain recommended products are clicked by the user, whereas negative samples indicate that the recommended products are skipped by the user which usually implies that the user is 'lack of interest' towards that certain item. By data cleaning and preprocessing of (1) merging duplicated histories; (2) removing users of too few records, the post-processed data set ends up with **895** users, **81223** products, **5535** brands with **380595** uniquely observed user-item pairs. In average, each user has **425** products records, ranging from **105** to **2048**, and **61.2**% of the users have fewer than **425** product clicking records. For each item, we encode the popularity and category features into a **20** dimensional feature vector; title and product property into a **50** dimensional feature vector. Combining with other features, the total dimension for each sample ends up with **140**.

4.2.1 Feature Analysis. The apparel data are composed of four types of features: (1) product popularity; (2) product category; (3) product title; (4) product properties. We briefly explain each feature's physical meaning and how we process the data as follows:

Product Popularity (POP): product popularity is a measure of the prevalence of a item in the dataset. In general, consumers have preferences for a particular product during a period of time. This phenomenon is pretty common for apparel products (*e.g.* apparels in certain styles may be of dominant prevalence especially when famous entertainers are wearing them). For a particular product i, the popularity is defined as: $\text{POP}_i := \frac{n_{x_i}}{\mathcal{N}_{\mathbf{x}}}$, 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): In e-commerce, items are clustered into different groups based on the alliance/ similarity of the item functionalities and utilities. In e-commerce websites, such an explicit hierarchy usually exists. We encode each item'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 [7]. The main idea is to average the wording weights in the title sentence based on the inverse document frequency (IDF) value of each individual word involved.

Product Property Features (PROP): other product meta data features are also provided in the apparel dataset. For instance, the color feature of items takes value like: 'black', 'white', 'red', etc, and the sizing feature takes value like 'S', 'M', 'L', 'XL', etc. Similar as category features, each product property is first encoded into 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 such property binary vectors into a fixed length (50) vector.

On one hand, by utilizing more features HBayes in general reaches better performance in terms of precision-recall metrics. We report PR-AUC in table (1) to prove this argument; on the other hand, more features needs much more training time. Figure (8) reports the changes in likelihood against training time cost (in minutes) for each feature use cases[1]. It is shown that by only taking POP features, model spends less than 5 minutes to converge while taking POP+CID+TITLE+PROP features, the model needs 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

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 2: NDCG on apparel recommendations

4.2.2 Performance Comparison. We first report the model performance regarding precision, recall, as well as F1-score for HBayes against other baselines in Figure (2,3,4). As shown when each method recommend fewer number of products (K = 5), HBayes does not show the superiority regarding with recalls, with the increment of recommended items, the recall for HBayes becomes much better against others, which implies HBayes is really efficient in terms of finding items that people tend to take interest in. In the sense of precision, HBayes is consistently better than other baseline methods which implies HBayes is much more accurate in terms of item classification. Given the performance of precisions and recalls, HBayes is much better regarding F1-score with different K for apparel recommendation.

[1]The experiment is conducted via the same Linux Qual core 2.8 GHz Intel Core i7 macbook with 16 Gigabytes of memory

Regarding the ranking quality, we use NDCG to report each method's performance in Table (2). HBayes is superior against other baseline methods through out different K recommended. Specially, HBayes beats the second best HPF at K = 5 by 10.3%, at K = 10 by 12.4%, at K = 25 by 14.7% and at K = 50 by 11.2%.

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 (9) shows the tSNE [21] representations for different apparel clusters learned by HBayes and we randomly pick 4 samples out of each cluster and display their product images at the right subfigure. As shown, cluster one taking the majority proportion of apparel items seems about stylish female youth garment. This intuitively makes sense since the majority of apparel customers are female youth for e-commerce websites; as a result, most apparels are also focusing on female customer audience. The second cluster seems about senior customers who are elder in age. Interestingly, the third cluster and the fourth cluster which are closely tied up are both about male customer youth. However, the third cluster seems focusing more on office business garment while the fourth cluster seems more about K-pop street styles. This indicates us that the HBayes indeed learns the meaningful intrinsic garment styles from apparel items by leveraging customer behavior data.

4.3 Recommendation on Last.fm Music Data

The second data set is collected from Last.fm dataset [6] and Free Music Archive (FMA) [8]. Last.fm is a publicly available dataset which contains the whole listening habits (till May, 5th 2009) for **1000** users. FMA is an open and easily accessible dataset providing **917** GiB and **343** days of Creative Commons-licensed audio from **106574** tracks, **16341** artists and **14854** 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, **16328** tracks and **36** genres.

4.3.1 Performance Comparison. We conduct similar experiments as we do for apparel data set and report precisions, recalls as well as F1-scores in Figure (5,6,7). Although HPF and HBayes share similar performance regarding recalls along with different K, HBayes is dominant for precision at different K, specially when K is small (5, 10), which indicates HBayes is very efficient and precise for helping users pick up the songs they prefer even when the recommended item lists are short. Combining the two, HBayes is superior in F1-scores for different K recommended.

In terms of ranking quality, we report the NDCG performance in Table (3). Similar as e-commerce apparel data, HBayes is the best approach and HPF is the second best one throughout different K items recommended. Specifically, HBayes beats HPF for 2.7% at K = 5, 4.3% at K = 10, 2.1% at K = 25, and 2.5% at K = 50 separately.

5 CONCLUSION

REFERENCES

[1] Shay Ben-Elazar, Gal Lavee, Noam Koenigstein, Oren Barkan, Hilik Berezin, Ulrich Paquet, and Tal Zaccai. 2017. Groove Radio: A Bayesian Hierarchical Model for Personalized Playlist Generation. In *Proceedings of the Tenth ACM*

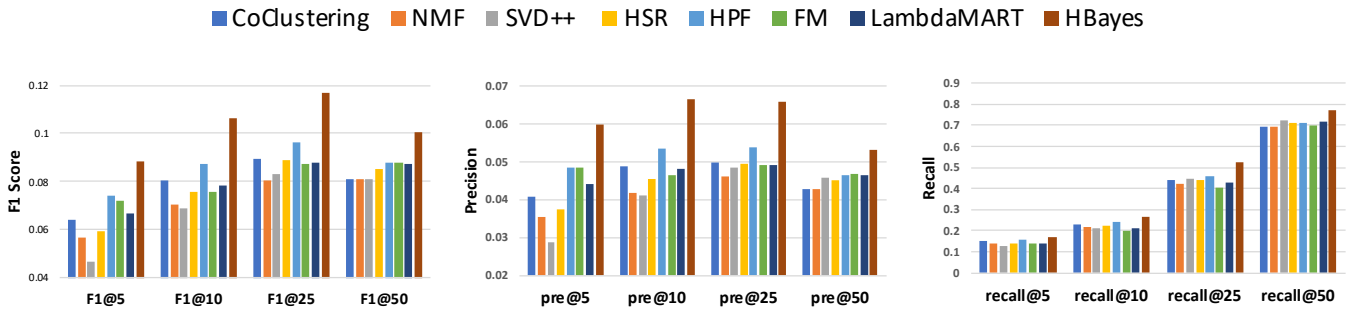

Figure 2: F1@K on *Apparel* data.

Figure 3: Precision@K on *Apparel* data.

Figure 4: Recall@K on *Apparel* data.

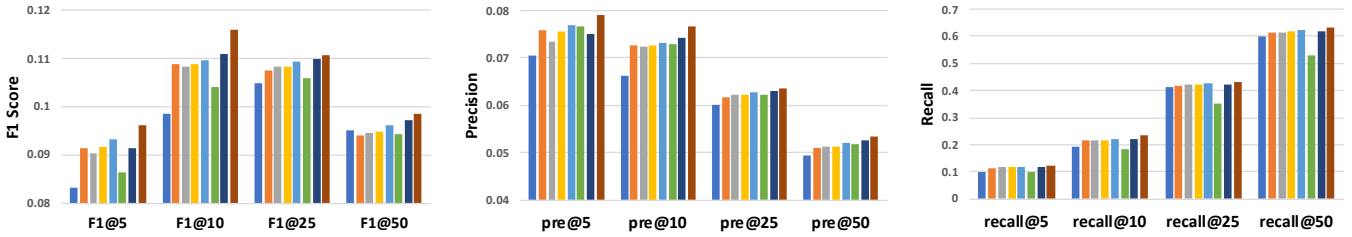

Figure 5: F1@K on *Music* data.

Figure 6: Precision@K on *Music* data.

Figure 7: Recall@K on *Music* data.

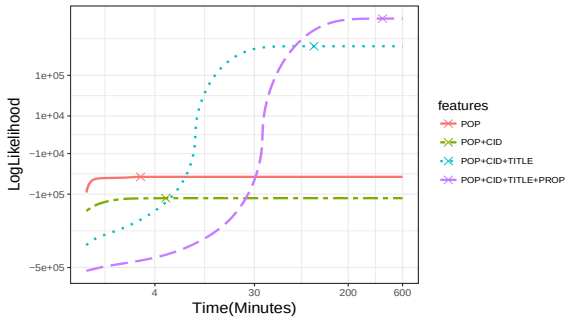

Figure 8: Training time under different feature combinations on e-commerce recommendations

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 3: NDCG on Last.fm recommendations

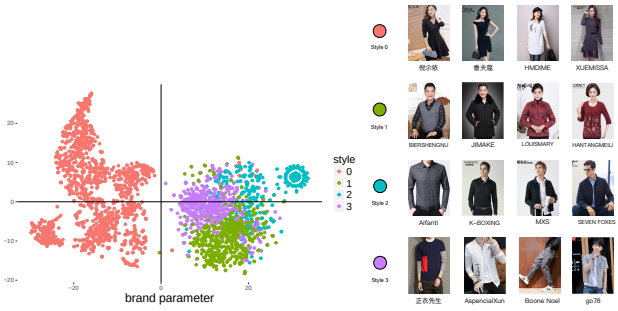

Figure 9: tSNE of four latent apparel style clusters

International Conference on Web Search and Data Mining (WSDM '17). ACM, New York, NY, USA, 445–453.

[2] Christopher M Bishop. 2006. Pattern recognition and machine learning (information science and statistics) springer-verlag new york. *Inc. Secaucus, NJ, USA* (2006).

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning.* ACM, 89–96.

[5] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

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

[7] Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters* 80 (2016), 150–156.

[8] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. 2017. FMA: A Dataset for Music Analysis. In *18th International Society for Music Information Retrieval Conference.*

[9] James M Dickey. 1983. Multiple hypergeometric functions: Probabilistic interpretations and statistical uses. *J. Amer. Statist. Assoc.* 78, 383 (1983), 628–637.

[10] Pinar Donmez, Krysta M Svore, and Christopher JC Burges. 2009. On the local optimality of LambdaRank. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.* ACM, 460–467.

[11] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE international conference on.* IEEE, 4–pp.

[12] Prem Gopalan, Jake M Hofman, and David M Blei. 2015. Scalable Recommendation with Hierarchical Poisson Factorization.. In *UAI*. 326–335.

[13] San Gultekin and John Paisley. 2014. A Collaborative Kalman Filter for Time-Evolving Dyadic Processes. In *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM '14)*. IEEE Computer Society, Washington, DC, USA, 140–149.

[14] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. 2009. Personalized Recommendation of Social Software Items Based on Social Relations. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. ACM, New York, NY, USA, 53–60.

[15] T Jaakkola and M Jordan. 1997. A variational approach to Bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, Vol. 82. 4.

[16] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning* 37, 2 (1999), 183–233.

[17] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[18] Yehuda Koren. 2010. Collaborative Filtering with Temporal Dynamics. *Commun. ACM* 53, 4 (April 2010), 89–97.

[19] Yiqun Liu, Junwei Miao, Min Zhang, Shaoping Ma, and Liyun Ru. 2011. How do users describe their information need: Query recommendation based on snippet click model. *Expert Systems with Applications* 38, 11 (2011), 13847 – 13856.

[20] P. Lops, M. de Gemmis, and G. Semeraro. 2011. *Content-based Recommender Systems: State of the Art and Trends*. 73.

[21] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[22] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[23] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

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

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 811–820.

[27] Håvard Rue, Sara Martino, and Nicolas Chopin. 2009. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the royal statistical society: Series b (statistical methodology)* 71, 2 (2009), 319–392.

[28] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295.

[29] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 259–266.

[30] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. in Artif. Intell.* 2009, Article 4 (Jan. 2009), 1 pages.

[31] Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: a review. *Social Network Analysis and Mining* 3, 4 (1 2013), 1113–1133. DOI: http://dx.doi.org/10.1007/s13278-013-0141-9

[32] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2015. Exploring Implicit Hierarchical Structures for Recommender Systems.. In *IJCAI*. 1813–1819.

[33] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2018. Exploring Hierarchical Structures for Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[34] Dawei Yin, Liangjie Hong, Zhenzhen Xue, and Brian D. Davison. 2011. Temporal Dynamics of User Interests in Tagging Systems. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI'11)*. AAAI Press, 1279–1285.

[35] Quan Yuan, Gao Cong, Kaiqi Zhao, Zongyang Ma, and Aixin Sun. 2015. Who, Where, When, and What: A Nonparametric Bayesian Approach to Context-aware Recommendation and Search for Twitter Users. *ACM Trans. Inf. Syst.* 33, 1, Article 2 (Feb. 2015), 33 pages.

[36] Yi Zhang and Jonathan Koren. 2007. 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*. ACM, 47–54.