CONTINUOUS PLSI AND SMOOTHING TECHNIQUES FOR HYBRID MUSIC RECOMMENDATION

Kazuyoshi Yoshii **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST)
{k.yoshii,m.goto}@aist.go.jp

ABSTRACT

This paper presents an extended probabilistic latent semantic indexing (pLSI) for hybrid music recommendation that deals with rating data provided by users and with content-based data extracted from audio signals. The original pLSI can be applied to collaborative filtering by treating users and items as discrete random variables that follow multinomial distributions. In hybrid recommendation, it is necessary to deal with musical contents that are usually represented as continuous vectorial values. To do this, we propose a continuous pLSI that incorporates Gaussian mixture models. This extension, however, causes a severe local optima problem because it increases the number of parameters drastically. This is considered to be a major factor generating "hubs," which are items that are inappropriately recommended to almost all users. To solve this problem, we tested three smoothing techniques: multinomial smoothing, Gaussian parameter tying, and artist-based item clustering. The experimental results revealed that although the first method improved nothing, the others significantly improved the recommendation accuracy and reduced the hubness. This indicates that it is important to appropriately limit the model complexity to use the pLSI in practical.

1. INTRODUCTION

The musical tastes of users of online music distribution services that provide millions of items are strongly influenced by the characteristics of the music automatically recommended by those services. Users often have difficulty retrieving unknown items they might like. In such case, users consider recommendations and get aware of what kinds of items are their favorites. When only popular items are always recommended, users are not exposed to items they might enjoy more and get used to enjoying only the "safe" recommendations. This in turn strengthens the tendency to recommend only popular items. In other words, there is a severe limitation in *serendipity* of music discovery. In fact, this negative-feedback phenomenon has been observed in many services based on collaborative filtering.

We aim to enhance the serendipity by transforming the passive experience in which users only receive "default" recommendations into an interactive experience in which users can freely customize (personalize) those recommendations. To achieve this, it is necessary to let users clearly understand and express their own musical tastes that are estimated as bases of making default recommendations. The conventional reasoning like "You like A, so you would like B because other users who like like A also like B" is a *relative* expression of musical tastes. We aim to obtain a *direct* expression of each user's musical tastes that is easy to use as a basis for interactive recommendation.

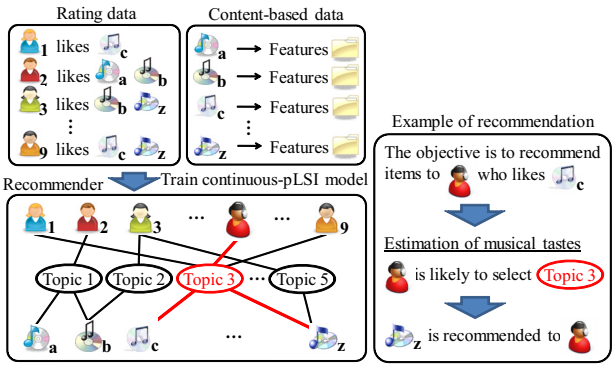

Figure 1. Hybrid recommender based on continuous pLSI.

A promising way to do this is to use probabilistic latent semantic indexing (pLSI) based on a multi-topic model, which has been originally used for document modeling [1]. The model includes latent variables corresponding to the concepts of topics. How likely a document and a word co-occur is predicted by stochastically associating each document and word with a topic. Documents and words that are strongly associated with the same topic are likely to occur jointly. The model can be applied to collaborative filtering based on rating histories by treating documents and words as users and items [2]. Given a user, we can predict how likely each item is purchased by estimating how likely the user chooses each topic. The musical tastes of users can be expressed as the strength of user-topic associations.

As shown in Figure 1, we propose continuous pLSI for hybrid recommendation that enhances serendipity by combining rating data with content-based data extracted from musical audio signals. Specifically, Gaussian mixture models (GMMs) are built into the collaborative filtering model of pLSI in order to address continuous vectorial data. Unlike the major collaborative methods relying on heuristics [3,4], the pLSI model can be extended in a consistent manner because it is flexible and has a theoretical basis.

The continuous pLSI, however, suffers from a serious local optima problem because a number of parameters linearly increases according to data size. This causes the *hub* phenomenon [5], in which specific items are almost always recommended to users regardless of their rating histories. Thus, the serendipity of recommendations is insufficient. Although a similar probabilistic model was proposed for genre classification [6], this problem was not addressed.

To solve this problem, we propose three smoothing techniques: multinomial smoothing, Gaussian parameter tying, and artist-based item clustering. The first technique is expected to avoid overfitting and the other two reduce the model complexity. We compared the effectiveness of these techniques experimentally.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 explains a model of the continuous pLSI. Section 4 describes the three smoothing techniques. Section 5 reports our experiments. Section 6 summarizes the key findings of this paper.

2. RELATED WORK

Music recommendation is an important topic today in the field of music information processing. Conventional studies on recommendation have been intended to deal with textual data (documents and words). In addition, many researchers have proposed various ideas to make the most of content-based data that is automatically extracted from musical audio signals. For example, Logan [7] proposed a content-based recommender based on the cosine distance between a user's favorite items and non-rated items. Magno and Sable [8] reported subjective experiments showing that a content-based recommender competes against Last.fm (a collaborative recommender) and Pandora (a recommender based on manual annotations) in terms of user satisfaction. These reports indicate the synergistic effect of integrating rating data with content-based data. Hybrid recommenders have been actively investigated recently. Celma *et al.* [9] used both content-based similarity and user profiles given in RSS feeds to choose suitable items. Tiemann *et al.* [10] integrated two weak learners (social and content-based recommenders) by using an ensemble learning method.

Another important issue is how to present recommendations to users. Donaldson and Knopke [11] visualized the relationships of recommended items in a two dimensional space. Lamere and Maillet [12] proposed a transparent and steerable interface for a recommender based on crowds of social tags. A common concept of these studies seems to be that users had better actively explore or control recommendations. This would result in enhanced serendipity.

The existence of hubs has recently been recognized as a serious problem. Interestingly, this problem was not reported in the field of text-based recommendation. In music recommendation and retrieval, GMMs are generally used to represent the distributions of acoustic features. Aucouturier *et al.* [5] pointed out that this kind of modeling tends to create hubs that are wrongly evaluated as similar to all other items. Berenzweig [13] concluded that the hub phenomenon is related to the curse of dimensionality. Chordia *et al.* [14] discussed content-based recommendation based on the Earth-Movers distance between GMMs of individual items. They empirically found that a homogenization method can improve performance [15]. Hoffman *et al.* [16] tried to solve this problem by using the hierarchical Dirichlet process (HDP) for modeling content-based data. Unlike the GMM, the HDP represents each item as a mixture of an unfixed number of Gaussians. The number is automatically adjusted to match the data complexity. In addition, the same set of Gaussians is used to model all items, with only the mixture weights varying from item to item. This is similar to Gaussian parameter tying.

3. CONTINUOUS PLSI

This section explains a continuous pLSI model and a training method suitable for efficient parallel processing.

3.1 Problem Statement

We define several symbols from a probabilistic viewpoint. Let $U = \{u_1, u_2, \cdots, u_{|U|}\}$ be the set of all users, where $|U|$ is the number of them, and let $V = \{v_1, v_2, \cdots, v_{|V|}\}$ be the set of all items, where $|V|$ is the number of them. Let u and v be *discrete* random variables respectively taking the values of one member of U and one member of V. Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{|V|}\}$ denote content-based data that is a set of D-dimensional feature vectors extracted from individual items. Let $\boldsymbol{x}$ be a *continuous* random variable in the D-dimensional space. Probabilistic distributions are represented as $p(\text{variable})$ or $p(\text{variable1}|\text{variable2})$, e.g., a discrete distribution $p(u)$ or a conditional continuous distribution $p(\boldsymbol{x}|u)$. For example, probabilities or probability densities are given by $p(u = u_i)$ or $p(\boldsymbol{x} = \boldsymbol{x}_j|u = u_i)$, which are simply described as $p(u_i)$ or $p(\boldsymbol{x}_j|u_i)$.

As to available rating data, we mainly assume implicit ratings such as purchase histories or listening counts, which are recorded automatically even when users do not explicitly express their preferences for individual items. In general, the number of implicit ratings tends to be much larger than that of explicit ratings. We thus think that the former are more suitable to probabilistic approaches because for them the sparseness problem is less serious.

The total available data (combinations of rating data and content-based data) is given by $\boldsymbol{O} = \{(u_{(1)}, v_{(1)}, \boldsymbol{x}_{(1)}), \cdots, (u_{(N)}, v_{(N)}, \boldsymbol{x}_{(N)})\}$, where $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)})$ $(1 \leq n \leq N)$ is a user-item-feature co-occurrence that user $u_{(n)}$ has purchased (viewed or listened to) item $v_{(n)}$ with feature $\boldsymbol{x}_{(n)}$ and N is the number of co-occurrences. Let $c(u, v)$ be the number of times that co-occurrence $(u, v, \boldsymbol{x})$ was observed. Obviously, $N = \sum_{u,v} c(u, v)$. An easy way to utilize explicit ratings (e.g., numerical rating scores such as the numbers of "stars" in an one-to-five scale rating system adopted by Amazon.com) is to set the value of $c(u, v)$ to one if a user u likes item v, i.e., the rating score is greater than a neutral score (three stars). Alternatively, we could use rating scores for weighting $c(u, v)$.

The final objective is to estimate the probabilistic distribution $p(v|u)$, which indicates how likely it is that user u likes item v. Recommendations are then made by ranking items not rated by user u in a descending order of $p(v|u)$.

3.2 Model Formulation

The graphical representation of a continuous pLSI model is shown in Figure 2. This is an extended version of three-way aspect models [17, 18] in which all variables are discrete. We assume that users, items, and features are conditionally independent through latent topics. In other words, once a latent topic is specified, there is no mutual information between three kinds of variables. Although this seems a strong assumption, it is a reasonable way to avoid the local optima problem. Introducing a dependency edge from items to features in order to model the real world accurately would increase the number of parameters drastically.

The pLSI model can explain the process generating co-occurrence $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)})$. Let $Z = \{z_1, \cdots, z_{|Z|}\}$ be a set of *topics*, where $|Z|$ is the number of them. Let z be a latent variable that takes the value of one of Z. Each topic can be regarded as a *soft* cluster that is simultaneously associated with users and items. That is, each user and each item stochastically belong to one of the topics. The model thus treats triplet $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)})$ as incomplete data that is latently associated with $z_{(n)} \in Z$. The complete data is given by quartet $(u_{(n)}, v_{(n)}, \boldsymbol{x}_{(n)}, z_{(n)})$. An interpretation of the generative process is that user $u_{(n)}$ stochastically selects topic $z_{(n)}$ according to his or her taste $p(z_{(n)}|u_{(n)})$, and $z_{(n)}$ stochastically generates item v and its features $\boldsymbol{x}$ in turn. For convenience, we let $\boldsymbol{S}$ be $\{z_{(1)}, \cdots, z_{(n)}\}$.

A unique feature of the continuous pLSI is that $p(\boldsymbol{x}|z)$ is modeled with a Gaussian mixture model (GMM) in order to deal with continuous observation $\boldsymbol{x}$. Let M be the number of mixtures (Gaussian components). Each topic $z_k \in Z$ has a GMM defined by the mixing proportions of Gaussians $\{w_{k,1}, \cdots, w_{k,M}\}$ and their means and covariances $\{\boldsymbol{\mu}_{k,1}, \cdots, \boldsymbol{\mu}_{k,M}\}$ and $\{\boldsymbol{\Sigma}_{k,1}, \cdots, \boldsymbol{\Sigma}_{k,M}\}$. As in the original pLSI, $p(u)$, $p(z|u)$, and $p(v|z)$ are multinomial distributions. We practically use an equivalent definition of the model obtained by focusing on $p(z)$, $p(u|z)$, and $p(v|z)$. The parameters of these multinomial distributions are simply given by (conditional) probability tables of target variables. Let $\boldsymbol{\theta}$ be the set of all parameters of $|Z|$ GMMs and $|Z|(1 + |U| + |V|)$ multinomial distributions.

3.3 Model Training

The training method we explain here uses the Expectation-Maximization (EM) algorithm [19] and is a natural extension of previous methods [17, 18] (c.f., discrete HMM v.s. continuous HMM). Instead of maximizing the incomplete log-likelihood, $\log p(\boldsymbol{O})$, the EM algorithm maximizes the expected complete log-likelihood $E_{\boldsymbol{S}}[\log p(\boldsymbol{S}, \boldsymbol{O})]$ iteratively, where $E_z[f(z)]$ means an expected value of function $f(z)$ with respect to $p(z)$; $E_z[f(z)] = \sum_z p(z) f(z)$.

The complete likelihood of $(u, v, \boldsymbol{x}, z)$ is given by

$$p(u, v, \boldsymbol{x}, z) = p(z)p(u|z)p(v|z)p(\boldsymbol{x}|z). \quad (1)$$

This can be easily calculated for given observations when parameters $\boldsymbol{\theta}$ are obtained.

In the E-step we define a Q function as

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{\text{current}}) = E_{\boldsymbol{S}}[\log p(\boldsymbol{S}, \boldsymbol{O})] \quad (2)$$

$$= \sum_{u,v} c(u, v) \sum_z p(z|u, v, \boldsymbol{x}) \log p(u, v, \boldsymbol{x}, z), \quad (3)$$

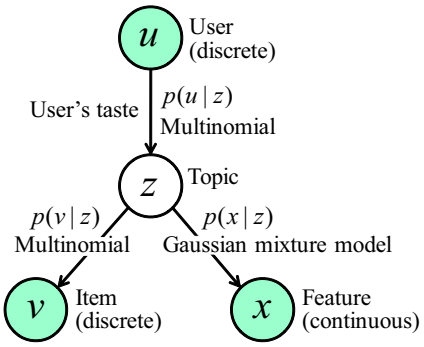

Figure 2. Graphical representation of continuous pLSI.

where $p(z|u, v, \boldsymbol{x})$ is a posterior distribution of latent variable z and can be calculated by using the current parameters $\boldsymbol{\theta}_{\text{current}}$ as follows:

$$p(z|u, v, \boldsymbol{x}) = \frac{p(u, v, \boldsymbol{x}, z)}{\sum_z p(u, v, \boldsymbol{x}, z)}. \quad (4)$$

In the M-step we update the current parameters by maximizing Eqn. (3). Note that $\log p(u, v, \boldsymbol{x}, z)$ can be decomposed into $\log p(z) + \log p(u|z) + \log p(v|z) + \log p(\boldsymbol{x}|z)$. This means that the parameters of each distribution can be updated independently. To update $p(z)$, for example, we only focus on a term related to $p(z)$ as follows:

$$Q_{p(z)} = \sum_{u,v} c(u, v) \sum_z p(z|u, v, \boldsymbol{x}) \log p(z). \quad (5)$$

Using a Lagrange multiplier λ for a constraint of probability standardization, we define a new function $F_{p(z)}$ as

$$F_{p(z)} = Q_{p(z)} + \lambda \left(1 - \sum_z p(z)\right). \quad (6)$$

We then calculate the differential of Eqn. (6) with respect to $p(z)$ and set it to zero as follows:

$$\frac{\partial F_{p(z)}}{\partial p(z)} = \frac{1}{p(z)} \sum_{u,v} c(u, v) p(z|u, v, \boldsymbol{x}) - \lambda \equiv 0. \quad (7)$$

The updated distribution $p(z)$ can be obtained by

$$p(z) = \frac{\sum_{u,v} c(u, v) p(z|u, v, \boldsymbol{x})}{\sum_{u,v,z} c(u, v) p(z|u, v, \boldsymbol{x})}. \quad (8)$$

The other two multinomial distributions $p(u|z)$ and $p(v|z)$ can be similarly updated as follows:

$$p(u|z) = \frac{\sum_v c(u, v) p(z|u, v, \boldsymbol{x})}{\sum_{u,v} c(u, v) p(z|u, v, \boldsymbol{x})}, \quad (9)$$

$$p(v|z) = \frac{\sum_u c(u, v) p(z|u, v, \boldsymbol{x})}{\sum_{u,v} c(u, v) p(z|u, v, \boldsymbol{x})}. \quad (10)$$

To update continuous distribution $p(\boldsymbol{x}|z)$, we focus on

$$Q_{p(\boldsymbol{x}|z)} = \sum_{u,v} c(u, v) \sum_z p(z|u, v, \boldsymbol{x}) \log p(\boldsymbol{x}|z) \quad (11)$$

$$= \sum_{u,v} c(u, v) \sum_{k=1}^{K} p(z_k|\cdot) \log \sum_{m=1}^{M} p(y_{k,m}) p(\boldsymbol{x}|z_k, y_{k,m}), \quad (12)$$

where to improve legibility we wrote $p(z|u, v, \boldsymbol{x})$ as $p(z|\cdot)$.

$y_k \in \{y_{k,1}, \cdots, y_{k,M}\}$ is a latent variable that indicates which Gaussian in the GMM of topic z_k generates $\boldsymbol{x}$. $p(y_k)$ represents a probability distribution over M Gaussians, i.e., $p(y_{k,m}) = w_{k,m}$, and $p(\boldsymbol{x}|z_k, y_{k,m})$ is the likelihood that feature $\boldsymbol{x}$ is generated from a Gaussian indicated by $y_{k,m}$. Because the logarithmic operation for the summation makes Eqn. (12) hard to maximize directly, we focus on the expected value of $Q_{p(\boldsymbol{x}|z)}$ with respect to y_k:

$$E_{y_k}[Q_{p(\boldsymbol{x}|z)}] = \sum_{u,v} c(u,v) \sum_{k=1}^{K} p(z_k|\cdot) \sum_{m=1}^{M} p(y_{k,m}|\boldsymbol{x}, z_k) \Big(\log w_{k,m} + \log \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{k,m}, \boldsymbol{\Sigma}_{k,m}) \Big), \quad (13)$$

where $p(y_{k,m}|\boldsymbol{x}, z_k)$ is a posterior probability given by

$$p(y_{k,m}|\boldsymbol{x}, z_k) = \frac{p(y_{k,m}) p(\boldsymbol{x}|z_k, y_{k,m})}{\sum_{m=1}^{M} p(y_{k,m}) p(\boldsymbol{x}|z_k, y_{k,m})}. \quad (14)$$

To obtain optimized $w_{k,m}$, we define the following function by introducing a Lagrange multiplier β:

$$F_{w_k} = E_{y_k}[Q_{p(\boldsymbol{x}|z)}] + \beta \left(1 - \sum_{m=1}^{M} w_{k,m} \right). \quad (15)$$

Calculating the partial partial differential of Eqn. (15) with respect to $w_{k,m}$ and setting it to zero, we obtain

$$w_{k,m} = \frac{\sum_{u,v} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k)}{\sum_{m=1}^{M} \sum_{u,v} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k)}. \quad (16)$$

Setting the partial differential of Eqn. (13) to zero, the mean and variance $\boldsymbol{\mu}_{k,m}$ and $\boldsymbol{\Sigma}_{k,m}$ are obtained by

$$\boldsymbol{\mu}_{k,m} = \frac{\sum_{u,v} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k) \boldsymbol{x}}{\sum_{u,v} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k)}, \quad (17)$$

$$\boldsymbol{\Sigma}_{k,m} = \frac{\sum_{u,v} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k) (\boldsymbol{x} - \boldsymbol{\mu}_{k,m})^2}{\sum_{u,v} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k)}. \quad (18)$$

Given a user u_i, recommendations are made by evaluating $p(v|u_i) = \sum_z p(v|z) p(z|u_i)$, where $p(z|u_i)$ is proportional to $p(z)p(u_i|z)$ and indicates the musical tastes of user u: how likely it is user u_i selects (likes) topic z.

3.4 MapReducing EM Algorithm

Computational efficiency, a very important issue in music recommendation when the database and model become large, is especially critical when used data cannot be loaded on the memory of a single machine. Elegant implementations, however, have scarcely been addressed.

A remarkable advantage of pLSI-based recommenders is that we can easily implement them in parallel processing environments that consist of multiple machines such as clusters. Google News, for example, uses a distributed computation framework called MapReduce [20].

We can implement the continuous pLSI by using MPI or Hadoop [21]. Suppose we have $G_U G_V$ machines (CPUs). Let $\{U_1, \cdots, U_{G_U}\}$ and $\{V_1, \cdots, V_{G_V}\}$ be exclusive sets of users and items, where $U_1 \cap \cdots \cap U_{G_U} = U$ and $V_1 \cap \cdots \cap V_{G_V} = V$. To update $p(z)$, for example, we calculate

$$c_z(U_i, V_j) = \sum_{u \in U_i, v \in V_j} c(u,v) p(z|u,v,\boldsymbol{x}). \quad (19)$$

This can be separately calculated in each machine. To calculate $p(z|u,v,\boldsymbol{x})$, we need only $p(z)$, $p(u|z)$ $(u \in U_i)$, $p(v|z)$ $(v \in V_j)$, and $p(\boldsymbol{x}|z)$. The number of parameters of these distributions is much smaller than the total number of parameters. Finally, we can get an integrated result by

$$p(z) \propto \sum_{1 \le i \le G_U, 1 \le j \le G_V} c_z(U_i, V_j). \quad (20)$$

4. SMOOTHING TECHNIQUES

To avoid overfitting, one needs to use appropriate smoothing techniques. In our study, we use three techniques to improve accuracy and reduce hubness: multinomial smoothing, Gaussian parameter tying, and artist-based item clustering. The first relaxes the excessive inclination of multinomial parameters, and the others limit model complexity.

4.1 Multinomial Smoothing

We add a conjugate prior called a Dirichlet distribution to a Q function as a regularization term. To estimate $p(z)$, for example, we consider the following function:

$$Q'_{p(z)} = Q_{p(z)} + \mathrm{Dir}(\boldsymbol{\alpha}), \quad (21)$$

where $\boldsymbol{\alpha}$ is a set of K parameters of a Dirichlet distribution. This results in the additive smoothing method. We set all parameters to 1.0001. Maximizing $Q'_{p(z)}$, we get

$$p(z) = \frac{\sum_{u,v} c(u,v) p(z|u,v,\boldsymbol{x}) + \alpha - 1}{\sum_z \left(\sum_{u,v} c(u,v) p(z|u,v,\boldsymbol{x}) + \alpha - 1 \right)}. \quad (22)$$

The updating formulas of the other multinomial distributions are similarly given by

$$p(u|z) = \frac{\sum_v c(u,v) p(z|u,v,\boldsymbol{x}) + \alpha - 1}{\sum_u \left(\sum_v c(u,v) p(z|u,v,\boldsymbol{x}) + \alpha - 1 \right)}, \quad (23)$$

$$p(v|z) = \frac{\sum_u c(u,v) p(z|u,v,\boldsymbol{x}) + \alpha - 1}{\sum_v \left(\sum_u c(u,v) p(z|u,v,\boldsymbol{x}) + \alpha - 1 \right)}. \quad (24)$$

4.2 Gaussian Parameter Tying

We force all GMMs to share the same set of Gaussians and differ from each other in the mixing proportions of those Gaussians. In the context of HMMs, this is called a tied mixture model. The new updating formulas are given by

$$\boldsymbol{\mu}_{k,m} = \frac{\sum_{u,v,m} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k) \boldsymbol{x}}{\sum_{u,v,m} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k)}, \quad (25)$$

$$\boldsymbol{\Sigma}_{k,m} = \frac{\sum_{u,v,m} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k) (\boldsymbol{x} - \boldsymbol{\mu}_{k,m})^2}{\sum_{u,v,m} c(u,v) p(z_k|\cdot) p(y_{k,m}|\boldsymbol{x}, z_k)}. \quad (26)$$

4.3 Artist-based Item Clustering

We replace *item-based* distribution $p(v|z)$ with *artist-based* distribution $p(a|z)$, where variable a represents one of the artists in the database. Let A be a set of items sung by artist a, That is, let all items be grouped according to their artist names. We train an *artist-based* model for users, artists, and features by iteratively updating $p(a|z)$ as follows:

$$p(a|z) = \frac{\sum_{u,v \in A} c(u,v) p(z|u,v,\boldsymbol{x})}{\sum_{u,v} c(u,v) p(z|u,v,\boldsymbol{x})}. \quad (27)$$

Score	5	4	3	2	1
Counts	**5336**	**1458**	457	211	333
Ratio	68.5%	18.7%	5.86%	2.71%	4.27%

Table 1. Distribution of rating scores.

To recommend items rather than artists, we then construct an *item-based* model by replacing $p(a|z)$ with $p(v|z)$. To do this, we use an incremental training method [18] that re-estimates a distribution of unknown items $p(v|z)$ without affecting other trained distributions $p(z)$, $p(u|z)$,$(\boldsymbol{x}|z)$:

$$p(v|z) = \frac{\sum_u c(u,v) \frac{p(z)p(u|z)p(\boldsymbol{x}|z)}{\sum_z p(z)p(u|z)p(\boldsymbol{x}|z)}}{\sum_{u,v} c(u,v) \frac{p(z)p(u|z)p(\boldsymbol{x}|z)}{\sum_z p(z)p(u|z)p(\boldsymbol{x}|z)}}. \quad (28)$$

5. EVALUATION

We experimentally evaluated the continuous pLSI in terms of accuracy and hubness by using various combinations of the smoothing techniques.

5.1 Data

The music items we used were Japanese songs recorded in single CDs that were ranked in weekly top-20 sales rankings from Apr. 2000 to Dec. 2005. To use these items, we need real implicit ratings $c(u, v)$ such as purchase histories and listening counts, but most online services do not release such data to the public. We therefore instead collected explicit ratings (numbers of "stars" ranging from one to five) from Amazon.co.jp by using official APIs [22] that let us download almost all the information available from Amazon.co.jp [22]. For reliable evaluation, we excluded users who had rated fewer than two items and excluded items that had been rated less than two times. As a result, $|U|$ was 1872 and $|V|$ was 1400. The number of artists was 471. If a rating score given to item v_j by user u_i was greater than three (the neutral score), we set $c(u_i, v_j)$ to the score. Otherwise, we set $c(u_i, v_j)$ to zero. In other words, we considered only positive ratings. A similar approach has been used previously [23]. Note that, as shown in Table 1, the distribution of rating scores was strongly skewed. The density of 6794 positive ratings (scores 4 and 5) was 0.259% in the user-item co-occurrence table.

With regard to the content-based data, we focused on vocal features because all the items included singing voices that strongly affected the musical tastes of users. To extract these features from polyphonic audio signals, we used a method proposed by Fujihara *et al.* [24]. We calculated a 13-dimensional feature vector at each frame where singing voices were highly likely to be included, concatenated the mean and variance of the feature vectors in each item into a 26-dimensional vector, and then used principal component analysis to compress the dimensionality to 20 ($D = 20$).

5.2 Protocols

To test all combinations of the three smoothing techniques, we prepared eight models of the continuous pLSI. For convenience, throughout Section 5, the multinomial smoothing, Gaussian parameter tying, and item clustering are respectively called SM1, SM2, and SM3. The number of latent variables was 256 ($|Z| = 256$). Although the number of mixtures was 32, when SM1 was disabled it was set to 1 in order to avoid overfitting.

	Disabled	SM1	SM2	SM1&2
Disabled	4.65	4.29	6.18	6.57
SM3	7.10	6.72	**19.4**	19.3

Table 2. Expected utility of recommendations: Higher scores indicate better performance.

	Disabled	SM1	SM2	SM1&2
Disabled	5.94	5.81	**6.39**	6.36
SM3	5.98	5.81	6.36	6.34

Table 3. Entropy of recommendations: Higher scores indicate better performance (fewer hubs).

We conducted 10-fold cross validation by splitting the positive explicit ratings into ten groups. Nine groups were used for making recommendations with the eight models. The other group was considered to be not observed and was used for evaluating the recommendations.

5.3 Measures

Recommendation results given as ranked lists of items were evaluated in terms of accuracy and hubness.

To calculate accuracy, we used the expected utility of a ranked list [25], which for each user is defined as

$$R_u = \sum_{r=1}^{|V|-\#(\text{rated items})} \frac{\max(\text{score}_{u,r} - 3, 0)}{2^{(r-1)/(\gamma-1)}}, \quad (29)$$

where $\text{score}_{u,r}$ is the rating score that user u *actually* gave the r-th ranked item although the item was considered a non-rated item (the score was hidden) in model training. When $\text{score}_{u,r}$ was not available, its value was set to 3. γ is a viewing half-life based on the assumption that the probability that a user views an r-th ranked item is twice the probability that the user views an $(r + \gamma)$-th ranked item. We set γ to 5 as in the literature [25]. R_u was not sensitive to the value of γ. The total score is given by

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^{\max}} \ (0 \le R \le 100), \quad (30)$$

where $R_u^{\max}$ is the maximum achievable utility if all items with available scores given by user u had been at the top of the ranked list in order of those scores. Basically, higher values indicate better performance, but note that the probability of recommending known items is high.

We propose the following hubness measure based the entropy of recommendations:

$$H = -\sum_{j=1}^{|V|} \frac{t(j)}{|U|} \log\left(\frac{t(j)}{|U|}\right), \quad (31)$$

where $t(j)$ is the number of times that item v_j was recommended with the highest (top 1) ranking. A larger H (higher entropy) indicates a smaller bias in how many times each item is recommended.

5.4 Results

As shown in Table 2, the accuracies of recommendations were greatly improved by using SM3. This can be explained from two aspects: the relationship between items and features and that between items and users. First, the items of each artist tend to be similar to each other in their musical features. Second, most users of Amazon.co.jp tend to like any of the items of the few artists they like. This would be a common tendency of the users of many online music distribution services. Therefore, SM3 reduced the complexity of the model while preserving almost all the information of the rating data.

SM2 improved the accuracy of recommendations made regardless of the combinations in which it was used. Interestingly, recommendations obtained by jointly using SM2 and SM3 were much more accurate than those made when these techniques were used independently. SM1, on the other hand, slightly reduced the accuracy because it is based on additive smoothing. It is known that its approximation errors are larger than those of the other smoothing methods such as the Good-turing method.

Table 3 shows hubness of recommendations. SM2 significantly reduced the hubness while the SM1 and SM3 had no gains. This is consistent with the results reported by Hoffman *et al.* [16], who found that HDP and vector quantization (VQ) did not produce many hubs. VQ can be considered as a *hard* clustering version of the tied GMM, which is a *soft* clustering model.

We conclude that combining SM2 and SM3 is the best approach to improving performance. In our experiments, it yielded accuracy comparable with that of conventional methods of collaborative filtering.

6. CONCLUSION

This paper has presented a continuous-pLSI-based model for hybrid music recommendation. The model uses GMMs to represent distributions of acoustic features extracted from musical audio signals. As in the original pLSI, users and items are assumed to follow multinomial distributions. We developed an algorithm for parameter estimation and implemented it in a parallel processing environment. Experimentally testing the abilities of three smoothing techniques —multinomial smoothing, Gaussian parameter tying, and artist-based item clustering—, we found that using the second and third techniques to adjust model complexity significantly improved the accuracy of recommendations and that the second technique could also reduce hubness.

In the future, we plan to introduce conjugate priors of all distributions (GMMs and multinomial distributions) into the continuous pLSI to enable full Bayesian estimation. Extending latent Dirichlet allocation (LDA) [23] and HDP-LDA [26] are worth considering.

Acknowledgement: This study was partially supported by Grant-in-Aid for Young Scientists (Start-up) 20800084.

7. REFERENCES

[1] T. Hofmann and J. Puzicha. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[2] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI*, pages 688–693, 1999.

[3] P. Resnick, N. Iacovou, M. Sushak, and P. Bergstrom. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.

[4] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, pages 21–23, 2005.

[5] J.-J. Aucouturier, F. Pache, and M. Sandler. The way it sounds: Timbre models for analysis and retrieval of polyphonic music signals. *IEEE Transactions of Multimedia*, 7(6):1028–1035, 2005.

[6] P. Ahrendt, J. Larsen, and C. Goutte, Co-occurrence models in music genre classification. In *MLSP*, pages 24–252, 2005.

[7] B. Logan. Music recommendation from song sets. In *ISMIR*, pages 425–428, 2004.

[8] T. Magno and C. Sable. A comparison of signal-based music recommendation to genre labels, collaborative filtering, musicological analysis, human recommendation, and random baseline. In *ISMIR*, pages 161–166, 2008.

[9] O. Celma, M. Ramírez, and P. Herrera. Foafing the music: A music recommendation system based on RSS feeds and user preferences. In *ISMIR*, pages 464–467, 2005.

[10] M. Tiemann, S. Pauws, and F. Vignoli. Ensemble learning for hybrid music recommendation. In *ISMIR*, pages 179–180, 2007.

[11] J. Donaldson and I. Knopke. Music recommendation mapping and interface based on structural network entropy. In *ISMIR*, pages 181–182, 2007.

[12] P. Lamere and F. Maillet. Creating transparent, steerable recommendations. Late breaking in ISMIR, 2008.

[13] A. Berenzweig. *Anchors and Hubs in Audio-based Music Similarity*. PhD thesis, Columbia University, 2007.

[14] P. Chordia, M. Godfrey, and A. Rae. Extending content-based music recommendation: The case of Indian classical music. In *ISMIR*, pages 571–576, 2008.

[15] M. Godfrey and P. Chordia. Hubs and homogeneity: improving content-based music modeling. In *ISMIR*, pages 307–312, 2008.

[16] M. Hoffman, D. Blei, and P. Cook. Content-based musical similarity computation using the hierarchical Dirichlet process. In *ISMIR*, pages 349–354, 2008.

[17] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *UAI*, pages 437–444, 2001.

[18] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally-trainable probabilistic generative model. *IEEE Transactions on ASLP*, 16(2):435–447, 2008.

[19] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society*, B-39(1):1–38, 1977.

[20] A. Das, M. Datar, and A. Garg. Google news personalization: Scalable online collaborative filtering. In *WWW*, pages 271–280, 2007.

[21] Hadoop. http://hadoop.apache.org/core.

[22] Amazon web services. http://aws.amazon.com.

[23] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Machine Learning Research*, 3:993–1022, 2003.

[24] H. Fujihara, T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. Singer identification based on accompaniment sound reduction and reliable frame selection. In *ISMIR*, pages 329–336, 2005.

[25] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[26] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. *JASA*, 101(476):1566-1581, 2006.