You Are What You Consume: A Bayesian Method for Personalized Recommendations

Konstantinos Babas
Electronic and Computer Engineering
Technical University of Crete
Chania, Greece
kbabas@isc.tuc.gr

Georgios Chalkiadakis
Electronic and Computer Engineering
Technical University of Crete
Chania, Greece
gchalkiadakis@isc.tuc.gr

Evangelos Tripolitakis
Electronic and Computer Engineering
Technical University of Crete
Chania, Greece
vtripolitakis@isc.tuc.gr

ABSTRACT

In this paper, we propose a novel Bayesian approach for personalized recommendations. In our approach, we model *both* user preferences *and* items under recommendation as *multivariate Gaussian* distributions; and make use of *Normal-Inverse Wishart* priors to model the recommendation agent beliefs about user types. We employ a lightweight agent-user interaction process, during which the user is presented with and asked to rate a small number of items. We then interpret these ratings in an innovative way, using them to guide a Bayesian updating process that helps us both capture a user's current mood, and maintain her overall user type. We produced several variants of our approach, and applied them in the movie recommendations domain, evaluating them on data from the MovieLens dataset. Our algorithms are shown to be competitive against a state-of-the-art method, which nevertheless requires a minimum set of ratings from various users to provide recommendations—unlike our entirely personalized approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

Keywords

Bayesian methods; personalized recommendations

1. INTRODUCTION AND RELATED WORK

Making decisions on what movie to see, what kind of music to listen to, or what book to read can be a hard problem when one is presented with a multitude of choices. Research in recommendation systems attempts to understand a user's needs, preferences and mood, and help her make a decision. Typically, however, most recommendation methods require pre-training on data gathered from many users, who are classified according to their inferred similarity in preferences. Moreover, many approaches require much user involvement in a potentially cumbersome, lengthy interaction with the system. This can impose serious limitations to the usability of a recommendation system, especially when a user wishes to make a fast decision that is also dependent on her current mood.

In more detail, most established recommendation systems exploit user ratings over a large number of items via the use of *collaborative filtering (CF)*, *content-based* methods, or a combination thereof. *Content-based methods* usually make recommendations by analyzing the content of textual information about an item. On the other hand, *collaborative filtering* is based on the assumption that, if two users rate n items similarly, they will probably rate other items similarly as well. So, collaborative filtering techniques use ratings from a specific user on some items (e.g., movies), and combine them with ratings of other users on a set of items in order to infer about the ratings of that user on unrated items.

Some examples of CF and content-based systems are those of [2, 18, 9, 25]. Many such systems find application in the movie recommendations domain. For instance, *Content-Boosted Collaborating Filtering* [18] uses a content-based predictor, enhanced with CF, to provide personalized movie recommendations; Debnath *et al.* [9] use a *collaborative social network graph* to assign weights on attributes of content-based recommendations, depending on their perceived importance; while Barbieri *et al.* [2] intertwine Bayesian methods with CF, in order to group users into "communities" based on their rating patterns. Another interesting application of CF methods is introduced in [25], which presents an interactive story generation system that employs *probabilistic principle component analysis* and *non-negative matrix factorization*, in order to compose a personalized story according to user storytelling preferences.

Now, utility theory-inspired *preference elicitation (PE)* techniques have also been tried out in various recommendation domains (see, e.g., [6, 23]). PE tries to collect user preferences in order to construct the user's *utility function*. To do so, most Preference Elicitation techniques set queries to the user asking her to evaluate, order, or constrain potential system outcomes; while others attempt to translate a user's interaction with the system to preferences. There also exist systems which use *semantics* and *social choice* or *voting theory* for making recommendations [24, 19, 12, 17]. For instance, Szomszor *et al.* [24] use movies *folksonomy* to enrich the current knowledge with descriptions of movies and interests of users; while [12] presents a website which uses *trust* in *social networks* for making movie recommendations. The website generates predictive personalized ratings based on a *trust inference algorithm*, which employs "trusted users" ratings in order to calculate the rating of a user for that movie. Others employ *WWW questionnaires* and *Bayesian networks* to model the collected data [22].

The work most relevant to ours is perhaps that of [16] and [26], which model users and items using a common representation. Both approaches use vectors for modelling purposes. In some detail, Langseth and Nielsen [16] propose a CF technique which uses a

RecSys '13, October 12 - 16 2013, Hong Kong, China

http://dx.doi.org/10.1145/2507157.2507158.

Bayesian network to support the inference procedure. They model items using *feature vectors*, and users using vectors which describe user's liking for each item feature. These vectors are incorporated in the Bayesian network in order to predict user's ratings about unrated items using her previous ratings on other items. On the other hand, Zhang and Koren [26] introduce a Bayesian hierarchical model for content-based recommendations. They model each user as a k-*dimensional vector* sampled randomly from a Gaussian distribution, and items as k-dimensional *feature vectors*. They then incorporate data *from all users* in the hierarchical model in order *to predict a label-rating* of an item for a specific user.

The recommendation technique we present in this paper differs to all aforementioned approaches in many ways. First of all, we neither set questions to the user, nor use textual information regarding an item so as to elicit user preferences. Moreover, we do not rely on any kind of user classification or other users' inferred preferences, but attempt to fine-tune recommendations over time for each specific individual. That is, we progressively build a user type for each individual, which gradually converges to the real one. In contrast to most social networks, social choice-inspired, and CF approaches, we *do not* attempt to estimate user ratings; but just recommend the item which matches the user preferences more closely.

Approach Overview and Contributions.

Indeed, in this paper we design and evaluate a *Bayesian recommendation agent*, based on a *simple*, *fast*, and *easy-to-use* elicitation and modelling process; and apply it in the movie recommendations domain. By means of this process, our agent is able to recommend a user items (e.g., movies) that best fit *both* her long-time preferences and current mood. To achieve its objective, the agent maintains *item types* (summarizing item characteristics); and *user types* (corresponding to modeled user preferences). Crucially, these are *both* represented as *multivariate Gaussian distributions* over ranges of values (ratings), describing the degree to which an item is composed of certain attributes (e.g., movie genres); and the degree to which a user likes the particular attributes, respectively. This allows for the establishment of a correspondence between user and item types: *intuitively, a user model is viewed by the agent as being an amalgamation of items this user likes*. We term this the *"you are what you consume"* idea. The employment of *Normal-Inverse Wishart (NIW)* conjugate priors to model agent beliefs about types guarantees that these beliefs can be readily updated.

Now, to learn about a specific user's current mood, we first *only* ask her to rate a small number of *demonstrative items* (*"demos"* for short; e.g., movie trailers), also represented by multivariate Gaussians (just like other items), and which are selected based on the current agent beliefs about user type. Each rating is treated as a unit, that is, we do not ask the user to rate different item attributes. We then employ these ratings in an innovative way, interpreting them as an indication of the *number of samples* to take from the corresponding demo type; i.e., to indicate a demo's "weight", or, to put otherwise, the "degree of correspondence" that the specific demo has with user preferences and current mood. This is reasonable, as a user's rating of, e.g., trailers, intuitively reflects her current mood—which is, nevertheless, not independent of her long-term tastes. The agent takes this into account: each demo presented was already selected according to perceived *user type*—thus, long-term user preferences *are* incorporated in the choice of demos.

The samples thus collected are then utilized by a Bayesian updating process that infers a *temporary demo-based user type*, given demo ratings. Having the demo-based type, we can then search for the best possible match (employing a *KL-divergence* metric) with an *item* to actually recommend—and which is *not* necessarily one of the demos shown earlier. The selected item is then presented to the user, who *rates* it, leading to an update of her overall *user type*.

The overall recommendation process effectively corresponds to a *Bayesian exploration* approach in this domain—since, when selecting demos or items to present to the user, our method optimises with respect to Bayes-updated beliefs, rather than taking explicit "exploration" actions. Moreover, intuitively our method allows for a better user experience, and implicitly helps the user "discover" more about her own real preferences, as it does not rigidly disallow suggestion possibilities or prune vast parts of the search space (as methods relying on explicit user statements about their preferences or ratings probably would), but only makes suggestions given its probabilistic beliefs regarding user preferences. For interest, we devised and tested certain additional *variants* of our approach, which employ alternative (Bayesian and non-Bayesian) exploration methods when selecting a demo or item to present to the user. One of these variants most probably constitutes the first adaptation of the well-known *Value of Perfect Information* Bayesian exploration heuristic [8, 7] in the recommendations domain.

Though to some extent conceptually straightforward, to the best of our knowledge an approach such as ours has not been used before in the literature. It is a *simple* yet *generic* approach that combines elements from various techniques. The idea of modelling the types of *both* the user and the object under recommendation *by a probability distribution of the same form* is a novel one. Representing users and items as *complete probability distributions* over a collection of features, instead of as, e.g., vectors of point-values corresponding to specific features' weights, enables the implicit inference of *latent* features, or *hidden and otherwise unrepresentable feature mixtures, combinations, and interconnections*. Moreover, our translation of "demo" ratings into weights for guiding the sampling process used during Bayesian updating is innovative, and allows us to build a *temporary user type* that captures both current user mood and long-time preferences simultaneously.

We evaluated our approach in the movie recommendations domain. As a note, the use of Multivariate Gaussians and Normal-Inverse Wishart distributions to model types and corresponding beliefs in this particular domain is novel, as is the use of "trailer" ratings for inference purposes. Our experimental results are highly encouraging, demonstrating as they do that the agent quickly learns to recommend movies that receive high user ratings. In particular, our agent manages to exhibit performance that is comparable to that of a popular, state-of-the-art, collaborative filtering-based method for movie recommendations. Importantly, it does so without the need of looking at pre-gathered/pre-processed data involving the current user or her peers. As such, the ever-present "cold start" problem, a constant challenge to CF methods due to ratings scarcity (see, e.g., [3] for a discussion), does not apply to our personalized method—since it does not employ any ratings of other users whatsoever, and does not aggregate past ratings to predict future ones.

2. BACKGROUND: BAYESIAN UPDATING

A key component of our agent is performing probabilistic inference regarding the types of the system's users, by means of *Bayesian updating*. In many cases, there exists the need of estimating the parameters of an unknown probability distribution—a *model*; and Bayesian updating can be employed to this end. Key to a computationally feasible application of Bayesian updating, is the use of *Bayes rule*, in conjunction with proper *conjugate priors* describing our beliefs about the model [10].

In some detail, in the face of new data (observations) regarding the unknown model, Bayes rule takes into account a *prior* distribution (reflecting our belief on the values of the model's parameters),

a *likelihood function*, and a marginal probability, in order to derive a *posterior*. The likelihood function brings together the prior and the observations and follows the form of the model. The posterior represents an updated *belief* about the prior, taking into account the new data. One can then use these posterior beliefs to derive new estimates of the parameters of the unknown distribution.

Both prior and posterior distributions must be of the same family– i.e., they must have the same algebraic form. If that is the case, they are termed *conjugate distributions*. Additionally, the family of the *likelihood function* gives rise to the choice of the prior's family. If the prior is appropriately selected to be *conjugate* for the *likelihood*, then the posterior will be of the same family as the prior [11]. *Conjugacy* offers a closed form for the posterior, allowing for the easy update of the prior—via straightforward manipulations of the prior's *hyperparameters* in the face of new evidence.

In our case, the data provided to the agent is in the form of a multivariate Gaussian, corresponding to *demo* or *item types*. We also need to maintain the *demo-based user type* or the overall *user type*, which will also be multivariate Gaussians. Since we are not aware of the underlying type parameters—determining its *mean* and *covariance matrix*—we can model the *conjugate prior* as a *Normal-Inverse Wishart* distribution *(NIW)* [20, 1]. We can then readily update the *prior* hyperparameters using samples drawn from the data[1] to get the *posterior* ones:

$$\mu_n = \frac{\kappa_0}{\kappa_0 + n}\mu_0 + \frac{n}{\kappa_0 + n}\overline{x} \tag{1}$$

$$\kappa_n = \kappa_0 + n \tag{2}$$

$$\nu_n = \nu_0 + n \tag{3}$$

$$\Lambda_n = \Lambda_0 + S + \frac{\kappa_0 n}{\kappa_0 + n}(\overline{x} - \mu_0)(\overline{x} - \mu_0)^T \tag{4}$$

$$S = \sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T \tag{5}$$

where $\overline{x}$ is the sample mean, *n* is the number of the samples and x_i are the samples from the data. Also, $\mu_0, \kappa_0, \nu_0, \Lambda_0$ are the known hyperparameters of the prior NIW distribution. Specifically, ν_0 represents its *degrees of freedom*, while μ_0 (the *mean vector*) and Λ_0 (the *precision matrix*) are the hyperparameters that specify the multivariate Gaussian component of the prior; and *S* is a *scatter matrix*, a statistic characterizing the model's covariance matrix and intuitively providing a measure of the samples' dispersion. Finally, the model's parameters, *mean* (μ) and *covariance matrix* (Σ), can be calculated ("integrated-out") given the updated beliefs, using an *Inverse Wishart* and a *Gaussian* as follows.

$$\Sigma \sim IW(\Lambda_n, \nu_n) \tag{6}$$

$$\mu \mid \Sigma \sim N(\mu_n, \Sigma/\kappa_n) \tag{7}$$

3. RECOMMENDATION PROCESS

We have already described the main intuitions and key ideas of our approach in Section 1 above. Here, we provide a more detailed picture of our system's architecture and overall recommendation process, also summarized in Figure 1(a).

When a user enters the system, she is shown and asked to rate a number of 5 demos.[2] The agent decides which demos to show based on its knowledge about the user—i.e., the stored *user type*. Thus, it presents demos that most closely match her preferences, as these have been embodied in the *user type* so far. In the case of *a new user*, the agent again shows 5 demos, but now those have few common features, since they are—out of necessity—selected randomly (as the long-term *user type* is first constructed after the user rates some *item*). Then, the user provides a rating for each demo, and a *Bayesian updating* process is used to create the *demo-based user type* from rated demo items.

Bayesian updating actually takes place two times during the recommendation process, firstly to infer a *temporary*, *demo-based user type*; and, secondly, to update the overall *user type*. The process takes into account user ratings regarding demos or items shown to the user, and, given the ratings, samples the respective *demo* or *item types* the appropriate number of times; and uses these samples to come up with an (updated) system-inferred type regarding the user. This can be done given the fact that the user type ("demo-based" or not) *is of the same form as the items' type*, and via the proper use of conjugate NIW priors. The intuition is that, in the absence of *explicit* data regarding the *user type* (which is the model whose parameters we need to infer), we utilize user ratings as an *implicit* way to indicate the extent to which a user "associates" herself with the demo or item she is shown. Thus, we can then sample the model this item (e.g., a trailer or a movie) originates from a number of times (proportional to the degree of the user's liking of that item), and treat these samples as new evidence for the Bayesian updating process. It is important to note that, to ensure the efficiency of the update, the number of samples must be high enough. To achieve this, we multiply the *rating* with 100, thus guaranteeing that hundreds of samples are used during an update.[3]

In more detail, the user observes and rates the first demo item (e.g., watches and rates the first trailer). Then, the system takes *rating*$\times$100 samples from the *demo type* distribution of that demo item, and updates the posterior's *hyperparameters* based on those *samples* and the *hyperparameters* of the *demo-based user type* prior (which has to be an *uninformative* one [20]). Subsequently, the user watches and rates the second demo and the system samples that *demo type* and updates the *hyperparameters*, in the same way as for the first one—but now the posterior of the first step has become the *informative* prior of the second step. After all such steps, the *hyperparameters* of the NIW distribution can be used to estimate the parameters (μ, Σ) of the multivariate Gaussian, which models the *demo-based user type*. Figure 1(b) summarizes this process.

Now, when the system has the final estimate of the *demo-based user type*, it has to recommend a specific item to the user. To do so, it must search the database to find the item matching that *demo-based user type* best. To this end, it uses the *Kullback-Leibler (KL) Divergence* criterion. The *KL-divergence* between a Gaussian t (modelling, e.g., a demo-based user type), and a Gaussian i (corresponding to some item), of dimension d each, is given by:

$$KL(t \parallel i) = \frac{1}{2}\log \mid S_t^{-1}S_i \mid + \frac{1}{2}tr((S_t^{-1}S_i)^{-1}) - \frac{d}{2} + \frac{1}{2}(m_t - m_i)^T S_i^{-1}(m_t - m_i) \tag{8}$$

[1] In our case, we will be utilizing user ratings to determine the number of samples to draw from different data distributions.

[2] The number of projected trailers (demos) used in our experiments is "5", as a tradeoff between receiving enough information for accurate inference of user preferences, and avoiding user distraction and frustration [14, 15]. Of course, when testing the agent with actual human subjects in real time, the number of trailers shown and their length will definitely have to be adjusted according to the users' reaction and perceived frustration levels. However, our main objective here is to evaluate the potential of our novel Bayesian user modelling method. Thus, while adjusting for specific real-time usage concerns in a given domain is important, current parameter choices are adequate for confirming the soundness of this approach.

[3] Taking a *rating*$\times$100 number of samples ensures that each *given* demo or item type is *represented* accurately enough to be combined with prior information regarding the user (who is also represented as a Gaussian) to estimate a user type via Bayesian updating. This is not an attempt to "estimate" a demo or item type via sampling.

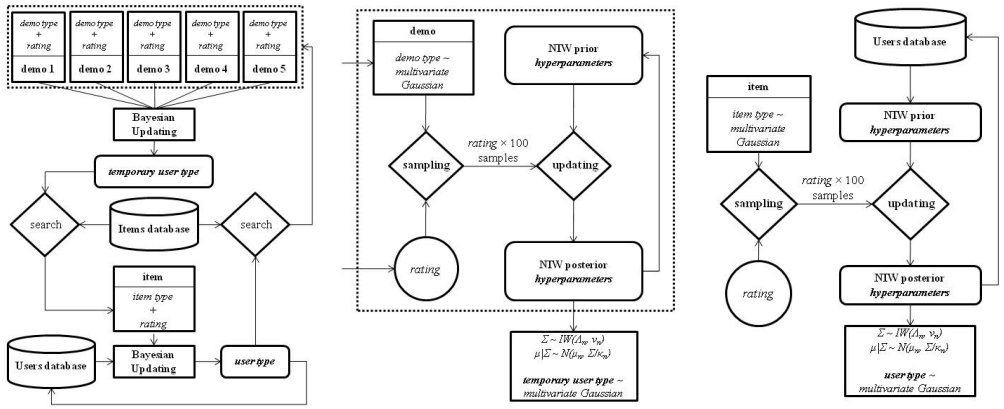

Figure 1: (a) The overall recommendation process. (b) Deriving the *demo-based user type*. (c) Deriving the *user type*.

where S_t, m_t, S_i and m_i are the distributions' parameters, and $tr(\cdot)$ is the *trace* of the corresponding matrix [21]. The lower the *KL-divergence* between two Gaussian types, the greater their proximity. This is the item (e.g., the movie) recommended to the user. Notice that this *might not* be one of the demos shown to the user before recommendation. This effectively corresponds to a *Bayesian* rather than a heuristic exploration approach in this domain: the system employs its probabilistic beliefs regarding (long-term) *user type* and (short-term) *temporary user type* to come up with a recommendation; and makes recommendations based on beliefs given sampled evidence, rather than, e.g., greedily matching a demo.

Subsequently, the user puts a rating on the item (after, e.g., watching the movie). Then, the system performs another Bayesian update, resulting to a new *user type* estimate. The Bayesian updating of the *user type* is similar to the *demo-based* one; but now the *rating*$\times 100$ samples collected are drawn from the distribution corresponding to the item the user was finally recommended and rated *only* (Figure 1(c)). Note that *user type* and corresponding beliefs are stored for future use, unlike the *demo-based user type* which lasts only for the current session. Thus, in the case of a future system use by a known user, the hyperparameters of the posterior from which the previous *user type* was inferred, will be the hyperparameters of the *(informative)* prior for deriving the next *user type*. Moreover, as stated, the updated and stored *user type* serves as a prior to guide the system to select demos. This is also done via using the *KL-divergence* minimization criterion.

3.1 Alternative action selection methods

The "basic" Bayesian method described above behaves "greedily" wrt. beliefs when selecting a demo or item: it just picks the one with minimum KL-divergence from the Bayes-updated user type. We also devised alternative *action exploration* techniques, to assess whether these would lead to improved recommendation decisions.

3.1.1 VPI-based selection of demos or items

The first of these techniques attempts to account for the expected *value of perfect information (VPI)* [7] characterizing the various "agent actions"—i.e., in this domain, potential recommendation choices. The rationale behind this technique is that a choice has a value not only because of its immediate benefit to the user, but also because of the information it relays with respect to user preferences. Intuitively, if a recommendation and observed user reaction leads to a reassessment of what the user really prefers, then this recommendation action carries a high information value. Thus, the *VPI exploration* technique, adapted for the recommendation domain, attempts to "simulate" various *alternative* user type reactions to future recommendations; calculates the value of information gained from these reactions when compared to the thus far modeled user type's expected behaviour; and averages out these results to come up with an *information gain estimate* that is used to "boost the desirability" of the recommendations to be made.

In more detail, let us suppose that the reward that some user derives from a specific item is $reward = f(KL\,divergence) = M - \lfloor KL\,divergence/M \rfloor$, where M is the maximum rating the user can give to a movie (e.g., $M = 10$). We define as i_1 the item with the highest reward r_1 for the user (given the model built for her so far), and as i_2 the second best with reward r_2. Consider now an item i selected for recommendation to a user type j (possibly different to the type modeled for the user so far), and assume that this in fact represents the "actual" type for the user—therefore, presenting this user type j with an item will lead to "perfect information" regarding the value of this item to the user. (Of course, this is just an assumption the method makes, but allows it to compute a value of information estimate, via "sampling" user types and averaging out their behaviour.) Assume that this recommendation results to a user reward of r_i for item i. We distinguish the following two cases (in all other cases, the gain due to perfect information is 0):

1. if i coincides with the item considered best for the user so far ($i = i_1$), then the *gain* from presenting the user with this item is either: *(a)* $\text{gain}_i^j = 0$, for $r_i > r_1$ (since we derived no new information from presenting the user with this item: the "perfect" information we got by fixing the user type to the assumed "actual" user type j coincides with what we had already estimated—i.e., that i_1 is the "best" item for the user); or *(b)* $\text{gain}_i^j = r_2 - r_i$, for $r_i < r_2$ (since we "learned" that the item i is actually worth less to the user than the item considered so far to be only second-best).

2. if i does not coincide with i_1, then the *gain* from presenting j with i is *(a)* $\text{gain}_i^j = 0$, for $r_i < r_1$ (since we only observed i to be sub-optimal, as expected); or *(b)* $gain_i^j = r_i - r_1$, for $r_i > r_1$ (since we now "learned" that i was actually better than the item considered best so far).

Given this, our *VPI exploration* method works as follows: We estimate the modeled-so-far *user type*[4] by "integrating out" our NIW

[4]This can be a long-term or a short-term user type, depending on whether we are attempting to select demos or actual items.

prior, as described in Section 2 (Eqs 6 & 7). Subsequently, for this integrated-out user type, we discover the i_1 item with the highest reward, and the second-best item i_2. We then sample from our NIW prior a number of s Gaussians, which represent "alternative" user types. (In our experiments, we set s=10.) After that, we calculate for each j user type (corresponding to one of the s sampled Gaussians), the rewards from presenting it with every i item, and compute the corresponding gain_i^j values, as outlined above.

The next step is the calculation of the *average gain*, $\overline{\text{gain}}_i$, for presenting an item i to our user; this is computed by averaging out, over all s samples (i.e., over all user types j sampled), the gain_i^j gains estimated for this item. Finally, the *VPI* method selects (and presents) the item I that maximizes the sum of the integrated-out user type's reward and the corresponding gain from selecting I:

$$I = \arg\max_i \{V_i = reward_i + \overline{\text{gain}}_i\} \quad (9)$$

3.1.2 Boltzmann selection

We also tried the well-known *Boltzmann exploration* [5] method to select the appropriate demo or item. At each time step t, it assigns a selection probability to all available i actions:

$$Pr(i) = \frac{e^{U_i/T}}{\Sigma_{j=1}^{n} e^{U_i/T}} \quad (10)$$

where $T = c \cdot \alpha^t$, with c a constant and $\alpha < 1$. An action i is chosen with probability proportional to its utility U_i; and with T decreasing over time, exploration is progressively reduced.

We tried Boltzmann selection with two different utility functions in our experiments. The first of these methods, simply called *Boltzmann*, employs a U function equal to the KL-divergence between the user type and the items. The second one uses as U the metric of the VPI method described above, i.e., sets $U_i = V_i$, where V_i is the quantity in Eq. 9. We call this method *Boltzmann-VPI*.

4. APPLICATION TO THE MOVIES DOMAIN

We chose to apply our method to movie recommendations, an important domain that has inspired much research in recommendation systems. Here, *items* correspond to *movies*; *demos* correspond to *movie trailers*; and we use the term *trailer-based user type* to refer to a *demo-based user type*.

Each *movie type* is modeled as a *multivariate Gaussian*, with each of its variables corresponding to a movie genre. The probabilities are distributed over ratings, which are provided in some scale of choice (e.g., $1-10$ or $1-5$). To create these multivariate Gaussians, we were inspired from the *MovieLens* (http://www.grouplens.org) datasets, which are actually used in our experiments below. These datasets comprise of ratings provided by thousands of users on thousands of movies. In the *MovieLens* dataset containing 1 *million* ratings, movies are characterized by 18 specific genres. We therefore define *movie types* to be *k-dimensional Gaussians*, with $k = 18$ in our experiments involving real MovieLens ratings.

Let us now describe the exact form (and attribute values) of such a Gaussian representing a movie or a trailer. The *mean* is essentially determined by the overall rating of that movie, so it is a $1 \times k$ vector which, on each dimension, contains values equal to that movie's rating—under the assumption that the movie rating corresponds to every genre available. Of course, one cannot be certain about the rating of a genre not associated with the movie, but this is taken care of by the way we construct the covariance matrix. Specifically, a movie's $k \times k$ covariance matrix is constructed as a *diagonal covariance matrix* [4], assuming that the movie genres are independent of each other. Each element on the diagonal is associated to a genre, and the element's value depends on whether the movie is characterized by that genre or not. We assume that the uncertainty about the rating of the actual genres of the movie is small, and thus set the values of the corresponding elements to $\sigma^2 = 1$. In contrast, uncertainty about those genres *not* associated to the movie is naturally *high*, and thus we assign a σ^2=20 (an empirically chosen value that is high enough so as to not "disturb" the distribution) to those diagonal elements. As an example, consider the movie 'Movie' with an overall rating of 7, whose genres are *action, sci-fi, thriller*. Its type is the following. The mean vector's entries all carry a value of 7; while its covariance matrix is a diagonal one, with diagonal entries corresponding to the *action, sci-fi and thriller* variables having a value of 1, and all other (diagonal) entries having the value of 20. The Gaussians for the *user* and *trailer-based user* types have the same form as that of movies (i.e., k-dimensional Gaussians).

In addition, the agent needs to store and update beliefs about user's types (*trailer-based* and overall). As mentioned, these beliefs take the form of Normal-Inverse Wishart priors, which can be easily manipulated to infer the corresponding types as k-dimensional Gaussians, and match them to movies as required by the system.

5. EXPERIMENTS

We ran several sets of experiments to validate our algorithm, which we call *BayesYouLikeIt*, and its variants, with very encouraging results. In all experiments, the *hyperparameters* of an uninformative NIW prior take the following values: $\kappa_0 = 0, \nu_0 = -1, \mid \Lambda_0 \mid = 0$. Thus, the *updated hyperparameters* of the NIW posterior after observing n samples become: $\mu_n = \overline{x}, \kappa_n = n, \nu_n = n - 1, \Lambda_n = S = \sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T$.

Now, to test the scalability of our technique, we initially ran experiments with *simulated* users on databases with *10,000*, *20,000*, and *40,000* randomly generated movies. Simulated movies' representation and ratings were based on those of the popular *IMDB* website (http://www.imdb.com). Specifically, we defined *movie types* to be 16-dimensional Gaussians, corresponding to the 16 most common genres available in *IMDB*. Also, we generated 50 simulated users that interact with the system. Each simulated user is characterized by its *real user type*, a 16-dimensional Gaussian distribution with a random mean in the range of $[1 - 10]$ on each dimension, corresponding to *IMDB* ratings, and a covariance matrix which is *spherical*, assuming that the *real user type* is confident regarding the degree to which it likes each genre [4].

To avoid the exhaustive search for trailers and movies within these large databases, we apply clustering methods on the set of stored movies. This reduces the number of comparisons between a *user type* and stored *movie types*. In our experiments, the movies were clustered in clusters with 1,000 movies each, on average (e.g., 20,000 movies clustered into 20 clusters), based on their similarity using the *Kullback-Leibler hard k-means*, a variation of the *Bregman hard k-means* clustering [21].[5] We also had to create a *rating function*, to be used by the simulated users to assign ratings. The function exploits the *KL divergence* between the *real user type* and each *trailer type* or *movie type* of movies in our system—the less the divergence, the higher the rating: $rating = f(KL\,divergence) = 10 - \lfloor KL\,divergence/10 \rfloor$ (where $0 < KL\,divergence < 100$).

In Figure *2*, we can see that *BayesYouLikeIt* recommends movies that constantly receive high ($> 8/10$) ratings, when tested on the *20,000* simulated movies database. The experimental results for the *10,000* and *40,000* movie databases are of the same quality. We

[5]Note that after a few hundreds of recommendations leading to movies being removed from their clusters, the need for reclustering arises. This is not a problem, as it can be executed off-line.

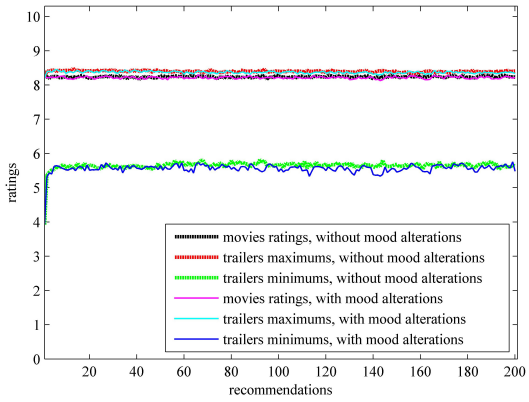

Figure 2: Comparison of *average per iteration ratings* from 50 *simulated* users on 200 movies for 10 runs, and *average range of corresponding trailer ratings*, with and without mood alterations simulation.

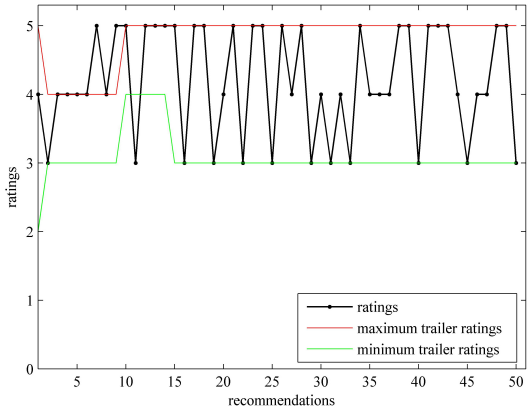

Figure 3: Typical run ratings of a *single*, *real* user on 50 movies, and *range* of corresponding trailer ratings.

also observe that, in an average iteration, the user is shown trailers whose ratings range is about 3 degrees wide; and whose average *maximum* rating is *almost always* higher than the (average) rating received for the movie shown during that iteration. This is because Bayesian exploration: *(1)* does not *necessarily* return a movie that matches the best trailer shown in an iteration; while, at any given iteration, there are still on average many "good" movies whose trailer the method can show; and *(2)* enables the system to actually project trailers of *movies already shown* at any iteration, and thus trailers of "preferred" movies might be shown to a user again—since these are just used to detect the *current mood* of the user. At no point is a *movie* already shown to the user actually recommended again; however, *BayesYouLikeIt* might re-use *trailers* derived from the "believed" user type to infer the *temporary* user type (though in practice this will occur only rarely in a large database).

We also evaluated the ability of our agent to capture temporary changes of a user's *mood*. We simulate such mood changes by periodically changing the mean values of the *"real user type"*'s multivariate Gaussian distribution (which represent the preferences of the assumed "real" user). Specifically, after every 10 recommendations, we randomly change the mean of each variable of the corresponding Gaussian (via sampling a normal distribution over the range of $[1-10]$). The "mood changes" last only for 5 recommendations, and then the real user type returns to its original form. Figure *2* confirms that our method is able to successfully capture the user's mood, since it constantly recommends movies that are subsequently rated highly. Indeed, our method's performance appears to be robust, and is not negatively affected by mood changes.

Following that, we ran experiments to test our agent on data coming from *real users*, that reflect actual human preferences and behaviour. This kind of data also offer an ideal testbed for comparison with other well known techniques. We used the *MovieLens* dataset with *1 million ratings* from *6,040* users on *3,952* movies. Thus, there is no longer a need to generate simulated users and ratings: we can now have a user's rating on a movie (or trailer) by just referring to the real-world dataset.[6] No clustering was used in these experiments. The movies inside the system's database are modeled as in Section 4, with a movie's *mean* being the average rating it has in the MovieLens dataset. Ratings are integers in $[1-5]$.

We executed different sets of experiments with *200 recommendations each*, using all aforementioned *variants* for trailer/movie selection. For each such experiment, we employed *5 sets* of *100 users* each, consisting of real users with *200 or more* ratings in the MovieLens dataset. In some detail, we took the histogram of the number of ratings per user in the MovieLens dataset, and sampled 100 different users for each experimental user set in accordance to that distribution. This ensures there is no bias in the ratings used as input in the experiments. Also, each experiment involving a specific user set was executed for *10 independent runs*. We then sequentially produced 200 recommendations to each individual user; and calculated the *average per iteration (true) ratings* assigned by the real users, across all user sets and experimental runs.

Now, as explained in Section 3, *BayesYouLikeIt* progressively builds the *user type*, and recommends the movie that best matches the *trailer-based user type* at each iteration—or the one selected according to some variant exploration criterion. Indeed, in addition to the "basic" *BayesYouLikeIt (BYLI)* algorithm, we ran experiments exploiting the VPI and Boltzmann exploration methods described in Section 3.1. We used these criteria in two different ways. First, we employed them only during the trailer selection phase of the algorithm; and second, during both the trailers and the movie selection phases. Boltzmann exploration parameters were set to: $c = 1, \alpha = 0.5$ and $t \leq 3$, with $t_0 = 0$ and $t_{i+1} = t_i + 1$.

We compare our algorithm with an established recommender engine, built on the *Apache Mahout* machine learning library, adapted to yield the *Myrrix* software.[7] The algorithm used by the engine is the *large, sparse matrix factorization (LSMF)* method [13, 27], implemented using a modified version of the *alternating least squares (ALS)* algorithm. LSMF requires a number of ratings to be entered in the system, so that matrix factorization can be performed.

Figure 3 helps us gain further insights in *BayesYouLikeIt* behaviour when recommending movies to a single (real) user. We can see in the figure that, while learning, *BayesYouLikeIt* might occasionally recommend a movie which the user ranks *lower* than the trailers projected to her during that iteration; however, some times the algorithm might also recommend a movie that the user prefers to all trailers shown to her during that iteration. Over time, *BayesYouLikeIt* returns movies that receive consistently good ratings.

[6]We remark that since we care about best "online" user experience and *do not* predict ratings, the use of comparison metrics like *Root Mean Square Error (RMSE)* is not very meaningful here. Instead, we compare the methods wrt. average per recommendation ratings.

[7]http://mahout.apache.org; http://www.myrrix.com/design

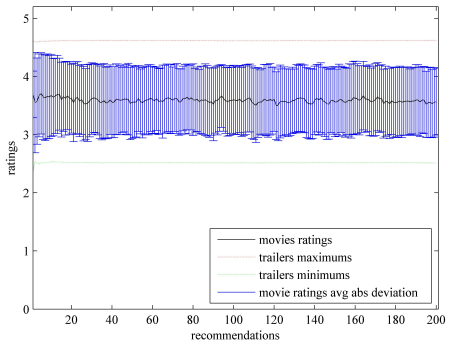

Figure 4: *Average per iteration ratings*, *average range* of corresponding trailer ratings and *movie ratings average mean absolute deviation* for *BYLI - Boltzmann-VPI on trailers & movies*.

Methods	Average ratings
LSMF - pretrained	3.6848
LSMF - untrained	3.6540
BayesYouLikeIt (BYLI)	3.6112
BYLI - VPI on trailers	3.5968
BYLI - VPI on trailers & movies	3.5911
BYLI - Boltzmann-VPI on trailers	3.5968
BYLI - Boltzmann-VPI on trailers & movies	3.5911
BYLI - Boltzmann on trailers	3.5920
BYLI - Boltzmann on trailers & movies	3.5592

Table 1: Comparison of average ratings of all methods (across all user sets, iterations and experimental runs).

This fact is confirmed by the *average per iteration ratings* results of Figs. 4 and 5. Fig. 4 shows that *BayesYouLikeIt* manages to recommend movies that are on average highly rated by the users.[8] Moreover, average ratings deviation can be observed to decrease over time, demonstrating an ability to "learn" and progressively become more confident on its assessment of users' preferences. We note that the over time decrease in ratings average absolute deviation is a bit sharper for the variants using *VPI* (or *Boltzmann-VPI*) on both trailers and movies. This figure also shows the (average) maximum and minimum trailer ratings received by users (per iteration). Maximum trailer ratings are consistently higher than ratings received for movies recommended, for the same reasons as in the experiments with simulated users above.

In Fig. 5(a), we observe that the VPI and Boltzmann exploration methods (and their combinations) are successfully intertwined with our "basic" Bayesian algorithm, since all *BYLI* variants return recommendations that receive consistently high ratings from the users. Average ratings across all iterations, depicted in Table 1, confirm that all *BYLI* variants perform very similarly, with the "pure" *BYLI* method achieving a slightly better average score than the rest. However, *BYLI - VPI on trailers* provides an advantage when it comes to sequential behavior (notice that on average it provides better recommendations at the initial steps). When Boltzmann is combined with VPI, VPI is the defining component, and thus *Boltzmann-VPI* and *VPI* exhibit identical behavior.

Fig. 5(b) then compares our method to *LSMF*. Note that *LSMF* was tested under two assumptions in these experiments. First, on the assumption that it is completely unaware of *any* user preferences at system start (this is marked as *"LSMF-untrained"* in the figures); and on the assumption that the system is *pre-trained* on ratings data from all other (i.e., the $6,040 - 100 = 5,940$ non-picked) users in the database (*"LSMF-pretrained"*). These results clearly demonstrate that *(a)* the average user ratings value of *BayesYouLikeIt*-recommendations is about $3.5 - 3.7$ (out of 5); and *(b)* although our method does not depend on other users' ratings in order to recommend movies to a user, ratings received are almost indistinguishable in average quality to those received when using the *LSMF* technique. Moreover, we can observe that, *LSMF* being a CF method, it has to first collect a small number of ratings from users in order to be able to return good recommendations, and thus cannot respond to a user request in a meaningful manner during the very first iteration. This is clearly visible in Fig. 5(c). In contrast, our personalised method can *immediately* suggest a good movie to a user. After the very first iterations, *LSMF* is able to exploit knowledge of "similar" users ratings and performs strongly—but as the number of "preferred" movies drops (since our dataset contains a "closed set" of movies), its performance drops (Fig. 5(b)). *BayesYouLikeIt* does not start as strong, but due to progressively converging to real user types, its performance is quite *stable* throughout all recommendations. As a final remark, running on MATLAB on a 2.20 GHz / 4 GB RAM PC, it takes a *BayesYouLikeIt* agent only about 0.3 sec on average to recommend a movie to a user.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel approach for making entirely personalized recommendations, and applied it in the movie recommendations domain. Our approach uses Bayesian updating to infer a *user model*, which is *a distribution of the same form* with the models of the items under recommendation. This is, we believe, key to enabling the implicit inference of an individual's latent or otherwise unrepresentable, complicated preferences; and enables us to devise and employ a kind of *Bayesian exploration* in this domain. We presented and tested several variants of our method, and evaluated their performance, with very promising results. In a sense, what we exploit in this work is the fact that modern-world data sets contain much "annotated" information regarding the nature of contained data (e.g., in the form of "genre"-specifying labels). This information allows us to properly define the dimensions of multivariate Gaussians representing users (according to our "you are what you consume" idea), without relying on the tastes of others—and without attempting to explicitly predict user ratings.

We remark that our method is generic, and not "fine-tuned" for movie recommendations. Nonetheless, our *personalized* method's performance almost matches that of a state-of-the-art movie recommendations method, which is able to exploit preference data originating from thousands of users. As our method proves to be competitive against algorithms that do have this "advantage", it is most probably especially well-suited for environments where user preferences data is scarse. In sparse datasets, our approach is expected to perform better than methods which require other users' ratings. Therefore, it could be used as a "bootstrapping" tool, generating recommendations until more data is available; moreover, it can be readily employed as a "training" component used during a more sophisticated system's initial operation period.

Regarding future work, we plan to conduct a user satisfaction survey, based on a number of users testing our system. We also intend to run experiments on larger movie datasets, and perform a more thorough evaluation of the *BayesYouLikeIt* variants. It is also worth testing our approach in other domains—e.g., use it for recommending scientific papers; studying nutritional habits; or em-

[8]Fig. 4 depicts the *BYLI-Boltzmann-VPI on trailers and movies* variant; results are similar for other *BayesYouLikeIt* variants.

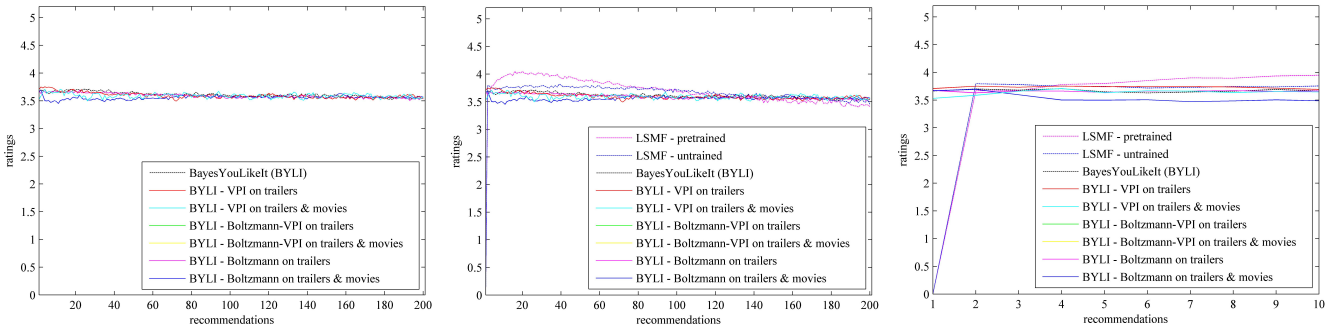

Figure 5: (a) Behaviour of *BayesYouLikeIt* and its variants. (b) Comparison between *LSMF* and *BYLI*. (c) Methods behaviour during the first 10 recommendations. All subfigures depict *average per iteration ratings* across all user sets and experimental runs.

ploy it in story-telling environments. Finally, we aim to incorporate multiagent systems techniques in our model, allowing us to exploit user-specific information coming from other types of online agents.

7. REFERENCES

[1] T. Anderson. *An introduction to multivariate statistical analysis*. John Wiley and Sons, 2003.

[2] N. Barbieri, G. Costa, G. Manco, and R. Ortale. Modeling Item Selection and Relevance for Accurate Recommendations: a Bayesian Approach. In *Proceedings of the 5th ACM Conference on Recommender systems*, RecSys '11, 2011.

[3] S. Berkovsky, T. Kuflik, and F. Ricci. Distributed Collaborative Filtering with Domain Specialization. In *Proceedings of the 1st ACM Conference on Recommender systems*, RecSys '07, 2007.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Corr. 2nd printing edition, October 2007.

[5] D. Carmel and S. Markovitch. Exploration strategies for model-based learning in multiagent systems. *Autonomous Agents and Multi-agent Systems*, 2(2):141–172, 1999.

[6] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of AAAI-2000*, 2000.

[7] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, 2003.

[8] R. Dearden, N. Friedman, and D. Andre. Model based Bayesian Exploration. In *Proceedings of Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

[9] S. Debnath, N. Ganguly, and P. Mitra. Feature Weighting in Content Based Recommendation System Using Social Network Analysis. *WWW 2008*, April 2008.

[10] M. DeGroot and J. Schervish. *Probability and Statistics*. 2002.

[11] D. Fink. A Compendium of Conjugate Priors. Technical report, 1997.

[12] J. Golbeck. Generating Predictive Movie Recommendations from Trust in Social Networks. In *iTrust-2006*, pages 93–104, 2006.

[13] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[14] K. E. Kendall and J. E. Kendall. *Systems Analysis and Design*. 2011.

[15] G. Kurtenbach, A. Sellen, and W. Buxton. Some Articulatory and Cognitive Aspects of "Marking Menus": An Empirical Study. *Journal of Human-Computer Interaction*, July 1991.

[16] H. Langseth and T. D. Nielsen. A latent model for collaborative filtering. *Int. J. Approx. Reasoning*, 53(4):447–466, June 2012.

[17] A. Liu, Y. Zhang, and J. Li. Personalized Movie Recommendation. In *Proc. of the 17th Intern. Conf. on Multimedia 2009*, October 2009.

[18] P. Melville, R. J. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. *In Proceedings of AAAI-2002*, pages 187–192, July 2002.

[19] R. Mukherjee, P. S. Dutta, and S. Sen. MOVIES2GO - A new approach to online movie recommendation. *In the IJCAI Workshop on Intelligent Techniques for Web Personalization*, 2001.

[20] K. P. Murphy. Conjugate bayesian analysis of the gaussian distribution. Technical report, University of British Columbia, 2007.

[21] F. Nielsen and R. Nock. Clustering Multivariate Normal Distributions. *ETVC 2008, LNCS 5416*, pages 164–174, 2009.

[22] C. Ono, M. Kurokawa, Y. Motomura, and H. Asoh. A Context-Aware Movie Preference Model Using a Bayesian Network for Recommendation and Promotion. *UM 2007, LNAI 4511*, 2007.

[23] S. Shearin and H. Lieberman. Intelligent Profiling by Example. *In Proc. of the 2001 ACM Conference on Intelligent User Interfaces (IUI-2001)*, Santa Fe, NM, January 2001.

[24] M. Szomszor, C. Cattuto, H. Alani, K. O'Hara, A. Baldassarri, V. Loreto, and V. D. Servedio. Folksonomies, the semantic web, and movie recommendation. In *4th European Semantic Web Conf.*, 2007.

[25] H. Yu and M. O. Riedl. A Sequential Recommendation Approach for Interactive Personalized Story Generation. *In Proceedings of AAMAS 2012*, pages 71–78, 2012.

[26] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th Annual intern. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 47–54. ACM, 2007.

[27] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proc. 4th Int. Conf. on Algor. Aspects in Information and Management, LNCS 5034*, pages 337–348. Springer, 2008.