Tied Boltzmann Machines for Cold Start Recommendations

Asela Gunawardana
Microsoft Research
One Microsoft Way
Redmond, WA 98052
aselag@microsoft.com

Christopher Meek
Microsoft Research
One Microsoft Way
Redmond, WA 98052
meek@microsoft.com

ABSTRACT

We describe a novel statistical model, the tied Boltzmann machine, for combining collaborative and content information for recommendations. In our model, pairwise interactions between items are captured through a Boltzmann machine, whose parameters are constrained according to the content associated with the items. This allows the model to use content information to recommend items that are not seen during training. We describe a tractable algorithm for training the model, and give experimental results evaluating the model in two cold start recommendation tasks on the MovieLens data set.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*; G.3 [**Probability and Statistics**]: *correlation and regression analysis*; I.2.6 [**Artificial Intelligence**]: Learning—*parameter learning*

General Terms

Algorithms, Performance

Keywords

recommender systems, collaborative filtering, content-based filtering, cold start, Boltzmann machines

1. INTRODUCTION

Recommender systems attempt to suggest items of interest to users based on information including their own previous usage patterns, the usage patterns of other users, and features of the items themselves [10]. Collaborative filtering techniques provide recommendations by using the preferences of other users that have historically had similar preferences to the target user [2, 9, 15]. These preferences may be expressed either explicitly (e.g. by rating movies at an online movie rental service) or implicitly (e.g. by browsing an item description or buying an item at an online retailer). For example, the familiar Amazon item-to-item recommender system [9] recommends items that were bought significantly more often by users buying a given item to a target user buying that given item. Such systems have the drawback that they suffer from the item cold start problem–an item cannot be recommended until it has been rated by a number of existing users.

This problem can be alleviated by using information about the content of items. This information can be quite broad, ranging from words in a document to metadata such as actors of movies. While many content-based techniques, as well as hybrid techniques that use both collaborative information and content information exist [10], we use the collaborative information to learn to predict preferences from content information. The work of [18] is a step in this direction, where user ratings and content information are used to learn user preferences for content. The content information of a new item is then used to predict the users' votes for that item. We present a novel hybrid approach that does this in an arguably more direct manner. The collaborative data is used to learn how well different content information predicts user ratings of items. We treat ratings of items by users as binary random variables indexed by users and items, and explicitly model correlations between these binary random variables using a Boltzmann machine. In order to alleviate the cold-start problem, we tie the parameters of this Boltzmann machine through constraints in terms of the content information associated with the items.

The remainder of the paper is organized as follows. Section 2 discusses some related work. Section 3 describes the details of our model. We present the use of Boltzmann machines for modeling pairwise correlations of item ratings in section 3.1, followed by our content information-based parameter-tying scheme in section 3.2. In section 3.3 we present the use of a regularized pseudo-likelihood criterion for tractable parameter estimation, along with a stratified sampling technique for further computational gains. Section 4 describes the benchmarking of our model against an aspect model based recommender system on two movie recommendation tasks. We conclude with some discussion in section 5.

2. RELATED WORK

In contrast to previous work such as [18], we treat ratings of items by users as binary random variables indexed by users and items, rather than as a contingency table to be modeled through a joint distribution over user-item pairs.

This distinction in probabilistic approaches to content-based recommendation systems is discussed in [12]. The approach in [18] extends the work of [7] which models the joint distribution of users and items through an aspect model that clusters users and items in a latent space. In order to deal with cold start items, the approach of [18] models the joint distribution of users and content features instead of the joint distribution of users and items. Then, a "folding-in" technique [6] is used to embed items into the latent space so that items rather than content features can be recommended. However, this approach can only yield a conditional distribution over users given an item, rather than the desired joint distribution over items and users. Our approach avoids this difficulty by modeling the joint distribution of ratings of all items by all users in terms of the content of the items. Since the model predicts preferences of items directly, we can avoid the difficulties associated with "folding-in."

We also contrast our approach to the superficially similar literature on the use of restricted Boltzmann machines for collaborative filtering [14]. Tied Boltzmann machines are fully connected, and have weights constrained through the use of content information. In contrast, restricted Boltzmann machines are constrained to have a bipartite topology, but the weights on legal links are unconstrained. Furthermore, restricted Boltzmann machines do not make use of content information, and as such, are vulnerable to cold start problems.

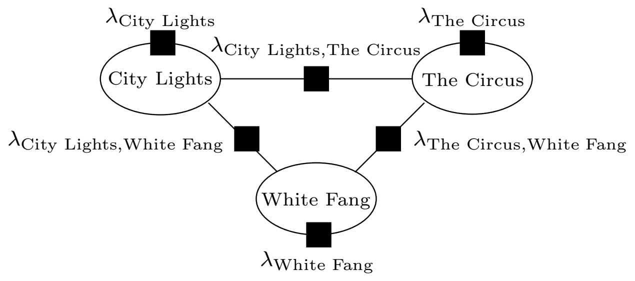
There has been some recent work on using filter-bots [4] for improving cold start recommendations [11]. This work attempts to use bots implementing various heuristics, some based on content information, to generate synthetic data for training recommender systems on. However [11] reports that the use of content based bots did not improve performance. Our approach uses data to learn how content data can be used to predict vote correlation, rather than using such heuristics.

Finally, we note that tied Boltzmann machines can also be viewed as dependency networks [5] where the conditional probabilities are logistic regressions instead of regression trees, and where the parameters are tied across nodes. It is this parameter tying that allows generalization to unseen items.

3. THE MODEL

We treat the problem of generating recommendations as one of predicting a user's vote on an item given his votes on other items as well as the votes of other users. In the case of implicit ratings, an action such as buying an item is taken to be an implicit vote for that item, and the problem is to predict what other items a user might act on in the future. In the case of explicit ratings, the user explicitly rates some subset of items, and the problem is to predict the user's ratings of other items.

We denote the vote for an item $i \in 1, \cdots, N$ by a user $u \in 1, \cdots, M$ by $v_i^{(u)}$. In the case of implicit ratings $v_i^{(u)} = 1$ when user u has acted on item i, and is 0 otherwise. In the case of explicit ratings , $v_i^{(u)}$ takes on a value given by their rating. In this paper, we will assume that votes are binary. Non-binary ratings can still be handled by binarizing as done in [18]–e.g. $v_i^{(u)} = 1$ if user u rates item i 4 or higher. We denote a user u's votes for all N items by a vector $v^{(u)}$. The key challenge is to model the interactions between the votes $v_i^{(u)}$.



$\lambda_{\text{City Lights}}$
$\lambda_{\text{The Circus}}$
$\lambda_{\text{City Lights,The Circus}}$
City Lights
The Circus
$\lambda_{\text{City Lights,White Fang}}$
$\lambda_{\text{The Circus,White Fang}}$
White Fang
$\lambda_{\text{White Fang}}$

Figure 1: Example Boltzmann machine for three movies. The ovals represent a user's votes on each of the movies. The solid squares represent the weights of the Boltzmann machine, which capture the popularity of each movie and how correlated a user's votes on each movie are.

3.1 Modeling Interactions with Boltzmann Machines

Boltzmann machines are a simple model for modeling interactions between variables. We assume that users' vote vectors are i.i.d., with the joint distribution of a user's vote vector v modeled using a Boltzmann machine:

$$p(v) = \frac{1}{z(\lambda)} \exp\left(\sum_i \lambda_i v_i + \sum_{i<j} \lambda_{ij} v_i v_j\right)$$

Here, we use λ to denote the collection of all λ_i and λ_{ij} and use $\sum_{i<j}$ to denote the sum over unordered pairs (i, j). Note that the per-item weights λ_i are distinct from the per-item-pair weights λ_{ij}. Since λ_{ij} is only defined for $i < j$, we abuse notation somewhat and use λ_{ij} and λ_{ji} interchangeably to denote the weight associated with the unordered pair (i, j) without introducing any ambiguity. The *partition function* $z(\lambda)$ is chosen to ensure that $p(v)$ is properly normalized over all configurations of the vote vector v. We note that the pairwise weights λ_{ij} capture pairwise collaborative effects–a high value of λ_{ij} indicates that users with $v_i = 1$ also tend to have $v_j = 1$. The per-item weights λ_i capture popularity effects–a high value of λ_i indicates a higher likelihood of $v_i = 1$ irrespective of the user's other votes.

Figure 1 shows an example of a Boltzmann machine for three movies, *The Circus*, *City Lights*, and *White Fang*. The ovals represent the random variables $v_{\text{The Circus}}$, $v_{\text{City Lights}}$, and $v_{\text{White Fang}}$, which are a randomly chosen user's votes for these three movies. The weights $\lambda_{\text{The Circus}}$, $\lambda_{\text{City Lights}}$, and $\lambda_{\text{White Fang}}$ capture the popularity of these movies–i.e. the likelihood that the user would vote for these movies, independently of what else they voted for. The pair-wise weights $\lambda_{\text{City Lights, The Circus}}$, $\lambda_{\text{City Lights, White Fang}}$, and $\lambda_{\text{The Circus, White Fang}}$ capture how likely the user's votes for these pairs of votes are to be correlated.

Note that the partition function $z(\lambda)$ is expensive to compute. It requires summing over all configurations of the vote vector. However, the conditional probability of a user's vote on a single item given his votes on all other items is easy to compute, and is given by

$$p(v_i|v_{-i}) = \frac{\exp\left(v_i\left(\lambda_i + \sum_{j\neq i|v_j=1} \lambda_{ij}\right)\right)}{1 + \exp\left(\lambda_i + \sum_{j\neq i|v_j=1} \lambda_{ij}\right)} \qquad (1)$$

This is natural in the case of implicit votes, where a value of v_j is always available for every item–either a user acted on an item or he didn't. In the case of explicit ratings, we typically observe the user's ratings for only a small set O of items, and do not have ratings for the other items (i.e. O^C). In this case, the conditional probability of v_i given the observed votes v_O is given by

$$p(v_i|v_O) \propto \sum_{v_{\{O^C \setminus i\}}} \exp\left(\sum_j \lambda_j v_j + \sum_{j<k} \lambda_{jk} v_j v_k\right)$$

Note that j and k range over all items, including i, all other items in O^C (i.e. all other items with unknown votes), and all the items in O. This probability is expensive to compute, because it involves a summation over all configurations of the unobserved votes. This difficulty can be averted by explicitly modeling unobserved votes in the values of v_i. For example, if $v_i = 1$ encodes a positive (e.g. 4-star or higher) rating for item i, and $v_i = 0$ encodes the lack of a positive rating (i.e. no rating or a negative rating), equation (1) remains valid for explicit ratings.

3.2 Content-Based Parameter Tying

One weakness of the model discussed in section 3.1 is that reliable estimation of the model parameters λ, in particular the pairwise weights λ_{ij}, requires sufficient observations. This is because the number of weights to be estimated scales quadratically with the number of items we need to be able to model correlations between. Since real data tends to be sparse, with many items having low probability of occurrence, these weights are difficult to estimate in practice. Thus, the Boltzmann machine model described above is not directly applicable to real large-scale applications. In particular, if an item is not seen during training, we have the item cold start problem, where none of the weights associated with that item can be reliably estimated.

We use parameter tying to alleviate these difficulties. In order to retain the ability to model the interactions between items, we use content information to guide the parameter tying. We assume that the content associated with item i is represented by a feature vector $f^{(i)} \in \mathbb{R}^D$ composed of D components. For example, features could be counts or TF-IDF weights of words in documents, or binary flags indicating whether specific actors appeared in a movie. Features with different semantics could be combined in a single vector. For example, some feature components could correspond to actors in a movie, while others could correspond to genres, while still others could take on numerical values such as movie length in minutes. We constrain λ to satisfy

$$\lambda_i = \mu^T f^{(i)}$$
$$\lambda_{ij} = {f^{(i)}}^T \eta f^{(j)}$$

where $\mu \in \mathbb{R}^D$ and $\eta \in \mathbb{R}^{D\times D}$ is symmetric. In this paper, we will only consider diagonal η, although none of our theoretical results depend on this simplification. In effect, this reparametrizes the model through the content, μ and η:

$$p(v;\ \mu,\eta) = \frac{1}{z(\mu,\eta)} \cdot \exp\left(\sum_k \mu_k \sum_i f_k^{(i)} v_i + \sum_k \eta_k \sum_{i<j} f_k^{(i)} f_k^{(j)} v_i v_j\right) \quad (2)$$

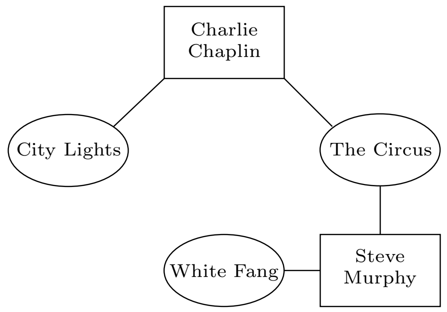


Figure 2: The example of Figure 1, showing how the movies are connected through content. The rectangles represent actors, and the links between actors and movies indicate that the actor appears in the movie.

where η_k denotes the kth diagonal component of η and $z(\mu,\eta) = z(\lambda(\mu,\eta))$.

Figure 2 shows the content associated with the movies considered in the example of Figure 1. In this example, the content information consists of actors appearing in the movies, and the feature vectors have a component for each actor, which is 1 when the actor appears in the movie and 0 otherwise. The fact that Charlie Chaplin appears in *City Lights* affects its popularity while the fact that Steve Murphy appears in *White Fang* affects its popularity. Our model assumes that the popularity weights of the movies are the sums of the effects of the actors appearing in them. Thus, the popularity weight $\lambda_{\text{The Circus}}$ is the sum of the popularity effect $\mu_{\text{Charlie Chaplin}}$ of Charlie Chaplin and the popularity effect $\mu_{\text{Steve Murphy}}$ of Steve Murphy. However, the model makes no assumption about the relative magnitude of the effects that each actor can have. Thus, $\mu_{\text{Charlie Chaplin}}$ may be higher than $\mu_{\text{Steve Murphy}}$ if movies including Charlie Chaplin happen to be more popular than movies including Steve Murphy. Similarly, the fact that Charlie Chaplin appears in both *The Circus* and *City Lights* may cause the votes for these movies to be correlated while the appearance of Steve Murphy in both *The Circus* and *White Fang* may cause the votes for those movies to be correlated. While the model assumes that the correlation weight of two movies is the sum of the correlation effects of the actors they have in common, it makes no assumption about the magnitude of these effects. If people who like Charlie Chaplin movies tend to like other Charlie Chaplin movies more than people who tend to like Steve Murphy movies tend to like other Steve Murphy movies, $\eta_{\text{Charlie Chaplin}}$ will be higher than $\eta_{\text{Steve Murphy}}$.

The conditional probability of v_i given all other votes v_{-i} is given by

$$p(v_i|v_{-i};\ \mu,\eta) = \frac{\exp\left(v_i\left(\sum_k \mu_k f_k^{(i)} + \sum_k \eta_k f_k^{(i)} \sum_{j\neq i} f_k^{(j)} v_j\right)\right)}{1+\exp\left(\sum_k \mu_k f_k^{(i)} + \sum_k \eta_k f_k^{(i)} \sum_{j\neq i} f_k^{(j)} v_j\right)} \quad (3)$$

We note that instead of constraining λ to be a deterministic function of the content feature vectors f and the parameters μ and η, we could have specified a distribution on λ, to yield a Bayesian model. However, we restrict attention to the more simple case of deterministic constraints on λ in this paper.

Generating recommendations for a user from this model once it is trained is simple. Equation (3) is used to compute the probability of $v_i = 1$ for an item i not yet seen by a user. This is the likelihood that the user would vote for item i. Ranking all items not yet seen by this probability gives a ranked list of recommendations. Thus, the remaining difficulty is in training the model.

3.3 Training the Model

We assume that we have a training set consisting of votes $v_i^{(u)}$ for all N items $i \in 1, \cdots, N$ by all M users $u \in 1, \cdots, M$. We also assume that the feature vectors $f^{(i)}$ are available for all items. We assume that missing votes are encoded by $v_i = 0$ as discussed above. The log likelihood of the training set is given by

$$- \log\big(z(\mu,\eta)\big) + \sum_u \Bigg(\sum_k \mu_k \sum_i f_k^{(i)} v_i^{(u)} + \sum_k \eta_k \sum_{i<j} f_k^{(i)} f_k^{(j)} v_i^{(u)} v_j^{(u)} \Bigg)$$

This likelihood is hard to optimize because it requires the evaluation of the partition function $z(\mu, \eta)$ for all candidate parameters.

Because of this, we approximate the log likelihood of the training set by its log pseudo-likelihood [1]:

$$\sum_u \log p(v^{(u)};\ \mu,\eta) \approx \sum_u \sum_i \log p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta)$$

As discussed above, the conditional likelihood $p(v_i|v_{-i})$ is easily computed using equation (3). If we treat each user u's vote for each item i as an independent training event given that user's other votes, the pseudo-likelihood can be interpreted as the conditional likelihood of the training set under a conditional model given by equation (3). In fact, equation (3) can be interpreted as the output probability of a logistic regression on v_i with inputs given by $f_k^{(i)}$ and $\sum_{j \neq i | v_j = 1} f_k^{(i)} f_k^{(j)}$. Thus optimizing the pseudo-likelihood of the tied Boltzmann machine model is equivalent to training a set of logistic regression classifiers with a very particular input parametrization and parameter tying scheme to predict individual votes. By the same argument, we note that the pseudo-likelihood is convex in the parameters, and therefore easily optimized via gradient methods.

In order to get robust estimates, we train the parameters μ, η by maximizing the regularized pseudo-likelihood

$$\mathcal{L}(\mu,\eta) = \sum_u \sum_i \log p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) + \alpha(||\mu||^2 + ||\eta||^2) \tag{4}$$

This can be viewed as MAP estimation of the set of tied logistic regressions mentioned above under a Gaussian prior [3].

The regularized pseudo-likelihood is alo convex in the parameters, and can easily be optimized using gradient methods. The gradients used can be computed according to:

$$\frac{\partial}{\partial \mu_k} \mathcal{L}(\mu,\eta) = \sum_u \sum_i f_k^{(i)} \left(v_i^{(u)} - p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \right) + 2\alpha\mu$$

and

$$\frac{\partial}{\partial \eta_k} \mathcal{L}(\mu,\eta) = \sum_u \sum_i f_k^{(i)} \left(v_i^{(u)} - p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \right) \cdot \sum_{j \neq i} f_k^{(j)} v_j^{(u)} + 2\alpha\eta$$

3.3.1 Sparse Training Through Stratified Sampling

With large data sets, optimizing the training criterion of equation (4) may be too expensive since it requires computing a conditional likelihood for each of $M \times N$ user-item pairs, where M and N may both be large. Since the set of items for which $v_i^{(u)} = 1$ is sparse, a further speedup is possible. Denote this sparse set of positive training items by $I_+^{(u)}$ and its complement by $I_-^{(u)}$. Sample items from $I_-^{(u)}$ with probability θ to yield a sampled set of negative items $I_-^{(u)}(\theta)$. Choose θ so that the total number of sampled negative items $\sum_u |I_-^{(u)}(\theta)|$ is on the order of the total number of positive items $\sum_u |I_+^{(u)}|$. The regularized pseudo-likelihood of equation (4) is then approximated by

$$\mathcal{L}(\mu,\eta) \approx \sum_u \Bigg(\sum_{i \in I_+^{(u)}} \log p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) + \frac{1}{\theta} \sum_{i \in I_-^{(u)}(\theta)} \log p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \Bigg) + \alpha(||\mu||^2 + ||\eta||^2) \tag{5}$$

Now, evaluating the training criterion only requires computing the conditional likelihood for $O(V)$ items, where V is the number of votes observed. Of course, the speedup depends on how sparse votes are, and the loss in accuracy is data dependent.

The gradients then become

$$\frac{\partial}{\partial \mu_k} \mathcal{L}(\mu,\eta) \approx \sum_u \Bigg(\sum_{i \in I_+^{(u)}} f_k^{(i)} \left(v_i^{(u)} - p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \right) + \frac{1}{\theta} \sum_{i \in I_-^{(u)}(\theta)} f_k^{(i)} \left(v_i^{(u)} - p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \right) \Bigg) + 2\alpha\mu$$

and

$$\frac{\partial}{\partial \eta_k} \mathcal{L}(\mu,\eta) \approx \sum_u \Bigg(\sum_{i \in I_+^{(u)}} f_k^{(i)} \left(v_i^{(u)} - p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \right) \sum_{j \neq i} f_k^{(j)} v_j^{(u)} + \frac{1}{\theta} \sum_{i \in I_-^{(u)}(\theta)} f_k^{(i)} \left(v_i^{(u)} - p(v_i^{(u)} | v_{-i}^{(u)};\ \mu,\eta) \right) \sum_{j \neq i} f_k^{(j)} v_j^{(u)} \Bigg) + 2\alpha\eta$$

4. COLD START MOVIE RECOMMENDATIONS

We evaluate our model on two movie recommender tasks, and compare performance with that of the aspect models described in [18]. In this case, the feature vectors will be as described for the examples in Figures 1 and 2–we will have a feature component for each actor, which will be 1 if the actor plays a headlining role in the movie and 0 otherwise.

We use the 1,000,000 rating MovieLens data set for all experiments (see www.movielens.org). This data set contains 1,000,209 ratings of 3706 movies by 6040 users. A rating consists of an assignment of one to five stars to a movie. Not all movies are rated by all users. The MovieLens movie description pages were crawled to obtain the headlining actors of each of these movies. This yielded 8406 actors.

We randomly divided the movies into 2962 training movies and 744 test movies, thereby simulating a cold start situation. We randomly selected 500 users to evaluate recommendations on, yielding a test set of 372,000 possible recommendations. We evaluate using two tasks on this data set. The first is to predict which of the test movies a user rated given the set of training movies he rated. There were 16,717 ratings in the test set. The second is to predict which of the test movies a user rated four stars or higher given the training movies he rated four stars or higher. There were 9592 ratings of four stars or higher in the test set. Thus, the first task is an implicit rating one and the second is an explicit rating one. In the terminology of [18], the first is a "weak occurrence prediction" task while the second is a "rating prediction" task.

We measure performance on these tasks using global ROC (GROC) and customer ROC (CROC) curves as described in [18]. The GROC is an ROC curve that measures performance when the system is allowed to recommend a different number of items (even no items) to some users. The CROC is a modified ROC curve that measures performance when the system is forced to make the same number of recommendations to each user. We recall that ROC curves plot the sensitivity vs. $(1 - \text{specificity})$ of a statistical test. While sensitivity is equivalent to the recall measure used in information retrieval, specificity differs from precision in that it measures the proportion of negative examples correctly identified (i.e., $1-$false positive rate) in contrast to precision which measures the proportion of examples labeled as positive that are correctly labeled. Thus, the GROC and CROC curves are subtly different measures than the precision-recall curves that are used in information retrieval problems. In addition to the areas under the curves (AUCs), [18] notes that it is important to focus attention on the left-hand portion of the GROC and CROC curves which corresponds to the low false positive region of operation.

We note that our evaluation differs from that of [18] in the following three ways. First, we evaluate on the full 1,000,000 rating MovieLens data set instead of the 100,000 rating MovieLens data set. Second, we use staring actors from MovieLens movie description pages instead of first billed actors from the Internet Movie Database (IMDB) for content information. MovieLens typically lists fewer staring actors per movie that IMDB lists as first billed actors. Since we use less content information, ours is arguably a more stringent evaluation. Third, we evaluate on a "rating prediction" task where the goal is to predict the rating a user would give a movie rather than a "rating imputation" task where the goal is to predict the rating a user gave a movie *given that they saw it.* Rating imputation is arguably less useful in recommender systems–the user gets little utility from a prediction of his rating *after* he has already watched the movie. However, we note that solutions of both the "weak occurrence prediction" and "rating prediction" tasks necessarily give a solution to the "rating imputation" task as well.

4.1 The Baseline Model

We use the user-actor aspect model described in [18] as our baseline. While more complicated models are discussed in [17] we used the user-actor model since the more complicated approaches were not reported to significantly improve performance. We now outline the approach of [18].

The user-actor aspect model fits a joint distribution of the form

$$p(u, a) = \sum_z p(z)p(u|z)p(a|z)$$

to a user-actor contingency table. The aspect model is trained using the tempered EM algorithm [6]. Given a novel movie i, it is "folded in" to the latent space through the following EM algorithm:

E-Step:

$$p(z|a, i) \propto p(a|z)p(z|i)$$

M-Step:

$$p(z|i) \propto \sum_z n(a, i)p(z|a, i)$$

where $n(a, i)$ is the number of times actor a appears in movie i, to yield $p(z|i)$. This enables the computation of

$$p(u|i) = \sum_z p(u|z)p(z|i)$$

Items i are ranked according to $p(u|i)$ in order to recommend items to user u. We used the PennAspect implementation for our experiments [16]. Runtime was on the order of hours on a 3 GHZ Xeon workstation with 16GB of memory.

4.2 Tied Boltzmann Machines for Movie Recommendations

In order to train tied Boltzmann machines for these tasks, we represent the movie content using vectors of actors. In other words, we set the dimensionality D of the content feature vectors to the total number of actors, and identify each dimension of the feature vectors with an actor. Then, we have

$$f_k^{(i)} = \begin{cases} 1 & \text{if actor } k \text{ stars in movie } i. \\ 0 & \text{otherwise} \end{cases}$$

Thus, our model has parameters μ_k that model the popularity of movies starring actor k and parameter η_k modeling the co-occurrence of movies starring actor k. While components of the feature vectors $f^{(i)}$ given here correspond to the same kind of content, this is not necessary in general. Because we train the model conditionally, it is easy to combine content features of different kinds in the same vector [8]. For example, we could have components corresponding to actors, directors, writers, and genres.

We train our model using the RPROP [13] algorithm to optimize the regularized sampled pseudo-likelihood training criterion of equation (5). This is a non-linear gradient technique that maintains an adaptive step size for each dimension. The step size during optimization was initialized to $1/D$, the inverse dimensionality of the feature vector, the step growth and shrinkage parameters of the RPROP algorithm were set to 1.2 and 0.5, while backtracking was enabled. The algorithm was run for at most 100 iterations. The regularization parameter α was set to 1000, but we observed that the results were not very sensitive to this value.

The sampling parameter θ was set to 0.05 for the "weak occurrence prediction" task and to 0.03 for the "rating prediction" task to balance the number of positive and negative examples in the training set. The sampling reduces the number of training examples considered, and therefore the training time, by a factor of about 100 for these data sets. Actual runtime was a few minutes on a 3 GHZ Xeon workstation with 16GB of memory. While formal evaluation of the resulting accuracy loss is left for future work, increasing θ by a factor of four, thus increasing training complexity by a factor of four, resulted in no change in the CROC and GROC curves. This strongly suggests that sampled training results in little accuracy loss.

4.3 Results

Figure 3 shows GROC and CROC curves comparing the tied Boltzmann machine model to the user-actor aspect model on the implicit rating prediction problem. The AUCs for the curves are also indicated in the figure. While the ideal GROC curve would follow the left and upper boundaries of the plot, having AUC 1.0, the ideal CROC curve is on the interior of the plot, having AUC less than 1.0. This is because not all the users rate the same number of items, while the CROC evaluation forces the system to recommend the same number of items to each user. For this reason, we also show the CROC curve of an ideal recommender to give an upper bound on performance. The GROC and CROC curves for random recommendations with AUC 0.5 are also shown.

The tied Boltzmann machine outperforms the aspect model in terms of area under the GROC and CROC curves, as well as in terms of the GROC and CROC performance in the low false positive region. The difference is particularly striking in the case of the CROC curves. Figure 4 repeats the analysis for the explicit rating prediction problem. Once again, the tied Boltzmann machine outperforms the aspect model, particularly in the case of the CROC curves. Although we did not perform a formal statistical significance test, we note that out test sets contained 372,000 test cases of which on the order of 1000 were positive for each task.

The aspect models tended to over-train on our sparse data set, as remarked upon in [18]. We found that when the annealing weight (inverse temperature) was reduced enough to prevent over-training, the resulting models did not discriminate between the latent classes. That is, at this point, the models were effectively using a single latent class. We verified this by training the model with various numbers of latent classes, including a single class. These models were identical in terms of performance. Since the aspect models rank recommendations by the probability of users given movies, the resulting degenerate models recommend all the movies to each user in turn, ordering users in terms of their weight in the training set.

5. DISCUSSION

We have shown that tied Boltzmann machines are effective at generating cold start recommendations, even on sparse data sets. We conjecture that the superior performance of tied Boltzmann machine over aspect models in the tasks we have presented can be attributed to two factors. First, tied Boltzmann machines have a relatively parsimonious parametrization that is easily regularized to combat over-training. Second, tied Boltzmann machines provide conditional distributions over ratings. In contrast, the folding-in procedure for new items means that content-user aspect models only provide a distribution over users, when what is desired is a distribution over items.

We also propose three directions for extending the ideas present in this paper. First, as mentioned briefly in section 4.2, tied Boltzmann machines provide a simple framework for combining different kinds of content information. In addition, given a recommendation, a sensitivity analysis of the conditional probability of the recommended item with respect to the different content features $f_k^{(i)}$ can be used to determine which content information was most responsible for producing the recommendation. This could be used in explaining the reasons behind the system's recommendations to users, making the system interpretable. Second, since Boltzmann machines can model pairwise collaborative effects directly while tied Boltzmann machines estimate collaborative effects from content information, it is natural to investigate the combination of these two approaches through softening the parameter-tying constraints. Third, the model can be extended to multinomial votes. Such an extension could attempt to model the product space of possible pairwise interactions, or could use interaction terms that depend on the difference of the vote pair.

6. ACKNOWLEDGMENTS

The authors thank David Maxwell Chickering and Guy Shani for their assistance, as well as the PennAspect team for making their software available online.

7. REFERENCES

[1] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, Sept. 1975.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predctive algorithms for collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence*, 1998.

[3] S. F. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University, 1999.

[4] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *National Conference on Artificial Intelligence*, 1999.

[5] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, Oct. 2000.

[6] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of ACM SIGIR*, 1999.

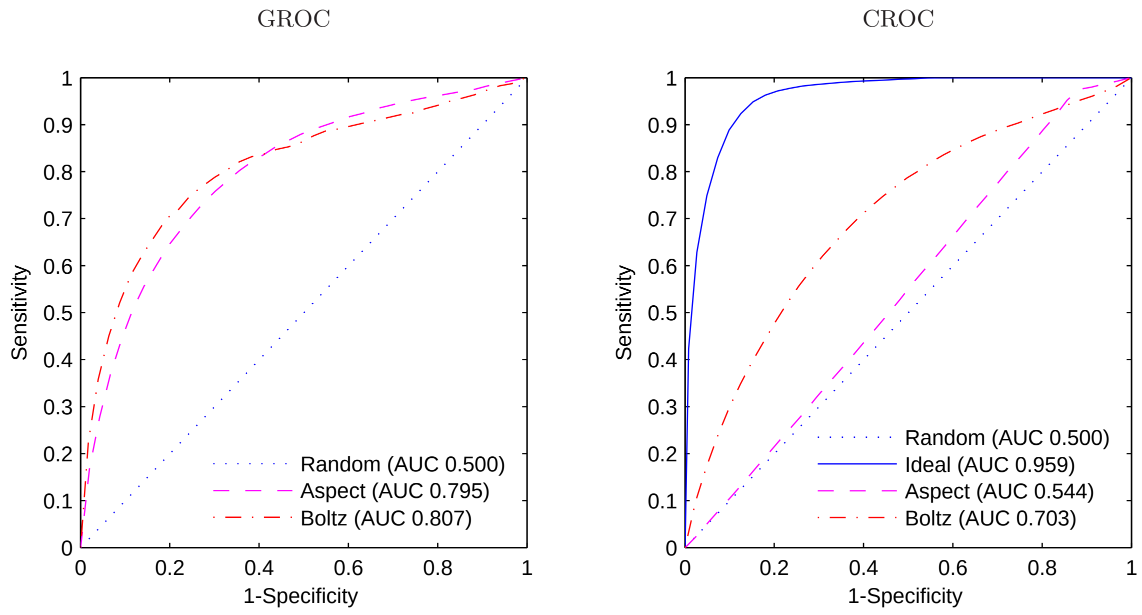


Figure 3: GROC and CROC curves for the implicit rating task.

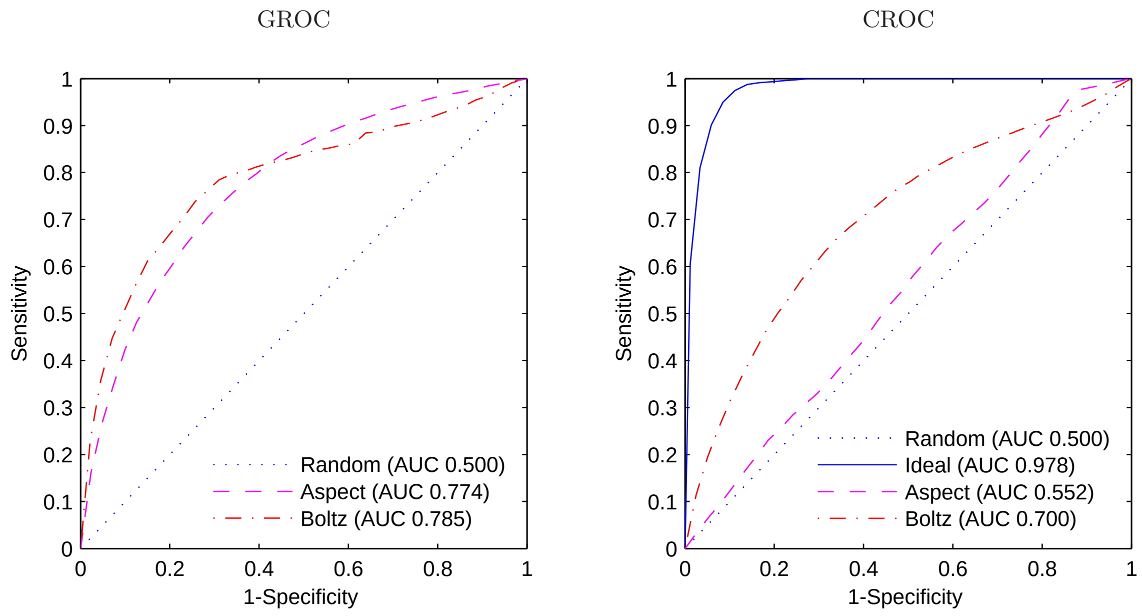


Figure 4: GROC and CROC curves for the explicit rating task.

[7] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *International Joint Conference on Artificial Intelligence*, 1999.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

[9] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing Magazine*, 7(1), 2003.

[10] M. Montaner, B. López, and J. L. de la Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19(4), 2003.

[11] S.-T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste. Na ive filterobts for robust cold-start recommendations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 699–705, 2006.

[12] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdel, editors, *The Adaptive Web: Methods and Strategies of Web Presentation*, Lecture Notes in Computer Science. Springer, 2007.

[13] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, volume 1, pages 586–591, 1993.

[14] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning*, 2007.

[15] B. Sarwar, G. K. nad Joseph Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web Conference*, 2001.

[16] A. I. Schein, A. Popescul, and L. H. Ungar. Pennaspect: A two-way aspect model implementation. Technical Report MS-CIS-01-25, University of Pennsylvania, 2003.

[17] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Generative models for cold-start recommendations. In *SIGIR Workshop on Recommender Systems*, 2001.

[18] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of ACM SIGIR*. ACM, Aug. 2002.