2010 IEEE International Conference on Data Mining

Learning Attribute-to-Feature Mappings for Cold-Start Recommendations

Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle and Lars Schmidt-Thieme
Machine Learning Group, University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany
Email: {gantner, ldrumond, freudenthaler, srendle, schmidt-thieme}@ismll.de

***Abstract*—Cold-start scenarios in recommender systems are situations in which no prior events, like ratings or clicks, are known for certain users or items. To compute predictions in such cases, additional information about users (user attributes, e.g. gender, age, geographical location, occupation) and items (item attributes, e.g. genres, product categories, keywords) must be used.**

We describe a method that maps such entity (e.g. user or item) attributes to the latent features of a matrix (or higher-dimensional) factorization model. With such mappings, the factors of a MF model trained by standard techniques can be applied to the new-user and the new-item problem, while retaining its advantages, in particular speed and predictive accuracy.

We use the mapping concept to construct an attribute-aware matrix factorization model for item recommendation from implicit, positive-only feedback. Experiments on the new-item problem show that this approach provides good predictive accuracy, while the prediction time only grows by a constant factor.

I. Introduction

Matrix and tensor factorization are well-suited methods for solving several problems in the field of recommender systems, like rating prediction for a given user and item (e.g. a movie or a book) [14], recommending a set of items to a given user [3], or predicting tags for a certain item to a user [23]. Because predictions from factorization models rely on the computation of simple dot products of latent feature vectors representing users, items, and possibly other entities in the application domain, they usually have good runtime performance. Training with regard to suitable optimization objectives usually leads to good predictive accuracy.

The downside of standard factorization methods is that feature vectors are only available for entities observed in the training data, e.g. users who rated a movie or bought a book, or movies rated by at least one user. Thus for entirely new users and items, such methods are not capable of computing meaningful recommendations. Even many hybrid systems that rely on both collaborative and content information cannot provide useful predictions for entirely new entities, i.e. ones that have no collaborative information associated with them.

In real-world recommender systems, such cold-start problems[1] are often solved by switching to a different, purely content-based method when encountering entirely new entities; other options are to present just the most popular items to new users and to randomly present new items to the users in order to gather collaborative information about those new entities.

Our approach is a modular one, with well-defined interfaces between its components: At the core of our model is a standard factorization model that only works for entities with collaborative training data. This factorization model is optimized for the given recommendation task. The additional components are mapping functions that compute adequate latent feature representations for new entities from their attribute representations.

For example, in the classical recommendation task of movie rating prediction, this approach would handle new users and new items by computing first the latent feature vectors for the unknown entities from attributes like the user's age or location and a movie's genres or main cast, and then by using those estimated latent feature vectors to compute the rating from the underlying matrix factorization (MF) model.

The training of such a combined model consists of learning the underlying standard model from the collaborative data, and then learning the mapping functions from the pairs of latent feature vectors and attribute vectors belonging to entities that are present in the collaborative data.

Note that this mapping approach is applicable to a variety of prediction tasks, underlying factorization models, and families of mapping functions. In the following, we describe the use of this framework for the task of item recommendation from implicit, positive-only feedback using a matrix factorization model optimized for Rendle et al.'s Bayesian Personalized Ranking (BPR) [22], and demonstrate its usefulness for the new-item recommendation task with a set of experiments.

The main contributions of this work are

1) a general, simple and straightforward method to make factorization models attribute-aware by plugging learnable mapping functions onto them, and,
2) based on that method, an extension of matrix factorization optimized for Bayesian Personalized Ranking (BPR-MF) that can deal with the cold-start problem, yielding accurate and fast attribute-aware item recommendation methods based on different families of

[1] In this article, we use the terms "cold start", "new item", and "new user" in the narrower sense; see section II-B.

1550-4786/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDM.2010.129

Table I
ATTRIBUTE EXAMPLE: MOVIE GENRES

ID	Movie	Genres
1	*The Usual Suspects*	Crime, Thriller
2	*American Beauty*	Comedy, Drama
3	*The Godfather*	Crime, Action
4	*Road Trip*	Comedy

mapping functions.

3) We also show empirically that it is worth training the mapping function for optimal model performance with respect to application-specific losses, instead of just trying to map the latent features as accurately as possible.

II. PROBLEM STATEMENT

In a classical recommender system, there are two types of entities, *users* (e.g. customers) and *items* (e.g. movies, books, songs). Throughout this paper, we use $U = \{1, \dots, |U|\}$ and $I = \{1, \dots, |I|\}$ to denote the sets of all user and all item IDs, respectively. For simplicity, we will not differentiate between the integer ID representing an entity and the entity itself.

We have different kinds of information about the entities:

1) information pertaining to one entity, *content information*, e.g. user attributes like age, gender, hobbies or item attributes like the price of a product, words in the title or description of a movie, editorial ratings, and
2) information that is linked to a user-item pair, *collaborative information*, e.g. the rating "4 stars" on a scale from one to five given to a movie by a specific user, the information that a user has purchased an item in an online shop or viewed a video in an IPTV system, or a tag in a collaborative tagging system.

There are several types of collaborative information. One important distinction is between *explicit* (e.g. ratings, up- and downvotes) and *implicit* expressions of user preferences (e.g. clicks, purchases). Depending on the type of system, implicit information may be *positive-only*, i.e. there may be no recorded negative preference observations.

We represent positive-only implicit feedback as a binary matrix $\mathbf{S} \in \{0,1\}^{|U|\times|I|}$, where s_{ui} is 1 iff user u has given positive feedback about item i. $\mathbf{A^U} \in \mathbb{R}^{|U|\times m}$ be the matrix of user attributes where a^U_{ul} is 1 iff user u has attribute l, $\mathbf{A^I} \in \mathbb{R}^{|I|\times n}$ be the matrix of item attributes, where a^I_{il} is 1 iff item i has attribute l. There are m different user attributes, and n item attributes. We assume that attributes are known for all users and items, e.g. because all items are movies where attributes like genre, participating actors etc. are known.

Example 1: Suppose we have the movies *The Usual Suspects*, *American Beauty*, *The Godfather*, and *Road Trip* in our recommender system. Each of those items is assigned to one or several of the genres *Crime*, *Thriller*, *Comedy*, *Drama*, and *Action*. If we assign consecutive IDs to the movie and genres, we can create the following item attribute matrix from the contents of Table I:

$$\mathbf{A^I} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

where the rows refer to the different movies, and the columns refer to the different genres. We will use this data in the following examples.

A. Item Recommendation from Implicit Feedback

The task of *item recommendation* from implicit, positive-only feedback [9], [10], [16], [17], [22] is to rank the items from a candidate set I^{cand} according to the probability of being viewed/purchased by a given user, based on the feedback matrix $\mathbf{S}$ and possibly additional data $\mathbf{A^U}, \mathbf{A^I}$.

We use $I^+_u := \{i \in I : s_{ui} = 1\}$ to refer to the items for which user u has provided feedback and $I^-_u := \{i \in I : s_{ui} = 0\}$ to refer to the items for which that user has not provided feedback.

Note that item recommendation is related to, but distinct from, *rating prediction*, where the task is to predict how much a user will like an item – or rather, what explicit rating the user will assign to the item.

Example 2: Suppose we have the users *Alice*, *Ben*, and *Christine*. None of them has watched *The Usual Suspects*; *Christine* has watched all three other movies, while *Alice* and *Ben* each have only watched *American Beauty* and *The Godfather*, respectively. If we assign IDs to all entities in order of their appearance, we have

$$\mathbf{S} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

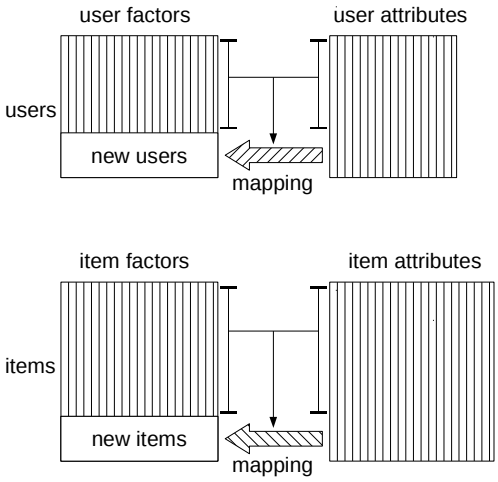
From $\mathbf{S}$, we can deduce the sets $I^+_1 = \{2\}$ and $I^-_1 = \{1, 3, 4\}$ for *Alice* (user 1). Note that we only see positive feedback here: We cannot deduce that *Alice* and *Ben* do not like the other movies only because they have not watched them.

B. Cold-Start Scenarios

In a wider sense, *cold-start scenarios* are those situations where we want to compute predictions for users or items that have few collaborative information [7], [21]; in the narrow sense, cold-start scenarios are exactly those scenarios in which there is no collaborative information at all for the given users or items [9], [10], [18]. In this article, we use the term in the latter sense.

Example 3: In our small example, *The Usual Suspects* would be a new (cold-start) item.

Figure 1. Mapping framework, see section III-A



III. Methods

In this section, we describe the framework we have sketched in the introduction, and use it for the task of item recommendation from implicit, positive-only feedback.

A. High-Level Framework

In factorization models, every entity (e.g. users, items, tags) is represented by a latent feature vector $\mathbf{f} \in \mathbb{R}^k$. Usually, the latent features of an entity can only be set to meaningful values during training if the entity occurs in the (collaborative) training data.

If this is not the case, one way to still make use of the factorization model for new entities is to estimate their latent features from the existing content data: to map from the attribute space to the latent feature space. The recommender system could then use the factorization model to compute scores for all kinds of entities; latent feature vectors for new entities would be computed from the content attributes and further on used as if they were normally trained latent features.

The mapping functions could theoretically take any form, although for practical purposes we will limit them to families of functions that allow the learning of useful mapping functions.

The training of a factorization model with a mapping extension consists of the following steps:

1) training the factorization model using the data $\mathbf{S}$, and then
2) learning the mapping functions from the latent features of the entities in the training data and their content attributes.

Figure 1 illustrates the framework for a domain involving users and items: The rectangles on the left-hand side represent the factor matrices, the ones on the right-hand side the attribute matrices. Attributes are supposed to be known for all entities (vertical hatching), while factors are initially only known for those entities that occur in the training data (vertical hatching). Entities without collaborative data have no factors (blank). The unknown entity factors are estimated using the corresponding mapping function. The mapping functions are learned from the factor and attribute values of the entities with complete information (thin arrows). Note that this framework can be extended to application domains with additional entity types besides users and items.

B. Matrix Factorization Model

To exemplify how attribute-to-feature mappings can be used for item recommendation from implicit data, we use BPR-MF, a matrix factorization model based on the Bayesian Personalized Ranking (BPR) framework [22]. Bear in mind that the general framework presented here can be applied to other matrix factorization models, as well as to any other model where the entities of the application domain are represented by latent feature vectors, like Tucker decomposition [27] or PARAFAC [11]. In the examples and experiments, we focus on new items; new users (or other kinds of entities) can be handled analogously.

1) Bayesian Personalized Ranking: BPR is a framework for optimizing different kinds of models based on training data containing implicit feedback or other kinds of implicit and explicit (partial) ranking information. It has been successfully applied to k-nearest-neighbor (kNN), matrix factorization, and different tensor factorization models for the tasks of item recommendation [22] and personalized tag prediction [23]. BPR's key ideas are to consider entity pairs instead of single entities in its loss function, which allows the interpretation of positive-only data as partial ranking data, and to learn the model parameters using a generic algorithm based on stochastic gradient descent.

2) Matrix Factorization based on BPR: BPR-MF approximates the score matrix $\hat{\mathbf{Y}}$ by the product of two low-rank matrices $\mathbf{W} \in \mathbb{R}^{|U|\times k}$ and $\mathbf{H} \in \mathbb{R}^{|I|\times k}$. For a specific user u and item i, the score estimation is

$$\hat{y}_{ui} = \sum_{f=1}^{k} w_{uf} h_{if} = \langle \mathbf{w}_u, \mathbf{h}_i \rangle. \tag{1}$$

For estimating whether a user prefers one item over another, we optimize for the BPR-Opt criterion:

$$\text{BPR-Opt} = \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda ||\Theta||^2, \tag{2}$$

where $\hat{x}_{uij} := \hat{y}_{ui} - \hat{y}_{uj}$ and $D_S = \{(u,i,j)|i \in I_u^+ \wedge j \in I_u^-\}$. $\Theta = (\mathbf{W}, \mathbf{H})$ represents the parameters of the model and λ is a regularization constant. σ denotes the logistic function:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{3}$$

Figure 2. Optimizing for BPR-OPT using stochastic gradient descent with replacement

procedure LEARNBPR(D_S)
  initialize Θ
  repeat
    draw (u, i, j) from D_S
    $\hat{x}_{uij} \leftarrow \hat{y}_{ui} - \hat{y}_{uj}$
    $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \theta} \hat{x}_{uij} - \lambda \theta \right)$
  until convergence
  return Θ
end procedure

3) Learning Algorithm: For learning the MF model, we use the LEARNBPR algorithm [22], which is a variant of stochastic gradient descent that samples from D_S. To apply LEARNBPR to MF, only the gradient of $\hat{x}_{uij}$ with respect to every model parameter has to be derived. The pseudocode of the generic algorithm is shown in Figure 2; α is the learning rate (step size). A more detailed description of BPR-MF and its theoretical background can be found in [22].

Example 4: Training a hypothetical factorization model with $k = 2$ yields two matrices consisting of the user and item factor vectors, respectively:

$$\mathbf{W} = \begin{pmatrix} 0.2 & 1.2 \\ 1.3 & 0.3 \\ 0.9 & 1.1 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} ? & ? \\ 0.9 & 1.0 \\ 1.1 & 0.2 \\ 0.1 & 1.2 \end{pmatrix}.$$

Every row in $\mathbf{W}$ corresponds to one user, which means that row 1 represents *Alice*, row 2 *Ben*, and row 3 *Christine*. In $\mathbf{H}$, each row corresponds to exactly one movie. Suppose that *The Usual Suspects* has not yet been added to the content catalog, so row 1 does not contain any meaningful values. We can compute item recommendations for *Alice* by ranking her previously unseen movies according to their predicted scores:

$$\hat{y}_{1,3} = \langle \mathbf{w}_1, \mathbf{h}_3 \rangle = 0.2 \cdot 1.1 + 1.2 \cdot 0.2 = 0.46$$
$$\hat{y}_{1,4} = \langle \mathbf{w}_1, \mathbf{h}_4 \rangle = 0.2 \cdot 0.1 + 1.2 \cdot 1.2 = 1.46.$$

Because the score for *Road Trip* is 1.46, the system would put it further on top of the result list than *The Godfather*, which only has a score of 0.46. If we want to make a prediction for *The Usual Suspects*, we need to estimate its factors from its attributes.

C. Attribute-to-Feature Mappings

In this section, we show how to design attribute-to-feature mappings for items; user-attribute-to-feature mappings can be designed accordingly.

The general form of score estimation by mapping from item attributes to item factors is

$$\hat{y}_{ui} := \sum_{f=1}^{k} w_{uf} \phi_f(\mathbf{a}^{\mathbf{I}}{}_i) = \langle \mathbf{w}_u, \phi(\mathbf{a}^{\mathbf{I}}{}_i) \rangle, \tag{4}$$

Table II
COSINE SIMILARITIES BETWEEN MOVIES

Movie	*US*	*AB*	*TG*	*RT*
The Usual Suspects	1	0	0.5	0
American Beauty	0	1	0	0.5
The Godfather	0.5	0	1	0
Road Trip	0	0.5	0	1

where $\phi_f : \mathbb{R}^n \to \mathbb{R}$ denotes the function that maps the item attributes to the factor with index f, and $\phi : \mathbb{R}^n \to \mathbb{R}^k$ denotes the vector-valued function that maps the item attributes to all item factors.

1) K-Nearest-Neighbor Mapping: One approach to map the attribute space to the factor space is to use weighted k-nearest-neighbor (kNN) regression [12] for each factor.

We determine the k nearest neighbors N_k as the most similar items according to the cosine similarity [15] of the attribute vectors. Each factor is then estimated by

$$\phi_f(\mathbf{a}^{\mathbf{I}}{}_i) := \frac{\sum_{j \in N_k(i)} sim(\mathbf{a}^{\mathbf{I}}{}_i, \mathbf{a}^{\mathbf{I}}{}_j) h_{jf}}{\sum_{j \in N_k(i)} sim(\mathbf{a}^{\mathbf{I}}{}_i, \mathbf{a}^{\mathbf{I}}{}_j)}. \tag{5}$$

vector too!

The cosine similarity is given by

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}. \tag{6}$$

Note that for other kinds of attribute data (e.g. strings, real numbers), other similarity metrics could be employed.

Example 5: The cosine similarities of the different items are given in Table II. The factors of *The Usual Suspects*, estimated by 1-NN, would be

$$\hat{\mathbf{h}}_1 := \phi(\mathbf{a}^{\mathbf{I}}{}_i) = \begin{pmatrix} \frac{0.5 \cdot h_{3,1}}{0.5} \\ \frac{0.5 \cdot h_{3,2}}{0.5} \end{pmatrix} = \begin{pmatrix} 1.1 \\ 0.2 \end{pmatrix}.$$

With this estimation, we can compute a score for the new item:

$$\hat{y}_{1,1} = \langle \mathbf{w}_1, \hat{\mathbf{h}}_1 \rangle = 0.2 \cdot 1.1 + 1.2 \cdot 0.2 = 0.46.$$

This result means that we would still recommend *Road Trip* to *Alice*.

2) Linear Mapping: For score estimation with a linear mapping to the item factors, we plug linear functions into equation (4):

$$\phi_f(\mathbf{a}^{\mathbf{I}}{}_i) = \sum_{l=1}^{n} m_{fl} a^I_{il} = \langle \mathbf{m}_f, \mathbf{a}^{\mathbf{I}}{}_i \rangle. \tag{7}$$

Each item factor is expressed by a weighted sum of the item attributes.

Example 6: Suppose we have trained a linear mapping model with the following weights:

$$\mathbf{M} = \begin{pmatrix} 0.7 & 0.0 & 0.1 & 1.0 & 0.7 \\ 0.1 & 0.0 & 0.8 & 1.1 & 0.0 \end{pmatrix}.$$

The rows in matrix $\mathbf{M}$ correspond to the different latent features, while the columns denote the influence of each attribute to the latent features. Then the latent feature estimates are

$$\hat{\mathbf{h}}_1 = \begin{pmatrix} 1 \cdot 0.7 + 1 \cdot 0.0 + 0 \cdot 0.1 + 0 \cdot 1.0 + 0 \cdot 0.7 \\ 1 \cdot 0.1 + 1 \cdot 0.0 + 0 \cdot 0.8 + 0 \cdot 1.1 + 0 \cdot 0.0 \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.1 \end{pmatrix},$$

and the score for *Alice* and *The Usual Suspects* is

$$\hat{y}_{1,1} = \langle \mathbf{w}_1, \hat{\mathbf{h}}_1 \rangle = 0.2 \cdot 0.7 + 1.2 \cdot 0.1 = 0.26.$$

Optimizing for Least Squares Error: One way to learn suitable parameters for the linear mapping functions is optimizing the model for the (regularized) squared error on the latent features, i.e. straightforward ridge regression [12]. Because the number of input variables (attributes) can be in the tens of thousands, we use stochastic gradient descent for training. This simple approach did not yield optimal results (see section IV-D), so we investigated another mapping method, which is explained next.

Optimizing for BPR-OPT*:* To optimize the parameters of the linear functions, $\Theta = \mathbf{M} \in \mathbb{R}^{k \times n}$, for the BPR-OPT criterion (of the prediction model in equation 4) is a more suitable approach, because it fits the parameters leading to optimal model performance, rather than just accurately approximating the latent feature values. As stated above, when optimizing for BPR-OPT, we are interested in the difference between two item scores for the same user:

$$\hat{y}_{ui} - \hat{y}_{uj} = \sum_{f=1}^{k} w_{uf} \sum_{l=1}^{n} m_{fl} a^I_{il} - \sum_{f=1}^{k} w_{uf} \sum_{l=1}^{n} m_{fl} a^I_{jl}. \quad (8)$$

Note that introducing a bias term m_{0f} (via an artificial attribute that is always set to 1) does not make sense for item mappings, because the bias part would be exactly the same for both sums.

This can be simplified to

$$\begin{aligned} \hat{y}_{ui} - \hat{y}_{uj} &= \sum_{f=1}^{k} \sum_{l=1}^{n} w_{uf} m_{fl} a^I_{il} - \sum_{f=1}^{k} \sum_{l=1}^{n} w_{uf} m_{fl} a^I_{jl} \\ &= \sum_{f=1}^{k} \sum_{l=1}^{n} w_{uf} m_{fl} (a^I_{il} - a^I_{jl}). \end{aligned}$$

For training with LEARNBPR (see section III-B3), we need the partial derivative with respect to m_{lf} for $f \in \{1 \dots k\}, l \in \{1 \dots n\}$:

$$\frac{\partial}{\partial m_{fl}} (\hat{y}_{ui} - \hat{y}_{uj}) = w_{uf} (a^I_{il} - a^I_{jl}). \quad (9)$$

The resulting learning algorithm for the linear mapping model optimized for BPR-OPT is shown in Figure 3.

Figure 3. Learning algorithm for linear mapping: The score difference $\hat{y}_{ui} - \hat{y}_{ui}$ is defined in equation 8.

procedure LEARNBPR($D_S, \mathbf{W}, \mathbf{H}, \mathbf{A^I}$)
  initialize $\mathbf{M}$
  repeat
    draw (u, i, j) from D_S
    $\hat{x}_{uij} \leftarrow \hat{y}_{ui} - \hat{y}_{uj}$
    for $1 \le f \le k$ **do**
      $\mathbf{m}_f \leftarrow \mathbf{m}_f + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot w_{uf} (\mathbf{a^I}_i - \mathbf{a^I}_j) - \lambda \mathbf{m}_f \right)$
    end for
  until convergence
  return $\mathbf{M}$
end procedure

D. Run-Time Overhead

Generally, the runtime overhead of adding mapping functions to an existing factorization model is low. For each new entity, the factors need to be estimated once, and can be either be stored in the pre-existing factor matrices, or in special data structures. After that, the computation of a prediction takes the same time as with just the underlying model. Note that factorization models themselves are among the fastest state-of-the-art methods. The experimental part of this paper contains a comparison in section IV-F that shows the method's advantage over classical content-based filtering.

IV. EXPERIMENTS

We performed experiments to confirm that our approach is able to produce useful new-item cold-start recommendations. We compare the two mapping methods described in section III to other approaches capable of solving the new-item cold-start problem (section IV-D). We also investigated how the number of attributes affects the prediction accuracy (section IV-E).

A. Datasets

For the experiments, we use the MovieLens 1M dataset[2], which is a commonly-used rating dataset [10], [18]. Like [10], we do not use the rating values, but just binary rating events, assuming that users tend to rate movies they have watched. To evaluate the performance of recommender algorithms in the presence of new items, we randomly split the items in the dataset into 5 groups of roughly equal size, and assign all corresponding rating events, to perform 5-fold cross-validation.

As attributes, we use the genre information included with the MovieLens dataset, and additional information from the Internet Movie Database (IMDB)[3]. Table III gives an overview of the attribute sets. All attributes used in the

[2] http://www.grouplens.org/
[3] http://www.imdb.com/interfaces, downloaded April 16, 2010

Table III
ATTRIBUTE SETS

Name	Source	Number	Sparsity
genres	MovieLens	18	90.83 %
directors	IMDB	479	99.59 %
actors	IMDB	16,149	99.91 %
credits	IMDB	17,739	99.91 %

evaluation are nominal (set-valued), their representation is binary, i.e. every possible attribute is represented by one number that is 1 if the item has the attribute, and 0 if not. *Number* refers to the number of attributes in the set, and *Sparsity* refers to the relative number of zero values in the movies' attribute representations, the matrix $\mathbf{A^I}$. Note that the methods described in this paper also would work for real-valued attributes. The *credits* attribute set contains actors, directors, producers, writers, cinematographers, etc. involved with the movies; it is a superset of the other two IMDB attribute sets.

B. Compared Methods

We compared three mapping methods with two baseline methods. For the mapping methods, we computed BPR-MF models (see section III-B) with $k = 32$ factors using hyperparameters that yielded satisfactory results in non-cold-start evaluations.[4]

We also performed the experiments with different numbers of factors ($k \in \{32, 56, 80, 120, 160\}$), and got similar results.

non-parametric

`map-knn`: The kNN-based mapping method described in section III-C1; we determined suitable values for k using 4-fold cross-validation on the training data.[5]

`map-lin`: A linear mapping method that uses ridge regression to estimate the latent features from the attributes, described in section III-C2. We determined suitable values for the hyperparameters (learning rate, regularization constant) using 4-fold cross-validation on the training data.

`map-bpr`: The linear mapping method that is optimized for BPR-OPT is described in section III-C2; Again, we determined suitable values for the hyperparameters (learning rate, regularization constant) using 4-fold cross-validation on the training data.

non-parametric

`cbf-knn`: Item-attribute based kNN, for example used in [5]; we used the cosine similarity between the items' binary attribute vectors as the similarity measure. We set $k = \infty$, so scores for user u and item i are computed by summing up the similarities of the item i with the items previously seen by user u:

$$\hat{y}_{ui} = \sum_{j \in I_u^+} sim(i, j). \tag{10}$$

Note that this is content-based filtering using kNN, not attribute-to-factor mapping via kNN as mentioned in section III-C.

`random`: To put the other methods in perspective, we also included the results for predicting a random set of items.

We do not compare against just recommending the most popular items, because in our evaluation protocol there are *only* previously unseen items in the test set, thus there is no popularity information about any of the candidate items.

Unified Boltzmann Machine: In the first experiment, we cite experimental results by Gunawardana and Meek [10], who used a comparable evaluation protocol.

C. Evaluation Metrics

prec@n (precision at n) measures the number of correctly predicted items in the top-n recommendations. It is commonly used in the area of information retrieval [15], is relevant to practice and easy to interpret. We report results for prec@5. We also measured prec@10 in all experiments, which gave the same results regarding the ranking of the methods. Additionally, we report AUC (the area under the ROC curve), which is a more general ranking measure.

$$\text{AUC} = \sum_{\substack{(u,i,j) \in U \times \\ I_u^{\text{test}} \times (I^{\text{cand}} - I_u^{\text{test}})}} z_u \, \delta(\hat{x}_{uij}), \tag{11}$$

where z_u and δ are defined as

$$z_u := \frac{1}{|U||I_u^{\text{test}}||I^{\text{cand}} - I_u^{\text{test}}|}, \quad \delta(x) := \begin{cases} 1, & x \geq 0 \\ 0, & \text{else} \end{cases}.$$

D. Experiment 1: Method Comparison

The comparison of the aforementioned methods on the attribute sets *genres*, *directors*, and a combination of the two sets can be seen in Figures 4 and 5. [10] used a similar evaluation protocol in their cold-start experiments – the same dataset, also an 80-20 split, but only evaluate on 500 randomly selected users, instead of all users. For *genres*, they report about 25% prec@5 for their primary method (Unified Boltzmann Machines). As you can see in Figure 4, the results for `map-bpr` also fall into this region, while `map-knn` and the two baseline methods perform considerably lower. For *directors*, `map-bpr`, `map-knn`, and `cbf-knn` are roughly on par. The comparison of `map-lin` and `map-bpr` shows that is really worth training the mapping function for overall recommendation performance, instead of least squares error on the latent features. Regarding the AUC metric (Figure 4), the results are similar.

Note that for `cbf-knn`, the results deteriorate when the two attribute sets are combined, while the two mapping

[4] $\alpha = 0.01, \lambda_u = 0.02125, \lambda_i = \lambda_j = 0.00355$, 265 iterations

[5] For this and the other methods, we picked the hyperparameter (combinations) with the best prec@5 performance.

Figure 4. Comparison: prec@5

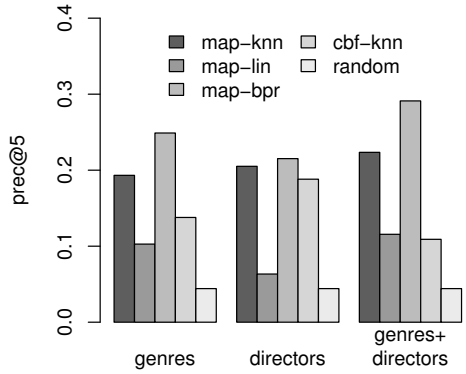


Figure 5. Comparison: AUC

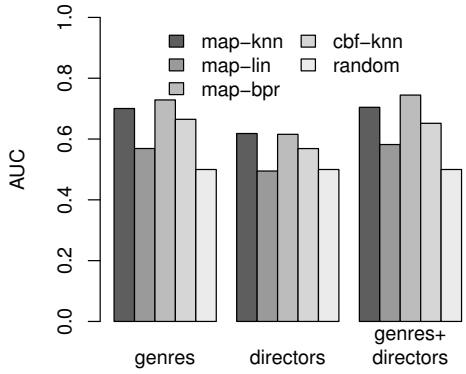


Figure 6. High-dimensional attribute sets: prec@5

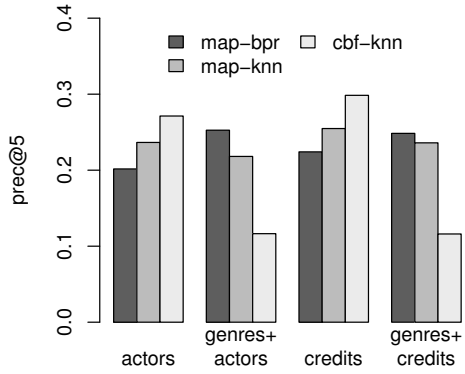
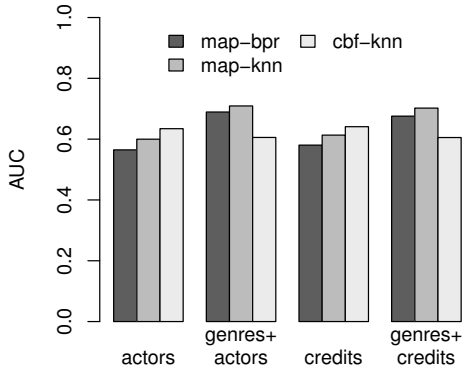


Figure 7. High-dimensional attribute sets: AUC



methods, and in particular `map-bpr`, profit from the additional data. We think that `cbf-knn`'s suboptimal results could be fixed by computing separate item similarities for the different attribute sets and then combining them, but doubt that this will be a stronger method than `map-bpr`.

E. Experiment 2: Large Attribute Sets

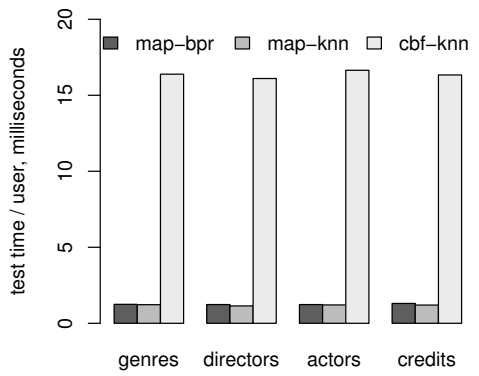
Next, we investigated the methods' performance on larger attribute sets (several thousand attributes). We notice (see Figure 6 and 7) that for large attribute sets the baseline method `cbf-knn` performs better than the mapping methods. Gunawardana and Meek [10] observed similar behavior for their models, Unified Boltzmann Machines and Tied Boltzmann Machines [9]: using only the genre data led to better results than using actor data (there: about 8,000 attributes) or the combined genres+actors data.

Again the combination of attribute sets leads to a deterioration of the prediction quality for `cbf-knn`, while the mapping methods do not suffer from more data.

F. Run-Time Comparison

Figure 8 shows the test times per user for the different methods. The number of factors per entity is again 32 for `map-bpr` and `map-knn`. One can clearly see that the mapping methods profit from the underlying fast matrix factorization model, while the kNN-based content-based filtering `cbf-knn` takes several times longer to compute the predictions.

Figure 8. Comparison: Test time per user



G. Discussion

The experiments have shown that for the new-item recommendation task, BPR-MF in combination with an attribute-to-feature mapping function yields accuracies comparable to state-of-the-art methods like Unified Boltzmann Machines (section IV-D).

The performance on large attribute sets could still be improved over content-based filtering with cosine similarity, however this is a problem that other methods in the literature also suffer from (section IV-E). One reason for this could be that cosine similarity works particularly well for high-dimensional sparse data, and that linear models like `map-bpr` and simple models like `map-knn` (without much adaption to the data) are not powerful enough to make use of large, sparse attribute sets. A remedy may be using a non-linear learned mapping function, e.g. based on multi-layer neural networks, or support vector regression [26].

Additionally, the mapping approaches have the advantage of being much faster (section IV-F) than content-based filtering using kNN.

H. Reproducibility of the Experiments

We have implemented all presented algorithms in the MyMediaLite recommender system algorithm library, which is available for download under http://ismll.de/mymedialite. The library is free/open source software, distributed under the terms of the GNU General Public License.

V. Related Work

Pazzani and Billsus [20] give an overview of content-based methods that can be used for new-item scenarios.

One of the MF variants described in [14] takes attributes into account for the rating prediction task; however it is assumed that for every entity there is also collaborative information available, which makes the model unsuitable for cold-start scenarios in the narrower sense.

Pilaszy and Tikk [21] propose an MF model for rating prediction that maps attributes to the factor space using a linear transformation, based on a method proposed by Paterek [19]. The method (NSVD1) can either handle user or item attributes; predictions are computed from item attributes by

$$\hat{y}_{ui} = \langle \mathbf{w}_u, \sum_{l=1}^{n} m_{fl} a^I_{il} \rangle. \tag{12}$$

This rating prediction method is similar to a special case of the framework presented here, but there are several differences, considering the concrete application as well as the model:

- NSVD1 is for rating prediction, while the models designed in the present paper deal with item recommendation (see section II-A).
- Pilaszy and Tikk's learning algorithm estimates all parameter at once, while we use a two-stage learning scheme (see section III-A).
- If NSVD1 uses user and item attributes at the same time, then there are no free latent features in the models – the rating is estimated entirely from the entities' attributes; our model only uses the entity attributes if no collaborative information is known about the given entity.

Pilaszy and Tikk learn the factors of one entity (e.g. the users) simultaneously with the mapping to the factors of the other entity (e.g. the items), which only exist implicitly via the mapping; the model is not based on a completely trained standard MF model, which is augmented by attribute-to-factor mappings like in our framework. In [21] there is also a generalization of NSVD1 that takes both user and item attributes into account, and which has free latent features. Because of the free latent features, this generalization is not capable of generating cold-start recommendations; it could, however, be enabled to do that using our framework.

fLDA [2] uses content data for rating prediction. It combines one-way and two-way user-item interactions and jointly learns the parameters for those interactions. The authors assume a bag-of-words-like structure for the content attributes of items such that latent feature extraction based on LDA [6] is possible. Thus, the fLDA approach is restricted to bag-of-word-features (i.e. nominal) whereas our approach can deal with any type of attributes (i.e. nominal, ordinal, metric); it is not applicable to new-user scenarios. The same authors also proposed Regression-based Latent Factor Models (RLFM) [1], a similar hybrid collaborative filtering method for rating prediction, which works also in cold-start scenarios. According to the authors, by assuming Bernoulli-distributed observations, fLDA and RLFM would also be suitable for item recommendation with positive and negative feedback; nevertheless, the suitability of the approach for

that task is not shown empirically.

Pairwise Preference Regression [18] is a regression model for rating prediction optimized for a personalized pairwise loss function. The two-way aspect model [25] is a variant of the aspect model [13] for the item recommendation and the rating prediction task. Filterbots [24] are a heuristic method to augment collaborative filtering systems with content data. Unified Boltzmann Machines [10] are probabilistic models that learn from collaborative and content information by combining Untied Boltzmann Machines, that capture correlations between items, with Tied Boltzmann Machines [9], that take content information into account.

VI. Conclusions

We presented a general and straightforward framework to make factorization models attribute-aware. The framework is applicable to both user and item attributes, and can deal with nominal/binary and real-valued attributes. We demonstrated the usefulness of the method by an extension of matrix factorization optimized for Bayesian Personalized Ranking (BPR) that is capable of making item recommendations for new items. The experimental evaluation on two different types of mappings – kNN and linear mappings optimized for BPR – showed that the method produces accurate predictions on par with state-of-the-art methods, and that it carries little run-time overhead.

We also showed empirically that it is worth training the mapping function for optimal model performance with respect to application-specific losses, instead of just trying to map the latent features as accurately as possible.

An appealing property of our framework is its simplicity and modularity: Because its components are only loosely coupled, it can be used to enhance existing factorization models to support new-user and new-item cold-start scenarios.

In the future, we will extend this work in several directions, among others with experiments on user attributes and real-valued (instead of binary) attributes. We also want to see whether the method produces similarly good results for other applications like rating or tag prediction. As stated before, we will investigate how to improve mapping and prediction accuracy for large attribute sets by employing non-linear learned mapping functions like multi-layer neural networks or support vector regression.

VII. Acknowledgments

We thank the anonymous reviewers for their comments and suggestions. This work was co-funded by the EC FP7 project “Dynamic Personalization of Multimedia” (MyMedia) under the grant agreement no. 215006. The second author is supported by a scholarship of CNPq, an institution of the Brazilian government for scientific and technological development.

References

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In J. F. E. IV, F. Fogelman-Souli, P. A. Flach, and M. Zaki, editors, *KDD*, pages 19–28. ACM, 2009.

[2] D. Agarwal and B.-C. Chen. fLDA: Matrix factorization through latent dirichlet allocation. In Davison et al. [8], pages 91–100.

[3] K. Ali and W. van Stam. Tivo: Making show recommendations using a distributed collaborative filtering architecture. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2004.

[4] L. D. Bergman, A. Tuzhilin, R. D. Burke, A. Felfernig, and L. Schmidt-Thieme, editors. *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*. ACM, 2009.

[5] D. Billsus, M. J. Pazzani, and J. Chen. A learning agent for wireless news access. In *Intelligent User Interfaces*, pages 33–36, 2000.

[6] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] P. Cremonesi and R. Turrin. Analysis of cold-start recommendations in iptv systems. In Bergman et al. [4], pages 233–236.

[8] B. D. Davison, T. Suel, N. Craswell, and B. Liu, editors. *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010*. ACM, 2010.

[9] A. Gunawardana and C. Meek. Tied boltzmann machines for cold start recommendations. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 19–26, New York, NY, USA, 2008. ACM.

[10] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In Bergman et al. [4], pages 117–124.

[11] R. Harshman. Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis. In *UCLA Working Papers in Phonetics*, pages 1–84, 1970.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd ed. corr. 3rd printing edition, January 2009.

[13] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, Stockholm, 1999.

[14] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[16] D. Oard and J. Kim. Implicit feedback for recommender systems. In *in Proceedings of the AAAI Workshop on Recommender Systems*, pages 81–83, 1998.

[17] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM 2008*, 2008.

[18] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In Bergman et al. [4], pages 21–28.

[19] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD Cup Workshop at KDD-07*, San Jose, California, USA, 2007.

[20] M. Pazzani and D. Billsus. Content-based recommendation systems. In *The Adaptive Web*, pages 325–341. 2007.

[21] I. Pilaszy and D. Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In Bergman et al. [4], pages 93–100.

[22] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI 2009*, 2009.

[23] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In Davison et al. [8], pages 81–90.

[24] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Computer Supported Cooperative Work*, pages 345–354, 1998.

[25] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002. ACM Press.

[26] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[27] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, pages 279–311, 1966.