Latent Factor Interpretations for Collaborative Filtering

Anupam Datta  Sophia Kovaleva
Piotr Mardziel  Shayak Sen
Carnegie Mellon University

ABSTRACT
Many machine learning systems utilize latent factors as internal representations for making predictions. However, since these latent factors are largely uninterpreted, predictions made using them are opaque. Collaborative filtering via matrix factorization is a prime example of such an algorithm that uses uninterpreted latent features, and yet has seen widespread adoption for many recommendation tasks.

We present Latent Factor Interpretation (LFI), a method for interpreting models by leveraging interpretations of latent factors in terms of human-understandable features. The interpretation of latent factors can then replace the uninterpreted latent factors, resulting in a new model that expresses predictions in terms of interpretable features. This new model can then be interpreted using recently developed model explanation techniques. In this paper, we develop LFI for collaborative filtering based recommender systems, which are particularly challenging from an interpretation perspective.

We illustrate the use of LFI interpretations on the MovieLens dataset demonstrating that latent factors can be predicted with enough accuracy for accurately replicating the predictions of the true model. Further, we demonstrate the accuracy of interpretations by applying the methodology to a collaborative recommender system using DBtropes and IMDB data and synthetic user preferences.

1. INTRODUCTION
Many machine learning systems utilize latent factors as internal representations for making predictions. However, since these latent factors are largely uninterpreted, predictions made using them are opaque. Recommender systems that perform collaborative filtering via matrix factorization are prime examples of such machine learning systems. These systems are state-of-the-art in important application domains, including movie and social recommendations [1], [2]. However, these models are difficult to interpret because they express user preferences and item characteristics along a set of uninterpreted latent factors trained from a sparse set of user ratings.

We present Latent Factor Interpretation (LFI), a method for interpreting models by leveraging interpretations of latent factors in terms of human-understandable features, and develop it in the particular setting of collaborative filtering. In order to interpret models that use uninterpreted latent factors, we address two challenges. The first challenge is that learnt latent factors are constants uninterpretable to humans; any explanations in terms of these factors would be unintelligible. In order to address this problem, we learn a mapping from interpretable features to these latent factors. In our setting, we compose the interpretation of item latent factors with user latent factors to make recommendations (see Fig. 1). In this composed model, which we call a shadow model, the structure of the original recommender system is kept intact, while predictions are expressed in terms of interpretable features. A second challenge is that this composed shadow model can be rather complex. However, since the shadow model expresses ratings in terms of interpretable features, we can leverage existing model explanation techniques [6], [15]. In particular, in this paper, we determine influential features using an existing technique [6] (see Fig. 3 for an example). Note that the purpose of the shadow model is not to supplant the recommender system, but to interpret its predictions.

For example, for an LFI interpretation of a movie recommender system, we predict the latent factors from interpretable features such as genre, year, and keywords associated with movies. This information is available in auxiliary data sources, such as IMDB and DBTropes [1], [10]. An interpretation for a given recommendation thus indicates important, human-understandable features behind it, e.g., a high recommendation for Star Trek arose for a particular user because its genre is sci-fi, and it has keywords indicating action in space. Since the recommendations of the shadow model are close to the latent model, this interpretation also serves as an interpretation of the recommendations of the latent model. Interestingly, prior
work with real users indicate that of the features most important to movie recommendations [21], most can be inferred from readily available attributes such as average rating and keywords. As a proof of concept, we apply our techniques to a movie recommendation system based on matrix factorization over the popular MovieLens dataset with data integrated from several other movie databases, producing interpretable explanations for recommendations.

This technique of training an approximate, but interpretable shadow model for LFI is similar in spirit to approaches to explaining other machine learning systems [4] [15] [18]. An important difference is that prior work has explored this idea using the features present in the task itself, or using pre-defined mappings to an interpretable space. Instead, we use externally available interpretable features and learn the mapping to an interpretable space. We also differ from existing approaches that try to attribute meaning to latent factors, e.g. with topic models [16], in that the constructed shadow model is itself a recommendation model, albeit with interpretable inputs, and is therefore amenable to existing explanation techniques for machine learning models. We demonstrate this point by applying a recently developed input influence measure [6] to build interpretable explanations for recommendations. We focus on this approach because it does not make assumptions about the complexity of the models involved and allows us to tailor explanations to individual users (or individual recommendations). From their user studies, Tintarev and Masthoff identify the latter as the most important aspect of effective explanations [21].

This paper makes the following contributions:

• We present Latent Factor Interpretation (LFI), a method for interpreting models by leveraging interpretations of latent factors in terms of human-understandable features, and develop it in the particular setting of collaborative filtering.

• We demonstrate how the approach applies to a real world use-case of a movie recommendation system trained from the MovieLens dataset and integrating auxiliary data from IMDB and DBTropes.

• We demonstrate the accuracy of the approach for matrix-factorized models by constructing movie recommendation explanations for synthetic individuals with known preferences.

This paper is structured with a brief background (§2) on matrix factorization for recommender systems, and quantitative input influence which serve as the building blocks of our approach or its evaluation later in this paper. We then describe the construction of the shadow model and computation of influence as means for interpreting recommendations (§3). We demonstrate the utility of our approaches using synthetic and real use-cases derived from the MovieLens [8] movie database augmented with various information sources (§4). We discuss related work (§5) and conclude with a summary of contributions and directions for future work (§6).

2. BACKGROUND

In a general sense recommender systems discover liked items, such as movies, previously not encountered by users. Numerous recommender systems have been proposed in literature, making use of varying forms of data and providing a variety of types of recommendations [3]. Two types of recommender systems are pertinent to this paper: content-based and collaborative.

In a content-based recommender system, new recommendations for a user are computed based on the items liked by that user. Such a system attempts to derive model that, given an item having some set of characteristics (in movies recommendations these would include genre, director, etc.), can predict whether (or to what extent) the user would enjoy an item they have not yet seen. In a collaborative system, however, recommendations for a user of an item are derived using the ratings given to this item by similar users. An important feature of such systems is that item information (genre, etc.) is not employed.

In this section we discuss in more detail a particular type of collaborative recommender system based on matrix factorization (§2.1) which will serve as the use-case for our methods, as well as the cold-start problem, the inability of such systems to make predictions about items (or users) not previously seen. We conclude the section with an overview of quantitative input influence, the main tool we will employ to construct explanations for recommendations (§2.2).

2.1 Matrix factorization for Recommendations

Recommendation systems, as the name implies, are models that give recommendations to users regarding items they would enjoy or prefer. Typically, construction of the models is a supervised learning task which assumes as a given a set of ground truth ratings the users have already given to the items and produces a model that can predict ratings of existing or new user/item pairs. Formally, we are given a set of $n$ users a set of $m$ items, and a sparse matrix of ground-truth ratings $R$ and need to fill in the missing elements of the matrix, that is, predict ratings.

A state of the art method for constructing recommendation models is via matrix factorization [11]. The technique associates with each user a set of preferences over some $k$ number of latent features and with each movie associates a measure of expression of those $k$ features. Formally, the model is composed of a $u$ by $k$ matrix $U$ and a $i$ by $k$ matrix $I$ and the predicted rating for each user/movie pair is described by the $u$ by $i$ matrix $R_{ui} = U^T_i I_u$. Thus each prediction for the rating of item $i$ by user $u$, or $R_{ui}$, is the dot product of the $k$-length vector $u_u$, expressing that user’s preferences for the $k$ latent factors, and the $k$-length vector $i_i$, expressing the extent with which item $i$ exhibits those latent factors. The model factors the ground truth matrix $R$ into the matrix product of $U$ and $I^T$. The choice of $k$, or rank, can vary.

\[
\begin{bmatrix}
\cdots & u_u & \cdots \\
\cdots & i_i & \cdots \\
\cdots & \cdots & \cdots 
\end{bmatrix} \times \begin{bmatrix}
\cdots \\
\cdots \\
\cdots 
\end{bmatrix} = \begin{bmatrix}
\cdots \\
\cdots \\
\cdots 
\end{bmatrix}
\]

User features $U$ Item features $I^T$ Ratings $R$

The matrix factorization method encodes a high dimensional sparse dataset (each user’s ratings over a large collection of items), into two $k$-dimensional dense datasets. The strength of this approach is that it does not rely on any a-priori knowledge about relevant attributes, it derives them from the relationships present among the users and movies in the rating data provided.

Setting $k \ll u, i$ provides some form of regularization though additional regularization can be employed such as the $L_2$ norm of the latent vectors. Overall the models minimize the following loss function:

\[
\min_{U,I} \sum_{(u,i) \in R} \left( R_{ui} - u_u i_i^T \right)^2 + \lambda (\|u_u\|^2 + \|i_i\|^2)
\]

Several algorithms exist for this task though this choice is not important in this paper. Our experimental results are based on the implementation of alternating least squares in the Apache Spark.
Our implementation and experiments are available in an anonymized form[5].

**Cold-start in collaborative filtering.** Recall from earlier, the matrix factorization technique uses nothing but a user/item rating matrix in order to construct the predicted ratings matrix. Though this method is designed to take advantage of a sparse set of ratings, it does not apply to users or items with no ratings. Collaborative recommender systems like those based on matrix factorization suffer from this cold-start problem; recommendations cannot be provided for new users or new items without an existing set of ratings by those users or for those items, respectively.

Several works address this problem by establishing connections between latent factors and content features, as we do in our approach for constructing explanations. Gantner et al., for example, establishes a mapping between user or item features to the latent factors within the matrix-factorized model so that new items or users can be recommended-to based solely on their characteristics and not on non-existent ratings[7].

### 2.2 Quantitative input influence

We briefly review a family of measures presented in Datta et al. [6] called Quantitative Input Influence (QII), that measures the influence of a feature on the outcomes of a model. QII can be tailored to a particular quantity of interest about the system, such as the outcomes of a model over a population, the outcome for a particular instance or other statistics of the system. We use this influence measure to identify influential metadata in the shadow model. In particular, in this work, we are interested in measuring the influence of metadata on the predicted ratings for a specific user and movie.

At its core, QII measures the influence of features by breaking their potential correlations with other input features. This focuses measurements on the explicit use of a particular feature and not on use via correlated other features.

Formally, given an a model $m$ that operates on a feature vector $x$, the influence of a feature $i$ is given by the expected difference in outcomes when feature $i$ is randomly perturbed:

$$t_{m,x}(i) \overset{def}{=} E_{y_i}[m(x) - m(x \ldots \hat{y_i})]$$

The expectation in the above quantity is over samples of the $i^{th}$ feature $y_i$, which is drawn independently from its marginal distribution.

### 3. METHODS

Our approach to interpreting recommender systems based on matrix factorization comprises of two steps. First, we use publicly available interpretable features (i.e., metadata) about items as interpretable features to predict latent factors of these items. We then compose these models for predicting latent factors into models that predict the outcomes for particular users. Second, this shadow model composed of predictors for the latent factors is used to generate human-understandable explanations of outcomes by identifying the most influential interpretable features.

#### 3.1 Metadata sources

In case of movies, we use several sources of publicly available metadata attributes such as genres, directors etc. Information about genres, directors, keywords, etc. is available from IMDB, information about tags is available from MovieLens, and information about tropes is available from DBTropes. For categorical features, these are one-hot encoded to obtain numerical features.

#### 3.2 Shadow Model

We assume that we are given a matrix of interpretable attributes $A$, with one row $a_i$ for each item $i$. For each item latent factor $j$, we train a predictor $f_j$ such that $f_j(a_i) \approx i_{ij}$. Composing these predictors, the final predicted recommendation for a user $u$ and item $i$ can be approximated as follows:

$$\tilde{r}_{ui} = u_u \cdot i_i = \sum_{j=1}^{k} u_u j f_j(a_i)$$

Consequently, we use the composed model $\tilde{r}_u(a) = \sum_{j=1}^{k} u_u j f_j(a)$ as a model that predicts the outcomes of the system for a movie with interpretable attributes $a$ and user $u$. This shadow model is more interpretable insofar as it maps interpretable attributes to ratings. However, it is still fairly complex. Therefore, to interpret the behavior of the shadow model $\tilde{r}_u$ on a point $a$, we examine the influences of interpretable attributes using QII.

The chosen regression model is a decision tree. We split the range of each item feature into a certain number of bins, and then treat it as a multiple-class classification problem, using the center of a predicted bin as the output of the regression model.

#### 3.3 Interpreting the Shadow Model

We interpret the shadow model by measuring the quantitative input influence of all metadata features (or just the metadata features present in the trees it consists of, as the influence of others will be zero) on its output. This can be measures either on the output of a particular user-item pair, in which case the question being answered is “why were you given this recommendation?” or the entirety of the model’s predictions for this user for all items, in which case it’s answering “what has the model inferred about your preferences in general?”.

The interpretation takes the form of a list of feature-influence pairs, which allows one to find out which of the metadata features are the most influential in the system’s decision.

#### 3.4 Measuring prediction accuracy

We will evaluate the accuracy of the shadow model on two measures First, we measure the accuracy of the overall predictions of the shadow model with respect to the baseline model as follows:

$$\frac{1}{N} \sum_{ui} (\tilde{r}_{ui} - r_u(a_{ui}))^2 = \frac{1}{N} \sum_{ui} (u_u \cdot i_i - \tilde{r}_u(a_i))^2$$

Second, in order to estimate the gains of predicting ratings using interpretable features, we compare against a random baseline where each item feature is randomized while predicting ratings. We call the ratio of accuracy of the recommendations with the predicted latent features and randomized latent features faithfulness.

### 4. RESULTS

We evaluated our methods on movie recommendations after integrating ratings data with several additional sources of movie metadata. We note some relevant details about the datasets we used in §4.1 and briefly describe our implementation in §4.2. We demonstrate the expression of latent factors in terms of collected metadata in §4.3 and present the recommendation interpretations we can derive using these predictions in §4.5. Finally, in §4.6, we describe some experiments on synthetic data.

#### 4.1 Datasets

The source of our data was MovieLens 20M Dataset[8], which contains approximately 20 million ratings of 27,000 movies by 138,000
distinct users. Ratings are on a 1-5 integer ⭐ range. Minimal movie metadata is also present, in particular, year of release, genres, and a list of user-generated tags.

Additionally we included various movie features from the Internet Movie DataBase (IMDB) [1], which contains keywords, directors, cast, production company, writer, among others. Finally, we integrated DBTropes data [10], a machine-readable snapshot of TV Tropes - a community-maintained wiki, that attempts to describe various media (movies, books, games, etc.) in terms of “tropes” – features frequently found in media, will it be a presence of a strong female character, the description or suburbia, the presence of a character who overcomes their troubles with determination, or many others.

Overall, the three sources of data contain a wealth of information about movies. Looking over the most relevant factors for recommendations as noted by Tintarev and Masthoff [21] in their user studies, a large portion of them could be conceivably determined to some extent by the metadata we have collected. Some of these are direct (for example cast and director information are metadata features) whereas some are less direct (for example visuals). For these we speculate that some metadata features could carry some information about the factors (locations and cinematographer might to some extent determine visuals).

**Pre-processing.**

We used all movie ratings from MovieLens 20M dataset for constructing a recommender system (see §4.3). For subsequent steps, however, we performed several pre-processing steps.

We encode nominal features via one-hot encoding, and use only two numeric metadata features: year of release and average rating (according to IMDB).

In a feature selection step, we dropped those not meeting a minimal entropy threshold, that is, indicators that were almost always true of a movie, or almost always false, resulting in approximately 1200 features. For training and evaluating explanations, we also pruned away movies with missing or negligible metadata.

**4.2 Implementation**

Our implementation is based on a set of Python programs that make heavy use of the Apache Spark library.

**4.3 Recommender**

The MovieLens ratings constituted the sparse user-item input matrix for the training of a recommender system. This data is also the ground truth for evaluation purposes later in this section. The ground truth was processed with alternating least squares matrix factorization algorithm, as implemented in Apache Spark MLlib, which outputs two matrices: user features and movie features, which encode user preferences and movie properties along low-dimensional space of latent features.

Most of our experiments were run with rank 12, which is substantially smaller than what could be expected in a deployed recommender system. However, having conducted a small number of experiments with higher ranks (up to rank 40), we have determined that the behavior on lower ranks generalizes sufficiently well on higher ranks.

The rank 12 recommender which we use throughout the rest of this section achieves an average error (MAE) of 0.53 ⭐ over the ground truth, or root mean squared error (RMSE) of 0.70 ⭐. Statistics concerning recommenders of several other ranks can be seen in Figure 2.

**4.4 Learning latent features**

We employed a several models in order to construct the predictors of latent features including linear ones as well as decision tree-based ones. We found empirically that regression trees (decision trees with numeric outputs) perform most consistently well when depth limits are imposed. Linear regression was notable in our experiments for its poor performance as compared to random latent feature predictors (the faithfulness measure as described in §3.4). A sample of the models attempted and their performance can be seen in Figure 2. Note that deeper trees are prone to overfit the latent features as error on the train data decreases with depth whereas error on test data increases.

**4.5 Interpreting recommendations**

To construct interpretations of the recommendations produced by the shadow model, we measure the influence of each of the metadata features on the rating the shadow model produces. For a particular recommendation (user,movie pair), the definition of influence of a metadata feature (see §2.2) in this setting measures the expected change in the output of the recommender (the rating) if we substitute a fresh value for only that metadata feature with one sampled independently from its marginal, while all the other metadata features are kept fixed.

Several examples of the resulting influence measures can be seen in Figure 3. For two users, we see the top 10 most influential features in their recommendation for two different movies. We order the influential metadata features on the y-axis and chart their influence (which can be measured in ⭐) on the x-axis.

Note that the metadata features listed are not indicative of whether the given movie has that feature. For example, Inspector Gadget is not a horror movie, hence the influence of movielens_genre:Horror on User 7’s recommendation about Inspector Gadget is negative and indicates that in this circumstance it would have been preferable for the movie to be of the horror genre.

Several features stand out in the interpretations presented in Figure 3. First, average rating tends to frequently have the highest impact on recommended rating. This is expected given that well regarded movies have a higher chance of being rated higher by an average user.

Second, the horror genre also appears in numerous influential positions.

Finally, notice that rare features can have large influence. For example, the most influential feature for User 7’s recommendation for Lake Placid is concerning the cancer keyword which is relatively rare.

**4.6 Validation against Known Preferences**
Since user affinities to explanations are hard to obtain, we evaluated our system on synthetic datasets for which we designed used personalities to be dependent on metadata features.

This evaluation on synthetic data provides some evidence that the latent factor predictors and our shadow models are indeed capturing preferences about individuals in a recommender system.

The synthetic dataset is generated by selecting two features \( pos_u, neg_u \) for each user \( u \), representing a positive and a negative criterion for that user. We call such a pair of features a profile. Then for a random selection of movies, ratings for the movie are generated by adding and removing a star for a negative and positive feature respectively as follows:

\[
    r_{i,u} = 3 + a_{i,pos_u} - a_{i,neg_u}.
\]

Once the data sets are generated, a recommender and its shadow model are trained in the same way they are with the real data set with the rank of the model chosen to be the same as the number of distinct profiles in the data set. Then, for randomly selected 20 user-movie pairs, we compute a sorted list \( q \) of QIIs as an explanation for that user’s recommendation for the movie.

We then measure the correspondence \( \sigma \) of the synthetic user profile \( (pos, neg) \) to the explanation vector using the following metric:

\[
    \sigma(q, pos, neg) = \frac{q_{pos} + q_{neg}}{q_{pos} + q_{neg}},
\]

where \( q_{pos} \) and \( q_{neg} \) are the reported QIIs of the positive and negative profile features, and \( q_0, q_1 \) are the top two QIIs in the vector. This measure is equal to 1 if \( pos \) and \( neg \) appear at the top of the vector, and 0 if their QII values are 0.

The correctness scores are then averaged between the 20 user-movie pairs, and this gives us per-experiment correctness scores. We then repeat the whole sequence \( N \) times to get a sample.

To evaluate correctness measures, we compare against two controls: (1) a profile sharing only one feature with the true profile (semi-random), and (2) a one sharing none (random). That is, we measure the correspondence metric in these controls with the profiles of the users changed from the ones used for training but with explanations kept fixed.

The hypotheses are that explanations correspond better with true profiles than with semi-random, and with semi-random better than random. To test the hypotheses, we ran the aforementioned experiments \( N \) times and performed a two-tailed t-test. The results are summarized in Table 1. As expected, the mean correctness scores align as true > semi-random > random, and this holds with a high statistical significance and large effect sizes in all cases, except for the one where the sample size was too small for statistical significance. In that last case a larger sample size with the same parameters achieved statistical significance.

5. RELATED WORK

Existing approaches to address the interpretability of latent factors either attempted to associate them with some item content, or to present them via the relationships they encode in the items and users of a system. We also discuss the relationship of our methods to other approaches for making machine learning interpretable via shadow models and interpretability constraints.
we can provide an explanation for such recommendations. We be-
with users and movies as nodes, arranged to designate proximity in
recommendations. Whereas they can recommend rating-less items,
items without ratings. Our focus, however are explanations for
start problem as their shadow models allow them to recommend
Gantner et al.[7] use externally provided interpretable features in
features and
handcrafted mappings. LFI uses externally provided interpretable
factorization techniques [18]. These approaches either use features
algorithms [19; 4; 13; 15], and has also been applied to matrix
recommendations by injecting the topic-latent matrix within the
usual matrix factor model.

| Parameters | l. | s.r. | r. | t. > s.r. | s.r. > r. |
|------------|----|-----|----|----------|---------|
| N=20, 3 pr, m, 3, 15 h.e.f. | 0.75 | 0.51 | 0 | 6e-11 | 3.3 | 1e-20 | 14.2 |
| N=20, 8 pr, m, 3, 40 h.e.f. | 0.26 | 0.2 | 0 | 0.03 | 0.8 | 8e-11 | 4.2 |
| N=20, 8 pr, m, 8, 40 h.e.f. | 0.4 | 0.3 | 2e-4 | 0.02 | 0.5 | 1e-23 | 4.5 |
| N=20, 10 pr, m, 15, any 250 | 0.22 | 0.19 | 0 | 0.1 | 0.5 | 7e-11 | 4.2 |
| N=49, 10 pr, m, 15, any 250 | 0.22 | 0.19 | 0 | 0.02 | 0.5 | 1.5e-27 | 4.6 |

Table 1: Synthetic data set hypotheses testing. The parameters of
the experiments include: sample size, number of user preference
profiles, rank of the matrix factorization model, and the strategy of
selecting features for generating profiles. “h.e.f.” stands for highest-
entropy features, “any 250” stands for any features with more than
250 non-zero values, “s.r.” stands for semi-random, “r.” stands for
random, “t.” stands for true, “e.s.” stands for effect size, “pr” stands for
profiles, “rn” stands for rank.

Associations. Rossetti et al.[16] use topic modeling to extract
topics from movie descriptions and then associate topics to latent
factors in a matrix factorized model. Their topics are of the form of
bags of words and are not as directly interpretable as movie features
we consider in our work. Further, they develop said association so
that the recommendation model can be portable; new users specify
their preferences on topics and the technique can then provide them
recommendations by injecting the topic-latent matrix within the
usual matrix factor model.

Presentation. Koren et al.[11] show that movies and users can
sometimes be understood in terms of their proximity to other movies
and users. Plotting users/movies according to their latent features
or certain projections can result in recognizable clusters. These
clusters can then be suggestive of user personalities and of movie
characteristics that may have not been part of their extrinsic char-
acteristics. For example, they show how groups of movies form
clusters that roughly correspond to movies with strong female leads
and fraternity humor.
In a related line of work, Hernando et al.[9] present a design of a tool
in which recommendation explanations are of a form of a graph
with users and movies as nodes, arranged to designate proximity in
the latent feature space.

Shadow Models. Our approach of training an interpretable shadow
model that mimics the behavior of the true uninterpretable model is
similar to a general approach for explaining machine learning
algorithms [19] [4] [13] [15], and has also been applied to matrix
factorization techniques [18]. These approaches either use features
present in the input space or map to an interpretable space using
handcrafted mappings. LFI uses externally provided interpretable
features and learns a mapping to the latent space. Similar to us,
Gantner et al.[17] use externally provided interpretable features in
order to train a shadow model. They do this to alleviate the cold-
start problem as their shadow models allow them to recommend
items without ratings. Our focus, however, are explanations for
recommendations. Whereas they can recommend rating-less items,
we can provide an explanation for such recommendations. We be-
lieve that explanations can further alleviate the cold start problem,
as explanations for recommendations of new items can encourage
users to rate them.

Interpretability Constraints. An orthogonal approach to adding
interpretable to machine learning is to constrain the choice of models
to those that are interpretable by design. This can either proceed
through regularization techniques such as Lasso [20] that attempt
to pick a small subset of the most important features, or by using
models that structurally match human reasoning such as Bayesian
Rule Lists [14], Supersparse Linear Integer Models [22], or Probabil-
istic Scaling [17]. For recommender systems, one approach that
belongs to this family is non-negative matrix factorization (NMF)
(see [12]), that enforces a level of interpretability by constraining
latent features to be positive. Even for NMF, the mapping to inter-
pretable features should be useful in order to discover the concepts
encoded in these latent factors.

6. CONCLUSIONS AND FUTURE WORK
We describe a method for interpreting recommendations of latent
factor models for collaborative filtering. We construct a shadow
model that agrees with the latent factor model in its predictions.
The shadow model predicts latent factors from interpretable features
available from auxiliary data sources and then uses the latent factors
to make recommendations like the original model. In contrast to
prior work, the shadow model is not interpretable by design. In fact,
it is more complex than the original model. However, since its input
features are interpretable, its recommendations can be explained
using input influence measures from prior work.
We apply this method to a movie recommendation system based
on matrix factorization over the popular MovieLens dataset with
auxiliary data from IMDB and TV Tropes, producing interpretable
explanations for recommendations. We find that the influence mea-
sures that quantify the impact of interpretable features on recom-
mendation ratings in the shadow model are a reasonable and concise
way of interpreting the functioning of the latent factor recommender
system.
There are several avenues for future work. One interesting direction
is the design of explanations for hybrid content/collaborative recom-
mender systems which use some interpretable features along with
user ratings, making them amiable to influence measures, though
only partially via their interpretable inputs. Other open questions
include formally characterizing conditions under which this inter-
pretation method effectively reveals user preferences as well limits
that arise from lack of informativeness in auxiliary data sources.
A related direction involves validating these explanations with real
users through user studies.

7. REFERENCES
[1] Imdb alternative interfaces. http://www.imdb.com/
Accessed: 2016-11-17.
[2] Recommending items to more than a billion people.
https://code.facebook.com/posts/861999383875667/recommending-items-to-
more-than-a-billion-people/ Accessed: 2016-
[3] Gediminas Adomavicius and Alexander Tuzhilin. Toward
the next generation of recommender systems: A survey of
the state-of-the-art and possible extensions. IEEE
transactions on
knowledge and data engineering, 17(6):734–749, 2005.
[4] Mark W. Craven and Jude W. Shavlik. Extracting tree-
structured representations of trained networks. In Proceedings
of the 8th International Conference on Neural Information Processing Systems, NIPS’95, pages 24–30, Cambridge, MA, USA, 1995. MIT Press.

[5] Anupam Datta, Sophia Kovaleva, Piotr Mardziel, and Shayak Sen. Supplementary code and experiments for submission. https://sites.google.com/site/explainfactors/

[6] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 598–617. IEEE, 2016.

[7] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In Data Mining (ICDM), 2010 IEEE 10th International Conference on, pages 176–185. IEEE, 2010.

[8] GroupLens. Movielens 20m dataset. https://grouplens.org/datasets/movielens/20m/ Accessed: 2017-03-24.

[9] Antonio Hernando, Jesús Bobadilla, Fernando Ortega, and Abraham Gutiérrez. Trees for explaining recommendations made through collaborative filtering. Information Sciences, 239:1–17, 2013.

[10] Malte Kiesel. Dbtropes. http://skipforward.opendfki.de/wiki/DBTropes Accessed: 2016-10-07.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8), 2009.

[12] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. Nature, 401:788–791, 1999.

[13] Jens Lehmann, Sebastian Bader, and Pascal Hitzler. Extracting reduced logic programs from artificial neural networks. Applied Intelligence, 32(3):249–266, June 2010.

[14] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. Ann. Appl. Stat., 9(3):1350–1371, 09 2015.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “why should i trust you?”: Explaining the predictions of any classifier. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, pages 1135–1144, New York, NY, USA, 2016. ACM.

[16] Marco Rossetti, Fabio Stella, and Markus Zanker. Towards explaining latent factors with topic models in collaborative recommender systems. In Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on, pages 162–167. IEEE, 2013.

[17] Stefan Rüping. Learning interpretable models. PhD thesis, Dortmund University of Technology, 2006. http://d-nb.info/997491736.