Position Paper: Exploring Explanations for Matrix Factorization Recommender Systems

Bashir Rastegarpanah
Boston University
bashir@bu.edu

Mark Crovella
Boston University
crovella@bu.edu

Krishna P. Gummadi
Max Planck Institute for Software Systems
gummadi@mpi-sws.org

ABSTRACT

In this paper we address the problem of finding explanations for collaborative filtering algorithms that use matrix factorization methods. We look for explanations that increase the transparency of the system. To do so, we propose two measures. First, we show a model that describes the contribution of each previous rating given by a user to the generated recommendation. Second, we measure the influence of changing each previous rating of a user on the outcome of the recommender system. We show that under the assumption that there are many more users in the system than there are items, we can efficiently generate each type of explanation by using linear approximations of the recommender system's behavior for each user, and computing partial derivatives of predicted ratings with respect to each user's provided ratings.

ACM Reference format:
Bashir Rastegarpanah, Mark Crovella, and Krishna P. Gummadi. 2017. Position Paper: Exploring Explanations for Matrix Factorization Recommender Systems. In *Proceedings of Workshop on Responsible Recommendation at RecSys 2017, Como, Italy, August 2017 (FATREC 2017),* 4 pages.
https://doi.org/10.18122/B2R717

1 INTRODUCTION

Recommender systems are taking on an increasing role in shaping the impact of computing on society, and it is consequently important to develop methods for explaining the recommendations made by such systems.

Among the many possible goals for explanation [6], we focus on user-oriented explanation (explanations that assume the system is fixed) rather than developer-oriented explanation (explanations that guide system development). Within the user-oriented domain, we focus on explanations that have as their goal *transparency:* providing the user with an understanding of how the system formulated a recommendation.

Among the broad class of recommender systems approaches, one can distinguish *neighborhood methods,* based on computing similarities between items or users, from *matrix factorization,* which assigns items and users to a latent space in which inner product captures the affinity of a user for an item. Neighborhood methods naturally lend themselves to explanation: witness Netflix's recommendations in which, for a given movie previously viewed, a set of recommended movies is proposed. In this context, the previously viewed movie is treated as an explanation.

FATREC 2017, August 2017, Como, Italy

DOI: 10.18122/B2R717

Matrix factorization (MF) methods, on the other hand, can be more accurate than neighborhood methods [2], but pose a greater challenge from an explanation standpoint. The associated challenges include:

- Matrix factorization methods make use of the *entire* set of previous recommendations – over all users and items – in formulating a single recommendation for a given user.
- Matrix factorization methods solve a non-convex optimization via heuristic methods, whose functioning can be quite opaque.

Our current work investigates two sets of corresponding questions:

(1) In a context where multiple items (and users) can be said to have contributed to forming a recommendation, what is the most meaningful or useful feedback to give a user to explain a single recommendation?
(2) Given the complexity of MF approaches, are there approximate representations of the behavior of MF algorithms that we can use to construct such useful feedback?

A general strategy, exemplified by [4], provides some guidance in addressing the two questions. First, explanations should be in terms that are familiar to users: broadly, they should be in terms of features rather than in terms of, e.g., latent vectors. Second, useful explanations can be in terms of *interpretable* models – for example, decision trees or linear models – which can be chosen as *local approximations* of a more complex nonlinear model (such as a neural network).

In the remainder of this position paper we use this general strategy to address the questions above. Our general approach is via the use of *gradients* of the rating function, as in recent work on classifiers (e.g., [1, 5]). First, we propose gradient based metrics appropriate for MF recommender systems; then we describe how, in a certain commonly encountered scenario, one may approximately compute those gradients.

2 EXPLANATIONS

Assume x_{ij} indicates the rating given by user j to item i. To formulate an explanation for a given recommendation, we consider the case in which the system has given user j a recommendation for item i with an estimated rating of $\hat{x}_{ij}$. That is, the system has formed a prediction that user j will rate item i at $\hat{x}_{ij}$ and has consequently proposed item i to the user.

In such a setting, user j may ask:

(1) Which previous ratings have contributed the most to the predicted rating $\hat{x}_{ij}$?

(2) Which previous ratings have the most influence over the predicted rating $\hat{x}_{ij}$? In other words, if things were different – e.g., different ratings had been provided in the past – which differences would matter most?

To answer these questions in terms familiar to users, it makes sense to use items and their ratings as the basic vocabulary (rather than, say, latent vectors). Furthermore, although an MF based recommender system implicitly takes into account the set of known ratings across all users and items in making its recommendation, other users' ratings are not under user j's control. Hence it does not seem helpful to express our explanations in terms of ratings other than user j's.

This leads us to propose the following kinds of explanations for MF recommender systems:

Impact We model each recommendation $\hat{x}_{ij}$ in terms of known ratings given by the same user. That is, we formulate a model

$$\hat{x}_{ij} \approx \sum_{k \in R(j)} \alpha_k x_{kj}.$$

where $R(j)$ is the set of items that have been previously rated by user j, and we term $\gamma_k = \alpha_k x_{kj}$ the *impact* of known rating x_{kj} on the predicted rating $\hat{x}_{ij}$. The model is linear to support interpretability.

Influence In order to explain the influence of the known rating x_{kj} on the predicted rating $\hat{x}_{ij}$, we define:

$$\beta_k = \frac{\partial \hat{x}_{ij}}{\partial x_{kj}}$$

and we call β_k the *influence* of x_{kj} on $\hat{x}_{ij}$.

We envision the use of these quantities as an interface element of the recommender system. For any given recommendation $\hat{x}_{ij}$, the interface can present the highest impact ratings (those with largest γ_k) and the highest influence ratings (those with largest β_k), along with their values, as explanations for that recommendation.

3 ALGORITHMS

We now seek to find ways to compute approximations to impact and influence as defined in Section 2.

To start, we formalize the setting. Assume $\mathbf{X} \in \mathbb{R}^{m \times n}$ is a partially observed, real-value matrix containing user ratings. Each column is associated with a user and each row is associated with an item. An MF recommender system attempts to estimate unknown elements of the rating matrix. To do so, it find factors $\mathbf{U} \in \mathbb{R}^{\ell \times m}$ and $\mathbf{V} \in R^{\ell \times n}$ such that $\mathbf{U}^T\mathbf{V}$ agrees with the known positions in $\mathbf{X}$. The unknown ratings are then estimated by setting $\hat{\mathbf{X}} = \mathbf{U}^T\mathbf{V}$.

More specifically, the recommender system finds factors $\mathbf{U}$ and $\mathbf{V}$ by applying an algorithm $\mathcal{A}$ to solve the following optimization problem:

$$\mathbf{U}, \mathbf{V} = \underset{\tilde{\mathbf{U}}, \tilde{\mathbf{V}}}{arg\,min} \sum_{(i,j) \in \Omega} (x_{ij} - \tilde{\mathbf{u}}_i^T \tilde{\mathbf{v}}_j)^2 \tag{1}$$

where Ω indicates the set of known entries in $\mathbf{X}$, $\tilde{\mathbf{u}}_i$ is column i of $\tilde{\mathbf{U}}$, and $\tilde{\mathbf{v}}_j$ is column j of $\tilde{\mathbf{V}}$.

3.1 U and V at a local optimum

To capture the effect of $\mathcal{A}$, let us define a function f such that for each user j, f returns the estimation of all item ratings for user j given the set of known item ratings of user j. In other words, f takes the column $\mathbf{x}_j$ of the *observed* matrix $\mathbf{X}$ as input, and returns the corresponding column $\hat{\mathbf{x}}_j$ of the *predicted* matrix $\hat{\mathbf{X}}$, i.e., $f(\mathbf{x}_j) = \hat{\mathbf{x}}_j$.

Now consider the properties of $\mathbf{U}$ and $\mathbf{V}$ at a local minimum of the objective function (1). In that case, each can be expressed as a linear function of $\mathbf{X}$. To see this, first note that for each user j we have $f(\mathbf{x}_j) = \mathbf{U}^T\mathbf{v}_j$. To capture the fact that only known entries matter in the solution of (1), we define $\mathbf{W}_j$ to be a binary matrix with 1s on the diagonal in positions corresponding the known entries of x_j. Then it follows that $\mathbf{v}_j$ is the least squares solution of

$$\mathbf{W}_j\mathbf{x}_j = \mathbf{W}_j\mathbf{U}^T\mathbf{v}_j$$

This implies that at a local minimum of (1), the following relationship holds between $\mathbf{x}_j$ and $f(\mathbf{x}_j)$:

$$f(\mathbf{x}_j) = \mathbf{U}^T\mathbf{v}_j = \mathbf{U}^T(\mathbf{U}\mathbf{W}_j\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{W}_j\mathbf{x}_j \tag{2}$$

3.2 A common case

To develop methods for approximating γ_k and β_k, we consider the case in which there are many more users in the system than there are items. For example, a movie recommendation system may have millions of users but only thousands of movies. In that case we formulate the following hypothesis:

Hypothesis 1. *Given matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, with $n \gg m$, let Ω be the set of known elements in $\mathbf{X}$ and $\mathcal{A}$ be an algorithm that computes $\mathbf{U}$ and $\mathbf{V}$, a local optimum of (1). Assume we change element x_{ij} to x'_{ij} and rerun algorithm $\mathcal{A}$ to find $\mathbf{U}'$ and $\mathbf{V}'$, then $\mathbf{U}'$ is approximately equal to $\mathbf{U}$ and the only significant differences between $\mathbf{V}'$ and $\mathbf{V}$ lie in column j.*

Informal justification for Hypothesis 1 is provided in Appendix A. We find that Hypothesis 1 holds consistently in empirical studies.

3.2.1 Influence. In cases where Hypothesis 1 holds, we can proceed as follows. We start by estimating influence. Our goal is to compute the Jacobian of the function $f()$ evaluated at $\mathbf{x}_j$. That is, we seek:

$$\mathbf{J}^{(j)} = \frac{\partial f(\mathbf{x}_j)}{\partial \mathbf{x}_j}$$

We call $\mathbf{J}^{(j)}$ the influence matrix of user j.

Assume $\boldsymbol{\varepsilon}_i$ is a vector of size m in which element i is ε and all the other elements are zero. In order to compute each element of the influence matrix of user j, we need to compute function f at $\mathbf{x}_j$ and at neighborhoods of $\mathbf{x}_j$ that are defined by $\mathbf{x}_j + \boldsymbol{\varepsilon}_i$ for $i \in \{1, \ldots, m\}$. Equation (2) provides a closed form formula of function $f()$ when the input is one of the user rating vectors. Moreover, under Hypothesis 1 we know that equation (2) provides an approximation for $f()$ when the input is a *modified* user rating vector in which only one of the elements is changed. Therefore we can state that Equation (2) holds not just at $\mathbf{x}_j$, but also within a small neighborhood around $\mathbf{x}_j$. Then:

$$\mathbf{J}^{(j)} = \frac{\partial f(\mathbf{x}_j)}{\partial \mathbf{x}_j} = \mathbf{U}^T(\mathbf{U}\mathbf{W}_j\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{W}_j$$

Interestingly, the influence matrix of each user j only depends on $\mathbf{U}$ and on the set of items that have been previously rated by user j. In particular, it does not depend on the actual ratings that user j has given to any previous items. So if two users a and b happen to have rated exactly the same set of items, although the actual rating values may differ, their influence matrices $\mathbf{J}^{(a)}$ and $\mathbf{J}^{(b)}$ will be identical.

In summary, when Hypothesis 1 holds, then for a given user j and recommended item i, the influence of item k is $\beta_k = \mathbf{J}^{(j)}_{ik}$.

3.2.2 Impact. Next, we turn to approximating impact. In section 2 we showed that the following linear model describes the output of an MF recommender system for user j as a function of his known ratings:

$$f(\mathbf{x}_j) = \mathbf{J}^{(j)}\mathbf{x}_j$$

where $\mathbf{J}^{(j)}$ is the influence matrix of user j. Therefore, the predicted rating for item i can be written as

$$\hat{x}_{ij} = \sum_{(k,j)\in\Omega} \mathbf{J}^{(j)}_{ik} x_{kj}.$$

We define γ_k, the impact of known rating x_{kj} on predicted rating $\hat{x}_{ij}$ as

$$\gamma_k = \mathbf{J}^{(j)}_{ik} x_{kj}$$

which is simply the product of the influence of item k on the prediction for item i and the actual rating given by user j to item k.

We emphasize that our proposed method for computing γ_k is only one way of quantifying impact. In other words, one may choose another linear combination of known ratings of user j that results in $\hat{x}_{ij}$ to define impact. While our method here has the interesting property that coefficients are identical to partial derivatives, one may choose another method to satisfy a different set of properties. For example, a recent work [5] studies the problem of attributing the prediction of a deep network to its input features. A similar approach can be adopted to define more elaborate measures of impact in the context of MF recommender systems.

4 EXAMPLE

To illustrate our proposal more concretely, we present an example using the MovieLens small dataset [3]. We choose the 650 most active users and the 50 most frequently rated movies. The resulting rating matrix has about 25% known entries. To this matrix we apply a well known matrix factorization algorithm (`LMaFit` [7]) with estimated rank 4 and obtain factors $\mathbf{U}$ and $\mathbf{V}$.

If Hypothesis 1 holds, then as discussed above, if users a and b have rated the same set of items, changing the rating given by user a to item i ($x'_{ia} \leftarrow x_{ia} + \varepsilon$) and changing the rating given by user b to item i by the same amount ($x'_{ib} \leftarrow x_{ib} + \varepsilon$) should have an identical effect on the predicted ratings for all other items. In other words, we have:

$$f(\mathbf{x}_a + \boldsymbol{\varepsilon}_i) - f(\mathbf{x}_a) = f(\mathbf{x}_b + \boldsymbol{\varepsilon}_i) - f(\mathbf{x}_b) \tag{3}$$

To illustrate this, we find two users a (user 16) and b (user 211) who happen to have rated the same set of five movies in our data. Figure 1 (left side) shows the ratings given by these two users to these five movies. We then add 1 to user a's rating for movie 4, rerun `LMaFit`, and compute the difference in predicted ratings for all movies. Next we repeat the same procedure, this time modifying only user b's rating for movie 4. The two vectors of rating differences are shown on the right side of Figure 1, and we see that the changes across all movie ratings are nearly identical.

Table 1: Explanation for `Terminator2`

Rated movie	Influence
Mission: Impossible (1996)	5.00
Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	1.01
Star Wars: Episode IV - A New Hope (1977)	-0.24
Fargo (1996)	-1.65
Independence Day (1996)	-2.74

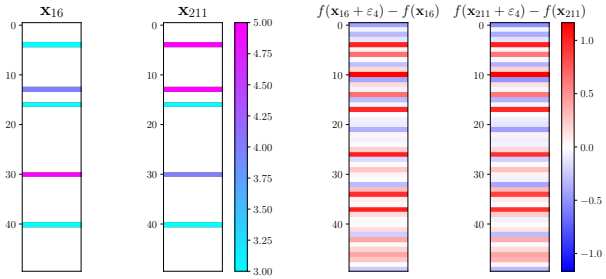

Figure 1: Users 16 and 211 have rated the same set of items. Left: original ratings; Right: changes to all predictions after modifying each user's rating for movie 4.

To illustrate the use of influence values in practice, we show in Table 1 a simple example drawn from our dataset. The table shows for user 16, the influence of each of the 5 movies that the user has rated on the system's predicted ratings for *Terminator2*. The Table shows that changing the user's previous ratings for *Star Wars* or *Fargo* would have much less influence on the predicted rating for *Terminator2* than would changing previous ratings for *Mission Impossible* or *Independence Day*.

5 CONCLUSION

In this position paper we've proposed two kinds of explanations for increasing the transparency of matrix factorization recommender systems: influence, and impact. We argue that these allow for interpretable responses to questions that are important to users: "What are the most important factors yielding this recommendation?" and "What are the factors whose change would most affect this recommendation?" The first question provides the users an understanding of how a recommendation is generated by the system based on the actions they have made in the past, while answering the second question provides the users with information that can be used to control the system's behavior in the future.

We have also shown that in the common case in which there are many more users than items (such as movie recommender systems), there are tractable computational approximations that can be used

to provide numerical values for influence and impact. Interestingly, we find that in this case influence is only determined by the set of movies rated, but not by the values of the ratings given.

We expect to develop these results in both theoretical and practical directions to explore the ultimate utility of these modes of explanation for matrix factorization recommender systems.

A JUSTIFICATION OF HYPOTHESIS 1

Here we present a justification for Hypothesis 1.

We consider the case in which $n \gg m$. In our analysis we make the assumption that a change to x_{ij} only results in changes to $\mathbf{u}_i$ and $\mathbf{v}_j$ (i.e, we focus on the first-order approximation to the effect of Algorithm $\mathcal{A}$). Let the updated latent vectors be $\mathbf{u}'_i$ and $\mathbf{v}'_j$.

Intuitively, our argument is as follows. Updating $\mathbf{u}_i$ to $\mathbf{u}'_i$ results in changes to errors only in row i, and updating $\mathbf{v}_j$ to $\mathbf{v}'_j$ results in changes to errors only in column j. The effect of updates yielding $\mathbf{u}'_i$ and $\mathbf{v}'_j$ will generally attempt to decrease error at position (i, j) and will consequently tend to increase errors on other elements of row i and column j. Since there are many more elements in row i than there are in column j, an update to $\mathbf{u}_i$ (to achieve a unit decrease in error at position i, j) will introduce more overall error than will an update to $\mathbf{v}_j$. Hence the bulk of change will occur in $\mathbf{v}_j$, while $\mathbf{u}_i$ will remain relatively constant.

More formally, let $e_{ij} = (x_{ij} - \mathbf{u}_i^T \mathbf{v}_j)$ and $L = \sum_{ij} e_{ij}^2$. Before the change to element x_{ij}, the effect of $\mathcal{A}$ has been to achieve $\frac{\partial L}{\partial \mathbf{u}_i} = \frac{\partial L}{\partial \mathbf{v}_j} = 0$. These partial derivatives are:

$$\frac{\partial L}{\partial \mathbf{u}_i} = -2 \sum_k^n e_{ik} \mathbf{v}_k \qquad \frac{\partial L}{\partial \mathbf{v}_j} = -2 \sum_k^m e_{kj} \mathbf{u}_k \tag{4}$$

Now, we introduce a change to the rating in position (i, j). Assuming that only $\mathbf{u}_i$ and $\mathbf{v}_j$ change during the subsequent optimization, then applying $\mathcal{A}$ leads to:

$$\mathbf{u}'_k = \begin{cases} \mathbf{u}_i + \tilde{\mathbf{u}} & k = i \\ \mathbf{u}_k & k \neq i \end{cases} \qquad \mathbf{v}'_k = \begin{cases} \mathbf{v}_j + \tilde{\mathbf{v}} & k = j \\ \mathbf{v}_k & k \neq j \end{cases} \tag{5}$$

Thus our goal becomes to establish that $\|\tilde{\mathbf{v}}\| \gg \|\tilde{\mathbf{u}}\|$.

Let e' be the new error values and L' be the new total squared error. At the new local optimum, we have $\frac{\partial L'}{\partial \mathbf{u}'_i} = \frac{\partial L'}{\partial \mathbf{v}'_j} = 0$.

$$\begin{aligned} \frac{\partial L'}{\partial \mathbf{u}'_i} &= -2 \textstyle\sum_k^n e'_{ik} \mathbf{v}'_k & \frac{\partial L'}{\partial \mathbf{v}'_j} &= -2 \textstyle\sum_k^m e'_{kj} \mathbf{u}'_k \\ &= -2(e'_{ij} \mathbf{v}'_j + \textstyle\sum_{k \neq j} e'_{ik} \mathbf{v}_k) & &= -2(e'_{ij} \mathbf{u}'_i + \textstyle\sum_{k \neq i} e'_{kj} \mathbf{u}_k) \end{aligned} \tag{6}$$

Subtracting corresponding eqns in (6) and (4) and dropping factors of -2, we get:

$$\frac{\partial L'}{\partial \mathbf{u}'_i} - \frac{\partial L}{\partial \mathbf{u}_i} = e'_{ij} \mathbf{v}'_j - e_{ij} \mathbf{v}_j + \sum_{k \neq j} (e'_{ik} - e_{ik}) \mathbf{v}_k \tag{7}$$

$$\frac{\partial L'}{\partial \mathbf{v}'_j} - \frac{\partial L}{\partial \mathbf{v}_j} = e'_{ij} \mathbf{u}'_i - e_{ij} \mathbf{u}_i + \sum_{k \neq i} (e'_{kj} - e_{kj}) \mathbf{u}_k \tag{8}$$

Note that:

$$e'_{ik} - e_{ik} = -\tilde{\mathbf{u}}^T \mathbf{v}_k \qquad k \neq j \tag{9}$$

$$e'_{kj} - e_{kj} = -\tilde{\mathbf{v}}^T \mathbf{u}_k \qquad k \neq i \tag{10}$$

So substituting (9) and (10) into (7) and (8):

$$\frac{\partial L'}{\partial \mathbf{u}'_i} - \frac{\partial L}{\partial \mathbf{u}_i} = e'_{ij} \mathbf{v}'_j - e_{ij} \mathbf{v}_j + \sum_{k \neq j} -(\tilde{\mathbf{u}}^T \mathbf{v}_k) \mathbf{v}_k = 0 \tag{11}$$

$$\frac{\partial L'}{\partial \mathbf{v}'_j} - \frac{\partial L}{\partial \mathbf{v}_j} = e'_{ij} \mathbf{u}'_i - e_{ij} \mathbf{u}_i + \sum_{k \neq i} -(\tilde{\mathbf{v}}^T \mathbf{u}_k) \mathbf{u}_k = 0 \tag{12}$$

Now subtracting (12) from (11) we get:

$$e'_{ij}(\mathbf{v}'_j - \mathbf{u}'_i) - e_{ij}(\mathbf{v}_j - \mathbf{u}_i) + \sum_{k \neq i}^{m} (\tilde{\mathbf{v}}^T \mathbf{u}_k) \mathbf{u}_k - \sum_{k \neq j}^{n} (\tilde{\mathbf{u}}^T \mathbf{v}_k) \mathbf{v}_k = 0 \tag{13}$$

In (13), we note that the terms $e'_{ij}(\mathbf{v}'_j - \mathbf{u}'_i)$ and $e_{ij}(\mathbf{v}_j - \mathbf{u}_i)$ are small compared to the two summation terms. Therefore we can approximately argue:

$$\sum_{k \neq i}^{m} (\tilde{\mathbf{v}}^T \mathbf{u}_k) \mathbf{u}_k \approx \sum_{k \neq j}^{n} (\tilde{\mathbf{u}}^T \mathbf{v}_k) \mathbf{v}_k \tag{14}$$

$$\tilde{\mathbf{v}}^T \sum_{k \neq i}^{m} \mathbf{u}_k \mathbf{u}_k^T \approx \tilde{\mathbf{u}}^T \sum_{k \neq j}^{n} \mathbf{v}_k \mathbf{v}_k^T \tag{15}$$

This establishes a relationship between $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{u}}$. To make quantitative predictions, we can assume, e.g., that $\mathbf{u}_k$ and $\mathbf{v}_k$ are i.i.d. multivariate Gaussian random variables $\mathcal{N}(0, \Sigma)$ with $\Sigma = E[\mathbf{u}_k \mathbf{u}_k^T] = \sigma^2 I$. Then in expectation:

$$E\left[\tilde{\mathbf{v}}^T \sum_{k \neq i}^{m} \mathbf{u}_k \mathbf{u}_k^T\right] \approx E\left[\tilde{\mathbf{u}}^T \sum_{k \neq j}^{n} \mathbf{v}_k \mathbf{v}_k^T\right] \tag{16}$$

$$\tilde{\mathbf{v}}^T \sum_{k \neq i}^{m} E\left[\mathbf{u}_k \mathbf{u}_k^T\right] \approx \tilde{\mathbf{u}}^T \sum_{k \neq j}^{n} E\left[\mathbf{v}_k \mathbf{v}_k^T\right] \tag{17}$$

$$(m-1)\sigma^2 \tilde{\mathbf{v}}^T \approx (n-1)\sigma^2 \tilde{\mathbf{u}}^T \tag{18}$$

So we have that $\|\tilde{\mathbf{v}}\| / \|\tilde{\mathbf{u}}\| \approx (n-1)/(m-1)$.

REFERENCES

[1] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. 2010. How to Explain Individual Classification Decisions. *J. Mach. Learn. Res.* 11 (Aug. 2010), 1803–1831. http://dl.acm.org/citation.cfm?id=1756006.1859912

[2] Yehuda Koren and Robert Bell. 2011. Advances in Collaborative Filtering. In *Recommender Systems Handbook*. Springer, 145–186.

[3] MovieLens [n. d.]. MovieLens dataset. https://grouplens.org/datasets/movielens/. ([n. d.]).

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. 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*. ACM, 1135–1144.

[5] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. *CoRR* abs/1703.01365 (2017). http://arxiv.org/abs/1703.01365

[6] Nava Tintarev and Judith Masthoff. 2011. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*. Springer, 479–510.

[7] Zaiwen Wen, Wotao Yin, and Yin Zhang. 2012. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation* 4, 4 (2012), 333–361. https://doi.org/10.1007/s12532-012-0044-1