Estimating Confidence of Individual User Predictions in Item-based Recommender Systems

Cesare Bernardis
Politecnico di Milano
Italy
cesare.bernardis@polimi.it

Maurizio Ferrari Dacrema
Politecnico di Milano
Italy
maurizio.ferrari@polimi.it

Paolo Cremonesi
Politecnico di Milano
Italy
paolo.cremonesi@polimi.it

ABSTRACT

This paper focuses on recommender systems based on item-item collaborative filtering (CF). Although research on item-based methods is not new, current literature does not provide any reliable insight on how to estimate confidence of recommendations. The goal of this paper is to fill this gap, by investigating the conditions under which item-based recommendations will succeed or fail for a specific user.

We formalize the item-based CF problem as an eigenvalue problem, where estimated ratings are equivalent to the true (unknown) ratings multiplied by a user-specific eigenvalue of the similarity matrix. We show that the magnitude of the eigenvalue related to a user is proportional to the accuracy of recommendations for that user.

We define a confidence parameter called the *eigenvalue confidence index*, analogous to the eigenvalue of the similarity matrix, but simpler to be computed. We also show how to extend the eigenvalue confidence index to matrix-factorization algorithms.

A comprehensive set of experiments on five datasets show that the eigenvalue confidence index is effective in predicting, for each user, the quality of recommendations. On average, our confidence index is 3 times more correlated with MAP with respect to previous confidence estimates.

ACM Reference Format:
Cesare Bernardis, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2019. Estimating Confidence of Individual User Predictions in Item-based Recommender Systems. In *27th Conference on User Modeling, Adaptation and Personalization (UMAP '19), June 9–12, 2019, Larnaca, Cyprus.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3320435.3320453

1 INTRODUCTION

Confidence in recommender systems is defined as the system's trust in its recommendations. Both algorithm and user-interface designers can benefit from observing confidence scores. From the algorithmic point of view, given two recommenders that perform similarly on some quality metrics, it can be desirable to choose the one that can provide more confident recommendations [18]. From the user-interface point of view, adding a confidence display to a list of recommended items has the potential to improve user trust in a recommender system [8, 19].

This work first focuses on confidence estimation in item-based (or item-item) collaborative filtering (CF) recommender algorithms. Results are later extended to model-based matrix-factorization (MF) algorithms.

Given a set of users, a set of items and a set of ratings (collected either implicitly or explicitly), item-based CF techniques use available ratings to build relationship between items in the form of an item *similarity matrix*, in order to predict ratings for the unknown user-item pairs, or provide a list of items that the user might find relevant.

Estimating confidence of rating predictions from CF algorithms has been a topic of interest in the literature [8, 18]. Most works relay on the empirical intuition that CF recommenders tend to improve their accuracy as the amount of data over users or items grows [3]. Other works design ad-hoc algorithms able to simultaneously predict ratings and estimate confidence of predictions [13]. However, little research has been performed toward identifying reliable and easy-to-derive confidence estimates for CF recommenders.

The goal of this paper is to complement existing works by investigating the theoretical foundation of confidence estimation in item-based CF recommendations and to experimentally explore the following research question:

> ***RQ**: are there any requirements a user profile must exhibit in order to receive "good" recommendations from an item-based algorithm?*

We show that item-based methods are analogous to an *eigenvalue problem*, where each user profile (i.e., the vector of ratings from a user) is a left eigenvector of the similarity matrix. Moreover, each user is associated with an eigenvalue, and the magnitude of the eigenvalue is correlated with the accuracy of recommendations for that user [7]. We call this analogy the *eigenvector analogy*.

The *eigenvector analogy* leads to a number of interesting properties of item-based methods, that will be empirically demonstrated through the rest of the paper:

(1) each user in the user-rating matrix is associated with a corresponding eigenvalue;
(2) the magnitude of the eigenvalue is proportional to the accuracy of recommendations for that user and it is a reliable measure of confidence for predicted ratings;
(3) users with eigenvalues close to zero will receive poor quality recommendations, regardless of the item-based model.

It is worth noting that the *eigenvector analogy* does not make any assumption on how the similarity matrix has been computed, but in this paper we will focus on item-based CF approaches. We later

UMAP '19, June 9–12, 2019, Larnaca, Cyprus

ACM ISBN 978-1-4503-6021-0/19/06…$15.00
https://doi.org/10.1145/3320435.3320453

extend the results to model-based matrix-factorization algorithms [4].

Thanks to the *eigenvector analogy* we introduce a new confidence parameter, analogous to the eigenvalue, that we call the *eigenvalue confidence index*. Experiments show that the correlation between quality of recommendations for a user and number of ratings in the user's profile, studied in past research work [3], is not always strong. We show that the *eigenvalue confidence index* is a much better predictor of the quality of recommendations than the number of ratings in then user profile.

In summary, we make the following contributions:

- We investigate some theoretical properties of item-based algorithms leveraging a similarity with the eigenvalue problem, which has never been explored by existing literature. To the best of our knowledge, this is the first attempt to provide a theoretical evidence of accuracy capabilities of *item-based* methods.
- We provide a reliable way to estimate confidence of recommendations to a given user for CF item-based algorithms.
- We extend these results to model-based *matrix-factorization* algorithms.
- We provide extensive empirical evidence on five datasets that the quality of recommendations to a given user is strongly correlated with the *eigenvalue confidence index* associated with that user.

The rest of the paper is organized as follows. In Section 2 the relevant existing methods are presented. Section 3 introduces the details of the eigenvalue analogy. Section 4 describes how to compute the *eigenvalue confidence index*. Sections 5 and 6 present the results of the experimental evaluation, together with the methodology and the datasets used for the evaluation. Finally, Section 7 provides some concluding remarks.

2 RELATED WORKS

Although there is a vast literature on item-based methods, the evaluation of their predicting capabilities, in terms of accuracy of recommendation lists, is purely empirical and has never been formally investigated.

Some works in the literature propose methods to empirically estimate the quality of recommendations (i.e., confidence of predictions). The estimated confidence is either used to improve the recommendation algorithm [10, 15] or to provide explanations to users [2, 8, 19].

Adomavicius et al. [1] observe that recommendations tend to be more accurate for users and items exhibiting lower rating variance. However, their approach can be applied only to explicit datasets. Cremonesi et al. [3] empirically show that there is some correlation between number of ratings in the user profile and recall of recommendations, while there is no correlation with perceived relevance or satisfaction. Shyong et al. [19] empirically show that the total number of ratings for each item is related to the mean-absolute-error (MAE) of predicted ratings. Koren et al. [13] empirically estimate whether a model's predicted rating is within one rating level of the true rating. Mazurowski [15] shows that re-sampling of ratings and injection of noisy ratings can be both used to empirically estimate confidence of prediction. However, this method is computationally expensive as it requires to retrain the model several times in order to have reliable confidence estimates.

A disadvantage of these approaches is that confidence estimation is based only on user and item statistics, without taking into account the properties of the prediction model. Most works focus on estimating confidence for individual rating predictions, and as such they are limited to explicit datasets.

In this work we introduce a different concept of confidence, which is more suitable for Top-N recommendation scenarios and works for both implicit and explicit datasets. We propose a method for estimating recommendation confidence which takes into account the ranking performance of the algorithm for each user. Our method does not take into account only the statistics of the ratings (the user-rating matrix), but also the structure of the model (the similarity matrix), and as such is able to provide better confidence estimates with respect to other approaches.

3 PROBLEM FORMULATION

3.1 Notation

In this paper, all vectors are represented by bold lower case letters and they are row vectors (e.g., $\mathbf{r}_u$), unless differently specified. All matrices are represented by bold upper case letters (e.g., $\mathbf{R}$, $\mathbf{S}$). A predicted value is denoted by having a ˜(tilde) over it (e.g., $\tilde{\mathbf{R}}$). We use calligraphic letters to denote sets (e.g., $\mathcal{R}^+$).

Matrix $\mathbf{R}$ will be used to represent the *user-item rating matrix* (*URM*) of size $N_U \times N_I$ (number of users × number of items).

Symbols u and i are used to denote individual users and items, respectively. An entry r_{ui} in $\mathbf{R}$ is used to represent the feedback information for user u on item i. User-rating matrices can be both explicit or implicit. In the first case, r_{ui} corresponds to the rating value given by user u to item i. In the second one, if user u has provided a positive feedback for item i, then the corresponding value r_{ui} is 1, otherwise it is 0 (i.e., $\mathbf{R}$ is a binary matrix). The u-th row of the user-rating matrix $\mathbf{R}$ is represented by the row vector $\mathbf{r}_u$, and it referred to as the *user profile*.

Matrix $\mathbf{S}$ will be used to represent an item-based similarity matrix of size $N_I \times N_I$. An entry s_{ij} in $\mathbf{S}$ measures how similar is item i with item j and it is somehow related to the probability that, if users like item i, they will also like item j. The j-th column of the similarity matrix $\mathbf{S}$ is represented by the column vector $\mathbf{s}^j$. Note that the similarity matrix is not necessarily symmetric, i.e., s_{ij} can be different from s_{ji}.

3.2 Motivation

The methods developed in this work are motivated by the analogy between the generic formulation of an item-based model and the eigenvector problem.

A generic item-based model predicts the ratings $\tilde{\mathbf{r}}_u$ of a user u for all items as

$$\mathbf{r}_u \mathbf{S} = \tilde{\mathbf{r}}_u \tag{1}$$

where $\mathbf{r}_u$ is the profile of the user and $\mathbf{S}$ is a similarity matrix.

Our goal is to find a solution $\mathbf{S}$ to equation (1) which is different from the identity matrix (in such a case, each item is identical to itself and different from all other items). The above equation can be used to model any type of item-based method, the only

difference being on how matrix $\mathbf{S}$ is computed. For instance, matrix $\mathbf{S}$ can be a traditional *KNN* similarity computed with the cosine between vectors of item ratings (CF) or item attributes (CBF) [6, 14]. Alternatively, matrix $\mathbf{S}$ can be directly estimated from the URM using a machine learning approach, as with SLIM methods [17].

We now assume to have an ideal item-based recommender, able to predict all the user ratings (both known and unknown).

Because of this assumption, we can rewrite (1) for an ideal item-based method as

$$\mathbf{r}_u\mathbf{S} = \mathbf{r}_u \tag{2}$$

The item-based model described by (2) is analogous to the left eigenvector problem

$$\mathbf{r}_u\mathbf{S} = \lambda_u\mathbf{r}_u \tag{3}$$

where $\mathbf{r}_u$ is a **left** eigenvector of matrix $\mathbf{S}$ and λ_u is the corresponding eigenvalue.[1] Eigenvectors of $\mathbf{S}$ are the only non-trivial solutions to (3), the trivial solution being $\mathbf{r}_u = 0$ [11].

There is one important difference between the original formulation (2) and the eigenvector formulation (3). In the original "ideal" formulation (2), predicted ratings are equivalent to the ratings in the user profile

$$\tilde{\mathbf{r}}_u = \mathbf{r}_u$$

In the eigenvector formulation (3), predicted ratings are equivalent to the ratings in the user profile *multiplied by the eigenvalue associated with the user*

$$\tilde{\mathbf{r}}_u = \lambda_u\mathbf{r}_u \tag{4}$$

The two formulations coincide if and only if $\lambda_u = 1$ for each user u.

The original formulation (2) has the property of providing recommendations which are perfect in terms of error metrics, such as RMSE and MAE.

The same property does not generally hold for the eigenvalue formulation, as eigenvalues can be smaller or larger than one, or they can even be complex numbers.

However, the eigenvector formulation (3) has still the property of providing recommendations which are perfect in terms of Top-N and ranking metrics, such as precision, recall, MAP and NCDG. This happens because, for a given user u, the corresponding eigenvalue λ_u is constant and the items in the predicted vector of ratings have the same ranking of the items in the user profile.

Another interesting property emerges from the analysis of (4): if, for a given user u, we have $\lambda_u = 0$, the predicted ratings for that user are all identical and equal to zero. For such a user, even a perfect item-based method that satisfies the eigenvector formulation described by equation (3) will not be able to make predictions for that user.

More in general, for each user, the corresponding eigenvalue can either flatten out (if very small) or amplify (if very large) the differences between predicted ratings. The closer an eigenvalue is to zero, the more difficult will be for the item-based method to correctly rank items and to distinguish between relevant and non-relevant items.

[1]Please, note that this problem is slightly different from the more classical **right** eigenvector problem, where there is a column vector that right multiplies a matrix. However, all the literature results on the right eigenvector problem applies seemingless to the left eigenvector problem with only minor modifications.

Thanks to the properties of eigenvalues in item-based methods, we can provide an answer to our research question, with the following observation:

Observation 3.1. *When using an ideal item-based method, eigenvalue λ_u can be used to predict the accuracy of recommendations for user u. The larger is the eigenvalue, the more accurate will be the predictions.*

In other words, we can say that the eigenvalue of a user can give us an estimation of the *confidence* we can have on the recommendation list generated for that user. In this case, with *confidence* we do not intend the ability to predict the correct rating of a user for an item, but instead how accurate we expect the recommendation list to be, where the accuracy depends on how many relevant items are recommended.

It is important to point out that, based on our observation, users with the same profile length but with different ratings might have different eigenvalues (i.e., different accuracy of recommendations).

4 FROM THEORY TO PRACTICE

From a practical point of view, given a similarity matrix $\mathbf{S}$, we can not guarantee to satisfy all the conditions necessary for the *eigenvector analogy*:

- matrix $\mathbf{S}$ is not guaranteed to be ideal, i. e. we are not sure that it preserves the correct ranking of each user profile,
- a user profile $\mathbf{r}_u$ is unlikely to be an eigenvector of matrix $\mathbf{S}$ and, consequently, the corresponding λ_u will hardly be an eigenvalue.

Anyway, the intuition about the correlation between the parameter that solves Equation 3 and the accuracy of the recommendation is still valid. For correctness, we do not call this parameter eigenvalue, because it is not guaranteed that it satisfies all the necessary conditions, but we will use the term *eigenvalue confidence index*.

Rewriting Equation 3 in matrix form we have

$$\mathbf{RS} = \mathbf{\Lambda R} \tag{5}$$

where $\mathbf{R}$ is the user-rating matrix, $\mathbf{S}$ is the similarity matrix and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalue confidence indices on its diagonal:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_{u_1} & & & \\ & \lambda_{u_2} & & \\ & & \ddots & \\ & & & \lambda_{u_{N_U}} \end{bmatrix}$$

Each eigenvalue confidence index λ_u in $\mathbf{\Lambda}$ is associated with a user and corresponds to a user profile $\mathbf{r}_u$ in $\mathbf{R}$.

A unique solution for $\mathbf{\Lambda}$ can be found by using the Moore-Penrose pseudoinverse[2] $\mathbf{R}^+$ of the user rating matrix $\mathbf{R}$ and solving the equation

$$\mathbf{\Lambda} = \mathbf{RSR}^+ \tag{6}$$

However, the computation of the pseudoinverse is very expensive in terms of memory and time consumption and this solution to the problem becomes intractable with large datasets.

[2]Note that the pseudoinverse is necessary because $\mathbf{R}$ almost never satisfies the conditions of invertibility

For this reason we approximate the computation of the *eigenvalue confidence indices* with a different approach. We consider the model described by (5) as a regression problem with loss function $E(\mathbf{\Lambda})$ defined as the sum of squared errors

$$E(\mathbf{\Lambda}) = \sum_{u}^{U} \sum_{j}^{I} (\mathbf{r}_u \mathbf{s}^j - \lambda_{\mathbf{u}} r_{uj})^2 \tag{7}$$

A unique solution for $\mathbf{\Lambda}$ is obtained by minimizing the convex optimization problem

$$\underset{\mathbf{\Lambda}}{\operatorname{argmin}}\, E(\mathbf{\Lambda}) \tag{8}$$

We solve the minimization problem using stochastic gradient descent. Deriving the loss function $E(\mathbf{\Lambda})$ we obtain the gradient

$$\frac{\partial E(\mathbf{\Lambda})}{\partial \lambda_u} = \sum_{j}^{I} (\mathbf{r}_u \mathbf{s}^j - \lambda_{\mathbf{u}} r_{uj}) r_{uj} \tag{9}$$

and the update rule

$$\lambda_u^{t+1} = \lambda_u^t - \eta \frac{\partial E(\mathbf{\Lambda})}{\partial \lambda_u}. \tag{10}$$

From equation (10) we observe two interesting properties:

- there are N_u independent problems, one for each user
- for each user, the parameter to estimate is updated only when the value of the interaction r_{uj} is non-zero.

These properties speedup consistently the solutions of (8), since they reduce the number of computational steps and divide the problem into several parallelizable tasks.

Note that $\mathbf{\Lambda}$ depends on both the user rating matrix $\mathbf{R}$ and the similarity matrix $\mathbf{S}$, which means that, even on the same dataset, we may have different values for the eigenvalue confidence index of a user, depending on the algorithm used to compute $\mathbf{S}$.

4.1 Extension to Matrix-Factorization

Matrix-Factorization (MF) algorithms factorize the user-item rating matrix $\mathbf{R}$ to a product of two lower rank matrices, one containing the so-called user factors ($\mathbf{A}$), while the other one containing the so-called item-factors ($\mathbf{B}$)

$$\mathbf{R} = \mathbf{AB}. \tag{11}$$

Examples of MF algorithms are SVD++ [12] and OrdRec [13]. MF algorithms can be re-designed so that user factors are approximated in terms of a new set of item factors $\mathbf{C}$

$$\mathbf{A} = \mathbf{RC} \tag{12}$$

providing a model-based formulation for MF algorithms

$$\mathbf{R} = \mathbf{RCB}. \tag{13}$$

Examples of model-based MF algorithms are AsySVD [12] and PureSVD [4].

By comparing (13) with (5) we can consider the product $\mathbf{CB}$ equivalent to a similarity matrix $\mathbf{S}$. As such, we can extract eigenvalue confidence indices also from the model-based formulation of MF algorithms.

Table 1: Details of datasets used

Dataset	Users	Items	Ratings	Density	Explicit
ML 20M	138K	27K	20M	0.53%	Yes
Netflix	473K	17.7K	100M	1.20%	Yes
LastFM	1.9K	17.6K	92.8k	0.28%	No
TVAudience	13.6k	17.5K	5.9M	2.49%	No
Xing2017	261K	1.3M	5M	<0.01%	No

5 EXPERIMENTAL EVALUATION

5.1 Datasets

We evaluated the correlation over five different real datasets:

ML20M is a dataset obtained from the MovieLens research project[3].

Netflix is the dataset used for the Netflix Prize[4].

LastFM is the dataset that contains music artist listening information collected from LastFM website [5].

TVAudience contains the TV viewing habits of 13k users over 217 channels during a period of 4 months in 2013. It includes either over-the-air (digital terrestrial broadcasting) or satellite, free or pay-TV [20][6].

Xing2017 is the dataset of the ACM RecSys challenges of 2017, containing users' interactions with job postings from the Xing website [7].

Some statistical properties of the datasets are reported in Table 1.

5.2 Evaluation Methodology

To evaluate the performance of the algorithms we employed leave-one-out. We kept only users with at least 2 ratings in the dataset and the test set was built by randomly selecting one of them among most popular items.[8]

All the other interactions compose the training set, that is used to build the model and get the similarity matrix $\mathbf{S}$. That matrix is used to generate a ranked list of N items for each user (top-N recommendation task), solve the regression problem and calculate the eigenvalue confidence indices. We performed several tests with different values of N, but, since the results were similar, in this paper we will report only the results with $N = 10$.

We evaluated the models by comparing the ranked list of recommended items with the item in the test set and we measured recommendation quality using both classification metrics and ranking-dependent metrics [9, 18]: Hit-Rate (HR), Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG).

To measure the correlation between quality of recommendations and confidence indices, we used Spearman's rank correlation coefficient and Pearson's correlation coefficient. Pearson's correlation is a very common measure of the linear correlation between two variables. However, in our case we are not trying to prove that

[3] https://grouplens.org/datasets/
[4] https://www.netflixprize.com/
[5] https://grouplens.org/datasets/hetrec-2011/
[6] http://recsys.deib.polimi.it/datasets/
[7] https://github.com/recsyschallenge/2017
[8] Popular items are defined as those cumulatively accounting for the 33% of all ratings in the dataset.

Table 2: Performance @10 of item-based algorithms

	Dataset	ItemKNN			PureSVD		
		HR	MAP	NDCG	HR	MAP	NDCG
Explicit	ML 20M	0.3618	0.1651	0.2110	0.4485	0.2190	0.2728
	Netflix	0.2786	0.1211	0.1578	0.3407	0.1501	0.1946
Implicit	LastFM	0.4601	0.2413	0.2927	0.4945	0.2672	0.3208
	TVAudience	0.3259	0.1636	0.2015	0.3393	0.1767	0.2147
	Xing2017	0.7363	0.5614	0.6052	0.8198	0.6176	0.6672

the correlation is explicitly linear, but, instead, we are interested to show that it is monotonically increasing, a behaviour better described by Spearman.[9]

As baseline confidence indices, we used the profile length, as proposed in [3], and the variance of the user ratings, as proposed in [1]. This last confidence index is available only on explicit datasets.

To measure the correlations we ordered and grouped users by increasing values of the confidence index. Each group is composed by 200 users and is represented by the average of both the metric value and the parameter value of the users in it. Note that LastFM is a peculiar dataset, because it has a low number of users, so we adopted smaller groups of 50 users each to avoid that a low number of points could influence too heavily the real value of the correlation.

5.3 Algorithms

To calculate the similarity matrix S we used an item-based CF algorithm and a model-based matrix-factorization algorithm.

ItemKNN is a neighborhood based method that computes the item-item similarities s_{ij} as the cosine of the ratings between item i and item j[5].

PureSVD is a machine learning approach based on Singular Value Decomposition of the user rating matrix[4]. It has been shown to be equivalent to an item-based method where the similarity matrix is obtained from the product between the items' latent factors matrix calculated by SVD and its own transpose[16].

Note that we did not tune the algorithms for best absolute performance, as our interest is not to maximize recommendation accuracy, but, rather, we wish to explore if a correlation exists between accuracy of recommendation and confidence indices.

Table 2 reports the accuracy of the different algorithms over the various datasets.

6 RESULTS AND DISCUSSION

In this section we compare the quality of the eigenvalue confidence index λ_u, with the profile length PL, described in [3], and with the user rating variance σ_u, described in [1]. We expect λ_u and PL to be positively correlated with accuracy of recommendations, while σ_u to be negatively correlated with accuracy.

Tables 3 and 4 show Pearson's and Spearman's correlation coefficient values for both the algorithms over implicit and explicit datasets, respectively.

[9]Please note that we also tested Kendall's coefficient, but the results were very similar to Spearman's ones, so, for brevity, we will not report them in this paper.

The results over implicit datasets in Table 3 show that the eigenvalue confidence index λ_u is a better confidence estimator than profile length in every tested configuration. For instance, λ_u is, on average, **3 times more correlated** with MAP@10 than the profile length, and this ratio is consistent among datasets and correlation metrics. The profile length, instead, exhibits an unexpected inverse (negative) correlation with accuracy on the LastFM dataset.

The results over explicit datasets in Table 4 confirm the eigenvalue confidence index λ_u as the best confidence estimator. Taking also in this case the MAP@10 performance as an example, on average λ_u is 4 times more correlated than the profile length and 2.5 times more correlated than the user rating variance. The user rating variance has a consistent behaviour among the different datasets, metrics and algorithms, even though it does not show a good correlation on the Netflix dataset. The profile length confirms an inconsistent behaviour, with negative correlations on the Netflix dataset.

In order to further investigate the predictive behaviour of the different confidence indices and the inconsistent behaviour of the profile length, Figures 1 and 2 plot MAP@10 on Movielens and Netflix datasets, respectively, as a function of the confidence indices.

Users have been sorted and grouped following the same procedure described in Section 5.2, but we used bigger groups of 2000 users each, in order to reduce the noise and have smoother curves. Note that, for an optimal visualization, in these plots the eigenvalue confidence indices have been normalized to have the same range of values.

We also fixed the superior limit of the horizontal axis in the profile length plot, Figure 1b, keeping out only the last point of the graph, in order to better show the behaviour of the correlation with shorter profiles, where the density of the points is sensibly higher.

The first plot of both figures highlights the monotonicity of the correlation between the eigenvalue confidence index λ_u and the MAP@10 performance. Moreover, the values of λ_u are uniformly distributed in the range between the minimum and the maximum.

The decreasing monotonicity of the user rating variance is less evident and noisier if compared to the λ_u increasing one, which explains the low values of the correlation coefficients.

More peculiar are the plots for the profile length PL in Figures 1b and 2b. For both algorithms on Netflix, and for PureSVD on Movielens, the plots are divided into three regions, based on the profile length, each one with different properties.

- **Short profiles – profiles with less than 20 ratings:** there is a very strong correlation between profile length and accuracy.

Table 3: Comparison on implicit datasets of the correlations between performance @10 and estimation parameters

Recommender	Dataset	Correlation	HR		MAP		NDCG	
			λ_u	PL	λ_u	PL	λ_u	PL
ItemKNN	LastFM	Pearson	0.6722	−0.3586	0.5967	−0.1589	0.6342	−0.2242
		Spearman	0.6727	−0.2700	0.6275	−0.1871	0.6440	−0.2197
	TVAudience	Pearson	0.7525	0.6339	0.7732	0.6517	0.7795	0.6662
		Spearman	0.7486	0.6326	0.7656	0.6976	0.7795	0.7024
	Xing2017	Pearson	0.5864	0.3814	0.3879	0.2148	0.4514	0.2648
		Spearman	0.7332	0.6481	0.4387	0.2734	0.4477	0.3194
PureSVD	LastFM	Pearson	0.5471	−0.2909	0.5699	−0.1097	0.5871	−0.1672
		Spearman	0.5070	−0.2616	0.5264	−0.1220	0.5843	−0.1546
	TVAudience	Pearson	0.5816	0.2109	0.4947	0.1759	0.5403	0.1967
		Spearman	0.5653	0.1114	0.3925	0.1393	0.4675	0.1390
	Xing2017	Pearson	0.4653	0.2267	0.2935	0.1527	0.3528	0.1811
		Spearman	0.6607	0.4380	0.3433	0.2193	0.3495	0.2589

Table 4: Comparison on explicit datasets of the correlations between performance @10 and estimation parameters

Recommender	Dataset	Correlation	HR			MAP			NDCG		
			λ_u	PL	σ_u	λ_u	PL	σ_u	λ_u	PL	σ_u
ItemKNN	ML 20M	Pearson	0.9534	0.6891	−0.6330	0.9465	0.7704	−0.5740	0.9547	0.7655	−0.6138
		Spearman	0.9554	0.4171	−0.5628	0.9516	0.5036	−0.5471	0.9594	0.4982	−0.5721
	Netflix	Pearson	0.8824	0.7476	−0.2475	0.8874	0.7544	−0.1747	0.8934	0.7619	−0.2075
		Spearman	0.8922	0.4335	−0.3041	0.8954	0.4535	−0.2060	0.9023	0.4516	−0.2528
PureSVD	ML 20M	Pearson	0.8258	−0.0106	−0.6013	0.7729	0.0154	−0.5232	0.8017	0.0050	−0.5752
		Spearman	0.7672	−0.5495	−0.5928	0.6858	−0.4863	−0.5533	0.7267	−0.5287	−0.5974
	Netflix	Pearson	0.8022	0.4862	−0.1511	0.8151	0.4640	−0.0966	0.8250	0.4816	−0.1201
		Spearman	0.8450	0.0178	−0.2333	0.8468	0.0694	−0.1592	0.8620	0.0473	−0.1960

- **Intermediate profiles – between 20 and 50 (Movielens), between 20 and 100 (Netflix):** there is a strong **negative** correlation between profile length and accuracy.
- **Longer profiles:** the correlation becomes again moderately positive.

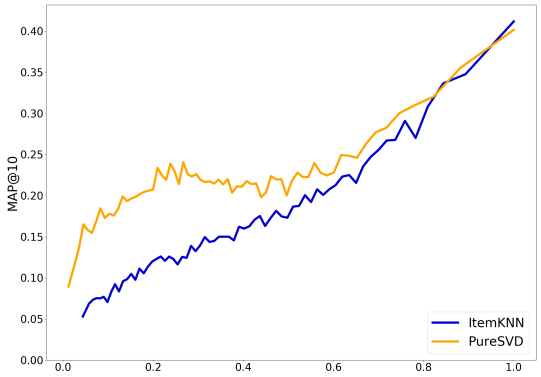
These unexpected results make the profile length an unreliable confidence estimator of the recommendation quality.

A possible explanation for this behaviour derives from the consideration that users with very short profiles will likely have rated mainly popular items or items within one specific category (for instance, only action movies) [4]. The recommender algorithm is able to well model these users, and the model improves its quality with the number of ratings in the user profile. Users with longer profiles will likely have rated more diverse items, and this diversity makes the recommendation task more difficult. For these users, the longer is the profile length, the greater is the confusion for the algorithm in correctly modelling users' taste. Only when the number of ratings in the profile is large enough, the algorithm is able to correctly capture the diverse interests of the user on to provide more and more accurate recommendations.

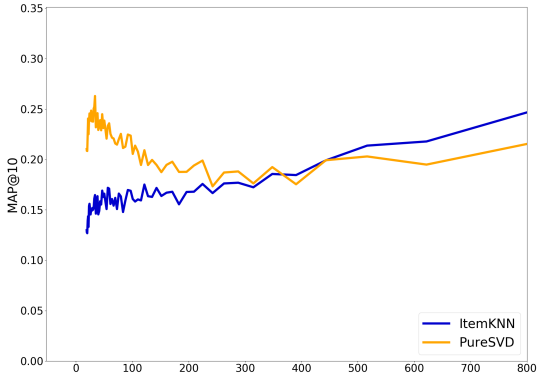
A similar *inversion* of correlation is present for the eigenvalue confidence index, limited to ItemKNN (Movielens) and PureSVD (Netflix). However, this inversion is limited in amplitude and does not affects significantly the overall correlation between eigenvalue confidence index and accuracy.

7 CONCLUSIONS

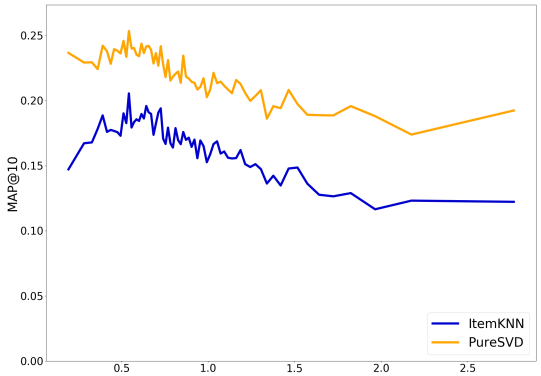
In this paper we investigate confidence estimation for item-based CF algorithms. We show that an ideal item-based method can be formulated as an eigenvalue problem, where estimated ratings are equivalent to the true ratings multiplied by a user-specific eigenvalue of the similarity matrix. We show that the magnitude of the eigenvalue is strongly correlated with the accuracy of recommendations for that user and it can provide reliable measure of confidence for predicted ratings. Thanks to the eigenvalue analogy, we present a new confidence index. Experiments show that the proposed confidence index outperforms other approaches in estimating reliability of recommendations. The results presented in this paper are not limited to item-based methods but can be used also with a broad class of model-based matrix-factorization algorithms.



(a) Eigenvalue confidence index

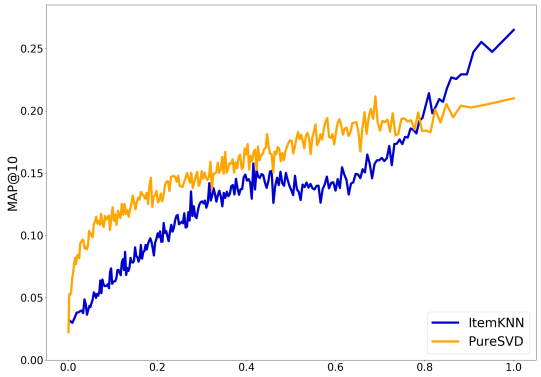


(b) Profile length

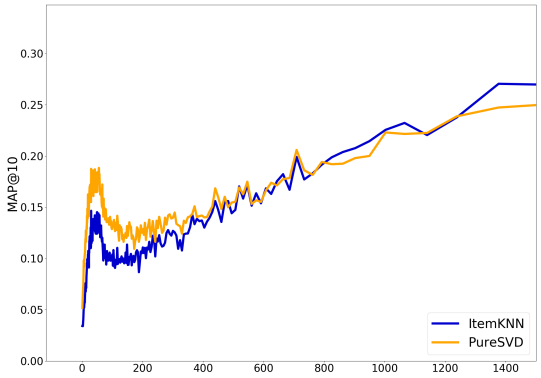


(c) User rating variance

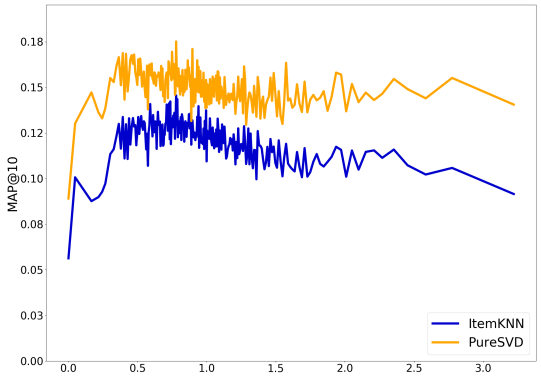
Figure 1: Comparison of MAP@10 and estimation parameters on the Movielens dataset.



(a) Eigenvalue confidence index



(b) Profile length



(c) User rating variance

Figure 2: Comparison of MAP@10 and estimation parameters on the Netflix dataset.

REFERENCES

[1] Gediminas Adomavicius, Sreeharsha Kamireddy, and YoungOk Kwon. 2007. Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance. In *17th Workshop on Information Technologies and Systems, WITS 2007*.

[2] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2013. User-centric vs. system-centric evaluation of recommender systems. In *IFIP Conference on Human-Computer Interaction*. Springer, 334–351.

[3] Paolo Cremonesi, Franca Garzottto, and Roberto Turrin. 2012. User Effort vs. Accuracy in Rating-based Elicitation. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12)*. ACM, New York, NY, USA, 27–34. https://doi.org/10.1145/2365952.2365963

[4] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 39–46.

[5] Mukund Deshpande and George Karypis. 2004. Item-based top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 143–177. https://doi.org/10.1145/963770.963776

[6] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. 2011. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction* 4, 2 (2011), 81–173.

[7] Maurizio Ferrari Dacrema and Paolo Cremonesi. 2018. Eigenvalue analogy for confidence estimation in item-based recommender systems. *Proceedings of the Late-Breaking Results of the 12th ACM Conference on Recommender Systems (RecSys 2018)*. http://arxiv.org/abs/1809.02052

[8] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00)*. ACM, New York, NY, USA, 241–250. https://doi.org/10.1145/358916.358995

[9] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

[10] Antonio Hernando, Jesús Bobadilla, Fernando Ortega, and Jorge Tejedor. 2013. Incorporating reliability measurements into the predictions of a recommender system. *Information Sciences* 218 (2013), 1 – 16. https://doi.org/10.1016/j.ins.2012.06.027

[11] Anton Howard. 1987. Aljabar Linear Elementer Edisi Kelima. *Terj. dari Elementary Linear Algebra, Fifth Edition, oleh Pantur Silaban & I Nyoman. Erlangga., Jakarta* (1987).

[12] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 426–434. https://doi.org/10.1145/1401890.1401944

[13] Yehuda Koren and Joe Sill. 2011. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 117–124. https://doi.org/10.1145/2043932.2043956

[14] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. *Content-based Recommender Systems: State of the Art and Trends*. Springer US, Boston, MA, 73–105. https://doi.org/10.1007/978-0-387-85820-3_3

[15] Maciej A. Mazurowski. 2013. Estimating Confidence of Individual Rating Predictions in Collaborative Filtering Recommender Systems. *Expert Syst. Appl.* 40, 10 (Aug. 2013), 3847–3857. https://doi.org/10.1016/j.eswa.2012.12.102

[16] Athanasios N Nikolakopoulos, Vassilis Kalantzis, Efstratios Gallopoulos, and John D Garofalakis. 2017. EigenRec: generalizing PureSVD for effective and efficient top-N recommendations. *Knowledge and Information Systems* (2017), 1–23.

[17] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*. 497–506. https://doi.org/10.1109/ICDM.2011.134

[18] Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender systems handbook*. Springer, 257–297.

[19] Sean Mcnee Shyong, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. 2003. Confidence Displays and Training in Recommender Systems. In *Proceedings of the 9th IFIP TC13 International Conference on HumanComputer Interaction (INTERACT*. 176–183.

[20] Roberto Turrin, Andrea Condorelli, Paolo Cremonesi, and Roberto Pagano. 2014. Time-based TV programs prediction. In *1st Workshop on Recommender Systems for Television and Online Video at ACM RecSys 2014*, Vol. 14. https://home.deib.polimi.it/pagano/portfolio/papers/Time-basedTVprogramsprediction.pdf Dataset: http://recsys.deib.polimi.it/datasets/.