Noname manuscript No.
(will be inserted by the editor)

Analyzing and Improving Stability of Matrix Factorization for Recommender Systems

Edoardo D'Amico* · **Giovanni Gabbolini*** ·
Cesare Bernardis · **Paolo Cremonesi**

Received: 15-04-2021 / Accepted: -

Abstract Thanks to their flexibility and scalability, collaborative embedding-based models are widely employed for the top-N recommendation task. Their goal is to jointly represent users and items in a common low-dimensional embedding space where users are represented close to items for which they expressed a positive preference.

The training procedure of these techniques is influenced by several sources of randomness, that can have a strong impact on the embeddings learned by the models. In this paper we analyze this impact on Matrix Factorization (MF). In particular, we focus on the effects of training the same model on the same data, but with different initial values for the latent representations of users and items. We perform several experiments employing three well known MF implementations over five datasets. We show that different random initializations lead the same MF technique to generate very different latent representations and recommendation lists. We refer to these inconsistencies as *instability of representations* and *instability of recommendations*, respectively. We report that stability of item representations is positively correlated to the accuracy of the model. We show that the stability issues affect also the items for which the recommender correctly predicts positive preferences. Moreover, we highlight that the effect is stronger for less popular items.

To overcome these drawbacks, we present a generalization of MF called Nearest Neighbors Matrix Factorization (NNMF). The new framework learns the embed-

Edoardo D'Amico*
Insight Centre for Data Analytics,
University College Dublin, Ireland E-mail: edoardo.damico@insight-centre.org

Giovanni Gabbolini*
Insight Centre for Data Analytics,
University College Cork, Ireland E-mail: giovanni.gabbolini@insight-centre.org

Cesare Bernardis
Politecnico di Milano, Italy E-mail: cesare.bernardis@polimi.it

Paolo Cremonesi
Politecnico di Milano, Italy E-mail: paolo.cremonesi@polimi.it

* The two authors have contributed equally to the work.

ding of each user and item as a weighted linear combination of the representations of the respective nearest neighbors. This strategy has the effect to propagate the information about items and users also to their neighbors and allows the embeddings of users and items with few interactions to be supported by a higher amount of information. To empirically demonstrate the advantages of the new framework, we provide a detailed description of the NNMF variants of three common MF techniques. We show that NNMF models, compared to their MF counterparts, largely improve the stability of both representations and recommendations, obtain a higher and more stable accuracy performance, especially on long-tail items, and reach convergence in a fraction of epochs.

Keywords Matrix Factorization · Nearest Neighbors · Stability · Popularity Bias

1 Introduction

Embedding-based models for collaborative filtering represent a wide family of approaches for top-N recommendation [12, 22, 49]. Their goal is to jointly represent users and items in a common, low-dimensional latent factor space, also called embedding space, so that users or items with similar profiles have matching latent representations, also called embeddings [32]. In order to learn these representations, various techniques have been proposed in literature, ranging from Matrix Factorization (MF) [29, 33, 17] to Deep Learning [22, 50, 49].

The training procedures of these approaches are influenced by several sources of randomness, such as the initial values of the embeddings and the order followed in the exploration of the available interactions. Using different random seeds or random generators might induce the models to learn different representations for users and items at convergence (instability of representations), and, consequently, lead to the generation of different recommendation lists (instability of recommendations). The magnitude of these differences determines whether an algorithm can be considered *stable* or not [43].

While several definitions of stability have been proposed in the recommender systems literature [41, 35, 4, 3], in this work we focus our analysis on one specific definition of stability: the same recommendation model is trained on the same dataset and in exactly the same experimental conditions (i.e., exploring the data points in the same order and using the same hyper-parameters configuration), but with a different random sequence used in the initialization of the latent factors, due to a different random seed. In other words, we vary only the initial values assigned to the latent factors that compose the representations of users and items, excluding all the other sources of randomness.

Instability of representations and recommendations strongly impacts several aspects of a recommender. First, the fact that different initializations converge to different local optimal solutions suggests that the learned model overfits the training data, reducing its ability to generalize [20, 9]. This also has an impact on the accuracy of a recommender, especially on niche items that have few interactions (i.e., long-tail items), where generalization plays a fundamental role. Second, the reliability of its recommendations becomes questionable, since the same technique trained on the same data would provide different predictions for the same user-item pair [34]. Third, the quality and the reliability of the explanations provided for

recommendations is compromised. Indeed, most approaches that face the explainability problem in embedding-based models rely on the similarities between the latent representations learned by the model [45, 1]. Fourth, an algorithm affected by this type of instability is not *repeatable*[1], since even in the same experimental setup it can provide different results.

Different techniques exist to improve the generalization capabilities of a model by leveraging or controlling the randomness of the training procedure. Bagging [44] is an ensemble method that trains different models from boostrap replica of the same dataset and average their predictions. However, it requires to retrain the model several times and it is designed to improve the generalization and not to stabilize the model. Stochastic Weight Averaging is another ensemble technique that averages the weights of the same model at different epochs of the training process [34, 25]. All these approaches are model agnostic and do not take into account the neighborhood properties of user and item embeddings: similar items and users have similar embeddings.

In literature, it is well know that that same MF algorithm, trained on the same data, can converge to different local optimal solutions during the training as the result of different initializations of the embeddings [52, 19]. However, the study of the consequences of converging to different local optima, in terms of the generalization capability of the model (as related to its stability) and, therefore, in terms of reduced accuracy, is unprecedented. Indeed, there are several works that define and measure the stability of recommendations, but no study that (i) measures the impact of stability on the quality of recommendations, (ii) defines and measures the stability of representations, (iii) suggests how to improve the stability and the ability of the learned representations to generalize and to provide more accurate recommendations.

The goal of our work is to fill this gap with a twofold contribution. First, we analyze the stability of Matrix Factorization, one of the most successful families of embedding-based models for top-N recommendation. In particular, we study the differences in the embeddings and the recommendations produced by the same model under different initializations of latent factors. Moreover, we analyze the correlation between stability and accuracy of MF models since, as already outlined in previous works [3], stability and accuracy are usually positively correlated. Second, we propose a new framework called Nearest Neighbors Matrix Factorization (NNMF) as a generalization of classic matrix factorization. In NNMF the embedding of each user, or item, is obtained as a weighted linear combination of the latent representations of its closest neighbors. This property allows exploiting the existing relationships, under the form of similarities, in the original interaction space, and to transduce them in the embedding space learned by the MF model. We provide the NNMF implementation of three well known MF approaches, namely BPR-MF [39], Funk-MF [15] and P-MF [42]. Beyond the stability, we also analyze the accuracy of the proposed models at different levels of popularity, comparing them with their MF counterparts. With an extensive set of experiments over five datasets, we show that:

- the stability of item representations is directly correlated with the accuracy of common MF instances;

[1] The definition of repeatability is reported in the ACM Artifact Review and Badging guidelines: `acm.org/publications/policies/artifact-review-and-badging-current`

- long-tail items are more affected by the stability issues outlined in this paper;
- the NNMF framework greatly improves stability over traditional MF, and it is particularly effective on less popular items;
- the NNMF variants achieve better accuracy, especially on the long-tail, with lower variance with respect to the original MF methods in almost all measures and datasets;
- propagating information in the neighborhood allows NNMF models to reach convergence in a fraction of the number of epochs required by MF.

For the sake of reproducibility, we also provide the source code used to perform all the experiments[2].

This work is an extension of [16]. Compared to the previous version, we provide more details on the new NNMF models we propose and a broader set of experiments. The rest of the paper is organized as follows. In Section 2 we report the most important works related to the main arguments treated in this paper. In Section 3 we introduce Matrix Factorization and the related stability issue. Then we present our new framework called Nearest Neighbor Matrix Factorization and we describe in detail three practical implementations based on three well known MF algorithms. In Section 4 we show and discuss the results of the stability and accuracy analyses. Finally, we conclude our paper with some remarks and future directions in Section 5.

2 Related Work

In the following two sections, we report relevant publications related to our work. The first section defines the concept of stability for a recommendation system algorithm and presents which are the techniques currently adopted to improve it. The second one report previous works that have tried to develop models inspired from both MF and Nearest Neighbors algorithms.

2.1 Stability

There exist different definitions of stability of a recommender system in the literature, and different ways to improve each of these definitions. Most works define the stability of a recommender system as the 'consistent agreement of predictions' made to the same user by the same algorithm, when new incoming interactions are added to the system in complete agreement to system's prior predictions [3, 37]. A refinement of this definition of stability has been addressed in [4], where the authors adopt bagging and iterative smoothing in conjunction with different traditional recommendation algorithms to improve their consistency. Specifically targeting matrix factorization algorithms, in [31] authors introduce a dynamic weighting strategy for negative samples. In [30], a new optimization process is proposed based on clustering data identifying harder set of samples in the training set.

Other works define stability as the ability of the recommender system to provide consistent recommendations when malicious perturbations are performed to

[2] `https://github.com/damicoedoardo/NNMF`

the dataset [3]. The work in [35] suggests hybrid collaborative and content-based filtering as the best solution to mitigate the effects of attacks on the consistency of recommendations. Finally, other works [40] relate the stability, or confidence, of a recommender system with the quality of a dataset, either at system level (the magic barrier described in [40]) or at user-level [8]. Our notion of stability – the consistency of both recommendations and latent representations of users and items when the same model is trained on exactly the same dataset with a different random sequence used to initialize latent factors of users and items – is different from the definitions used in the literature.

There are also several works that try to control (or leverage) the randomness intrinsic in machine learning algorithms in order to improve the generalization capabilities of a model. Bagging is the most widely adopted black-box method used to leverage randomness in the input data in order improve the classification accuracy of a model [44]. Bagging builds an ensemble of models by (i) running the same training algorithm on different boostrap replica of the same dataset and (ii) by aggregating their predictions. Training multiple model for prediction averaging, as with bagging, is computationally expensive. Therefore, other works train a single model and save the model parameters (snapshots) along the optimization path. The predictions of the snapshot models are later combined to produce the final prediction [24]. Differently from bagging and snapshots, that build ensambles in the model space, other works build ensembles in the weights space. For instance, the works in [34] and [25] use two variants of the same technique, Stochastic Weight Averaging (SWA), to compute a running average of the model weights during the last epochs of the training process.

2.2 Matrix Factorization and Nearest Neighbors

There is a family of algorithms which mixes ideas from MF and from Nearest-Neighbors (NN) techniques. The models belonging to this category embed information from both the two sub families and provide predictions which are, at least ideally, different from the simple union of the recommendations coming from a NN and from a MF. As reported in [27], MF and NN methods perform well in complementary scenarios. Experimentally, the author observes that NN techniques are able to grasp strong associations among a small set of closely related items, while they are weak when it comes to estimate the overall structure that relates simultaneously to most of or all items. MF, instead, works the other way around. This observation justifies the existence of the hybrid category of models introduced in this section. In the following, we cite notable attempts of MF models extended with Neighbors information.

Koren [27] introduces a modified version of an item based NN model where the similarity among items is learned solving an optimization problem. Then, he introduces SVD++, which is a MF extension able to handle implicit feedback. Last, he sums the prediction rules of those two techniques, formulating a new model and an associated objective function, which is then optimized by gradient descent. In [28, 46], the authors introduce NN models in which the similarity matrix is learned from data. In order to lower the number of model parameters, they propose to learn a factorized similarity, *i.e.* two low-dimensional matrices that, through a multiplication, reconstruct the complete $|\mathcal{I}| \times |\mathcal{I}|$ similarity matrix. Bell et. al [7] introduce

a similarity model inspired by [27] and a novel procedure for learning latent factors in MF. Then, they combine the predictions trying to exploit the strengths of both the NN and MF models by means of a confidence coefficient. Paterek [38] proposes a NN variation which computes the similarity matrix as the dot product between the rows of the latent factors learned by Funk SVD [15]. Lastly, Zhang [51] proposes a modified version of matrix factorization where information about similar users is used to create new regularization terms to be considered during the latent factor optimization process. Besides the many attempts done in the direction of formulating a unique model that takes advantage of both the MF and NN approaches, in the literature we can find many examples of the increments on the perfomance that can be achieved by combining the recommendations provided by independent MF and NN models [6].

Combining the benefits of MF and NN is at the basis of NNMF. As shown in this section, the idea to merge MF and NN is not new, and it has proved to be effective in improving accuracy and quality of recommendations [27]. However, differently from all previous approaches, NNMF is not a unique model that combines MF and NN. It is, instead, a framework that can be applied to any shallow or deep model based on the factorization of users and items, with the aim to improve the final accuracy of standard factorization models, enhancing their stability and generalization capability. As such, our goal is to compare traditional matrix factorization methods with the respective NNMF variants, to assess the improvements brought by NNMF over the traditional approaches. Such a comparison is not possible with the other methods in the literature.

3 Models

In this section, we provide the basics of matrix factorization and we discuss about the stability issues that affect it. We propose the Nearest Neighbors Matrix Factorization framework and we argue about its advantages over traditional MF. Finally, we provide a detailed description of the NNMF variant of three widely known collaborative MF approaches for top-N recommendation.

3.1 Preliminaries

In this paper, we denote the sets of users and items as $\mathcal{U}$ and $\mathcal{I}$ respectively. We use lower case letters to refer to single entities that belong to these sets. In particular, u, v are used to indicate users, while i, j, k indicate items.

Lower case, bold letters denote vectors in column format, unless differently specified, while uppercase bold letters denote matrices. The User Rating Matrix (URM) is represented with letter $\mathbf{R}$ and each cell r_{ui} contains the value of the preference, either explicit or implicit, that a user provided for an item. If no feedback is available, r_{ui} is set to 0. Symbol $\mathbf{r}_u$ indicates the user profile, intended as the u-th row of matrix $\mathbf{R}$, while $\mathbf{r}_i$ indicates the item profile, intended as the i-th column of $\mathbf{R}$. We finally define κ as the set of user-item couples (u, i) for which r_{ui} is known.

3.2 Matrix Factorization

The main goal of Matrix Factorization is to decompose the original User Rating Matrix into the product of two dimensionally lower matrices. These two matrices contain the representations of users and items, also called *embeddings*, in a common, low-dimensional latent factor space. In this space, users are represented close to items for which they expressed a positive preference, using the dot product as proximity measure.

Formally, given a latent space of dimension f, users and items are represented respectively in matrices $\mathbf{P} \in \mathbb{R}^{|\mathcal{U}| \times f}$ and $\mathbf{Q} \in \mathbb{R}^{|\mathcal{I}| \times f}$. Each row $\mathbf{p}_u$ of $\mathbf{P}$ contains the representation of user u in the latent factor space. Each row $\mathbf{q}_i$ of $\mathbf{Q}$ handles the representation of item i in the same space. The dot product between a user and an item vectors provides an estimation of the rating for the respective user-item couple:

$$\overline{r}_{ui} = \mathbf{p}_u \cdot \mathbf{q}_i^T \qquad \overline{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T \tag{1}$$

Over the years, researchers proposed several implementations of MF, varying how matrices $\mathbf{P}$ and $\mathbf{Q}$ are learned [15, 39, 42], in order to improve the quality of recommendations under different aspects. Most of them learn the parameters of the model optimizing an objective function through stochastic gradient descent, iterating over the available data. The training phases of such algorithms share a common schema, which is partially altered from case to case. Firstly the two latent factor matrices $\mathbf{P}$ and $\mathbf{Q}$ are initialized with random values, then an iterative learning procedure begins. Within each iteration, one or more interactions are selected among the available ones, and the objective function is computed alongside its respective gradient. Finally, the latent factors of the sampled items and users are updated accordingly.

The iterative exploration of the available interactions leads classic MF approaches to follow their distribution during the training procedure. A well known issue that affects common recommender system datasets is the popularity bias: most interactions are referred to a small set of popular items. As a consequence, factors belonging to popular items are updated far more often than niche ones during the training procedure of an algorithm. The same holds for the users with long profiles. Due to the low number of updates performed on the latent representations of unpopular items and short-profile users, we expect that the initial values of the latent factors, that are randomly assigned at the beginning of the training process, have a strong impact on their final representations at convergence. In other words, the scarcity of interactions available for items (e.g., for unpopular items) or users can lead MF techniques to learn very different representations for the same item or user, just depending on their initial values. We refer to this issue as *instability of representations*. The same problem directly affects also the recommendation lists generated by the algorithms, as they are strictly connected to the mutual representations of users and items. Indeed, a Matrix Factorization algorithm recommends the nearest items to a user, according to the respective representations in the latent factor space learned. If the representation of a user is unstable, the same holds also for the closest items in the latent space, leading to the generation of different recommendation lists for the same user, based on different random initial conditions. We refer to this issue as *instability of recommendations*.

Table 1: Stability of Top-10 recommendation lists generated by three MF techniques expressed with Jaccard index. Higher values indicate a better stability. Datasets names abbreviations are reported in Section 4.1.

Algorithm	**LFM**	**M1M**	**BCR**	**PIN**	**CUL**
BPR-MF	0.72	0.80	0.31	0.53	0.51
Funk-MF	0.75	0.65	0.06	0.33	0.33
P-MF	0.64	0.65	0.24	0.44	0.39

As an example, in Table 1 we show the stability of the top-10 recommendations for three common MF techniques, expressed with the Jaccard index, that indicates how much the generated lists overlap. We compare the recommendations provided by 10 instances of the same model. All the instances are trained on the same data, explored in the same order, and with the same configuration. The only difference we allowed between the training procedures of different instances was in the initial values of the latent factors. We obtained this difference by changing the random seed used at the beginning of the initialization[3]. The results show a worrying trend, since the recommendation lists of all the three algorithms overlap by less than 50% on three datasets out of five. From another perspective, this means that more than half of the items recommended by the same algorithm, trained on the same data, vary by only modifying the initial random seed, i.e. by simply altering the initial values of the latent factors. For the BookCrossing dataset the instability is even more dramatic, since lists overlap by less than 30%. More details about the experimental procedure and a wider range of experiments are reported in Section 4.

3.3 Nearest Neighbors Matrix Factorization

As discussed in Section 3.2, MF learns users and items representations, also called embeddings, in a new latent space where users are mapped close to items they have expressed positive preference for. One of the main drawbacks of MF algorithms is that they treat individual users and items as independent entities. However, relationships among users and among items do exist in the original interaction space and transduce them in the new latent space would be beneficial. We hence propose a new framework called *Nearest Neighbors Matrix Factorization* (NNMF), which is able to let Matrix Factorization algorithms leverage knowledge about users and items relationships, under the form of similarities, during the algorithm learning procedure. Given the two latent factor matrices $\mathbf{P}$ and $\mathbf{Q}$ presented in Section 3.1, we define the new neighborhood-aware latent representations for users $\mathbf{P}^*$ and items $\mathbf{Q}^*$ as:

$$\mathbf{P}^* = \mathbf{S}^U\mathbf{P} \qquad \mathbf{Q}^* = \mathbf{S}^I\mathbf{Q}$$

where $\mathbf{S}^U$ and $\mathbf{S}^I$ are a user similarity matrix of size $|\mathcal{U}|\times|\mathcal{U}|$ and an item similarity matrix of size $|\mathcal{I}| \times |\mathcal{I}|$, respectively. Each element s_{xy} stores the value of the similarity between entity x and y, being them either users or items (i.e. $x, y \in \mathcal{U}$

[3] Note that instances that use the same random seed for the initialization would converge to the exact same results.

or $x, y \in \mathcal{I}$). In the implementation of NNMF that we propose, these similarity matrices are constant matrices that can be pre-computed by using any traditional nearest-neighbor collaborative-filtering approach. As such, NNMF can be easily applied to almost any MF algorithm. Notice that if both similarity matrices $\mathbf{S}^U$ and $\mathbf{S}^I$ are identity matrices, NNMF collapses to classic MF. It follows that the new NNMF framework is a generalization of MF.

An important characteristic of the new technique can be highlighted by exploding the neighborhood-aware representations:

$$\mathbf{p}_u^* = \sum_{v}^{\mathcal{U}} s_{uv}^U \mathbf{p}_v = s_{uu}^U \mathbf{p}_u + \sum_{v}^{\mathcal{U}\setminus\{u\}} s_{uv}^U \mathbf{p}_v \tag{2}$$

$$\mathbf{q}_i^* = \sum_{j}^{\mathcal{I}} s_{ij}^I \mathbf{q}_j = s_{ii}^I \mathbf{q}_i + \sum_{j}^{\mathcal{I}\setminus\{i\}} s_{ij}^I \mathbf{q}_j \tag{3}$$

Clearly, every user/item embedding is composed by an independent component that directly refers to the user/item being represented, and a second component defined by a weighted representation of the neighbors. The magnitude of the contribution of each neighbor is proportional to the similarity with the user or the item we are considering: the more similar they are, the stronger the contribution will be. Moreover, note that, with the proposed formulation, $\mathbf{P}$ and $\mathbf{Q}$ do not directly contain users and items embeddings. They contain, instead, vectors that form a generating set, not necessarily a basis, for the vector space where users and items representations are projected. Embeddings for users and items are now contained in $\mathbf{P}^*$ and $\mathbf{Q}^*$, respectively.

Finally, we can modify Equation (1), used to estimate the preferences and provide recommendations, with the new formulation of the latent representations, rewriting it as:

$$\overline{\mathbf{R}} = \mathbf{P}^* \mathbf{Q}^{*T} = \mathbf{S}^U \mathbf{P} \mathbf{Q}^T {\mathbf{S}^I}^T \tag{4}$$

$$\overline{r}_{ui} = \mathbf{p}_u^* \cdot \mathbf{q}_i^{*T} = \left(\sum_{v}^{\mathcal{U}} s_{uv}^U \mathbf{p}_v \right) \cdot \left(\sum_{j}^{\mathcal{I}} s_{ij}^I \mathbf{q}_j \right) \tag{5}$$

The main advantage of NNMF over traditional MF is that the knowledge we have concerning a user or an item is also spread to the respective closest neighbors. This translates into two important benefits. First, it allows having a larger amount of information supporting the latent representations of items and users. This aspect is particularly important for users and items that have scarce data available, and it leads to a globally higher stability of recommendations and representations, as the empirical experiments in Section 4 demonstrate. The second benefit can be easily highlighted looking at the differences between Algorithm 1 and 2. In Algorithm 1 we report the pseudocode that implements the training procedure of a classic MF approach. In Algorithm 2 we show the pseudocode of the training process referred to a NNMF implementation. In the latter, the updates made to the embeddings during the learning procedure are not restricted to the user and the item associated to the sampled interaction, but they are also propagated to the representations of users and items in the neighborhoods of u and i. This allows disseminating the information learned on a user or an item also to the neighbors, resulting in a faster convergence of the model.

Algorithm 1 MF

1: Randomly initialize $\mathbf{P}$ and $\mathbf{Q}$
2: **while** not converged **do**
3: Extract a training sample randomly
4: Compute gradients
5: Update embedding of user u
6: Update embedding of item i

Algorithm 2 NNMF

1: Randomly initialize $\mathbf{P}^*$ and $\mathbf{Q}^*$
2: **while** not converged **do**
3: Extract a training sample randomly
4: Compute gradients
5: **for** $v \in \mathcal{U}$ **do**
6: Update embedding of user v according to s^U_{uv}
7: **for** $j \in \mathcal{I}$ **do**
8: Update embedding of item j according to s^I_{ij}

3.4 Similarity

In the NNMF algorithm, relationships among users and items are modeled in the form of similarity matrices $\mathbf{S}^U$ and $\mathbf{S}^I$. Even though in the definition of the framework we did not make any assumption on how these matrices are obtained, for the NNMF instances proposed in this paper we assume that the similarity values are calculated using the *shrinked cosine similarity* function:

$$s_{uv} = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u|| \cdot ||\mathbf{r}_v|| + h_U} \qquad s_{ij} = \frac{\mathbf{r}_i \cdot \mathbf{r}_j}{||\mathbf{r}_i|| \cdot ||\mathbf{r}_j|| + h_I} \tag{6}$$

where $\mathbf{r}_u$ and $\mathbf{r}_v$ are user profiles, $\mathbf{r}_i$ and $\mathbf{r}_j$ are item profiles and h_U and h_I are the shrink terms. Moreover, for every item and user we kept only a small number of the nearest neighbors, since we noticed that this approach led to the best performance.

Note that the choice of the cosine has two main advantages. The first is that it is simple and fast to compute. The second is that it allows a fairer comparison, since the NNMF model has the same complexity and the same number of parameters to learn compared to the original MF variant.

3.5 Instances

We implement three NNMF algorithms as generalizations of three common MF algorithms: Funk-MF [15], BPR-MF [39] and P-MF [42].

3.5.1 Funk-NNMF

Simon Funk [15] proposed one of the earliest implementations of MF as a simplified version of Singular Value Decomposition. Users and items are represented in a common, low-dimensional latent space, and the rating prediction for a user-item couple is performed through the dot product between the respective latent representations, as described in Equation 1. The optimization procedure minimizes the following regularized MSE loss function:

$$J = \sum_{(u,i)}^{\kappa} \left(r_{ui} - \overline{r}_{ui}\right)^2 + \lambda_p ||\mathbf{P}||^2 + \lambda_q ||\mathbf{Q}||^2 \tag{7}$$

where λ_p and λ_q are the variables that control the regularization.

The NNMF version of FUNK-MF can be obtained replacing the original prediction rule, using the variant proposed in Equation (5). The loss function to minimize consequently becomes:

$$J = \sum_{(u,i)}^{\kappa} \left(r_{ui} - \left(\sum_{v}^{\mathcal{U}} s^U_{uv} \mathbf{p}_v \right) \cdot \left(\sum_{k}^{\mathcal{I}} s^I_{ik} \mathbf{q}_k \right) \right) + \lambda_p ||\mathbf{P}||^2 + \lambda_q ||\mathbf{Q}||^2 \tag{8}$$

Algorithm 3 FUNK-NNMF

```
Randomly initialize matrices P* and Q*
κ = {(u, i)|r_ui is known }
while not converged do
    Randomly sample (u, i) ∈ κ
    for v ∈ U do
        p_v ← p_v + α[(r_ui − r̄_ui) s^U_uv Σ_k^I s^I_ik q_k − λ_p p_v]
    for k ∈ I do
        q_k ← q_k + α[(r_ui − r̄_ui) s^I_ik Σ_v^U s^U_uv p_v − λ_q q_k]
```

3.5.2 BPR-NNMF

Bayesian Personalized Ranking (BPR) is a generic optimization criterion for user personalized ranking of items [39]. This approach learns to rank couples of items correctly, according to the user preferences, instead of scoring them singularly. Each couple is composed by an item that belongs to the set of items for which the user u provided a positive feedback ($\mathcal{I}_u^+$), and an item that does not belong to that set. Defining the training data as $\mathcal{D}_S = \{(u,i,j)|u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$, the generic optimization criterion is to maximize:

$$\text{BPR-Opt} = \sum_{(u,i,j)}^{\mathcal{D}_S} \ln \sigma(\overline{x}_{uij}(\Theta)) - \lambda_\Theta \left\|\Theta\right\|^2 \tag{9}$$

The term $\overline{x}_{uij}(\Theta)$ represents an arbitrary real valued function of the model parameters Θ. It captures a specific relationship between user u and items i and j, so that $\sigma(\overline{x}_{uij}(\Theta))$ estimates the probability that user u prefers item i over j. In the case of BPR-MF, $\overline{x}_{uij}(\Theta)$ is defined as the difference between the preference values that a Matrix Factorization model predicts for user u on items i and j. In particular:

$$\overline{x}_{uij}(\Theta) = \overline{r}_{ui} - \overline{r}_{uj} \tag{10}$$

The NNMF version can be derived redefining the estimator $\overline{x}_{uij}$ according to the prediction rule reported in Equation (5):

$$\overline{x}_{uij}(\Theta) = \left(\sum_{v}^{\mathcal{U}} s^U_{uv} \mathbf{p}_v \right) \cdot \left(\sum_{k}^{\mathcal{I}} s^I_{ik} \mathbf{q}_k \right) - \left(\sum_{v}^{\mathcal{U}} s^U_{uv} \mathbf{p}_v \right) \cdot \left(\sum_{k}^{\mathcal{I}} s^I_{jk} \mathbf{q}_k \right) \tag{11}$$

Algorithm 4 BPR-NNMF

Randomly initialize matrices $\mathbf{P}^*$ and $\mathbf{Q}^*$
$D_S = \{(u,i,j)|i \in I_u^+ \wedge j \in I \setminus I_u^+ \wedge u \in U\}$
while not converged **do**
  Randomly sample $(u,i,j) \in D_S$
  $A_{ij} = \sum_k^{\mathcal{I}} s_{ik}^I \mathbf{q_k} - \sum_k^{\mathcal{I}} s_{jk}^I \mathbf{q_k}$
  $B_u = \sum_v^{\mathcal{U}} s_{uv}^U \mathbf{p_v}$
  $\overline{x}_{uij} = A_{ij} \cdot B_u$
  for $v \in \mathcal{U}$ **do**
    $\mathbf{p_v} \leftarrow \mathbf{p_v} + \alpha\left(\frac{e^{-\overline{x}_{uij}}}{1+e^{-\overline{x}_{uij}}} s_{uv}^U A_{ij} + \lambda_p \mathbf{p_v}\right)$
  for $k \in \mathcal{I}$ **do**
    $\mathbf{q_k} \leftarrow \mathbf{q_k} + \alpha\left(\frac{e^{-\overline{x}_{uij}}}{1+e^{-\overline{x}_{uij}}} \left(s_{ik}^I - s_{jk}^I\right) B_u + \lambda_q \mathbf{q_k}\right)$

3.5.3 P-NNMF

P-MF is a Probabilistic variant of Matrix Factorization introduced by Mnih *et al.* [42], that aims to improve scalability and accuracy in sparse scenarios of classic collaborative filtering approaches. The authors propose to train the model minimizing the following sum-of-squared-errors objective function with quadratic regularization terms:

$$J = \sum_{(u,i)}^{\kappa} (r_{ui} - \sigma(\overline{r}_{ui}))^2 + \lambda_p ||\mathbf{P}||^2 + \lambda_q ||\mathbf{Q}||^2 \tag{12}$$

where λ_p and λ_q are the model parameters that control the regularization. Note that the rating prediction $\overline{r}_{ui}$ is passed through the sigmoid function σ, in order to bound the predictions to the probability range $[0,1]$. The same must hold also for all the known ratings r_{ui}. The NN version of P-MF can be derived by adapting the prediction rule according to Equation 5. The loss consequently becomes:

$$J = \sum_{(u,i)}^{\kappa} \left(r_{ui} - \sigma\left(\sum_{v}^{\mathcal{U}} s_{uv}^U \mathbf{p}_v \cdot \sum_{k}^{\mathcal{I}} s_{ik}^I \mathbf{q}_k\right)\right)^2 + \lambda_p ||\mathbf{P}||^2 + \lambda_q ||\mathbf{Q}||^2 \tag{13}$$

Algorithm 5 P-NNMF

Randomly initialize matrices $\mathbf{P}^*$ and $\mathbf{Q}^*$
$\kappa = \{(u,i)|r_{ui} \text{ is known }\}$
while not converged **do**
  Randomly sample $(u,i) \in \kappa$
  $\mathbf{q}_i^* \leftarrow \sum_k^{\mathcal{I}} s_{ik}^I \mathbf{q}_k$
  $\mathbf{p}_u^* \leftarrow \sum_v^{\mathcal{U}} s_{uv}^U \mathbf{p}_v$
  $\overline{r}_{ui} \leftarrow \mathbf{q}_i^* \cdot \mathbf{p}_u^*$
  $g \leftarrow (r_{ui} - \sigma(\overline{r}_{ui}))\sigma(\overline{r}_{ui})(1 - \sigma(\overline{r}_{ui}))$
  for $v \in \mathcal{U}$ **do**
    $\mathbf{p}_v \leftarrow \mathbf{p}_v + \alpha[g s_{uv}^U \mathbf{q}_i^* - \lambda_p \mathbf{p}_v]$
  for $k \in \mathcal{I}$ **do**
    $\mathbf{q}_k \leftarrow \mathbf{q}_k + \alpha[g s_{ik}^I \mathbf{p}_u^* - \lambda_q \mathbf{q}_k]$

Table 2: Statistics of the datasets used for the experiments.

Dataset	Users	Items	Inter.	Density	Avg. Inter. per user	Avg. Inter. per item
LFM	1859	2823	42798	0.81%	23.0	15.2
M1M	6038	3307	501114	2.51%	83.0	151.5
BCR	13975	33925	314499	0.06%	22.5	9.3
PIN	55186	9637	877796	0.16%	15.9	91.1
CUL	5536	15429	119919	0.14%	21.7	7.8

4 Experiments

It is known that a MF algorithm, even if trained on the same data, can converge to different local optimal solutions and learn very different representations for users and items due to different initial values of latent factors [52, 19]. In this paper, our first goal is to study the consequences of converging to different local optima in terms of the ability of the model to generalize (as related to the stability of the model) and to assess the respective impact on the accuracy of the recommendations. The second goal is to propose NNMF as a remediation to the stability issues, providing empirical evidence of its benefits on both stability and accuracy with several experiments on five research datasets.

In this section, as first we report on the stability of the studied algorithms in both MF and NNMF variants. We focus on studying the stability of representations learned, showing its correlation with the recommendation accuracy and highlighting how the popularity bias has an impact on it. Afterwards, we analyze the stability of the set of items recommended with two experiments, the first taking into account the entire set of recommendations and the second one using only relevant ones. Then we shift our focus on analyzing the accuracy performance of the algorithms with a particular attention on the influence of the popularity bias. We first report a comparison on the long-tail performance with respect to relevant baselines, followed by an analysis of the performance restricted to item popularity clusters. Finally we report on the convergence of the proposed algorithms.

4.1 Datasets

We carry out experiments employing a number of research datasets:

LastFM (LFM) Implicit interactions gathered from the music website Last.fm. In particular, user *listened* artist relations expressed as listening counts. [10]

Movielens1M (M1M) Explicit interactions gathered from the website MovieLens. In particular, user *rated* movie relations. [21]

BookCrossing (BCR) Explicit interactions gathered from the online book club BookCrossing. In particular, user *rated* book relations. [53]

Pinterest (PIN) Implicit interactions gathered from the social network Pinterest. In particular, user *pin-to-own-board* image relations. [18]

CiteULike (CUL) Implicit interactions gathered from the online scientific community CiteULike. In particular, user *saved-to-own-library* paper relations. [47]

We employ datasets with densities of interactions that range from 0.04% to 2.51%, as we want to take into account the effect of a varying density of the datasets on both the accuracy and stability of the models. The statistical details of the datasets are described in Table 2.

The datasets used for the experiments have been preprocessed in two steps. First, we brought all the interactions to either 0 or 1 by means of thresholding, since BPR requires binary preference values. Among the implicit datasets, only LFM needs thresholding: we use threshold value equal to 1, hence we convert every interaction to one if the user has listened at least once to an artist. In explicit datasets we use threshold value equal to 6 for 1 to 10 ratings and equal to 3 for 1 to 5 ratings. Second we applied a filtering procedure keeping only users and items with at least 5 interactions, in order to remove entities with a too scarce amount of information.

Each dataset has been randomly partitioned performing a standard holdout procedure in three sets: train, validation and test sets accounting for 60, 20 and 20 percent of the available interactions.

4.2 Hyper-parameter tuning

For all algorithms and datasets, we conduct a hyper-parameter tuning procedure. We cannot resort to a grid search, due to the substantial number of parameters, algorithms and datasets. Instead, we employ a Bayesian search, using the implementation of Scikit-Optimize[4]. We set the search space empirically. We let the number of neighbors in NNMF algorithms to vary from two to 50. That is, we constrain NNMF to use at least two neighbors for users or items, so that it is not equivalent to MF, as elaborated in Section 3.3. We let the learning rate vary from 10^{-6} to 10^{-1}, the number of latent factors from 50 to 300, the batch size from 10^2 to 10^5, and the regularization terms from 10^{-6} to 10^{-1}. We run the Bayeasian search with 100 parameters configurations and 30 initial random restarts. We monitor the accuracy on the validation set every 5 epochs, as measured by MAP@5. We also apply early stopping, useful to speed up the search. Early stopping interrupts the training with a parameter configuration if continuing is not promising. In particular, if the metric on the validation set does not improve within a certain number of epochs, also known as patience, the training stops. We set the patience to 10.

4.3 Stability

The main focus of our stability study, is on the representations of items (and users) learned by the models, since they are directly related to both the instability of representations and the instability of recommendations issues introduced in Section 3.2. Assessing the stability of representations requires to use a technique which is invariant to transformations on the vector space, such as permutation or rotation of coordinates. As an example, consider an arbitrary latent space and apply a permutation to the order of the dimensions that compose it (i.e. permute the order

[4] `https://scikit-optimize.github.io/stable/`

of the factors that compose each embedding). Formally, the new space is radically different from the original one, but the relationships between users and items are perfectly preserved, which is the most important aspect. Indeed, as already discussed in Section 3.2, MF learns to represent, in a f-dimensional latent factor space, users close to items for which they expressed a positive preference, without any additional constraint on the shape of the space learned. For this reason, we define that the representation of an item (user) is stable if, in every new latent space, it is close to the same items (users) in order to assess if the relationships among items (users) are maintained in the different embedding spaces. We check this condition by ensuring that the neighborhoods formed in the new spaces are composed by the same set of items (users).

We execute each algorithm ten times, using the best hyper-parameter configuration found on the validation set, as described in Section 4.2. In each execution we change the latent factor initialization by simply using different seeds for the random number generator used to assign the initial values, but we ensure that the order in which the training data samples are explored during the different runs of the algorithms is kept constant. Considering the latent factor representations learned, we create a list of the closest items to every item and users to every user, by means of a cosine similarity on the latent factor space. We compute these lists of K nearest neighbors for the first of the ten models. Then, we measure the degree of similarity of the K nearest neighbors of the other nine models against the first according to the Jaccard index, a common statistic used to asses the similarity between sets. Higher similarity of the nearest neighbors in the latent space across different runs suggests a higher similarity of the latent factor representations. The results show that:

- there is a strong, positive correlation between the stability of item representations and the accuracy of a MF model;
- the representations and the recommendations of MF models are noisy and strongly impacted by the random initialization of the latent factors, since the available information about the real user's taste is often poor and marginally exploited;
- NNMF models are overall more stable then their counterparts by a large margin, especially when considering non-popular items, where classic MF approaches particularly suffer the scarcity of interactions.

4.3.1 Stability and accuracy correlation

As already outlined in previous works [3], stability and accuracy are usually positively correlated. The goal of this section is to verify if, in the MF techniques studied in this paper, a positive correlation between the stability of item representations and the accuracy exists, and to assess its strength. To perform this experiment:

- We split the dataset into training and test, by using the same strategy described in Section 4.1.
- We compute the stability of representation of each item based on the procedure discussed at the beginning of Section 4.3, using lists of $K = 10$ nearest neighbors.

Table 3: Spearman correlation between the stability of item representations and the accuracy of the algorithm expressed as the MAP@10.

Algorithms	**LFM**	**M1M**	**BCR**	**PIN**	**CUL**
BPR-MF	0.83	0.84	0.97	0.97	0.90
Funk-MF	0.92	0.90	0.95	0.99	0.93
P-MF	0.77	0.83	0.12	0.94	0.20

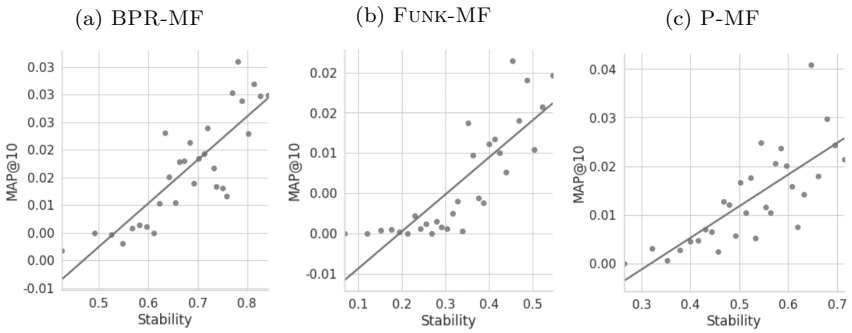

Fig. 1: Comparison between the stability of item representations and accuracy in terms of MAP@10 in MF approaches on the M1M dataset. The plots include the regression lines of the data points.

- We divide the items in 30 groups with increasing stability of representations, so that: the first group includes the items with lowest stability; the second group includes the items with lowest stability that do not appear in the first group; and so on.
- We compute the stability of each group as the average of the stability of the items that compose the group.
- For each group of items, we compute the respective average accuracy as follows:
  - For each user, we keep only the test interactions related to items contained in the group; all other test interactions are discarded; all the training interactions are maintained, regardless of the group.
  - We train the model on the training interactions;
  - We compute MAP and Recall at cutoffs 5 and 10 on the test interactions (i.e., the ground truth contains only interactions within the group of items).
  - We average the metric over all the users.

In Table 3 we show the Spearman's rank correlation coefficient value between stability and MAP@10 of the item groups, to assess if there's a monotonic correlation between them. For brevity, we do not report the results obtained with other metrics and cutoffs, since they exhibit the same results. The table shows a strong, positive correlation between stability and MAP@10. In 13 cases out of 15 the Spearman's coefficient value is above 0.75, and in 9 of these cases it is higher than 0.9, demonstrating the existence of a very strong correlation between the two factors. Funk-MF in particular shows the highest correlation scores in 4 out of 5 datasets, with values always greater or equal to 0.9. The only two exceptions are P-MF on BCR and CUL, where the correlation between MAP@10 and sta-

Table 4: Stability of representations @10 as Jaccard index. Underline indicates the most stable algorithm. Bold indicates the most stable between MF and NNMF.

Algorithms	**LFM**		**M1M**		**BCR**		**PIN**		**CUL**	
	Item	User	Item	User	Item	User	Item	User	Item	User
BPR-MF	0.73	0.67	0.70	0.70	0.47	0.32	0.45	0.30	0.57	0.58
BPR-NNMF	**0.82**	**0.87**	**0.93**	**0.91**	**0.55**	**0.57**	**0.66**	**0.51**	**0.69**	**0.69**
FUNK-MF	0.66	0.61	0.36	0.33	0.17	0.13	0.34	0.19	0.48	0.48
FUNK-NNMF	**0.81**	**0.83**	**0.95**	**0.92**	**0.42**	**0.25**	**0.68**	**0.58**	**0.64**	**0.61**
P-MF	0.62	0.56	0.55	0.47	0.38	0.26	0.32	0.19	0.60	0.40
P-NNMF	**0.76**	**0.78**	**0.81**	**0.68**	**0.57**	**0.62**	**0.63**	**0.46**	**0.75**	**0.71**

Table 5: Stability of representations @100 as Jaccard index. Underline indicates the most stable algorithm. Bold indicates the most stable between MF and NNMF.

Algorithms	**LFM**		**M1M**		**BCR**		**PIN**		**CUL**	
	Item	User	Item	User	Item	User	Item	User	Item	User
BPR-MF	0.72	0.75	0.82	0.80	0.48	0.38	0.67	0.43	0.63	0.61
BPR-NNMF	**0.85**	**0.91**	**0.96**	**0.95**	**0.53**	**0.54**	**0.81**	**0.63**	**0.71**	**0.72**
FUNK-MF	0.73	0.71	0.57	0.53	0.14	0.08	0.60	0.33	0.48	0.43
FUNK-NNMF	**0.87**	**0.88**	**0.97**	**0.95**	**0.26**	**0.17**	**0.84**	**0.70**	**0.63**	**0.50**
P-MF	0.60	0.63	0.73	0.61	0.23	0.28	0.56	0.31	0.62	0.38
P-NNMF	**0.78**	**0.84**	**0.87**	**0.76**	**0.57**	**0.53**	**0.80**	**0.60**	**0.75**	**0.74**

bility is rather week. Note that, according to Table 2, BCR and CUL are the two datasets with the lowest average number of interactions per item. It is well known [5, 23] that accuracy of recommendations is related the the number of available interactions. Therefore, these two datasets are the noisiest in terms of accuracy of recommendations.

In Figure 1 we compare stability of item representations and accuracy, expressed as the MAP@10, of the item groups on the M1M dataset. We show the stability, on the horizontal axis, and the MAP@10, on the vertical axis, of each group of items using a scatter plot. We also include the regression line of the data points. For brevity, we omit other datasets, since we obtained similar results (excluding the two exceptional cases previously highlighted). The plots confirm the results shown in Table 3, since a positive correlation between stability of item representations and MAP@10 is quite evident for all the MF approaches employed.

4.3.2 Stability of representations

In Tables 4 and 5 we report the stability of user and item representations, expressed as the average Jaccard index, computed as described in the introduction of Section 4.3. We use lists of nearest neighbors of length $K = 10$ for Table 4, and of length $K = 100$ for Table 5. We compare MF and NNMF in a pairwise manner, considering the differences in the stability of representations.

The results obtained with the two values of K are similar and show an evident trend: applying the new NNMF technique leads to a large improvement in the stability of the recommendations in every configuration. The stability of NNMF

approaches is over 50% in almost all the experiments, and well above 80% on two out of five datasets. MF approaches, on the contrary, struggle to reach 50% of stability for both users and items on three datasets out of five.

Analyzing the results, we observe higher stability for denser datasets, while it drops when the density of interactions is really low. Indeed lowest values of stability on both users' and items' latent factors, are recorded on the three least dense datasets, namely BookCrossing, CiteULike and Pinterest, while the highest values are obtained on the two most dense datasets, Movielens1M and LastFM. This is an expected behavior and confirms our statements reported in Section 3.2. A lower density of interactions translates into a smaller amount of updates on the representations of users and items. This effect is particularly amplified for items with a very low popularity and users with short profiles. Consequently, the dependency of their representations on the respective initial values is strong, leading to low values of stability across different runs. We also notice how the stability tends to be higher for small datasets. Intuitively, the neighborhoods tend to have less variety when there are less elements in the latent factor space. Finally, among the MF algorithms, it is interesting to highlight that the BPR-MF approach and its NNMF variant have a higher stability than the other methods in almost all the experiments.

Another difference raises between the stability of users and items. MF algorithms tend to have more stable latent representations of items with respect to users' ones, especially on the three most sparse datasets that we previously mentioned. However, this trend is not replicated in NNMF versions, where the stability scores of items and users are very often comparable. It follows that the stability improvements for users representations are usually larger than the items' counterparts.

4.3.3 Stability of representations per item popularity

In this experiment we want to analyze the stability of the item representations at different levels of item popularity. Our goal is to understand if a correlation between the popularity of an item and its stability exists. We divide the items in 6 bins, depending on their popularity, using 1, 0.66, 0.4, 0.3, 0.2, 0.1 and 0 as thresholds. The first bin contains all the most popular items that account for the $1 - 0.66 = 34\%$ of the interactions in the dataset, i.e. the short-head[5]. The second bin contains all the most popular items, excluding those in the first bin, that account for the $0.66 - 0.4 = 26\%$ of interactions. Then, all the other bins are obtained accordingly. For each bin, we average the stability of the items that belong to it.

In Figure 2 we show the results of the experiment performed on the LFM dataset. For brevity, we omit the results on the other datasets, since we obtained very similar results on all of them. Clearly, the stability of NNMF models is globally higher than the MF counterparts at all levels of popularity. However, the plots show another evident trend: the representations of popular items are much more stable than unpopular ones. This behavior is not surprising, and confirms our intuition discussed in Section 3.2: the scarcity of interactions available for unpopular

[5] The long-tail is the set of least popular items that account for the 66% of the interactions. The short-head is defined as complementary to the long-tail: it is the set of most popular items that account for the 34% of the interactions

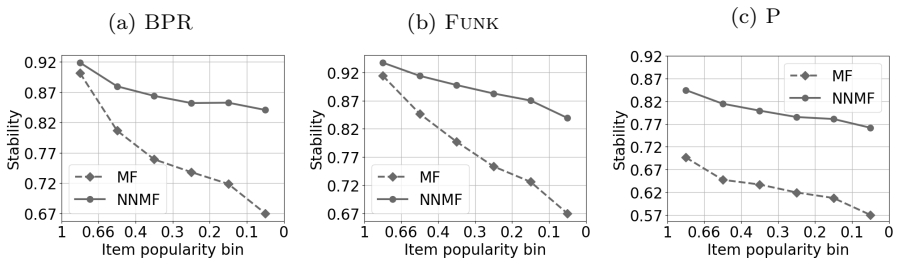

Fig. 2: Stability of item representations for popularity ranges on the LFM dataset expressed as Jaccard @10. Every point represents the average stability value of the items in a bin. A bin contains all the items that belong to a range of popularity.

items impacts the stability of item representations at convergence. Higher popularity determines a higher number of updates and, consequently, a more detailed representation supported by a higher amount of information. Instead, niche items representations are subject of few updates during the learning process, resulting in more fuzzy representations even at model convergence, where the impact of the initialization values used is still strong. It is also evident that MF is subject to higher drops of stability, compared to NNMF, when passing from popular to niche items.

4.3.4 Stability of recommendations

As further experiment, we confront MF and NNMF in order to assess the differences in the stability of recommendations. Moreover, for a more sound comparison, we include the SVD_KNN model proposed in [38], as representative of MF-based approaches that exploit neighborhood information. According to the original definition, the item factors learned through Funk-MF (called Regularized SVD in [38]) are used to compute an item similarity matrix using cosine as similarity measure. For each item, only the k nearest neighbors are used for the prediction of user preferences, which are computed like a common item-based model, through the dot product between the user profile and the item similarity matrix [6]. For completeness, in our experiments we also included two variants of this model, using different MF approaches to learn the item factors, namely BPR-MF, and P-MF. Therefore, we obtained three different models, that we call respectively Funk-MF-KNN (that corresponds to the original implementation of the model proposed in [38]), BPR-MF-KNN, and P-MF-KNN, depending on the algorithm used to generate the item factors. The goal of this experiment is to assess whether NNMF is able to more effectively take advantage of neighborhood information, in order to increase the stability of MF models.

The experimental procedure is similar to the one described in Section 4.3.2: every algorithm is trained ten times, changing the random seed that controls the latent factor initialization, but keeping constant the order in which samples are explored during the training procedure. For each algorithm we perform a pairwise

[6] In our experiments, the number k of nearest neighbors was treated as a hyperparameter of the model, and optimized on the validation set.

Table 6: Stability of recommendations @10 as measured by Jaccard, indicated as J, and RBO indices. Underline indicates the most stable algorithm. Bold indicates which is more stable among MF, MF-KNN and NNMF.

Algorithms	**LFM**		**M1M**		**BCR**		**PIN**		**CUL**	
	J	RBO	J	RBO	J	RBO	J	RBO	J	RBO
BPR-MF	0.72	0.80	0.80	0.85	0.31	0.39	0.53	0.62	0.51	0.62
BPR-MF-KNN	0.79	0.83	0.78	0.83	0.49	0.56	0.48	0.56	0.51	0.59
BPR-NNMF	**0.86**	**0.91**	**0.95**	**0.97**	**0.54**	**0.65**	**0.71**	**0.79**	**0.66**	**0.76**
FUNK-MF	0.75	0.81	0.65	0.72	0.06	0.08	0.33	0.40	0.33	0.42
FUNK-MF-KNN	0.77	0.82	0.37	0.43	**0.20**	**0.25**	0.41	0.48	0.43	0.52
FUNK-NNMF	**0.89**	**0.92**	**0.96**	**0.97**	0.19	**0.25**	**0.78**	**0.84**	**0.53**	**0.64**
P-MF	0.64	0.74	0.65	0.72	0.24	0.34	0.44	0.52	0.39	0.50
P-MF-KNN	0.58	0.65	0.54	0.61	0.22	0.27	0.39	0.45	0.53	0.60
P-NNMF	**0.81**	**0.87**	**0.79**	**0.85**	**0.63**	**0.72**	**0.79**	**0.86**	**0.68**	**0.76**

Table 7: Stability of recommendations @100 as measured by Jaccard, indicated as J, and RBO indices. Underline indicates the most stable algorithm. Bold indicates which is more stable among MF, MF-KNN and NNMF.

Algorithm	**LFM**		**M1M**		**BCR**		**PIN**		**CUL**	
	J	RBO	J	RBO	J	RBO	J	RBO	J	RBO
BPR-MF	0.79	0.86	0.88	0.92	0.45	0.53	0.65	0.73	0.59	0.70
BPR-MF-KNN	0.85	0.90	0.89	0.91	0.48	0.65	0.61	0.71	0.64	0.73
BPR-NNMF	**0.90**	**0.94**	**0.97**	**0.98**	**0.62**	**0.72**	**0.80**	**0.86**	**0.71**	**0.81**
FUNK-MF	0.82	0.88	0.80	0.84	0.10	0.14	0.57	0.61	0.38	0.51
FUNK-MF-KNN	0.86	0.90	0.57	0.64	**0.23**	**0.35**	0.67	0.70	0.50	0.63
FUNK-NNMF	**0.92**	**0.95**	**0.98**	**0.98**	0.22	0.31	**0.87**	**0.90**	**0.56**	**0.70**
P-MF	0.70	0.80	0.76	0.83	0.29	0.41	0.60	0.67	0.46	0.59
P-MF-KNN	0.65	0.76	0.67	0.75	0.27	0.38	0.51	0.62	0.62	0.73
P-NNMF	**0.85**	**0.91**	**0.87**	**0.91**	**0.67**	**0.77**	**0.84**	**0.90**	**0.74**	**0.83**

comparison between the first model (first random seed) with all the other nine. We compare the top-N recommendations of the two versions of the same model, measuring the degree of similarity of the recommendations retrieved. The similarity is expressed resorting to the Jaccard index and to the Rank Biased Overlap (RBO) [48]. Differently from the experimental setting in Section 4.3.2 we report the RBO index to account for the ranking of the elements of the two lists, since the order in which recommendations are retrieved can be crucial in some settings. The results are averaged over the nine different comparisons performed. The experiment has been repeated using two different recommendation cutoff values 10 and 100, and the stability scores are reported in Table 6 and Table 7 respectively.

The results in the stability of recommendations are compatible with respect to what we observed for the representations in Tables 4 and 5. The recommendation lists generated by NNMF models are always widely more stable than their MF counterparts from both the two different points of view of the Jaccard and RBO indexes on the two cutoffs analyzed. Exploiting the neighborhood as proposed in MF-KNN variants is not clearly effective for the improvement of recommendation stability. Indeed, they obtain inconsistent results across the datasets, and they

improve the stability of the respective MF model used to learn the item factors in only 7 cases over 15. In the other scenarios, they obtain stability scores that are similar or worse than the respective MF model. Finally, the MF-KNN variants obtain comparable results with NNMF in only one single case (Funk-MF-KNN on BCR), while NNMF proves to be largely more effective in improving stability exploiting neighborhood information in all the other scenarios. The MF implementations fail to reach the 50% threshold for the Jaccard index in many configurations of the recommendation cutoff @10, with a negative spike of 6% for Funk-MF on BookCrossing. When the cutoff is shifted to 100, the values of the indexes increase for the standard MF algorithms, but they still do not reach the 50% threshold in multiple cases. The same holds also for the MF-KNN models. NNMF techniques largely mitigate this issue, since they are able to outperform the respective MF implementations and the MF-KNN variants with a wide margin.

As for the representations, also in this case the difference between dense datasets and sparse ones is evident, with a higher stability in the first scenario. For example, on Movielens, that is the most dense dataset, the algorithms have stability scores which are from two to ten times higher than those obtained on BookCrossing, which is the least dense dataset. Moreover, notice how the stability in both experiments tends to be higher for small datasets. In fact, having fewer items to take into account also reduces the different available options in the recommendation phase. As final observation, among the MF algorithms note that the BPR-MF approach is steadily better than the others on all the datasets, while among the NNMF this trend is not evident anymore.

4.3.5 Stability of relevant recommendations

The stability of an algorithm is particularly important for the relevant items. Indeed, if an algorithm is stable in selecting the top-N relevant items, we can argue that the algorithm is confident about them, while, on the contrary, the stability with which an algorithm ranks the less relevant items has lower value.

The experiment in Section 4.3.4 assesses the stability of the whole recommendation lists generated by the different models, but it does not consider the relevancy of the recommended items. For this reason, we perform the same experiment reported in Section 4.3.4, taking into account only the relevant items. Also in this case, every algorithm is executed ten times, changing the initial values of the latent factors. We compare the relevant items in the top-10 recommendations for a user generated by the first model against the relevant items in each of the top-10 recommendation lists provided by the other nine in a pairwise manner. If both lists in a comparison do not contain any relevant item, the comparison is not considered. In Table 8 we report the average Jaccard index.

The results show stability scores in line with those reported in Table 6, with a steady improvement in stability of NNMF over MF of about 10% in every configuration. MF-KNN variants confirm their inconsistency in the improvement of stability, and the results corroborate the insights discussed in Section 4.3.4. Clearly, NNMF variants are largely more stable, and they outperform their MF counterparts with improvements that range from 10% to 40%. Moreover, they are able to exceed the 75% Jaccard score threshold in 12 cases over 15. Basic MF approaches, instead, can reach the same threshold in only 4 scenarios, while they struggle in

Table 8: Stability of relevant recommendations @10 as measured by Jaccard index. Underline indicates the most stable algorithm. Bold indicates which is more stable among MF, MF-KNN and NNMF.

Algorithms	**LFM**	**M1M**	**BCR**	**PIN**	**CUL**
BPR-MF	0.83	0.83	0.44	0.64	0.66
BPR-MF-KNN	0.86	0.84	0.64	0.57	0.64
BPR-NNMF	**0.92**	**0.96**	**0.69**	**0.77**	**0.77**
FUNK-MF	0.84	0.71	0.19	0.42	0.52
FUNK-MF-KNN	0.85	0.43	**0.44**	0.49	0.59
FUNK-NNMF	**0.93**	**0.97**	**0.44**	**0.82**	**0.65**
P-MF	0.75	0.71	0.40	0.55	0.58
P-MF-KNN	0.68	0.59	0.50	0.47	0.68
P-NNMF	**0.88**	**0.82**	**0.77**	**0.83**	**0.77**

all the others. In particular, on the most sparse datasets, namely BookCrossing, Pinterest and CiteULike, they obtain stability values between 50% and 60%.

Note that relevant items directly impact the accuracy performance of the algorithms. Low stability scores obtained in this experiment imply that relevant items largely vary, just depending on the initial values of the latent factors. It follows that the performance of the algorithms is originated by different correct predictions, an effect that raises a non-negligible reliability issue.

4.4 Accuracy

We want to study the accuracy of MF and NNMF models, paying special attention to the impact of item popularity. We compare the performance of the proposed models in the top-N recommendation task at different levels of popularity. The results show that while MF and NNMF obtain similar results on top-popular items, NNMF variants outperform their counterparts on the long-tail.

4.4.1 Top-n Recommendation

In this section we focus on the accuracy performance on the long-tail i.e. on non-popular items. We are particularly interested in comparing MF and NNMF on the long-tail because it is where they differ the most on stability, as pointed out in Section 4.3.2. Moreover, recommending non-popular items adds novelty and serendipity to the users, and it is usually a more difficult task compared to the recommendation of popular items [12]. The experimental design follows [12]. The test set T has been further partitioned in T_{head} and T_{tail} such that the items in T_{head} are in the short-head and the items in T_{tail} are in the long-tail, following the definitions of short-head and long-tail provided in Section 4.3.3 (i.e., 34% of the items for the short-head and 66% for the long-tail). We compute the top-n performance using as ground truth T_{tail}. We evaluate the behavior of NNMF and MF approaches, carrying out pairwise comparisons. To provide context to our results, we additionally score other competitive collaborative baselines [13]:

Table 9: Long-tail accuracy @5. Bold indicates the best between MF and NNMF. Underline indicates the best performing algorithm.

Algorithm	LFM		M1M		BCR		PIN		CUL	
	MAP	Recall	MAP	Recall	MAP	Recall	MAP	Recall	MAP	Recall
ItemKNN	0.028	0.040	0.005	0.006	0.014	0.018	0.013	0.023	0.047	0.065
UserKNN	0.020	0.032	0.005	0.007	0.007	0.010	0.010	0.019	0.047	0.071
SLIM	0.024	0.038	0.002	0.003	0.004	0.006	0.010	0.017	0.045	0.071
PureSVD	0.027	0.045	0.022	0.020	0.002	0.003	0.005	0.010	0.018	0.022
BPR-MF	0.035	0.050	**0.022**	**0.025**	0.006	0.009	0.010	0.018	0.037	**0.060**
BPR-NNMF	**0.039**	**0.058**	0.017	0.018	**0.009**	**0.011**	**0.012**	**0.023**	**0.039**	**0.060**
Funk-MF	**0.033**	0.050	0.017	0.018	0.004	0.006	0.010	0.019	0.044	0.069
Funk-NNMF	**0.033**	**0.052**	**0.034**	**0.024**	**0.006**	**0.010**	**0.011**	**0.021**	**0.053**	**0.080**
P-MF	0.019	0.030	0.008	0.010	0.001	0.002	0.009	0.016	0.033	0.052
P-NNMF	**0.034**	**0.054**	**0.027**	**0.025**	**0.007**	**0.009**	**0.011**	**0.020**	**0.047**	**0.064**

ItemKNN Nearest neighbors item-based approach for top-n recommendation. Given a user, this technique recommends the items most similar to those in the user profile. Proximity is computed with shrinked cosine similarity [14].

UserKNN Nearest neighbors user-based approach. For each a user, this algorithm recommends the items that are more frequently present in the profiles of users similar to the considered one. Also in this case, vicinity is computed with shrinked cosine similarity [14].

SLIM Sparse LInear Method is a linear regression model for top-n recommendation. It learns a sparse aggregation coefficient matrix for items exploiting user interaction profiles only. The model is trained using Bayesian Personalized Ranking (BPR) as optimization function [36].

PureSVD It is a basic Matrix Factorization model that performs conventional Singular Value Decomposition on the user rating matrix [12]. Differently from the other MF methods considered in this paper, it has an exact mathematical solution that can not include the similarity matrices introduced by NNMF.

For all the algorithms employed in this comparison, including the collaborative baselines and all the MF and NNMF implementations, we performed a hyperparameter optimization procedure as described in Section 4.2.

In Tables 9 and 10 we report the performance obtained by the different algorithms, expressed by the Mean Average Precision and the Recall at two cutoffs, 5 and 10. In almost 90% of the measures, the NNMF algorithms perform better than or equal to the corresponding MF versions, and in more than 80% of the measures, the improvement provided by NNMF is consistent. NNMF models achieve highest accuracy on three datasets over five and in the remaining cases they try to fill the gap between the best performing model (usually ItemKNN) and classic MF algorithms, proving to be at least competitive against the other models across the datasets. Indeed, notice that even when a NNMF model is not the most accurate model on the long-tail, it is the second best performing algorithm.

The poor long-tail accuracy of MF obtained in many scenarios is quite surprising, especially if we consider that MF approaches are known for being less popularity biased than other CF approaches [2, 11, 26]. We can conclude that these models are able to recommend niche items, but they are often noisy rec-

Table 10: Long-tail accuracy @10. Bold indicates the best between MF and NNMF. Underline indicates the best performing algorithm.

Algorithm	LFM		M1M		BCR		PIN		CUL	
	MAP	Recall	MAP	Recall	MAP	Recall	MAP	Recall	MAP	Recall
ItemKNN	0.030	0.073	0.005	0.013	0.014	0.027	0.016	0.044	0.051	0.110
UserKNN	0.022	0.065	0.006	0.022	0.007	0.016	0.013	0.037	0.052	0.116
SLIM	0.026	0.068	0.003	0.010	0.005	0.010	0.012	0.035	0.050	0.114
PureSVD	0.028	0.084	0.021	0.047	0.002	0.004	0.007	0.022	0.019	0.046
BPR-MF	0.037	0.087	**0.022**	**0.051**	0.007	0.014	0.012	0.035	0.042	**0.099**
BPR-NNMF	**0.040**	**0.099**	0.016	0.039	**0.008**	**0.016**	**0.015**	**0.046**	**0.043**	**0.099**
Funk-MF	0.035	0.085	0.015	0.034	0.004	0.009	0.013	0.037	0.049	0.108
Funk-NNMF	**0.036**	**0.094**	**0.028**	**0.046**	**0.007**	**0.016**	**0.014**	**0.041**	**0.058**	**0.127**
P-MF	0.020	0.058	0.009	0.027	0.001	0.003	0.011	0.033	0.038	0.090
P-NNMF	**0.036**	**0.095**	**0.024**	**0.048**	**0.007**	**0.014**	**0.013**	**0.038**	**0.049**	**0.107**

ommendations, resulting from the low number of updates on the latent factors of non-popular items, rather than a real evidence of the user's taste. NNMF models, instead, leverage the knowledge of the neighborhood to construct more consistent item representations on the long-tail, transforming its part of non-popular recommendations in higher quality long-tail accuracy.

4.4.2 Accuracy per item popularity

To further evaluate how the model performance is affected by the item popularity, we analyze the accuracy of the models over different item popularity bins. We partition items in 5 bins, based on popularity, such that each bin contains 20% of all the interactions. The first bin contains all the most popular items that account for 20% of the interactions in the dataset. The second bin contains all the most popular items, excluding the ones in the first bin, accounting for another 20% of the interactions, and so on. For each bin, we compute the accuracy performance, expressed as the MAP@10, of all the MF and NNMF techniques. We set the hyper-parameters as described in Section 4.2.

In Figure 3 we show the results on all the studied datasets. On the x-axis we report the item bins ordered by decreasing popularity, so that the most popular items are contained in the left-most bin, while the least popular in the right-most bin. On the y-axis we report the MAP@10 (on the right, represented with lines) and the number of items contained in each bin (on the left, represented with bars). We indicate the MF versions with dotted lines, while with contiguous lines we indicate the NNMF algorithms.

Clearly, NNMF algorithms perform better on low popularity items compared to their MF counterparts. In the last bin, accounting for the least popular items, NNMF algorithms perform better with respect to the corresponding MF variants 13 times over 15. The opposite happens 2 times on Movielens1M, which is the most dense dataset. In this case, the advantage of propagating knowledge in the neighborhoods brought by NNMF is not evident in terms of accuracy, as the singular items already have enough information to leverage during learning. Indeed, considering the most popular bins, where more feedbacks are available for the

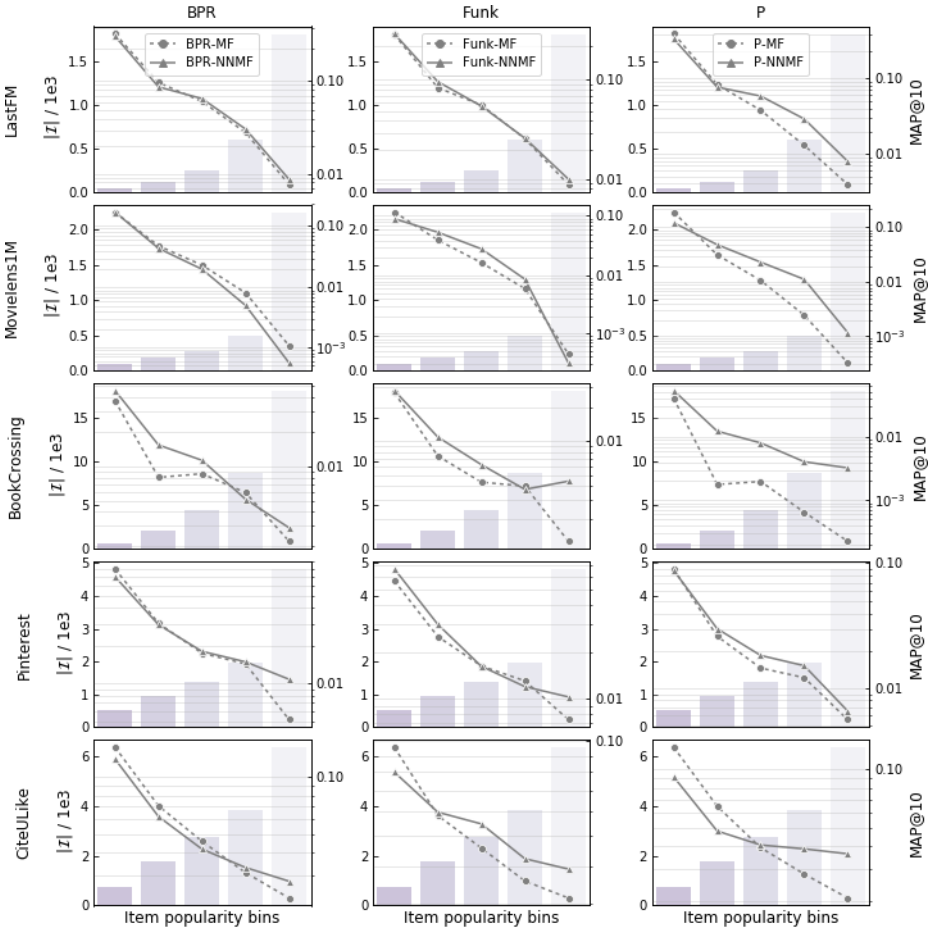

Fig. 3: Performance comparison when clustering items in different popularity bins. On the x-axis, item bins are ordered by decreasing popularity from left to right. Bar plots show the number of items in each bin, according to the scale on the left. Line plots show the MAP@10, according to the scale on the right.

items, it is not possible to declare a unique winner between MF and NNMF, since the results are dataset dependent. On Movielens and CiteULike, MF approaches obtain a slightly higher MAP score in most scenarios. On LastFM and Pinterest, both the variants perform similarly. On Bookcrossing, NNMF outperforms MF also in popular bins.

Overall, MF algorithms suffer a stronger performance drop in correspondence of less popular item bins in almost any configuration. This global trend suggests that MF algorithms benefit from the abundancy of information, but their performance is strongly impacted by the density of interactions. Indeed, considering bins composed by items with gradually lower popularity, the balance in accuracy between NNMF and MF algorithms moves in favor of the firsts, until reaching the

Table 11: Stability of performance as measured by the coefficient of variation on accuracy @5. Bold indicates which is more stable between MF and NNMF. Underline indicates the most stable algorithm.

Algorithm	LFM		M1M		BCR		PIN		CUL	
	MAP	Recall	MAP	Recall	MAP	Recall	MAP	Recall	MAP	Recall
BPR-MF	0.012	0.013	0.013	0.016	0.029	0.036	0.008	0.009	0.017	**0.012**
BPR-NNMF	**0.009**	**0.007**	**0.005**	**0.012**	**0.017**	**0.029**	**0.007**	**0.006**	**0.009**	0.013
FUNK-MF	0.014	0.018	0.088	0.106	0.085	0.058	0.008	0.012	0.012	0.017
FUNK-NNMF	**0.009**	**0.010**	**0.004**	**0.010**	**0.035**	**0.056**	**0.004**	**0.008**	**0.009**	**0.012**
P-MF	0.034	0.026	**0.023**	0.033	0.026	0.056	**0.013**	0.013	0.017	0.021
P-NNMF	**0.012**	**0.014**	0.025	**0.014**	**0.019**	**0.024**	0.014	**0.010**	**0.014**	**0.015**

least popular items where NNMF techniques are able to outperform standard MF algorithms.

4.4.3 Stability of accuracy

In Section 4.3 we show that NNMF algorithms have higher stability than MF algorithms with a number of experiments. One experiment is concerned with stability of recommendations. We expect the stability of recommendations to be associated with the stability of accuracy, as the accuracy of an algorithm is a function of the recommendations of an algorithm. To measure the stability of accuracy, we consider an experimental procedure similar to the one described in Section 4.3.2: every algorithm is trained ten times, changing the random seed that controls the latent factor initialization, but keeping constant the order in which samples are explored during the training procedure. For each random seed, we consider the accuracy of the algorithms, and we compute the coefficient of variation. The coefficient of variation measures the variability of accuracy while accounting for different average values of accuracy. This analysis excludes the baselines as some of them are not based on latent factors. We compute the coefficient of variation on MAP and Recall @5, and we report the results in Table 11. We observe similar results for MAP and Recall @10, which we omit for brevity.

As expected, NNMF algorithms exhibit in general higher stability of performance (i.e., lower coefficient of variantion). We observe up to one order of magnitude decrease on the coefficient of variation. We do not observe a decrease in coefficient of variation in 3 cases over 15. However, in those cases the stability of performance of MF and NNMF is comparable.

4.5 Model training

As last experiment, we investigate the behavior of the models during the training procedure. In particular, we evaluate the capability of the models to fit the training data and the performance on the validation set, as the epochs evolve. We set the hyper-parameters as described in Section 4.2. For brevity, we show in Figures 4 and 5 the comparison between MF and NNMF on CUL, but we could observe the same trend on almost all the other datasets and algorithms. Figure 4 depicts

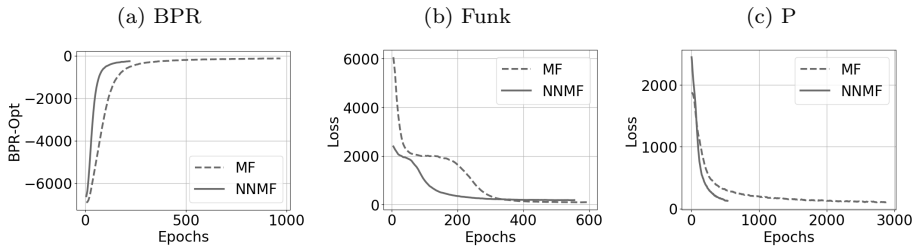

Fig. 4: Comparison between the training procedures of MF and NNMF on the CUL dataset. We report the loss, for the Funk and P variants, and the likelihood estimator value (BPR-Opt), for the BPR, computed on the training set.

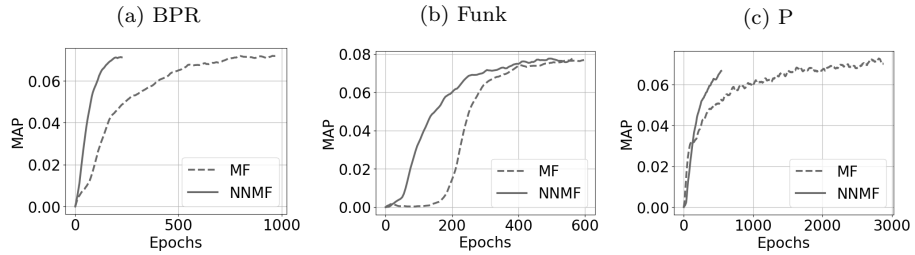

Fig. 5: Comparison between the training procedures of MF and NNMF on the CUL dataset. We show the MAP@5 obtained on the validation set.

the capability of the model to fit the training data, as measured by the objective function. Figure 5 shows performance on the validation set as measured by MAP@5. Even if the starting and the convergence values of MF and NNMF are quite similar in every comparison, NNMF variants reach convergence in a smaller amount of epochs. As an example, BPR-NNMF converges after about 200 epochs and the training is interrupted by the early stopping technique, while BPR-MF version needs about 1000 epochs. To summarize the results obtained on all the datasets, in Table 12 we report the number of epochs required to the algorithms to converge in all the datasets, according to the best hyper-parameter configurations found as described in Section 4.2. We observe that NNMF takes a fraction of iterations to converge compared to MF in 13 cases out of 15. The only two exceptions are Funk-MF on M1M and P-MF on BCR, where the NNMF variants require a considerably higher number of epochs to converge. The reason of this unexpected behavior is related to the wide accuracy gap between the MF and the NNMF variants. As shown in Table 9, Funk-NNMF on M1M reaches twice the MAP@5 with respect to Funk-MF, and the MAP@5 of P-NNMF on BCR is 7 times higher than the same metric score achieved by P-MF. The accuracy of NNMF variants continuously increases during the training procedure. Therefore, the early stopping procedure does not interrupt the training for a high number of epochs.

The results reported in this section prove that the propagation of the information we have about a user or an item also to its neighbors is effective and

Table 12: Number of epochs required for convergence.

Algorithms	**LFM**	**M1M**	**BCR**	**PIN**	**CUL**
BPR-MF	900	4040	960	945	975
BPR-NNMF	540	525	600	555	225
Funk-MF	480	500	980	960	600
Funk-NNMF	480	1820	600	570	560
P-MF	640	2280	1670	885	2880
P-NNMF	420	500	4170	555	540

useful, allowing the model to reach the optimal performance in a lower number of iterations over the training data in almost every scenario.

5 Conclusions and Future Work

In this work we assess the stability of widely-used Matrix Factorization recommenders, according to a novel notion of stability. We elaborate on how instability can undermine real world recommender systems, such as the reliability and the explainability of their recommendations. Moreover, we show that the accuracy of common MF techniques is positively correlated with the stability of item representations.

The results of extensive experiments on five different datasets show that three common MF algorithms are particularly affected by instability. By simply changing the initial values assigned to the latent factors, the same model, trained on the same data and with the exactly same configuration, provides very different recommendations and latent representations at convergence, two issues that we call, respectively, instability of recommendations and instability of representations.

We present Nearest Neighbors Matrix Factorization, a generalization of classic MF. The new framework merges Nearest Neighbors and Matrix Factorization techniques in order to exploit similarity information during the training procedure. To promote the validity of the new technique, we propose the NNMF extensions of three common MF algorithms.

We assess the stability of the algorithms with a variety of experiments. We highlight that both relevant and non-relevant recommendations suffer from instability, and we show that there exists a correlation between item popularity and stability. The results highlight that NNMF provides large improvements over MF in terms of stability of recommendations and representations. We also measure the accuracy of the algorithms as a function of item popularity. NNMF scores best results, especially on the long-tail. Finally, we show that the new models are able to reach convergence in a fraction of the epochs necessary to MF, thanks to the propagation of the information through the neighborhood relations.

The NNFM framework described in this paper has shown promising results in improving both stability and accuracy of classic MF algorithms. However, there are several research directions that can be subject of future work. This paper considers only three instances of matrix factorization, but there is no evidence that other models based on user and item latent factors could benefit from the NNMF

approach. More studies with other latent factor models should be performed to provide a stronger validation of the results presented in Section 4. In our experiments we adopted a single type of similarity function (i.e., shrinked cosine similarity) to model the relations between users and between items for the NNMF framework. It would be interesting to explore different similarity functions obtained with more powerful techniques, and to evaluate their impact on both stability and accuracy.

Acknowledgement

Edoardo D'Amico and Giovanni Gabbolini would like to acknowledge a grant from Science Foundation Ireland (SFI) under Grant Number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund, that partially supported this research.

Declarations

Funding

Edoardo D'Amico and Giovanni Gabbolini would like to acknowledge a grant from Science Foundation Ireland (SFI) under Grant Number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund, that partially supported this research.

Conflict of interest

The authors declare that they have no conflict of interest.

Availability of data and material

The datasets used in the experiments are publicly available in the online repository.

Code availability

The source code used to perform all the experiments, including data splitting, hyper-parameter optimization, stability and accuracy performance evaluation, is publicly available. The link is the following: https://github.com/damicoedoardo/NNMF

References

1. Abdollahi B, Nasraoui O (2017) Using explainability for constrained matrix factorization. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp 79–83

2. Abdollahpouri H, Mansoury M, Burke R, Mobasher B (2019) The unfairness of popularity bias in recommendation. In: Burke R, Abdollahpouri H, Malthouse EC, Thai KP, Zhang Y (eds) Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments co-located with the 13th ACM Conference on Recommender Systems (RecSys 2019), Copenhagen, Denmark, September 20, 2019, CEUR-WS.org, CEUR Workshop Proceedings, vol 2440, URL http://ceur-ws.org/Vol-2440/paper4.pdf
3. Adomavicius G, Zhang J (2012) Stability of recommendation algorithms. ACM Trans Inf Syst 30(4):23:1–23:31, DOI 10.1145/2382438.2382442, URL https://doi.org/10.1145/2382438.2382442
4. Adomavicius G, Zhang J (2014) Improving stability of recommender systems: a meta-algorithmic approach. IEEE Transactions on Knowledge and Data Engineering 27(6):1573–1587
5. Anand D, Bharadwaj KK (2011) Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. Expert Systems with Applications 38(5):5101–5109, DOI https://doi.org/10.1016/j.eswa.2010.09.141, URL https://www.sciencedirect.com/science/article/pii/S0957417410010985
6. Bar A, Rokach L, Shani G, Shapira B, Schclar A (2013) Improving simple collaborative filtering models using ensemble methods. In: Multiple Classifier Systems, Springer Berlin Heidelberg, pp 1–12
7. Bell R, Koren Y, Volinsky C (2007) Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '07, pp 95–104, DOI 10.1145/1281192.1281206, URL http://doi.acm.org/10.1145/1281192.1281206
8. Bernardis C, Ferrari Dacrema M, Cremonesi P (2019) Estimating confidence of individual user predictions in item-based recommender systems. In: Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization, Association for Computing Machinery, New York, NY, USA, UMAP '19, p 149–156, DOI 10.1145/3320435.3320453, URL https://doi.org/10.1145/3320435.3320453
9. Bousquet O, Elisseeff A (2002) Stability and generalization. J Mach Learn Res 2:499–526, URL http://jmlr.org/papers/v2/bousquet02a.html
10. Cantador I, Brusilovsky P, Kuflik T (2011) Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In: Proceedings of the fifth ACM conference on Recommender systems, ACM, New York, NY, USA, pp 387–388
11. Channamsetty S, Ekstrand MD (2017) Recommender response to diversity and popularity bias in user profiles. In: Rus V, Markov Z (eds) Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017, AAAI Press, pp 657–660, URL https://aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15524
12. Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, Association for Computing Machinery, New York, NY, USA, RecSys '10, p 39–46, DOI 10.1145/1864708.1864721, URL https://doi.org/10.1145/1864708.1864721

13. Dacrema MF, Cremonesi P, Jannach D (2019) Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Bogers T, Said A, Brusilovsky P, Tikk D (eds) Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019, ACM, pp 101–109, DOI 10.1145/3298689.3347058, URL https://doi.org/10.1145/3298689.3347058
14. Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender Systems Handbook, Springer, pp 107–144, DOI 10.1007/978-0-387-85820-3_4, URL https://doi.org/10.1007/978-0-387-85820-3_4
15. Funk S (2006) Netflix update: Try this at home. http://sifterorg/ simon/journal/20061211html
16. Gabbolini G, D'Amico E, Bernardis C, Cremonesi P (2021) On the instability of embeddings for recommender systems: the case of matrix factorization. In: Proceedings of the 36th ACM/SIGAPP Symposium on Applied Computing, DOI 10.1145/3412841.3442011
17. Gemulla R, Nijkamp E, Haas PJ, Sismanis Y (2011) Large-scale matrix factorization with distributed stochastic gradient descent. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 69–77
18. Geng X, Zhang H, Bian J, Chua TS (2015) Learning image and user features for recommendation in social networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp 4274–4282
19. Gong L, Nandi AK (2013) An enhanced initialization method for non-negative matrix factorization. In: IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2013, Southampton, United Kingdom, September 22-25, 2013, IEEE, pp 1–6, DOI 10.1109/MLSP.2013.6661949, URL https://doi.org/10.1109/MLSP.2013.6661949
20. Guidotti R, Ruggieri S (2019) On the stability of interpretable models. In: International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019, IEEE, pp 1–8, DOI 10.1109/IJCNN.2019.8852158, URL https://doi.org/10.1109/IJCNN.2019.8852158
21. Harper FM, Konstan JA (2016) The movielens datasets: History and context. ACM Trans Interact Intell Syst 5(4):19:1–19:19, DOI 10.1145/2827872, URL https://doi.org/10.1145/2827872
22. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182
23. Huang CB, Gong SJ (2008) Employing rough set theory to alleviate the sparsity issue in recommender system. In: 2008 International conference on machine learning and cybernetics, IEEE, vol 3, pp 1610–1614
24. Huang G, Li Y, Pleiss G, Liu Z, Hopcroft JE, Weinberger KQ (2017) Snapshot ensembles: Train 1, get M for free. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, URL https://openreview.net/forum?id=BJYwwY9ll
25. Izmailov P, Podoprikhin D, Garipov T, Vetrov DP, Wilson AG (2018) Averaging weights leads to wider optima and better generalization. In: Glober-

son A, Silva R (eds) Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018, AUAI Press, pp 876–885, URL http://auai.org/uai2018/proceedings/papers/313.pdf
26. Jannach D, Lerche L, Gedikli F, Bonnin G (2013) What recommenders recommend – an analysis of accuracy, popularity, and sales diversity effects. In: Carberry S, Weibelzahl S, Micarelli A, Semeraro G (eds) User Modeling, Adaptation, and Personalization, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 25–37
27. Koren Y (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, ACM, New York, NY, USA, KDD '08, pp 426–434, DOI 10.1145/1401890.1401944, URL http://doi.acm.org/10.1145/1401890.1401944
28. Koren Y (2010) Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Trans Knowl Discov Data 4(1):1:1–1:24, DOI 10.1145/1644873.1644874, URL https://doi.org/10.1145/1644873.1644874
29. Koren Y, Bell RM, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37, DOI 10.1109/MC.2009.263, URL https://doi.org/10.1109/MC.2009.263
30. Li D, Chen C, Lv Q, Yan J, Shang L, Chu S (2016) Low-rank matrix approximation with stability. In: Balcan MF, Weinberger KQ (eds) Proceedings of The 33rd International Conference on Machine Learning, PMLR, New York, New York, USA, Proceedings of Machine Learning Research, vol 48, pp 295–303, URL http://proceedings.mlr.press/v48/lib16.html
31. Li D, Miao C, Chu S, Mallen J, Yoshioka T, Srivastava P (2018) Stable matrix approximation for top-n recommendation on implicit feedback data. In: Proceedings of the 51st Hawaii International Conference on System Sciences
32. Liang D, Altosaar J, Charlin L, Blei DM (2016) Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In: Proceedings of the 10th ACM conference on recommender systems, pp 59–66
33. Luo X, Zhou M, Xia Y, Zhu Q (2014) An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Transactions on Industrial Informatics 10(2):1273–1284
34. Madhyastha P, Jain R (2019) On model stability as a function of random seed. In: Bansal M, Villavicencio A (eds) Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019, Association for Computational Linguistics, pp 929–939, DOI 10.18653/v1/K19-1087, URL https://doi.org/10.18653/v1/K19-1087
35. Mobasher B, Burke RD, Bhaumik R, Sandvig JJ (2007) Attacks and remedies in collaborative recommendation. IEEE Intell Syst 22(3):56–63, DOI 10.1109/MIS.2007.45, URL https://doi.org/10.1109/MIS.2007.45
36. Ning X, Karypis G (2011) Slim: Sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining, pp 497–506
37. Olaleke O, Oseledets IV, Frolov E (2021) Dynamic modeling of user preferences for stable recommendations. In: Masthoff J, Herder E, Tintarev N, Tkalcic M (eds) Proceedings of the 29th ACM Conference on User Model-

ing, Adaptation and Personalization, UMAP 2021, Utrecht, The Netherlands, June, 21-25, 2021, ACM, pp 262–266, DOI 10.1145/3450613.3456830, URL https://doi.org/10.1145/3450613.3456830
38. Paterek A (2007) Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, vol 2007, pp 5–8
39. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2012) BPR: bayesian personalized ranking from implicit feedback. CoRR abs/1205.2618, URL http://arxiv.org/abs/1205.2618, 1205.2618
40. Said A, Bellogín A (2018) Coherence and inconsistencies in rating behavior: estimating the magic barrier of recommender systems. User Modeling and User-Adapted Interaction 28(2):97–125
41. Said A, Jain BJ, Narr S, Plumbaum T (2012) Users and noise: The magic barrier of recommender systems. In: International Conference on User Modeling, Adaptation, and Personalization, Springer, pp 237–248
42. Salakhutdinov R, Mnih A (2007) Probabilistic matrix factorization. In: Proceedings of the 20th International Conference on Neural Information Processing Systems, Curran Associates Inc., USA, NIPS '07, pp 1257–1264, URL http://dl.acm.org/citation.cfm?id=2981562.2981720
43. Shriver D, Elbaum SG, Dwyer MB, Rosenblum DS (2019) Evaluating recommender system stability with influence-guided fuzzing. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, pp 4934–4942, DOI 10.1609/aaai.v33i01.33014934, URL https://doi.org/10.1609/aaai.v33i01.33014934
44. Skurichina M, Duin RPW (1998) Bagging for linear classifiers. Pattern Recognit 31(7):909–930, DOI 10.1016/S0031-3203(97)00110-6, URL https://doi.org/10.1016/S0031-3203(97)00110-6
45. Tintarev N, Masthoff J (2007) A survey of explanations in recommender systems. In: 2007 IEEE 23rd international conference on data engineering workshop, IEEE, pp 801–810
46. Töscher A, Jahrer M, Legenstein R (2008) Improved neighborhood-based algorithms for large-scale recommender systems. In: Proceedings of the 2Nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, ACM, New York, NY, USA, NETFLIX '08, pp 4:1–4:6, DOI 10.1145/1722149.1722153, URL http://doi.acm.org/10.1145/1722149.1722153
47. Wang H, Chen B, Li W (2013) Collaborative topic regression with social regularization for tag recommendation. In: Rossi F (ed) IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI, pp 2719–2725, URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/7006
48. Webber W, Moffat A, Zobel J (2010) A similarity measure for indefinite rankings. ACM Trans Inf Syst 28(4), DOI 10.1145/1852102.1852106, URL https://doi.org/10.1145/1852102.1852106
49. Xue HJ, Dai X, Zhang J, Huang S, Chen J (2017) Deep matrix factorization models for recommender systems. In: IJCAI, Melbourne, Australia, vol 17, pp 3203–3209

50. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR) 52(1):1–38
51. Zheng Y (2016) Adapt to emotional reactions in context-aware personalization. In: EMPIRE@ RecSys, pp 1–8
52. Zheng Z, Yang J, Zhu Y (2007) Initialization enhancer for non-negative matrix factorization. Eng Appl Artif Intell 20(1):101–110, DOI 10.1016/j.engappai.2006.03.001, URL https://doi.org/10.1016/j.engappai.2006.03.001
53. Ziegler CN, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In: Proceedings of the 14th International Conference on World Wide Web, Association for Computing Machinery, New York, NY, USA, WWW '05, p 22–32, DOI 10.1145/1060745.1060754, URL https://doi.org/10.1145/1060745.1060754