Accuracy-Diversity Trade-off in Recommender Systems via Graph Convolutions

Accuracy-Diversity Trade-off in Recommender Systems via Graph Convolutions

Thesis

to obtain the degree of Master in Computer Science with Specialization in Data Science
and Technology at Delft University of Technology,
to be publicly defended on Monday, August 31st 2020 at 12:00

by

Matteo Pocchiari
Born in Potenza, Italy.

Multimedia Computing Group,
Faculty of Electrical Engineering, Mathematics and Computer Science (Faculteit
Elektrotechniek, Wiskunde en Informatica),
Delft University of Technology,
Delft, The Netherlands.

Keywords: Accuracy-diversity, collaborative filtering, graph filters, graph convolutional neural networks, graph signal processing.

Contents

Summary

Recommender Systems assist the user by suggesting items to be consumed based on the user's history. The topic of diversity in recommendation gained momentum in recent years as additional criterion besides recommendation accuracy, to improve user satisfaction. Accuracy and diversity in recommender systems coexist in a delicate trade-off due to the complexity in capturing user tastes through a limited amount of interactions. Graphs have been employed for recommendation, given their ability to efficiently represent user-item interactions. Graph convolutions, as learning over graphs tools, have reached state-of-the-art accuracy on recommender system benchmarks. However, the potential of graph convolutions to improve the accuracy-diversity trade-off is unexplored. Here, we develop a model that learns from a nearest neighbor and a furthest neighbor graph via a joint convolutional model to establish a novel accuracy-diversity trade-off for recommender systems. In detail, the nearest neighbor graph connects entities (users or items) based on their similarities and is responsible for improving accuracy, while the furthest neighbor graph connects entities based on their dissimilarities and is responsible for diversifying recommendations. The information between the two convolutional modules is balanced already in the training phase through a regularizer inspired by multi-kernel learning. Numerical experiments on three benchmark datasets showed the joint convolutional model can improve substantially the catalog coverage or the diversity among recommended items; or boost both by a lesser amount. We compared our model against state-of-the-art accuracy-oriented algorithms, showing diversity gains up to seven times by trading as little as 1% in accuracy. We also compared the joint model against algorithms proposing a different accuracy-diversity trade-off, evidencing our model achieves better accuracy while preserving a wide diversity range. Our findings highlight that the joint convolutional model offers a balance in each setting that is difficult to be achieved with a single model.

Acknowledgements

I would like to thank the persons who supported me along this amazing journey and were unconditionally by my side.

First of all, I would like to thank Elvin for his time and energy spent in supervising me during these months. You taught me hard work always repays in the end, and this is a lesson I will never forget. Every suggestion, comment, or word from you brought me a step closer to the final result, and you always encouraged me to not give up and always give my best. Thank you for the kind and wise words during the lockdown. None of this would have been possible without you. I would also like to acknowledge Pablo, Claudia, and Jesse for accepting to be part of my thesis committee.

Next, I would like to thank all my friends that have been with me during these months and even more, all over the world. My friends from the Netherlands, Bianca, Colm, Gabriele, Panos, Rafal, Jody, and Kyriakos. When I left Italy, I was scared of not being able to meet new people and make new friends, but thankfully I met an amazing group of people. Thank you, Bianca, Colm and Francesca, for being so supportive especially during the pandemic. All of you, together with the tea breaks and movie nights, will always have a special place in my heart. My friends from Italy, my second family. I would like to thank all my bois, Alessandro, Fabrizio, Marco, Pasquale, Rocco, Stefano, Valerio and Vito, and Liliana, Teodora, Monica, and Benedetta. You believed in me from the very beginning and I am extremely grateful for that. I could not have asked for better friends in my life, I love you guys. Panda and Rosetta, we shared three amazing years in Torino, and even if we are far apart I know I can always count on you. My love for you is endless, even if have never told you so. A special thanks to Luca: even if you have not been with me in Delft during the second year, all the memories from our first year are undoubtedly priceless. Valentina, you changed me since I met you. You are the strongest person I know and you showed me life from a totally different perspective. Your support is the most precious thing I could have asked for.

Finally, I would like to thank my family. Mom, Dad, you made me the man I am today. Words cannot describe the love I feel for you. I wish I could give you back even an infinitesimal part of the love and affection you gave me throughout all these years, and that would still be immense. I would not be here without you. Martina, you are my greatest inspiration. Thank you for all the kind words and attention you always gave me. You always protected me, and there are not enough words to properly thank you. I love you with all my heart.

1 INTRODUCTION

The development of web services granted access to an unprecedented amount of data which users cannot surf in its entirety, contributing to the ascent of Recommender Systems (RS) [69]. RS assist users in traversing the catalog by predicting items the user might like based on past preferences. Typically, designing a RS was an accuracy-oriented task. The underlying assumption of the most used algorithms (e.g. collaborative filtering [76] and content filtering [54]) is to match the user preferences to build a recommendation as accurate as possible. However, in recent years, research steered towards a diversity-aware recommendation [49, 7, 6], as diversity became crucial for the sake of user interests [35]. On one hand, a highly accurate recommendation is more likely to comply with user personal taste, but it narrows down the choices to items alike to previously consumed items (i.e. items from the same category) [62]. On the other hand, diversity in recommendations brings a wider spectrum of alternatives for the user, and this is to some extent preferred to improve user satisfaction [23]. Accuracy and diversity coexist in a delicate trade-off which is critical for the performance of the system, known as the accuracy-diversity trade-off, dilemma, or balance [96]. Interpreting user's tastes from limited interactions (e.g. click, view, like, rating, etc.) makes it very hard to achieve a solution to the trade-off.

The complexity in modeling user-item relationships is at the basis of the accuracy-diversity trade-off [98]. A direction to provide a better trade-off is generalizing accuracy-oriented algorithms, to incorporate also diversity. One example is the nearest neighbor (NN) collaborative filter, which relies only on similarities to build the recommendation (*similar users like similar items*). For instance, [96] extends the vanilla NN collaborative filter to the antipode side and considers the influence of furthest neighbors (FNs), i.e., a subset of the k most dissimilar users w.r.t the target user in terms of preferences. The assumption here is that recommending items FNs disliked the most could bring more diversity while preserving an acceptable level of accuracy (*the enemy of my enemy is my friend*) [70]. Alternative approaches aiming at trading accuracy with diversity include re-ranking [97, 6], leveraging side information [38, 66], or merging different models operating with different criteria [96, 98]. We shall review this in detail in Chapter 3.

The user-item interactions, at the basis of the trade-off in RS, are efficiently captured by non-Euclidean data structures such as graphs. Graph-based RS flourished in recent years. Examples of graph-based approaches are diffusion-based recommendations [64], random walks [2], and graph neural network-based recommendations [61, 83, 95], to name a few. Moreover, as a result of the joint efforts of signal processing and machine learning communities, advanced tools are available to leverage information over irregular domains [65]. The building block in these areas is the graph convolution, a generalization to graphs of the convolution operation in temporal and spatial signals processing [27]. Graph convolutions are important in irregular data because they allow processing and learning in an efficient manner by taking into account the geometry of these data. Graph convolutional filters enjoy also an equivalent spectral analysis as conventional convolutional filters do. The latter allows characterizing the behavior of the graph convolutional filters in two domains: in the vertex domain –how users or items interact with each other– and in the graph spectral domain –how graph spectral components are exploited to learn recommendations. In RS, graph convolutions have already reached state-of-the-art performance in terms of accuracy [37, 61, 95]. However, literature has only examples of graph convolutions for RS working for accuracy-oriented purposes, without considering diversity [37, 61]. That is, the exploited interactions are mainly similarity-based to aid accuracy. For this reason, the potential of graph convolutions for increasing the diversity and improving accuracy-diversity trade-off is unexplored. To fill this gap, we propose to use graph convolutions for establishing a novel accuracy-diversity trade-off for RS. In specific, the research questions this thesis aims to answer are:

(RQ1) *How can we employ graph convolutions to tweak the accuracy-diversity trade-off for recommender systems?*

(RQ2) *How can we learn the graph convolutional parameters to model jointly accuracy and diversity?*

(RQ3) *How can we explain the accuracy-diversity trade-off achieved by the graph convolutional model in the spectral domain?*

To answer these questions, we propose a novel accuracy-diversity trade-off framework for RS via graph convolutions. The model jointly operates on a NN graph to improve accuracy and on a FN graph to improve diversity. Each graph can capture user-user or item-item relationships, allowing to also include the hybrid settings, such as a user-NN and an item-FN graph. We develop design strategies that estimate the joint model parameters in view of both accuracy and diversity. These design strategies are versatile to both rating and ranking frameworks. In the rating case, they optimize the mean square error between the prediction and the true rating and consider the accuracy-diversity trade-off as a regularizer. In the ranking case, they optimize the Bayesian personalized ranking criterion proposed by [67], to account for the final order in the recommendation list, and the accuracy-diversity trade-off is also considered the regularizer. Finally, we study the effects of our model in the graph frequency domain. Such analysis helps our understanding of how graph convolutions are able to filter out irrelevant features and focus on discriminative aspects in both NN and FN graphs. Our discussion leads to the following research question:

(RQ4) *How does the proposed accuracy-diversity trade-off model behave in recommender system datasets?*

To address this research question, we conduct extensive numerical experiments on three real world datasets with increasing sparsity level. We measure the performance of the proposed model with two accuracy metrics, one for rating and one for ranking, and two diversity metrics, to inspect diversity at the system and the user level, respectively. We compare our results against accuracy-optimized state-of-the-art baselines, and against algorithms proposing a solution to accuracy-diversity trade-off.

By answering the research questions, this thesis provides the following contributions:

1. A new framework in RS by using graph convolutions to provide a wider spectrum of choices in the accuracy-diversity trade-off. The model aims at improving diversity without a significant drop in accuracy.

2. Versatile design choices to embed diversity at learning phase via joint optimization. The contributions of NN and FN graphs are balanced via a tunable parameter controlling the trade-off.

3. A graph-spectral analysis of the accuracy-diversity trade-off to explain the inner mechanisms of the joint convolutional model. The analysis highlights the importance of multi-hop neighbors in both NN and FN graphs.

4. Analysis of two accuracy-diversity trade-offs: (i) a broader view, by analyzing aggregated diversity, (ii) a narrower inspection, by analyzing individual diversity. Numerical experiments show the more importance is given to NNs (FNs), the higher is the accuracy (diversity).

The findings of this thesis contributed to the following research article, which we attach in the end of the thesis.

- Elvin Isufi, **Matteo Pocchiari**, and Alan Hanjalic, "Accuracy-Diversity Trade-off in Recommender Systems via Graph Convolutions", submitted to *Advances in Graph Representation Learning for Large-scale Multimedia Analytics* (2020).

The remainder of this thesis proceeds as follows. Chapter 2 covers the background knowledge about RS and graph convolutions, which we will use throughout the thesis. Chapter 3 discusses relevant literature. Chapter 4 introduces our proposed method and its spectral analysis. Chapter 5 contains the numerical experiments. Chapter 6 concludes the thesis.

Notation. We use plain letters a or A to denote scalar variables, bold lowercase letters $\mathbf{a}$ to denote vector variables, and bold uppercase letters $\mathbf{A}$ to denote matrix variables. The (i,j)th entry of matrix $\mathbf{A}$ is denoted by A_{ij} or $[\mathbf{A}]_{ij}$ and, likewise, the ith entry of vector $\mathbf{a}$ is denoted by a_i or $[\mathbf{a}]_i$. We will denote sets with calligraphic letters, e.g., $\mathscr{A}$ and their cardinality as $|\mathscr{A}|$. The transpose operator is denoted by $^\top$. The two norm of vector $\mathbf{a}$ is denoted by $\|\mathbf{a}\|_2$.

2

Background

In this chapter, we introduce the required material we will use throughout the rest of the thesis. The structure of the chapter is the following: Section 2.1 introduces the RS setting and collaborative filtering; Section 2.2 introduces Graph Signal Processing, the field that studies how to process information from irregular domains; Section 2.3 shows how collaborative filtering can be seen in the optic Graph Signal Processing; Section 2.4 concludes the chapter by highlighting the key concepts.

2.1. Recommender Systems

2.1.1. Setting

In the RS setting, we work with a user set $\mathcal{U} = [u_1, \dots, u_U]$ with cardinality $|\mathcal{U}| = U$ and an item set $\mathcal{I} = [i_1, \dots, i_I]$ with cardinality $|\mathcal{I}| = I$. Users interact with items by giving a rating (often a scalar from one to five). This setting is represented with the user-item matrix (UIM) $\mathbf{X} \in \mathbb{R}^{U \times I}$, where the rating user u gave to item i is the entry X_{ui}. If user u did not rate item i, we set $X_{ui} = 0$. See Table 2.1.

Table 2.1: Example of UIM. Entries with a dash indicate absence of rating, e.g. X_{34} means user 3 has not a rating for item 4. The goal of the RS is to predict the missing entries, e.g. predict X_{34}.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	3	4	-	5	1
User 2	3	-	2	4	-
User 3	5	2	2	-	5

A typical scenario in RS sees $\mathbf{X}$ having high sparsity. The goal is to use the information provided in the known ratings to predict the missing values in the UIM. I.e., if $X_{ui} = 0$, the output of the recommendation is the prediction $\hat{X}_{ui}$. We obtain a matrix $\hat{\mathbf{X}}$ containing the predictions for all possible user-item pairs $(u, i) \in \mathcal{U} \times \mathcal{I}$. Ultimately, the N the highest predicted ratings in the u-th row of $\hat{\mathbf{X}}$ will be the N items recommended to user u.

Collaborative filtering with its two variations (user-based and item-based collaborative filtering) is the most used algorithm to leverage information from ratings in $\mathbf{X}$ [76, 8]. Regardless of the variation, a notion of similarity is needed. For this thesis, we use the Pearson correlation coefficient, as it is widely adopted in RS field [37, 29, 79], but other measures can be employed such as cosine similarity [75] or Jaccard similarity [16].

Considering the user-based scenario, the symmetric user-user similarity matrix $\mathbf{B} \in \mathbb{R}^{U \times U}$ contains the Pearson correlation coefficient for all possible user-user pairs. Given two users u and v, let $\mathcal{S}_{uv}$ be the set of items rated by both u and v. Furthermore, let

$$\mu_u = \frac{1}{|\mathcal{S}_{uv}|} \sum_{i \in \mathcal{S}_{uv}} X_{ui} \tag{2.1}$$

be the user u mean; that is, the average rating that user u gives to items $i \in \mathcal{S}_{uv}$. The Pearson correlation coefficient B_{uv} between users u and v is

$$B_{uv} = \frac{\sum_{i \in \mathcal{S}_{uv}} (X_{ui} - \mu_u)(X_{vi} - \mu_v)}{\sqrt{\sum_{i \in \mathcal{S}_{uv}} (X_{ui} - \mu_u)^2 \sum_{i \in \mathcal{S}_{uv}} (X_{vi} - \mu_v)^2}}. \tag{2.2}$$

The scalar B_{uv} measures the similarity between two users u and v, based on the subsample of items they both interact. The denominator normalizes the correlations, as $B_{uv} \in [-1, 1]$ for all $(u, v) \in \mathcal{U}$. In other words, if users u and v have similar ratings for common items (high ratings to the same items, low ratings to the same items) the correlation will be close to one; discordant ratings (user u gives a high rating to item i while user v gave a low rating to the this item) give correlation close to negative one.

Likewise, in the item scenario we can assess item similarities with analogous reasoning. The symmetric item-item similarity matrix $\mathbf{C} \in \mathbb{R}^{I \times I}$ contains the Pearson correlation measured among all item-item pairs. Given two items i and j, let $\mathcal{T}_{ij}$ be the set of users that have rated both i and j. Let also

$$\mu_i = \frac{1}{|\mathcal{T}_{ij}|} \sum_{u \in \mathcal{T}_{ij}} X_{ui} \tag{2.3}$$

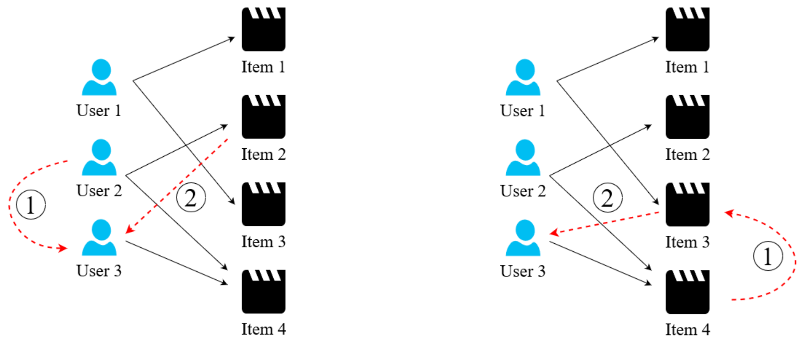
be the item i mean, namely, the average rating given to item i by the users that have also interacted with item j. The correlation coefficient C_{ij} between two items i and j is

$$C_{ij} = \frac{\sum_{u \in \mathcal{T}_{ij}} (X_{ui} - \mu_i)(X_{uj} - \mu_j)}{\sqrt{\sum_{u \in \mathcal{T}_{ij}} (X_{ui} - \mu_i)^2 \sum_{u \in \mathcal{T}_{ij}} (X_{uj} - \mu_j)^2}}. \tag{2.4}$$

We shall see how to use $\mathbf{B}$ and $\mathbf{C}$ in user-based and item-based scenarios to predict ratings of the target user u.

2.1.2. Nearest Neighbor Collaborative Filtering

In user-based nearest neighbor (NN) collaborative filtering, the rationale is that similar users share similar tastes [76]. Once the similarity is assessed for all users, the predicted rating $\hat{X}_{ui}$ uses only neighbors with the highest correlation coefficient (hence, the NN in the name); see Fig. 2.1(a). Let $\mathcal{N}_{ui}$ be the set of $|\mathcal{N}_{ui}|$ users which are most similar to u



(a) Example of user-based collaborative filtering.

(b) Example of item-based collaborative filtering.

Figure 2.1: Examples of collaborative filtering variations. In both cases, we consider user 3 to be the target user. (a) First, similarity between users is assessed and only the most similar users are kept. In our case, user 2. Then, we look at items consumed by user 2 to build the recommendation. In this case item 2 is recommended. (b) First, we compute the similarity between items consumed by target user and the other items, based on the ratings the community provided. Finally, we find the most similar item to recommend. In this case item 3.

(the entries of the u-th row of $\mathbf{B}$ with the highest value) and rated item i. The predicted rating for item i is

$$\hat{X}_{ui} = \frac{\sum_{v \in \mathcal{N}_{ui}} B_{uv} X_{vi}}{\sum_{v \in \mathcal{N}_{ui}} B_{uv}}. \tag{2.5}$$

In item-based collaborative filtering, we work with item similarities to perform the recommendation; see Fig. 2.1(b). Let $\mathcal{N}^{iu}$ the set of $|\mathcal{N}^{iu}|$ items which are the most similar to i (the entries of the i-th row of $\mathbf{C}$ with the highest value) and rated by user u. The predicted rating is

$$\hat{X}_{ui} = \frac{\sum_{j \in \mathcal{N}^{iu}} C_{ij} X_{uj}}{\sum_{j \in \mathcal{N}^{iu}} C_{ij}}. \tag{2.6}$$

The normalization is present in both NN estimators to reduce the bias of rating patterns of different users [57]. Some users might be more "tolerant" and give higher ratings, while some other users more "conservative" and tend to stay on low ratings [44].

Although being the foundation of many recommendation algorithms, NN collaborative filtering considers only the influence of direct neighbors. In the following section, we shall see how to express rating prediction as an application of graph signal processing. We will extend the rationale of NN estimators to include multi-hop neighbors' contribution, by means of graph convolutional filters acting on similarity graphs (user-similarity graph or item-similarity graph).

2.2. GRAPH SIGNAL PROCESSING

2.2.1. BASICS

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph of N vertices (or nodes) $\mathcal{V} = \{v_1, ..., v_N\}$ and a set of edges $\mathcal{E}$. Each edge is associated with a weight $W_{ij} > 0$ indicating the strength of the relationship between nodes i and j. If there is no specified direction for the tuples $(v_i, v_j) \in \mathscr{E}$, the graph is undirected. The adjacency matrix $\mathbf{W}$ of an undirected weighted graph is an $N \times N$ symmetric matrix, whose entry $W_{ij} > 0$ is the weight of the edge $e_{i,j}$ connecting nodes i and j.

A graph signal $x : \mathscr{V} \to \mathscr{R}$ associates to each node a real value; see Fig.2.2. We express the signal in the vector form $\mathbf{x} = [x_1, x_2, \ldots, x_N]$, where entry x_i is the signal value at node v_i. As $\mathscr{G}$ can represent complex relations between entities in nodal domain, the graph signal $\mathbf{x}$ depicts information on top of such graph. In NN RS, we work with similarity graphs (user-graph or item-graph) and use the known ratings as a graph signal.

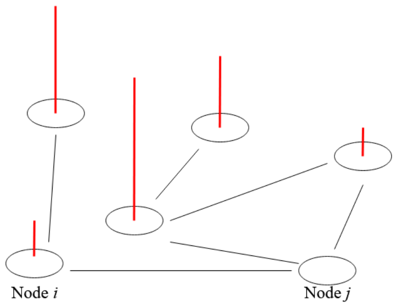


Figure 2.2: A graph signal $\mathbf{x}$. Node j has no value associated with it. Using information of neighboring nodes we can predict the signal value at node j.

2.2.2. GRAPH CONVOLUTIONAL FILTERS

The information contained in the graph signal $\mathbf{x}$ propagates through $\mathscr{G}$ by *shifting*. The graph shift operator (GSO) is a matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ whose sparsity matches the structure of $\mathscr{G}$ [27]. An example of S is the adjacency matrix W of the graph. Another example is the graph Laplacian matrix. Multiplying the GSO $\mathbf{S}$ with the graph signal $\mathbf{x}$ diffuses the signal over $\mathscr{G}$, resulting in the shifting operation

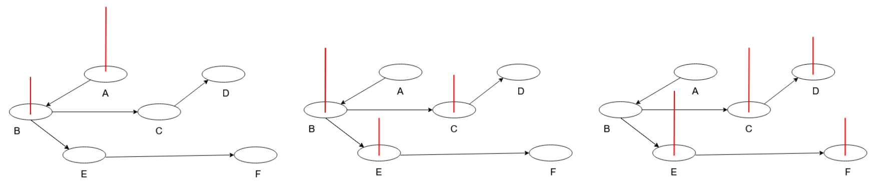
$$\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x} \tag{2.7}$$

which is a vector $\mathbf{x}^{(1)} = [x_1^{(1)}, \ldots, x_N^{(1)}]^\top$. Being $\mathscr{N}_i$ the set of direct neighbors to node i, the signal value at node i after the shift is

$$x_i^{(1)} = \sum_{j \in \mathscr{N}_i} S_{ij} x_j. \tag{2.8}$$

In words, the signal value $x_i^{(1)}$ at node i is the linear combination of the signal values of its neighbors. We highlight shifting acts *locally*: a node i does not need to know the values of the nodes that are not immediate neighbors. This has computational benefits for operation (2.8), which is of order $\mathscr{O}(|\mathscr{E}|)$; i.e., on the order of the edges.

By applying the GSO up to k times, we can reach up to k-hop neighborhoods. The output of the recursive relation $\mathbf{x}^{(k)} = \mathbf{S}^k\mathbf{x} = \mathbf{S}\mathbf{S}^{(k-1)}\mathbf{x} = \mathbf{S}\mathbf{x}^{(k-1)}$ is another graph signal where the influence of the k-hop neighborhoods are considered. Put differently, the value $x_i^{(k)}$ contains information stored in nodes that are k-hops away from i. The computational complexity to obtain $\mathbf{x}^{(k)}$ is of order $\mathcal{O}(\mathrm{k}|\mathcal{E}|)$. Fig. 2.3 depicts multiple shifts.



(a) Original graph signal $\mathbf{x}$ (b) Signal after one shift $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}$ (c) Signal after two shifts $\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)}$

Figure 2.3: Example of the shifting a signal on a directed graph. The original graph signal has nonzero values only at nodes A and B, as we notice from (a). After shifting, the result is $\mathbf{x}^{(1)}$ in (b). The value that was previously stored in node A now is in node B. Likewise the value that was previously in node B now is in its one-hop neighbors, C and E. By shifting $\mathbf{x}^{(1)}$, we obtain the two-hops shifted signal $\mathbf{x}^{(2)}$ in (c). Nodes C and E perceive the influence of their two-hops neighbor node A. Similarly nodes D and F perceive the influence of their two-hops neighbor B.

The notion of shift operator relates to graph convolutional filters. A graph convolutional filter is a matrix $\mathbf{H}(\mathbf{S}) \in \mathbb{R}^{N \times N}$, written as a polynomial of the shift operator $\mathbf{S}$ as

$$\mathbf{H}(\mathbf{S}) = h_0\mathbf{I} + h_1\mathbf{S} + h_2\mathbf{S}^2 + \ldots + h_K\mathbf{S}^K = \sum_{k=0}^{K} h_k\mathbf{S}^k \tag{2.9}$$

where K is the filter order. Filtering a graph signal $\mathbf{x}$ means performing the matrix multiplication between a graph convolutional filter $\mathbf{H}(\mathbf{S})$ and the signal itself. The output $\hat{\mathbf{x}}$ is

$$\hat{\mathbf{x}} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} h_k\mathbf{S}^k\mathbf{x}. \tag{2.10}$$

The filter coefficients (or parameters) $\mathbf{h} = [h_0, h_1, \ldots, h_K]^\top$ weight the information coming from different resolutions $\mathbf{x}, \mathbf{S}\mathbf{x}, \ldots, \mathbf{S}^K\mathbf{x}$. The filter parameters in $\mathbf{h}$ define the behavior of the filter. Parameters can be learned in a data-driven fashion, by optimization of a loss function defined for the system. This procedure is called *filter design* and will play a central role in how we approach the recommendation problem in Sections 4.2 and 4.3. For detailed properties of graph filters refer to [73].

The advantages of employing a graph convolutional filter are: (i) a limited number of parameters (i.e. $K+1$ parameters to describe a filter of order K), which reduces the load of the learning phase; and (ii) high interpretability of the output $\hat{\mathbf{x}}$, as the filter coefficients $h_1, \ldots, h_K$ explain the importance of different neighborhoods. On the other hand, graph filters lack in expressivity, as they are the result of linear combinations of matrices. The limit in expressivity led to the graph convolutional neural networks, which introduce nonlinearity.

2.2.3. Graph Convolutional Neural Networks

Graph convolutional filters are the building block for graph convolutional neural networks (GCNNs) [1]. To build a GCNN with the graph convolutional filter in (2.10), we

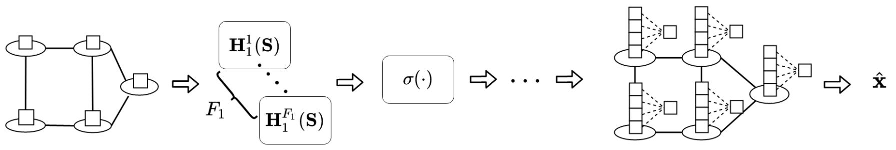

$\mathbf{H}_1^1(\mathbf{S})$
F_1
$\mathbf{H}_1^{F_1}(\mathbf{S})$
$\sigma(\cdot)$
$\hat{\mathbf{x}}$

Figure 2.4: Structure of a GCNN. Each layer is composed by a bank of graph convolutional filters followed by a nonlinearity. The final convolutional features are aggregated with a shared fully connected layer (to reduce the number of parameters w.r.t a pure fully connected layer fashion).

consider the composition of a set of L layers. The first layer $\ell = 1$ comprises a bank of F_1 filters $\mathbf{H}_1^f(\mathbf{S})$ each defined by coefficients $\{h_{1k}^f\}_k$. Each of these filters outputs graph signals $\mathbf{u}_1^f = \mathbf{H}_1^f(\mathbf{S})\mathbf{x}$, which are subsequently passed through a pointwise nonlinearity $\sigma(\cdot)$ to produce a collection of F_1 features $\mathbf{x}_1^f$ that constitute the output of layer $\ell = 1$, i.e.,

$$\mathbf{x}_1^f = \sigma\left[\mathbf{u}_1^f\right] = \sigma\left[\mathbf{H}_1^f(\mathbf{S})\mathbf{x}\right] = \sigma\left[\sum_{k=0}^{K} h_{1k}^f \mathbf{S}^k \mathbf{x}\right] \quad \text{for } f = 1, \ldots, F_1. \tag{2.11}$$

At subsequent intermediate layers $\ell = 2, \ldots, L-1$, the output features $\{\mathbf{x}_\ell^g\}_g$ of the previous layer $\ell - 1$ become inputs to a bank of $F_\ell F_{\ell-1}$ convolutional filters $\mathbf{H}_\ell^{fg}(\mathbf{S})$ with coefficients $\{h_{\ell k}^{fg}\}_k$ each of which produces the output features $\mathbf{u}_\ell^{fg} = \mathbf{H}_\ell^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^g$. To avoid exponential growth, the filter bank outputs obtained from a common input feature $\mathbf{x}_{\ell-1}^g$ are aggregated and the result is passed through a nonlinearity $\sigma(\cdot)$ to produce the F_ℓ output features $\mathbf{x}_\ell^f$ of layer ℓ, i.e.,

$$\mathbf{x}_\ell^f = \sigma\left[\sum_{g=1}^{F} \mathbf{u}_\ell^{fg}\right] = \sigma\left[\sum_{g=1}^{F} \mathbf{H}_\ell^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^g\right] = \sigma\left[\sum_{g=1}^{F}\sum_{k=0}^{K} h_{\ell k}^{fg} \mathbf{S}^k \mathbf{x}_{\ell-1}^g\right] \quad \text{for } f = 1, \ldots, F_\ell. \tag{2.12}$$

The operation in (2.12) is the propagation rule for a generic layer ℓ of the GCNN and it is repeated until the last layer $\ell = L$ which collection of F_L features $\mathbf{x}_L^1, \ldots, \mathbf{x}_L^{F_\ell}$ forms the GCNN output. These final convolutional features are then passed through a shared multi-layer perceptron per node to map the F_L features per node n, $[x_{Ln}^1; \ldots; x_{Ln}^{F_\ell}]$ into the final output $\hat{\mathbf{x}}$; see Figure 2.4.

In essence, the GCNN is a map $\mathbf{\Phi}(\cdot)$ that takes as input a graph signal $\mathbf{x}$, a GSO $\mathbf{S}$ describing a graph $\mathcal{G}$, and a set of parameters $\mathcal{H} = \{h_{\ell k}^{fg}\}$ for all layers ℓ, orders k, feature pairs (f, g), and final multi-layer perceptron. This map produces the output

$$\mathbf{\Phi}(\mathbf{x}; \mathbf{S}; \mathcal{H}) := \hat{\mathbf{x}}. \tag{2.13}$$

The GCNN inherits the numerical benefits of graph convolutions. Denoting by $F = \max_\ell F_\ell$ the maximum number of features for all layers, the number of parameters defining the GCNN is of order $\mathcal{O}(F^2KL)$ while its computational complexity amounts to $\mathcal{O}(F^2KL|\mathcal{E}|)$. The GCNN parameters are again learned in a data-driven way by minimizing a loss function. We stress the relevance of this characteristic for its importance in Section 4.2 and 4.3.

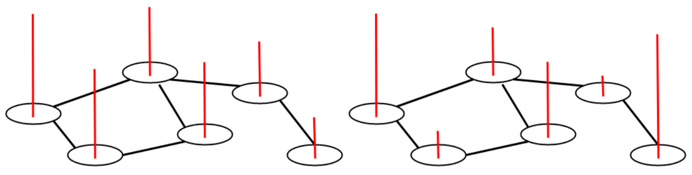

(a) Slow variation graph signal. (b) High variation signal

Figure 2.5: Example of (a) low variation graph signal and (b) high variation graph signal.

Lastly, we remark the linear graph convolutional filter [cf. (2.10)] is a particular GCNN map $\boldsymbol{\Phi}(\cdot)$ [cf. (2.13)] limited to the linear space and composed of a single filter. For the rest of this thesis, we use notation $\boldsymbol{\Phi}(\cdot)$ to generalize between the graph convolutional filters and the GCNN, since both can be employed to perform rating prediction. We shall see the application of such tools to NN collaborative filtering and its extension to multi-hop neighbors.

2.2.4. SPECTRAL INTERPRETATION

Beside nodal domain, we can study graph convolutional filters in the spectral (or frequency) domain, to gain more insights about the filters' behavior. To discuss the notion of frequency for graph signals, we introduce the concept of Fourier transform in directed graphs [74]. Assuming the GSO $\mathbf{S}$ is diagonalizable, it admits the eigendecomposition $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$ with eigenvector matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ and complex eigenvalues $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$. Then, the graph Fourier transform (GFT) of a graph signal $\mathbf{x}$ is

$$\tilde{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}. \tag{2.14}$$

I.e., it is the projection of $\mathbf{x}$ onto the shift operator eigenspace. The ith GFT coefficient $\tilde{x}_i$ of $\tilde{\mathbf{x}}$ quantifies the contribution of the ith eigenvector $\mathbf{u}_i$ to expressing the variability of signal $\mathbf{x}$ over the graph. The latter is analogous to the discrete Fourier or cosine transform for temporal or spatial signals if the graph is particularized to a directed line or to a rectangular grid, respectively. In this analogy, the complex eigenvalues $\lambda_n \in \boldsymbol{\Lambda}$ are referred to as the graph frequencies [73, 80]. The inverse transform is $\mathbf{x} = \mathbf{U}\tilde{\mathbf{x}}$. To measure the graph signal variability, we follow [72] and order the graph frequencies λ_i based on their distance from the maximum eigenvalue $\lambda_{\max}(\mathbf{S})$. This ordering is based on the notion of total variation (TV), which for the eigenpair $(\lambda_n, \mathbf{u}_n)$ is defined as

$$\text{TV}(\mathbf{u}_n) = \left|1 - \frac{\lambda_n}{\lambda_{\max}(\mathbf{S})}\right| \|\mathbf{u}_n\|_1 \tag{2.15}$$

where $\|\cdot\|_1$ is the ℓ_1-norm. The closer eigenvalue λ_i is to the maximum eigenvalue $\lambda_{\max}(\mathbf{S})$, the smoother is the corresponding eigenvector $\mathbf{u}_i$ over the graph (i.e., values on neighboring nodes are similar). If signal $\mathbf{x}$ changes little over the graph, the corresponding GFT $\tilde{\mathbf{x}}$ has nonzero entries mostly in entries $\tilde{x}_i$ which index corresponds to a low graph frequency $\lambda_i \to \lambda_{\max}(\mathbf{S})$ (low TV); see Figure 2.5(a). Contrarily, if signal $\mathbf{x}$ varies substantially in nodes sharing an edge, the GFT $\tilde{\mathbf{x}}$ has nonzero values also in those entries $\tilde{x}_i$ which index corresponds to a high graph frequency $\lambda_i \gg \lambda_{\max}(\mathbf{S})$ (high TV); see Figure 2.5(a). Refer to [72, 65] for further detail.

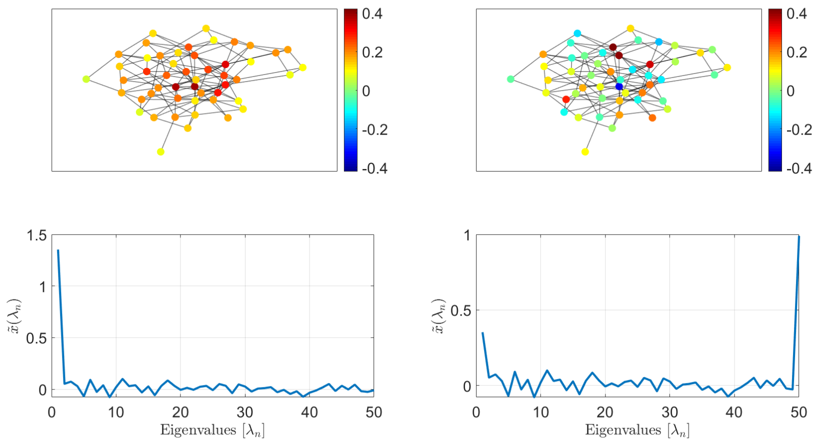


Figure 2.6: Two graph signals with different total variation in their nodal (Top) and frequency (Bottom) domain representation, on a sample graph of 50 nodes. (Left) In a slow-varying graph signal $\mathbf{x}$, if nodes i and j share an edge, graph values x_i and x_j are close to each other; in spectral domain, it means low frequency components are the most contributing ones. (Right) In a high-varying graph signal $\mathbf{x}$, adjacent nodes i and j have discordant graph signal values x_i and x_j; in spectral domain, this translates to high frequency components being the most contributing ones.

With this analogy in place, we substitute the eigendecomposition $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$ into the graph convolutional filter (2.10) and obtain the filter input-output relationship in the spectral domain

$$\tilde{\hat{\mathbf{x}}} = \sum_{k=0}^{K} h_k \boldsymbol{\Lambda}^k \tilde{\mathbf{x}} := \mathbf{H}(\boldsymbol{\Lambda})\tilde{\mathbf{x}} \tag{2.16}$$

where $\tilde{\hat{\mathbf{x}}} = \mathbf{U}^{-1}\hat{\mathbf{x}}$ is the GFT of the output and $\mathbf{H}(\boldsymbol{\Lambda}) = \sum_{k=0}^{K} h_k \boldsymbol{\Lambda}^k$ contains the filter frequency response on the main diagonal. Relation (2.16) shows in first place graph convolutional filters respect the convolutional theorem as they act in the spectral domain as a pointwise multiplication between the filter transfer function $\mathbf{H}(\boldsymbol{\Lambda})$ and the input GFT $\tilde{\mathbf{x}}$. Therefore, analyzing the diagonal of $\mathbf{H}(\boldsymbol{\Lambda})$ shows how the graph convolutional filter acts on the input signal $\mathbf{x}$ to build $\hat{\mathbf{x}}$.

We remark any graph convolutional filter can be subject to spectral analysis. The concepts about spectral domain introduced here will be employed in Section 4.4. We will study the characteristics of plain graph convolutional filters and filters from GCNNs to gain insights about the way recommendation is carried out, with particular regard to the accuracy-diversity balance.

2.3. NEAREST NEIGHBORS AS GRAPH CONVOLUTIONS

The notions presented in the previous section are valid for any graph $\mathcal{G}$. Here we transfer the knowledge to recommender systems. We introduce the user-similarity graph and we explain how the columns of $\mathbf{X}$ can be seen as signals acting on top of such graph. The same is valid for the item-similarity graph and the rows of $\mathbf{X}$.

2.3.1. User-based Scenario

Consider the symmetric user-user similarity matrix $\mathbf{B} \in \mathbb{R}^{U\times U}$, whose entries $\mathbf{B}_{uv}$ are computed as in (2.2). We use matrix $\mathbf{B}$ as the adjacency matrix of an undirected graph $\mathcal{G}_u = \{\mathcal{U}, \mathcal{E}_u\}$, where each user $u \in \mathcal{U}$ is a node and two users u and v are connected with an edge $e_{uv} \in \mathcal{E}_u$ if $B_{uv} \neq 0$. From $\mathcal{G}_u$, we can to construct a similarity graph for each item $i \in \mathcal{I}$, whose adjacency matrix will be denoted with $\mathbf{B}_i$. This is done by taking into account that some users might not have interacted with item i. Hence, the graph topology depends on the specific item i.

Consider a specific item i and let $\mathcal{T}_i \subseteq \mathcal{U}$ be the set of users who interacted with item i. The item-i specific user-similarity matrix $\mathbf{B}_i$ is obtained from $\mathbf{B}$ as:

1. Treat undirected edges as bidirectional edges.
2. Remove the edges starting from users who did not rate item i. Given two users u and v, where user v did not rate item i ($v \notin \mathcal{T}_i$) and user u rated it ($u \in \mathcal{T}_i$), remove the edge e_{vu}. That is, set B_{vu} to zero. Keep edge e_{uv}, that is, do not modify B_{uv} since user v has relevant information about item i for user v.
3. Keep only the k most similar users. Given a user u, keep only the k most similar users, i.e. the k highest entries of u-th row of $\mathbf{B}$.
4. Normalize the weights to guarantee a normalization factor as in (2.5).

Fig. 2.7 illustrates the first three steps.

While working with item i, we introduce $\mathbf{x}^i \in \mathbb{R}^U$ as graph signal containing the ratings of all the users to item i. It coincides with the i-th column of the UIM $\mathbf{X}$. Since not every user rated item i, graph signal $\mathbf{x}^i$ is sparse by constitution. We use GSP tools, namely the graph convolutional filter [cf.(2.10)] and GCNN [cf.(2.13)], to propagate the information contained in $\mathbf{x}^i$ throughout $\mathbf{B}_i$. The goal is to get $\hat{\mathbf{x}}^i \in \mathbb{R}^U$ which stores the predictions for all the users about item i.

A straightforward approach is to shift $\mathbf{x}^i$ to direct neighbors, employing the adjacency matrix $\mathbf{B}_i$ as GSO. In other words, we implement user-based NN collaborative filtering [cf.(2.5)] as one-hop shifting of the graph signal $\mathbf{x}^i$. The prediction $\hat{\mathbf{x}}^i$ is obtained as

$$\hat{\mathbf{x}}^i = \mathbf{B}_i \mathbf{x}^i \tag{2.17}$$

where only the influence of the most positively correlated users is examined. Equation (2.17) and NN user-based collaborative filtering [cf. (2.5)] yield the same result.

We can generalize to higher resolutions by exploiting the properties of graph convolutional filters. We design a filter $\mathbf{H}(\mathbf{B}_i)$ of order K to include contribution of users that are at most K-hops away. The predicted ratings are

$$\hat{\mathbf{x}}^i = \mathbf{H}(\mathbf{B}_i)\mathbf{x}^i = \sum_{k=1}^{K} h_k \mathbf{B}_i^k \mathbf{x}^i, \tag{2.18}$$

where we account for positive correlations among users that are at most K-hops away[1]. In words, $\mathbf{B}_i\mathbf{x}^i$ is a weighted average of the ratings produced by one-hop neighboring

[1] The term for $k = 0$, $\mathbf{S}^0\mathbf{x} = \mathbf{x}$, does not contribute to predicting ratings [37]. Hence, considering a filter order K, we have $K+1$ filter coefficients $\mathbf{h} = [0, h_1, \ldots, h_K]$.

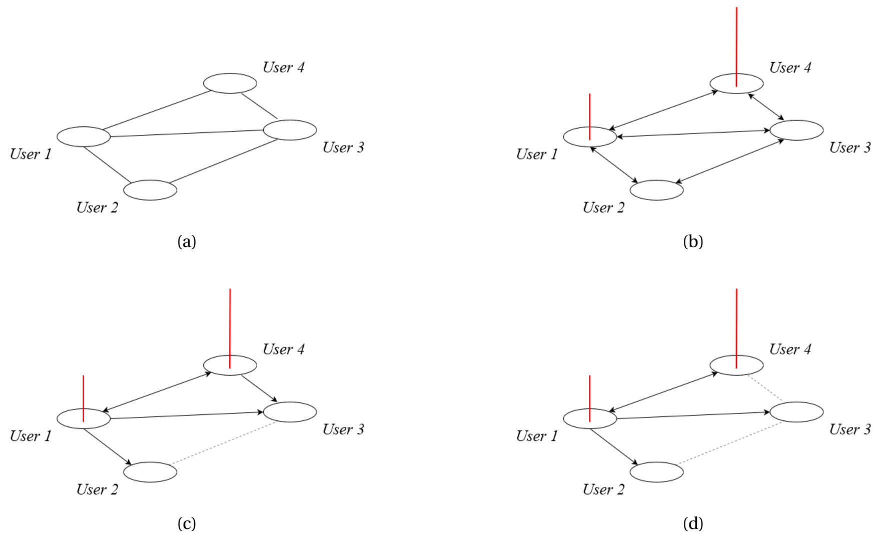


Figure 2.7: Procedure to obtain $\mathbf{B}_i$ from $\mathbf{B}$. In (a) we have the starting user similarity graph $\mathscr{G}_u$, whose structure matches $\mathbf{B}$. In (b) we consider item i with its corresponding graph signal $\mathbf{x}_i$ and treat each edge as bidirectional. Users 1 and 4 rated item i, while users 2 and 3 did not. In (c) we remove the edges starting from users who did not rate item i, thus edges starting from users 2 and 3. In (d) we keep only the k most similar users for the prediction. The figure depicts the case $k = 1$. This is item i-specific user-similarity graph with adjacency matrix $\mathbf{B}_i$, which will be the GSO of the specific user graph for item i.

users, $\mathbf{B}_i^2\mathbf{x}^i$ is a weighted average accounting for the influence of two-hop neighboring users, and, in general, $\mathbf{B}_i^K\mathbf{x}^i$ is aware of interactions happening among users in the K-hop neighborhood. Each information $\mathbf{B}_i\mathbf{x}^i, \ldots, \mathbf{B}_i^K\mathbf{x}^i$ is weighted in a different way via the filter coefficients $h_1, \ldots, h_K$.

Finally, in the same way we generalized from graph convolutional filters to GCNNs, we can predict $\hat{\mathbf{x}}^i$ with a GCNN. The GCNN learns a map $\mathbf{\Phi}(\mathbf{x}^i, \mathbf{B}_i, \mathscr{H})$ depending on the item i-specific signal $\mathbf{x}^i$, the GSO $\mathbf{B}_i$, and a set of parameters $\mathscr{H}$. Thus, the prediction $\hat{\mathbf{x}}^i$ is given by

$$\hat{\mathbf{x}}^i = \mathbf{\Phi}(\mathbf{x}^i, \mathbf{B}_i, \mathscr{H}). \tag{2.19}$$

Either we use graph convolutional filter [cf. (2.18)] or GCNN [cf. (2.19)], by repeating this process for all $i \in \mathscr{I}$ we complete the original UIM $\mathbf{X}$. The resulting matrix is $\hat{\mathbf{X}} \in \mathbb{R}^{U \times I}$ where the entry $\hat{\mathbf{X}}_{ui}$ contains the rating prediction of item i based on similarities user u shares with other users. From this point on, we shall generalize the user-based prediction with the mapping $\mathbf{\Phi}(\mathbf{x}^i, \mathbf{B}_i, \mathscr{H})$, recalling GCNNs (nonlinear models) are a natural generalization of graph convolutional filters (linear models). We specify the model employed when discussing the parameter learning phase in Sections 4.2 and 4.3.

2.3.2. ITEM-BASED SCENARIO

Item-based recommendation follows the same principles of the user-based scenario, though we work now with the item-similarity graph and a different signal. In details, consider the symmetric item-item similarity matrix $\mathbf{C} \in \mathbb{R}^{I\times I}$ computed as in (2.4). It can be seen as the adjacency graph of the item-similarity graph $\mathscr{G}_i = \{\mathscr{I}, \mathscr{E}_i\}$, where each node is an item, and two items i and j are connected with an edge $e_{ij} \in \mathscr{E}_i$ if $C_{ij} \neq 0$.

We build user u-specific item-similarity graphs for all $u \in \mathscr{U}$, whose adjacency matrices $\mathbf{C}_u$ are extracted from $\mathbf{C}$ with the same procedure presented in 2.3.1. Consider the set $\mathscr{T}_u \subseteq \mathscr{U}$ containing all the items rated by user u. We get $\mathbf{C}_u$ from $\mathbf{C}$ as:

1. Treat undirected edges as bidirectional edges.
2. Remove the edges starting from items who have not been rated by user u. Given two items i and j, where user u did not rate item i ($i \notin \mathscr{T}_u$) and rated item j ($j \in \mathscr{T}_u$), remove the edge e_{ij} and keep edge e_{ji}.
3. Keep only the k most similar items. Given an item i, keep only the k most similar items, i.e. the k highest entries of i-th row of $\mathbf{C}$.
4. Normalize the weights to guarantee a normalization factor as in (2.6).

Then, we consider the graph signal $\mathbf{x}_u \in \mathbb{R}^I$, collecting the ratings user u gave to all items. Signal $\mathbf{x}_u$ coincides with the u-th row of $\mathbf{X}$. Employing $\mathbf{C}_u$ as GSO, we can express the item-based NN collaborative filtering [cf.(2.6)] as one-hop shifting of $\mathbf{x}_u$:

$$\hat{\mathbf{x}}_u = \mathbf{C}_u \mathbf{x}_u. \tag{2.20}$$

The natural generalization is to consider multi-hop neighboring items, using a graph convolutional filter $\mathbf{H}(\mathbf{C}_u)$. The prediction becomes

$$\hat{\mathbf{x}}_u = \mathbf{H}(\mathbf{C}_u)\mathbf{x}_u = \sum_{k=1}^{K} h_k \mathbf{C}_u^k \mathbf{x}_u, \tag{2.21}$$

where the influence of items that are K-hops away is embedded in the prediction. Finally, the GCNN in the item-based scenario uses a map $\mathbf{\Phi}(\mathbf{x}_u, \mathbf{C}_u, \mathscr{H})$ depending on the graph signal $\mathbf{x}_u$, the GSO $\mathbf{C}_u$ and a set of learnable parameters $\mathscr{H}$. The prediction with GCNNs is

$$\hat{\mathbf{x}}_u = \mathbf{\Phi}(\mathbf{x}_u, \mathbf{C}_u, \mathscr{H}). \tag{2.22}$$

For the rest of this thesis, we will refer to item-based models (either linear [cf.(2.21)] or nonlinear [cf.(2.22)]) with the general map $\mathbf{\Phi}(\mathbf{x}_u, \mathbf{C}_u, \mathscr{H})$, exploiting again the relation between graph convolutional filters and GCNNs.

For the sake of generalization, we consider notation $\mathbf{\Phi}(\mathbf{x}, \mathbf{S}, \mathscr{H})$, which denotes a model (either graph convolutional filter or GCNN) working with a graph signal $\mathbf{x}$, a GSO $\mathbf{S}$ and a set of parameters $\mathscr{H}$. If we work in a user-based scenario, we will set $\mathbf{x} = \mathbf{x}^i$ and $\mathbf{S} = \mathbf{B}_i$; similarly, if we work in an item-based scenario will set $\mathbf{x} = \mathbf{x}_u$ and $\mathbf{S} = \mathbf{C}_u$. This choice will be helpful when working with optimization problems on multiple graphs in Sections 4.2 and 4.3.

2.4. CONCLUSION

This Section summarizes the content covered in this Chapter. In Section 2.1 we presented the principles of RS and the basic NN estimator, which works with similarities between users or items. Then, in Section 2.2 we saw the tools at the base of graph signal processing, namely shifting, graph convolutional filters, and the most general case with GCNNs. More insights about filters can be analyzed in the spectral domain via GFT. Finally, in Section 2.3 we explained how all these concepts can be applied to perform rating prediction in the recommendation problem. The user graph and the item graph are supports where NN collaborative filtering can be translated as shifting of specific graph signals, where only most similar direct neighbors are considered. Graph convolutional filters and GCNNs offer a way to account for multi-hop contributions from similar neighbors in a linear and nonlinear manner, respectively.

3

Related Works

This chapter discusses the relevant literature about diversity in RS and graph signal processing techniques to perform recommendation. We will use the concepts touched in Chapter 2 to present the literature. The chapter is organized as follows: Section 3.1 discusses the importance of diversity in RS and the metrics to measure it; Section 3.2 highlights the solutions for the accuracy-diversity dilemma; Section 3.3 presents works employing graph convolutional filters in RS, both in linear and nonlinear setups; Section 3.4 highlights the knowledge gap.

3.1. Diversity in RS

Diversity and User Satisfaction. The importance of aspects beyond accuracy in RS gained momentum in recent years, as evidenced by the large number of publications addressing diversity [87]. Not always an accurate recommendation is synonym of user satisfaction, as low diversity discourages a further use of the system [99] or limits the number of exposed items, resulting in the so-called *filter bubble* [62, 59, 31, 68]. Diversity has a direct impact on user experience, as shown in several works. One of the first studies in this direction is [35], which investigated how different users perceive diversity. Authors in [84] took a step forward and performed a study with focus on different personality traits: users prone to new experiences prefer a more diversified recommendation. The authors in [10] continued along this path, by giving control to the user to choose the amount of diversity in the recommendation and studied the perceived satisfaction. Authors in [23] asked feedback not only on diversity, but also on other aspects of recommendation, such as novelty, accuracy, satisfaction, and level of personalization: their work highlighted the positive correlation between user satisfaction and diversity. These works prove diversity to be a crucial asset in RS. They also highlight the existence of a tradeoff between accuracy and diversity, which we will discuss in the following. For a broader view on diversity in RS, refer to [49].

Diversity Metrics. Having clear the benefits of diversity, a metric to quantify the diversity within a RS is needed. The first to tackle this problem is [14], by defining diversity as the opposite of similarity. Also [50, 17] adopt this view. Later, metrics from document information retrieval have been adapted in RS. One of the first works following this rationale is [21], which sees diversity as a part of calculating normalized discounted cumulative gain (NDCG). They capture novelty, ambiguity, redundancy, and diversity all in one metric. This idea is extended by [86], which accounts for the user's requirements. The flaw in these metrics is the attempt of casting multiple aspects into a single number. We would rather have multiple diversity-related metrics where we capture different extents of diversity. For instance, we can study diversity at the system level. Metrics looking at item popularity distribution help the cause of quantifying diversity at the system level. A diverse recommendation is a recommendation able to leverage more items from the long-tail, in other words, less popular items. The authors of [25] use the Gini coefficient as a measure of diversity. Also authors in [3] employ the Gini coefficient, and extend the analysis by introducing the Entropy diversity, and Herfindahl diversity, to analyze whether recommendations are concentrated on few popular items or are equally spread. The authors show these dispersion metrics are highly correlated between them, thus working only with one of them, e.g. entropy-diversity, is sufficient to get insightful measurements. Another system-level metric is the aggregate diversity, often referred to as catalog coverage [4, 3, 5, 63, 82, 34]. It counts the number of items leveraged by the system throughout the recommendations to all the users. Coupled with the entropy-diversity, it gives a snapshot of the catalog's portion covered by the system. However, even if used as a standalone metric without the entropy-diversity, it is a good indicator of system diversity.

Diversity at user-level is also crucial. Individual diversity measures the average dissimilarity in a recommendation list [14, 97, 99]. Computing the dissimilarity is however domain dependent, as requires additional metadata to estimate the difference between two items. Authors in [87] propose a domain agnostic solution, which embeds also relevancy and novelty in the computation. The result is again an all-in-one metric, which might fail in capturing a single aspect. To overcome the domain dependency, we will measure dissimilarity between two items as the Euclidean distance between their latent features vectors as in [85]. We will explain the implementation details in Section 5.1.

3.2. ACCURACY-DIVERSITY TRADEOFF IN RS

Many works in literature faced the accuracy-diversity tradeoff. A popular direction to tweak this trade-off is by two-step approaches, in which a re-ranking is applied to a retrieved list for boosting diversity [97, 7]. The work in [6] re-ranks items based on the rating variance in the neighborhood of a node, while [3] proposes re-ranking approaches to cover a larger portion of the catalog. Among the same lines, [32, 24] diversify items to improve coverage in a user-personalized manner. The authors of [39], instead, propose a new metric to quantify diversity within a list and develop an optimization algorithm to improve it. Works in this category rely heavily on the provided list, therefore, it is difficult to attribute to which extent the accuracy-diversity trade-off is due to the re-ranking method or due to the issues present in the lists.

A second category of works leverages side information such as metadata or context to improve diversity with a single algorithm. The work in [38] builds a dissimilarity item-item matrix from item features and accounts for the latter in a learning-to-rank framework. Also the work in [28] uses item features to provide a single accuracy-diversity design method in a matrix completion framework. Differently, [66] leverages context and evaluates different pre-filtering, post-filtering, and modeling schemes in terms of accuracy and diversity. Worth noticing that also some re-ranking approaches leverage item features. Our method instead shows that it is possible to balance accuracy with diversity without relying on side-information in both a learning-to-rate and a learning-to-rank frameworks.

A third category of works, tweaks conventional accuracy-oriented algorithms to improve diversity. Authors in [53] investigate approaches to build user similarities that avoid the influence of high-degree users. The rationale behind these strategies is to reduce the bias induced by popular objects or high-degree users. The latter changes the direction of random walk collaborative filtering which is corroborated heuristically to improve diversity. Also [26] changes the way user similarities are calculated to improve diversity in user-based collaborative filtering. The work in [93] follows up on [38] and regularizes the learning-to-rank objective function to improve diversity. In our view, the latter may be problematic as it overloads the regularizer with an additional objective. That is, since the primarily goal of the regularizer is to generalize the model to unseen data, leveraging it also to improve diversity leads to a triple accuracy-diversity-generalization trade-off, which becomes challenging to handle; and likely one of the three objectives will be treated as byproduct. Differently, [70, 71] connect users based on their dissimilarities and propose the so-called furthest neighbor collaborative filtering –contrarily to the conventional nearest neighbor collaborative filtering. By using information from neighbors that a user disagrees with, these works show it is possible to improve diversity without substantially affecting accuracy. Yet, the degree to which the FN affect the accuracy-diversity trade-off remains little investigated. We shall leverage both the NNs and the FNs to provide an accuracy-diversity trade-off.

Working with the inner-working mechanisms or the hyperparameters of a single model to bias the output towards diversity remains definitively a choice. But a single model will often suffer to capture the complex relationships in highly-sparse RS data. This fourth category of works overcomes the latter by working with a mixture of models, also referred to as joint, hybrid, or ensemble models. The latter have improved descriptive power to balance the accuracy-diversity tradeoff at expenses of a higher complexity, which in order of magnitude remains the same. The work in [96] works with a joint collaborative filtering algorithm that leverages the influence of both similar and dissimilar users. To compute the dissimilarity, it counts the items two users have consumed individually but not jointly. The predicted rating of the similar and dissimilar users are merged into a final score and the influence of each group is controlled by a scalar. In our view, the way the dissimilarity is computed ignores that users may have consumed the same item but rated it differently. Also building dissimilarities based on non-consumed items ignores that a user may also like an item the other user has consumed individually. Our approach accounts for the rating when building the dissimilarity between users. The authors of [98] follow a similar strategy as [96] and mix a heat spreading with a random walk to provide an accuracy-diversity trade-off. A probabilistic model is further proposed in [42]. The latter considers the order in which items are consumed and proposes a joint model, which on one branch maximizes accuracy (e.g., precision) while on the other diversity (e.g., specification). Differently, we train the whole model jointly.

3.3. Graph Convolutional Filters for Recommendation

Recent studies showed the importance of graphs in recommender systems due to their ability of capturing information hidden in the data structure [20, 81, 94]. Graph-based recommender systems first build a graph to represent the data and then employ tools to tailor recommendation [78]. Advances in graph signal processing proved to be crucial for this task, by translating ratings into graph signals.

Linear Models. The first work to perform recommendation by means of graph signal processing tools is [11], which designs a RS via non-negative matrix factorization using graph TV [cf.(2.15)] as regularization parameter. This choice is proved to be more beneficial to accuracy compared to using Tikhonov regularization as in [15]. Later, graph convolutional filters entered the recommendation scene in [56]. The authors focus on similarity graphs and introduce *diffusion filtering*, where they consider signal propagation through the graph as a heat diffusion process. They show accuracy can be improved by applying diffusion filtering on similarity graphs prior to calculating cosine similarity. Authors in [37] continue with graph filters and show the NN collaborative filter is a graph convolutional filter of order one w.r.t. the NN graph adjacency matrix [73]. This work also showed higher order graph convolutional filters (therefore, higher-order neighbor information) improve rating prediction, as we discussed in Section 2.3. Also the work in [18] uses the same graph convolutions as [37] –although building from a different standpoint and named differently, from a technical perspective the two approaches are identical– but trains them for ranking rather than rating. These approaches are limited to linear transformations, neglecting nonlinear solutions which may capture complex relationships among entities involved.

Nonlinear Models. The authors of [61] extract spatial features from similarity graphs by means of graph convolutional filters. Then, they feed such features into a recurrent neural network to complete the UIM. Later, authors in [12] use a graph convolutional encoder followed by a bylinear decoder to predict recommended items in the form of labeled edges in the user-item bipartite graph. The user-item bipartite graph is also the starting point in [83], where the authors deployed a GCNN with filters of order one on an augmented graph comprising the user-item bipartite interaction graph and also user-user and item-item proximal graphs. The work in [92] learns also from the user-item interaction bipartite graph through an order one GCNN but the propagation rule is augmented to promote message passing from similar items. Authors in [95] combine random walks with graph convolutions and develop a scalable framework to billion node graphs.

Knowledge graph is another graph type widely employed in RS to boost the recommendation with additional information [89]. The first to use GCNN methods to leverage information from the knowledge graph is [88], which uses a GCNN to generate high order embeddings to capture structural proximity among entities. This idea is further analyzed in [88], which regularizes the loss function by a smoothness prior to enforce similar user scores in adjacent items. Authors in [91] add an attention mechanism to the pipeline, learning embeddings that are aware of the importance of neighbors.

All in all, these works show the large potential of graph convolutions and GCNNs to change the RS landscape. However, all approaches mainly focus on accuracy while ignoring diversity along with the respective trade-off. This thesis, instead, shows how to use graph convolutions and GCNNs for establishing an accuracy-diversity trade-off in both a rating and a ranking setup.

3.4. DISCUSSION

Although difficult to sum up all the literature in RS, diversity and graph convolutions, we highlighted the most important works to help the reader understanding the state-of-the-art. We started in Section 3.1 with an overview of diversity in RS and its importance in the design phase; in Section 3.2, we discussed the accuracy-diversity trade-off intrinsic of RS and some of relevant literature offering a better balance; finally in Section 3.3, we presented the works proposing linear and nonlinear solutions with graph convolutional filters in RS.

The goal of this thesis is to build and analyze graph convolutional methods that explore both strong *positive* and *negative* correlations between users and items to allow trading between recommendation accuracy and diversity. Our rationale is that, if graph convolutions are employed to capture hidden relationships between entities connected based on their similarity, their learning will be skewed towards an accuracy-oriented map, ultimately, undermining diversity. Our approach consists of deploying two different graph convolutions over two different graphs: one graph built from the strong similarities between entities and another graph build from strong dissimilarities between entities. The similarity-based graph will capture the influence of similar entities to build recommendations, while the dissimilarity graph will capture the influence of dissimilar entities to include that information that cannot be described by similarity connections. Together, they increase the descriptive power of graph convolutional methods to represent the data. However, the information coming from dissimilar connections needs to be accounted properly to affect recommendation accuracy as little as possible. We tackle this issue in two ways: first, we balance the information provided by the two graphs through a tuneable parameters, which serves also as our handle to establish an accuracy-diversity trade-off; second, we train jointly the ensemble of graph convolutional methods deployed over the two graphs w.r.t. an accuracy-oriented metric. This latter aspect is critical to learn representations that enrich recommendation accuracy with information coming from dissimilar connections.

4 METHODS

In this chapter we present our proposed approach, where we perform rating prediction employing a similarity graph and a dissimilarity graph. Contrarily to accuracy-centered systems, we learn parameters via joint optimization of two graph convolutional filters, one for each graph. With this approach we embed diversity already in learning. We divide this chapter as follows: Section 4.1 introduces the notion of dissimilarity graph as diversity source; then, we discuss joint rating optimization of filters in a rating oriented and a ranking oriented fashion in Sections 4.2 and 4.3, respectively; in Section 4.4 we show the bandlimitedness of the graph convolutional filters we use in linear and nonlinear models; finally, Section 4.5 concludes the chapter by highlighting the key aspects of our approach.

4.1. LEVERAGING NEGATIVE CORRELATIONS

We work with a NN similarity-graph $\mathcal{G}_\mathrm{s} = \{\mathcal{V}, \mathcal{E}_\mathrm{s}\}$ [cf. Sec. 2.3.1; Fig. 2.7] and a FN dissimilarity-graph $\mathcal{G}_\mathrm{d} = \{\mathcal{V}, \mathcal{E}_\mathrm{d}\}$. The dissimilarity graph is built by following the opposite principles of NNs, i.e., connecting each entity to its top-$\bar{n}$ most negatively related ones. To illustrate the latter, consider $\mathbf{C}$ captures item-item correlations while vector $\mathbf{x}_u$ contains the ratings of user u to all items. The user-specific dissimilarity graph has the adjacency matrix $\bar{\mathbf{C}}_u$ obtained from $\mathbf{C}$ by:

1. Removing any edge starting from an item not rated by user u.

2. Keeping for each item i the $\bar{n}$ most *negatively* connected items.

3. Normalizing the resulting matrix to make $\bar{\mathbf{C}}_u$ right stochastic.

In other words, defining $\overline{\mathcal{N}}_{iu}$ as the set containing the $\bar{n}$ most dissimilar items to i rated by user u, entry (i, j) of $\bar{\mathbf{C}}_u$ is

$$[\bar{\mathbf{C}}_u]_{ij} = \begin{cases} C_{ij} / \sum_{j' \in \overline{\mathcal{N}}_{iu}} C_{ij'} & \text{if } j \in \overline{\mathcal{N}}_{iu} \\ 0 & \text{if } j \notin \overline{\mathcal{N}}_{iu} \end{cases} . \tag{4.1}$$

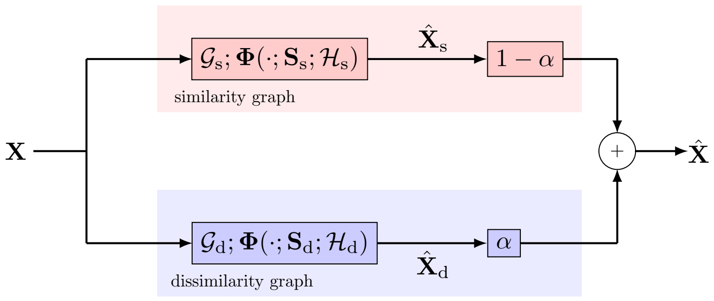

$\mathcal{G}_s; \mathbf{\Phi}(\cdot; \mathbf{S}_s; \mathcal{H}_s)$
$\hat{\mathbf{X}}_s$
$1-\alpha$
similarity graph
$\mathbf{X}$
$+$
$\hat{\mathbf{X}}$
$\mathcal{G}_d; \mathbf{\Phi}(\cdot; \mathbf{S}_d; \mathcal{H}_d)$
α
$\hat{\mathbf{X}}_d$
dissimilarity graph

Figure 4.1: Rating prediction with similar and dissimilar Graphs. We construct a NN graph $\mathscr{G}_s$ capturing similarities between entities; and a FN graph $\mathscr{G}_d$ capturing dissimilarities between entities. On each graph, we run a graph convolutional module $\mathbf{\Phi}(\cdot)$ with respective parameter set $\mathscr{H}_{(\cdot)}$ [cf. (2.10), (2.13)]. The estimated outputs are combined through a parameter α to obtain the final joint estimate $\hat{\mathbf{X}}$.

Table 4.1: Combinations between the similarity-graph $\mathscr{G}_s$ and the dissimilarity-graph $\mathscr{G}_d$.

Similar $\mathscr{G}_s$		Dissimilar $\mathscr{G}_d$	
		User	Item
	User	User-user (UU)	User-item (UI)
	Item	Item-user (IU)	Item-item (II)

The normalization step is identical to the NN approach and ensures a similar magnitude of signal shifting [cf.(2.7)] on both NN and FN graphs. This normalization implies the entries of $\bar{\mathbf{C}}_u$ are positive, i.e., a larger value indicates a stronger dissimilarity. In the considered datasets, positive correlations go up to 1.0 while the negative correlations down to -0.2.

On each graph $\mathscr{G}_s$ and $\mathscr{G}_d$ we have a convolutional module $\mathbf{\Phi}_s(\mathbf{x};\mathbf{S}_s;\mathscr{H}_s)$ and $\mathbf{\Phi}_d(\mathbf{x};\mathbf{S}_d;\mathscr{H}_d)$, outputting an estimate of the user-item matrix $\hat{\mathbf{X}}_s$ and $\hat{\mathbf{X}}_d$, respectively. We combine the two outputs in the joint estimate

$$\hat{\mathbf{X}} = (1-\alpha)\hat{\mathbf{X}}_s + \alpha\hat{\mathbf{X}}_d \tag{4.2}$$

where scalar $\alpha \in]0,1[$ balances the influence of the similar and dissimilar connections; see Figure 4.1. Each graph $\mathscr{G}_s$ or $\mathscr{G}_d$ can be a user or an item graph and the graph convolutional modules $\mathbf{\Phi}(\cdot)$ can be linear [cf. (2.10)] or nonlinear [cf. (2.13)]. This framework yields eights combinations to investigate the trade-off. We limit ourselves to situations where the graph convolutional modules are the same on both graphs and focus on the four combinations in Table 4.1. To ease exposition, we shall discuss the theoretical methods with the hybrid combination user NN graph (i.e., $\mathscr{G}_{s,u}$ with adjacency matrix $\mathbf{B}_i$ for item i) and item FN graph (i.e., $\mathscr{G}_{d,i}$ with adjacency matrix $\bar{\mathbf{C}}_u$ for user u). This setting implies we predict rating $\hat{X}_{ui}$ by learning, on one side, from the coupling $(\mathscr{G}_{s,u}, \mathbf{x}^i)$, and, on the other side, from the coupling $(\mathscr{G}_{d,i}, \mathbf{x}_u)$.

Joint models like the one we consider are popular beyond the RecSys literature. The works in [36, 77] consider two different shift operators of the same graph to model signal diffusion with graph convolutional filters [cf. (2.10)]. This strategy is subsequently extended to GCNNs in [22]. Instead, [19, 40] exploit different relationships between data to build GCNNs. The common motivation in all these works is that a model based on a single graph (often capturing similarities between nodes [58]) or a single shift operator is insufficient to represent the underlying relationships. Therefore, we argue a joint model capturing different interactions helps representing better the data. A model based only on NNs fails giving importance to items that differ from the main trend. FNs account for this information and aid diversity. However, the information from FNs should be accounted for properly during training to keep the accuracy at the desired level. We detail this aspect in the upcoming two sections.

4.2. LEARNING FOR RATING

In this section, we estimate the joint model parameters w.r.t. the mean squared error (MSE) criterion. Analyzing the MSE quantifies also the trade-off for all items in the dataset (unbiasing the results from the user preferences in the list)[1]. The MSE also provides insights into the role played by the FNs. To this end, consider a training set of user-item pairs $\mathcal{T} = \{(u,i)\}$ for the available ratings in $\mathbf{X}$. Consider also the user-similarity graph $\mathcal{G}_{\mathrm{s,u}}$, the item-dissimilarity graph $\mathcal{G}_{\mathrm{d,i}}$, and their respective graph convolutions $\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_{\mathrm{s}})$ and $\mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_{\mathrm{d}})$. We estimate parameters $\mathcal{H}_{\mathrm{s}}$ and $\mathcal{H}_{\mathrm{d}}$ by solving the regularized problem

$$\begin{aligned} \underset{\mathcal{H}_{\mathrm{s}},\mathcal{H}_{\mathrm{d}}}{\text{minimize}} \quad & \frac{1}{\mu}\mathrm{MSE}_{(u,i)\in\mathcal{T}}\Big(\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_{\mathrm{s}}) + \mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_{\mathrm{d}});\mathbf{X}_{\mathcal{T}}\Big) + \frac{1}{2}\Big(\frac{\|\mathcal{H}_{\mathrm{s}}\|_2^2}{1-\alpha} + \frac{\|\mathcal{H}_{\mathrm{d}}\|_2^2}{\alpha}\Big) \\ \text{subject to} \quad & 0<\alpha<1 \end{aligned} \tag{4.3}$$

where $\mathrm{MSE}_{(u,i)\in\mathcal{T}}(\cdot;\cdot)$ measures the fitting error w.r.t. the available ratings $\mathbf{X}_{\mathcal{T}}$, while the second term acts as an accuracy-diversity regularizer[2].

Scalar α controls the information flow from the NN and the FN graph. For $\alpha \to 0$, we have $\|\mathcal{H}_{\mathrm{d}}\|_2^2/\alpha \gg \|\mathcal{H}_{\mathrm{s}}\|_2^2/(1-\alpha)$, therefore, problem (4.3) forces parameters $\mathcal{H}_{\mathrm{d}}$ to a smaller norm rather than using them to fit the data. Hence, this setting mainly leverages information from the similar graph $\mathbf{\Phi}_{\mathrm{s}}(\cdot)$, ultimately, reducing the ensemble to a single NN graph convolutional model. For $\alpha \to 1$, we have the opposite case $\|\mathcal{H}_{\mathrm{s}}\|_2^2/(1-\alpha) \gg \|\mathcal{H}_{\mathrm{d}}\|_2^2/\alpha$, which implies the information from the similar graph plays little role in fitting since parameters $\mathcal{H}_{\mathrm{s}}$ are forced towards zero. Hence, problem (4.3) mainly exploits information from FNs to reduce the MSE. Intermediate values of α closer to zero than to one lead to models where most information is leveraged from the NNs to keep the MSE low, while some information is taken from FNs to add diversity. We refer to α as the trade-off parameter. Scalar μ balances the fitting error with the overall regularization and allows generalizing the model to unseen data.

4.2.1. GRAPH CONVOLUTIONAL FILTER

Recall the graph convolutional filter in (2.10) and consider graphs $\mathcal{G}_{\mathrm{s,u}}$ and $\mathcal{G}_{\mathrm{d,i}}$ can have different number of nodes. To account for this technicality in the design phase, we

[1]We shall see that often methods trained with the MSE perform better in the list w.r.t. trained for ranking.

[2]In (4.3), we allowed ourselves a slight abuse of notation and indicated with $\|\mathcal{H}_{(\cdot)}\|_2^2$ the ℓ_2−norm squared of the vector containing the coefficients in set $\mathcal{H}_{(\cdot)}$.

first transform the filters into a more manageable form. The filter output on the user-similarity graph $\mathbf{B}_i$ can be written as

$$\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}^i;\mathbf{B}_i;\mathbf{h}_{\mathrm{s}}) = \sum_{k=0}^{K} h_{s,k}\mathbf{B}_i^k\mathbf{x}^i := \mathbf{B}_{\mathrm{s},i}\mathbf{h}_{\mathrm{s}} \tag{4.4}$$

where the $U \times (K+1)$ matrix $\mathbf{B}_{\mathrm{s},i} = [\mathbf{B}_i^0\mathbf{x}^i,\dots,\mathbf{B}_i^{K_s}\mathbf{x}^i]$ contains the shifted ratings of item i over graph $\mathbf{B}_i$ and vector $\mathbf{h}_{\mathrm{s}} = [h_{s,0},\dots,h_{s,K_s}]^\top$ the parameters. The uth row of $\mathbf{B}_{\mathrm{s},i}$ is the $1\times(K+1)$ vector $[\mathbf{B}_{\mathrm{s},i}]_{u:}$ containing the shifted ratings of user u for item i. We then stack the $|\mathcal{T}|$ row vectors $[\mathbf{B}_{\mathrm{s},i}]_{u:}$ for all pairs $(u,i)\in\mathcal{T}$ in

$$\mathbf{M}_{\mathrm{s}} = \big[\dots;[\mathbf{B}_{\mathrm{s},i}]_{u:};[\mathbf{B}_{\mathrm{s},j}]_{u:};\dots;[\mathbf{B}_{\mathrm{s},k}]_{v:};[\mathbf{B}_{\mathrm{s},l}]_{v:};\dots\big] \in \mathbb{R}^{|\mathcal{T}|\times(K+1)}.$$

The τth row of $\mathbf{M}_{\mathrm{s}}$ corresponds to the τth (u,i) tuple. Denoting by $\mathbf{x}_{\mathcal{T}} = \mathrm{vec}(\mathbf{X}_{\mathcal{T}})$ the $|\mathcal{T}|\times 1$ vector of available ratings, we can write the filter output for all training samples as $\hat{\mathbf{x}}_{\mathrm{s},\mathcal{T}} = \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}}$. Likewise, we can write the filter output over the item-dissimilarity graph $\bar{\mathbf{C}}_u$ as

$$\mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathbf{h}_{\mathrm{d}}) = \sum_{k=0}^{K} h_{d,k}\bar{\mathbf{C}}_u^k\mathbf{x}_u := \bar{\mathbf{C}}_{\mathrm{d},u}\mathbf{h}_{\mathrm{d}} \tag{4.5}$$

where matrix $\bar{\mathbf{C}}_{\mathrm{d},u} = [\bar{\mathbf{C}}_u^0\mathbf{x}_u,\dots,\bar{\mathbf{C}}_u^{K_d}\mathbf{x}_u] \in \mathbb{R}^{I\times(K+1)}$ collects the shifted ratings of user u w.r.t. graph $\bar{\mathbf{C}}_u$ and vector $\mathbf{h}_{\mathrm{d}} = [h_{d,0},\dots,h_{d,K}]^\top$ the filter parameters. Then, we construct the $|\mathcal{T}|\times(K+1)$ matrix $\mathbf{M}_{\mathrm{d}}$ by collecting the rows $[\bar{\mathbf{C}}_{\mathrm{d},u}]_{i:}$ for all $(u,i)\in\mathcal{T}$ so that to write $\hat{\mathbf{x}}_{\mathrm{d},\mathcal{T}} = \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}$.

With these in place, the design problem (4.3) particularizes to

$$\begin{aligned} \underset{\mathbf{h}_{\mathrm{s}},\mathbf{h}_{\mathrm{d}}}{\text{minimize}} \quad & \frac{1}{2\mu}\left\|\mathbf{x}_{\mathcal{T}} - \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}} - \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}\right\|_2^2 + \frac{1}{2}\left(\frac{\|\mathbf{h}_{\mathrm{s}}\|_2^2}{1-\alpha} + \frac{\|\mathbf{h}_{\mathrm{d}}\|_2^2}{\alpha}\right). \\ \text{subject to} \quad & 0<\alpha<1 \end{aligned} \tag{4.6}$$

which is a regularized-least squares problem in the filter coefficients $\mathbf{h}_{\mathrm{s}}$ and $\mathbf{h}_{\mathrm{d}}$. The closed-form solution for (4.6) can be found by setting the gradient to zero, i.e.,

$$-\frac{1}{\mu}\mathbf{M}_{\mathrm{s}}^\top\big(\mathbf{x}_{\mathcal{T}} - \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}} - \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}\big) + \frac{1}{1-\alpha}\mathbf{h}_{\mathrm{s}} = \mathbf{0} \tag{4.7a}$$

$$-\frac{1}{\mu}\mathbf{M}_{\mathrm{d}}^\top\big(\mathbf{x}_{\mathcal{T}} - \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}} - \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}\big) + \frac{1}{\alpha}\mathbf{h}_{\mathrm{d}} = \mathbf{0} \tag{4.7b}$$

or equivalently solving the linear system of equations

$$\frac{1}{\mu}\begin{bmatrix}\mathbf{M}_{\mathrm{s}}^\top\mathbf{x}_{\mathcal{T}}\\ \mathbf{M}_{\mathrm{d}}^\top\mathbf{x}_{\mathcal{T}}\end{bmatrix} = \begin{bmatrix}\mathbf{M}_{\mathrm{s}}^\top\mathbf{M}_{\mathrm{s}} - \frac{1}{1-\alpha}\mathbf{I} & \mathbf{M}_{\mathrm{s}}^\top\mathbf{M}_{\mathrm{d}}\\ \mathbf{M}_{\mathrm{d}}^\top\mathbf{M}_{\mathrm{s}} & \mathbf{M}_{\mathrm{d}}^\top\mathbf{M}_{\mathrm{d}} - \frac{1}{\alpha}\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{h}_{\mathrm{s}}\\ \mathbf{h}_{\mathrm{d}}\end{bmatrix}. \tag{4.8}$$

If the matrix inversion in (4.8) is ill-conditioned, we can always solve (4.6) with of-the-shelf iterative methods. The above procedure leads to an optimal balance between the information coming from the NNs and the FNs.

4.2.2. GRAPH CONVOLUTIONAL NEURAL NETWORK

We now consider models $\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathscr{H}_{\rm s})$ and $\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_{\rm d})$ are GCNNs running respectively over graphs $\mathbf{B}_i$ and $\bar{\mathbf{C}}_u$. Particularizing (4.3) to this setting implies solving

$$\begin{aligned}&\underset{\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}}{\text{minimize}} && \frac{1}{2\mu}\sum_{(u,i)\in\mathscr{T}}\left|X_{ui}-\left[\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathscr{H}_{\rm s})\right]_u-\left[\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_{\rm d})\right]_i\right|^2+\frac{1}{2}\left(\frac{\|\mathscr{H}_{\rm s}\|_2^2}{1-\alpha}+\frac{\|\mathscr{H}_{\rm d}\|_2^2}{\alpha}\right)\\&\text{subject to} && 0<\alpha<1\end{aligned} \tag{4.9}$$

where $[\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathscr{H}_{\rm s})]_u$ is the user-similarity GCNN output for user u and $[\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_{\rm d})]_i$ is the item-dissimilarity GCNN output for item i. Problem (4.9) preserves the trade-offs of the general version (4.3), but it is non-convex and little can be said about its global optimality. However, because of the compositional form of the GCNN, we can estimate paramters $\mathscr{H}_{\rm s}$ and $\mathscr{H}_{\rm d}$ via standard backpropagation since the graph convolutional filters are linear operators in the respective parameters [30]. The following remark is in order.

Remark 1 In (4.9), we considered the accuracy-diversity parameter α only in the regularizer and not also in the fitting part as in (4.2). We found that including the latter to the MSE term leads to a more conservative solution towards diversity. We have consistently seen that keeping α only in the regularizer allows for a better trade-off. Furthermore, the regulariser in (4.9) does not need be rational in α, but can be in any form as long as it balances the NNs with the FNs. An alternative is $\Omega(\mathscr{H}_{\rm s};\mathscr{H}_{\rm d};\alpha)=\frac{1}{2}\big(\alpha\|\mathscr{H}_{\rm s}\|_2^2+(1-\alpha)\|\mathscr{H}_{\rm d}\|_2^2\big)$.

4.3. LEARNING FOR RANKING

This section designs the joint model for ranking. We considered the Bayesian personalized ranking (BPR), which is a state-of-the-art learn-to-ranking framework [67]. BPR considers the rating difference a user u has given to two items i and j. Let symbol $i>_u j$ indicate user u rated item i more than item j and augment the training set as $\mathscr{T}\subseteq\mathscr{U}\times\mathscr{I}\times\mathscr{I}$ to contain triplets of the form $\mathscr{T}=\{(u,i,j)|i>_u j\}$. For each available tuple (u,i) we created four triplets $\{(u,i,j)\}_j$ such that $X_{ui}>X_{uj}$ following [67]. Subsequently, the estimated ratings for tuples (u,i) and (u,j) are respectively

$$\begin{aligned}\hat{X}_{ui}\big(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}\big)&=\left[\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathscr{H}_{\rm s})\right]_u+\left[\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_{\rm d})\right]_i\\\hat{X}_{uj}\big(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}\big)&=\left[\mathbf{\Phi}_{\rm s}(\mathbf{x}^j;\mathbf{B}_j;\mathscr{H}_{\rm s})\right]_u+\left[\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_{\rm d})\right]_j\end{aligned} \tag{4.10}$$

and the utility function is

$$\hat{X}_{uij}\big(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}\big)=\hat{X}_{ui}\big(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}\big)-\hat{X}_{uj}\big(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}\big) \tag{4.11}$$

which expresses the rating difference as a parametric relationship between user u, item i, and item j. The utility function is used to estimate parameters $\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}$ by maximizing the likelihood

$$p(i>_u j|\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}):=\sigma\Big(\hat{X}_{uij}\big(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d}\big)\Big)=\Big(1+e^{-\hat{X}_{uij}(\mathscr{H}_{\rm s},\mathscr{H}_{\rm d})}\Big)^{-1} \tag{4.12}$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the logistic sigmoid function [67]. By applying the natural logarithm (monotonic increasing) to (4.12) and regularizing it, we can estimate the joint convolutional model parameters by solving the regularized optimization problem

$$\begin{array}{ll} \underset{\mathscr{H}_\mathrm{s},\mathscr{H}_\mathrm{d}}{\text{minimize}} & -\frac{1}{\mu}\sum_{(u,i,j)\in\mathscr{T}} \ln\sigma\big(\hat{X}_{uij}\mathscr{H}_\mathrm{s},\mathscr{H}_\mathrm{d}\big) + \alpha\|\mathscr{H}_\mathrm{s}\|_2^2 + (1-\alpha)\|\mathscr{H}_\mathrm{d}\|_2^2 \\ \text{subject to} & 0<\alpha<1 \end{array}. \tag{4.13}$$

Differently from (4.2), the regularizer in (4.13) is linear in α. We opted for this choice because the linear was more robust to μ. Nevertheless, the regulariser in (4.13) respects the same trend as that in (4.2): for $\alpha \to 0$, NNs are mainly used for fitting since $\alpha\|\mathscr{H}_\mathrm{s}\|_2^2 \to 0$; vice-versa, for $\alpha \to 1$ the FNs are mainly used for fitting since $(1-\alpha)\|\mathscr{H}_\mathrm{d}\|_2^2 \to 0$.

4.3.1. GRAPH CONVOLUTIONAL FILTER

Particularizing the convolutional models to filters [cf. (2.10)], (4.10) becomes

$$\begin{aligned} \hat{X}_{ui}\big(\mathscr{H}_\mathrm{s},\mathscr{H}_\mathrm{d}\big) &= \big[\mathbf{B}_{\mathrm{s},i}\big]_{u:}\mathbf{h}_\mathrm{s} + \big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_{i:}\mathbf{h}_\mathrm{d} \\ \hat{X}_{uj}\big(\mathscr{H}_\mathrm{s},\mathscr{H}_\mathrm{d}\big) &= \big[\mathbf{B}_{\mathrm{s},j}\big]_{u:}\mathbf{h}_\mathrm{s} + \big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_{j:}\mathbf{h}_\mathrm{d} \end{aligned} \tag{4.14}$$

where $\big[\mathbf{B}_{\mathrm{s},i}\big]_{u:}$ is the uth row of the similar user-NN graph matrix $\mathbf{B}_{\mathrm{s},i}$ [cf. (4.4)] and $\big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_{i:}$ is the ith row of the dissimilar item-FN graph matrix $\bar{\mathbf{C}}_{\mathrm{d},u}$ [cf. (4.5)]. Substituting (4.14) into (4.13) leads to

$$\begin{array}{ll} \underset{\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}}{\text{minimize}} & -\frac{1}{\mu}\sum_{(u,i,j)\in\mathscr{T}} \ln\sigma\big(\hat{X}_{uij}\big(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}\big) + \big(\alpha\|\mathbf{h}_\mathrm{s}\|_2^2 + (1-\alpha)\|\mathbf{h}_\mathrm{d}\|_2^2\big) \\ \text{subject to} & \hat{X}_{uij}\big(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}\big) = \big(\big[\mathbf{B}_{\mathrm{s},i}\big]_{u:}\mathbf{h}_\mathrm{s} + \big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_{i:}\mathbf{h}_\mathrm{d}\big) - \big(\big[\mathbf{B}_{\mathrm{s},j}\big]_{u:}\mathbf{h}_\mathrm{s} + \big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_{j:}\mathbf{h}_\mathrm{d}\big) \\ & 0<\alpha<1 \end{array}. \tag{4.15}$$

Function $-\ln\sigma(\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}))$ is convex since it involves a log-sum-exp of an affine function [13]. Consequently, problem (4.15) is convex in $\mathbf{h}_\mathrm{s}$ and $\mathbf{h}_\mathrm{d}$. Convexity guarantees we can find a minimizer for (4.15) but not a closed-form solution. In fact, finding an analytical solution for logistic fitting problems is notoriously difficult except for particular instances [52]. However, we can get the optimal parameters for (4.15) through the stochastic gradient descent updates

$$\mathbf{h}_\mathrm{s} \leftarrow \mathbf{h}_\mathrm{s} + \frac{\gamma}{\mu}\Big[e^{-\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d})}\sigma\big(\hat{X}_{uij}\big(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}\big)\big)\big(\big[\mathbf{B}_{\mathrm{s},i}\big]_u^\top - \big[\mathbf{B}_{\mathrm{s},j}\big]_u^\top\big) - 2\alpha\mathbf{h}_\mathrm{s}\Big] \tag{4.16a}$$

$$\mathbf{h}_\mathrm{d} \leftarrow \mathbf{h}_\mathrm{d} + \frac{\gamma}{\mu}\Big[e^{-\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d})}\sigma\big(\hat{X}_{uij}\big(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}\big)\big)\big(\big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_i^\top - \big[\bar{\mathbf{C}}_{\mathrm{d},u}\big]_j^\top\big) - 2(1-\alpha)\mathbf{h}_\mathrm{d}\Big] \tag{4.16b}$$

where γ is the stepsize. These optimal parameters guarantee the best learning-to-rank solution for any balance between the NNs and FNs (α) and between fitting and generalization (μ).

4.3.2. GRAPH CONVOLUTIONAL NEURAL NETWORK

When $\mathbf{\Phi}_\mathrm{s}(\mathbf{x}^i;\mathbf{B}_i;\mathscr{H}_\mathrm{s})$ and $\mathbf{\Phi}_\mathrm{d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_\mathrm{d})$ are GCNNs, the BPR optimization problem is that in (4.13). Because of the nonlinearity, it is difficult to establish if a global minimum exists and we should seek for a satisfactory local minima. Since cost (4.13) is differentiable w.r.t. $\mathscr{H}_\mathrm{s}$ and $\mathscr{H}_\mathrm{d}$, we can achieve this local minima through conventional backpropagation.

Either estimated for rating or ranking, the coefficients of the joint model dictate the filter behavior (either directly or within the GCNN layers) on the NN and FN graphs. Besides analyzing the filter behavior in the node domain (as multi-hop rating aggregation) and in the respective cost functions (as accuracy-diversity trade-off), we can also get insight on the trade-off by analyzing the graph convolutional modules in the graph spectral domain [65]. We discuss this aspect next.

4.4. SPECTRAL EXPLANATION

In this section we use the concepts discussed in Section 2.2.4 to study the spectral behavior of the graph convolutional filters in our model. We start with plain graph convolutional filters [cf. (2.10)] and show they act as bandstop filters [37]. Consequently, we run the same analysis on filters employed in GCNNs, and highlight they follow the same spectral behavior.

4.4.1. GRAPH CONVOLUTIONAL FILTERS

Recall the joint model with a user-similarity filter $\mathbf{H}(\mathbf{B}_i) = \sum_{k=0}^{K} h_{s,k}\mathbf{B}_i^k\mathbf{x}^i$ [cf. (4.4)] and an item-dissimilarity filter $\mathbf{H}(\bar{\mathbf{C}}_u) = \sum_{k=0}^{K} h_{d,k}\bar{\mathbf{C}}_u^k\mathbf{x}_u$ [cf. (4.5)]. By substituting the respective eigendecompositions $\mathbf{B}_i = \mathbf{U}_{\mathrm{s},i}\mathbf{\Lambda}_{\mathrm{s},i}\mathbf{U}_{\mathrm{s},i}^{-1}$ and $\bar{\mathbf{C}}_u = \bar{\mathbf{U}}_{\mathrm{d},u}\bar{\mathbf{\Lambda}}_{\mathrm{d},u}\bar{\mathbf{U}}_{\mathrm{d},u}^{-1}$ we can write the estimates in the graph frequency domain as

$$\hat{\tilde{\mathbf{x}}}^i = \sum_{k=0}^{K} h_{\mathrm{s},k}\mathbf{\Lambda}_{\mathrm{s},i}^k\tilde{\mathbf{x}}^i \quad \text{and} \quad \hat{\tilde{\mathbf{x}}}_u = \sum_{k=0}^{K} h_{\mathrm{d},k}\bar{\mathbf{\Lambda}}_{\mathrm{d},u}^k\tilde{\mathbf{x}}_u. \tag{4.17}$$

In (4.17), $\hat{\tilde{\mathbf{x}}}^i$ and $\hat{\tilde{\mathbf{x}}}_u$ are the GFT of the filter output on the similar and dissimilar graphs, respectively; likewise, $\mathbf{H}(\mathbf{\Lambda}_{\mathrm{s},i}) := \sum_{k=0}^{K} h_{\mathrm{s},k}\mathbf{\Lambda}_{\mathrm{s},i}^k$ and $\mathbf{H}(\bar{\mathbf{\Lambda}}_{\mathrm{d},u}) := \sum_{k=0}^{K} h_{\mathrm{d},k}\bar{\mathbf{\Lambda}}_{\mathrm{d},u}^k$ are the filter responses of $\mathbf{H}(\mathbf{B}_i)$ and $\mathbf{H}(\bar{\mathbf{C}}_u)$. To estimate the responses, we first get the filter coefficients from (4.6) (rating design) or (4.13) (ranking design) and order the eigenvalues $\lambda_{n,i}$ (resp. $\lambda_{n,u}$) of each $\mathbf{B}_i$ (resp. $\bar{\mathbf{C}}_u$) as per the total variation in (2.15). Subsequently, for each $\mathbf{B}_i$ (resp. $\bar{\mathbf{C}}_u$) we record the transfer functions $\{\mathbf{H}(\mathbf{\Lambda}_{\mathrm{s},i})\}_i$ (resp. $\{\mathbf{H}(\bar{\mathbf{\Lambda}}_{\mathrm{d},u})\}_u$) and average them across all items I (resp. users U) to get a single transfer functions over the user-similarity graph (resp. item-dissimilarity graph). The transfer functions are shown in Figure 4.2 for different values of α.

In all cases, we observe a band-stop behavior since more than 90% of the middle graph frequencies are zero. The latter corroborates the behavior of the vanilla NN graph filter and that of the graph convolutional filters operating on the similarity graph [37]. This is because all models operate over NN-based (or FN-based) graphs which have a clustered structure; hence, they induce a low-rank representation. Another behavior inherited from NN/FN graphs is that filters preserve the extreme low and high graph frequencies. Low graph frequencies are signals with a small total variation, while high graph frequencies are signals with a high total variation.

- *In the user-similarity graph,* low graph frequencies represent signals where similar

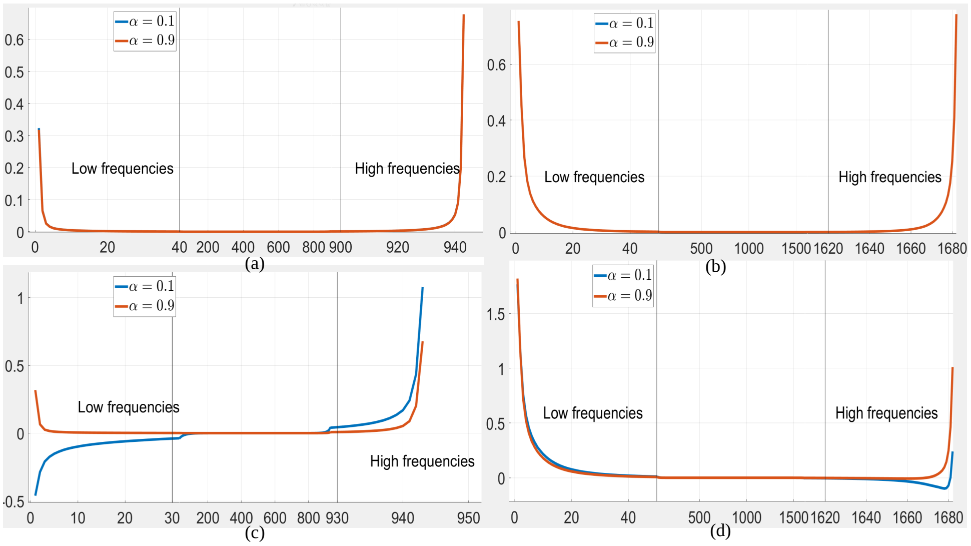


Figure 4.2: Frequency responses of the graph convolutional filters over a user-similarity and an item-dissimilarity graph. The horizontal axis is the graph frequency index, while the vertical axis is the estimated frequency response. (Top) Filters designed w.r.t. the mean squared error criterion in (4.6). (Bottom) Filters designed w.r.t. the Bayesian personalized ranking criterion in (4.13). (a-c) Frequency response of the user-similarity graph filter $\mathbf{H}(\mathbf{\Lambda}_{s,i})$. (b-d) Frequency response of the item-dissimilarity graph filter $\mathbf{H}(\bar{\mathbf{\Lambda}}_{d,u})$.

users tend to give similar ratings. This part provides the global trend of preferences among similar users that is leveraged to predict ratings. The high graph frequencies represent discordant ratings between similar users for a particular item and can be seen as a primitive source for diversity.

- *In the item-dissimilarity graph,* the spectral behavior is the same but the implications are different. Low frequencies represent ratings with small difference in *dissimilar* neighboring items; implying, a user u gave similar ratings to items that are dissimilar between them. These low frequencies may also be because users rate negatively a subset a dissimilar items and positively another subset of dissimilar items. The high pass components represent ratings changing significantly between neighboring dissimilar items; e.g., one of the two dissimilar items sharing an edge is rated positively while the other negatively. This part will in turn contribute towards keeping high the recommendation accuracy while relying on strongly negative correlations between items.

These insights show the joint linear model eliminates irrelevant features (band-stop behavior), smoothes out positive and negative ratings (low frequencies), and preserves discriminative features to aid diversity (high frequencies). This phenomenon is observed for different values of α (importance on the similarity vs. dissimilarity graph) and design criteria (MSE [cf. (4.6)] vs. BPR [cf. (4.15)]). The frequency response changes less with α in the MSE (lines differ by 10^{-3}) than in BPR design. This might be because the MSE focuses on the average rating prediction error for all items (preferred or not), while the

BPR prioritises a subset of most preferred items. In BPR, we also observed a stronger band-stop behavior for $\alpha \rightarrow 1$ meaning the joint model focuses even more on extreme frequencies to predict ratings. This suggests the model relies on the average trend on both graphs (lower frequencies) and focuses explicitly on highly dissimilar values in adjacent nodes (higher frequencies).

We can also understand the joint model behavior by inspecting parameters $\mathbf{h}_\mathrm{s}$ and $\mathbf{h}_\mathrm{d}$. For rating, the parameters vary little with α (e.g., 10^{-2}), while there is a more insightful behavior in ranking design which we discuss next. For the similar filters, we have $\mathbf{h}_{\mathrm{s},\alpha=0.1} = [0, 1.867, -0.371, -0.288]^\top$ and $\mathbf{h}_{\mathrm{s},\alpha=0.9} = [0, 10.601, -0.012, 0.065]^\top$. This indicates that the more information we force from the dissimilar FN into the joint model (i.e., $\alpha = 0.9$), the more the latter relies on the one-hop similar NN; i.e., the gap in the coefficients increases by one order of magnitude. The latter is also reflected in the wider band-stop filter behavior. For the dissimilar filters, we have $\mathbf{h}_{\mathrm{d},\alpha=0.1} = [0, -0.849, 1.105]^\top$ and $\mathbf{h}_{\mathrm{d},\alpha=0.9} = [0, -0.474, 1.549]^\top$. The information from the two-hop FN neighbors has a double importance compared with the one-hop FN neighbors. These weights are because we design the coefficient w.r.t an accuracy-oriented metric for a few relevant items. That is, in $\mathscr{G}_{\mathrm{d},i}$ the one-hop FN neighbors i and j are highly dissimilar while the two-hop FN neighbors i and k are less dissimilar since they do not share an edge. Consequently, the two-hop FN neighbor k brings in less dissimilar information than the one-hop FN neighbor j, which aids ranking accuracy. At the same time, the information from the two-hop FN neighbors is more diverse than the information obtained from the two-hop NNs. The joint model compensates the degraded accuracy coming from the FN by increasing the importance of its one-hop NN.

4.4.2. GRAPH CONVOLUTIONAL NEURAL NETWORKS

We now analyze the graph frequency response of the convolutional filters in the joint GCNN model [cf.(2.12)]. Figure 4.3 illustrates the latter for a one-layer GCNN with $F = 2$ filters over each graph. We observe again the strong band-stop behavior. In the NN graph, the stopped band is narrower than in the FN graph, and it is narrower if the GCNN is learned w.r.t. rating than ranking. The band-stop behavior and the increased focus on the extreme low and high graph frequencies suggest the GCNN leverages the information on the similar and dissimilar graph in a similar way as the linear counterpart. We refer to the previous section to avoid repetition.

4.5. CONCLUSION

This chapter introduced our proposed approach to tackle the accuracy-diversity dilemma via graph convolutions. In Section 4.1 we described the dissimilarity graph and its role in recommendation; in Sections 4.2 and 4.3 we discussed the joint optimization problem from a rating and ranking perspective, respectively, in linear and nonlinear scenarios. We performed in-depth analysis of optimality in linear optimization in both cases. Finally, in Section 4.4, we studied the model in the graph-frequency domain, with insights about the recommendation mechanisms.

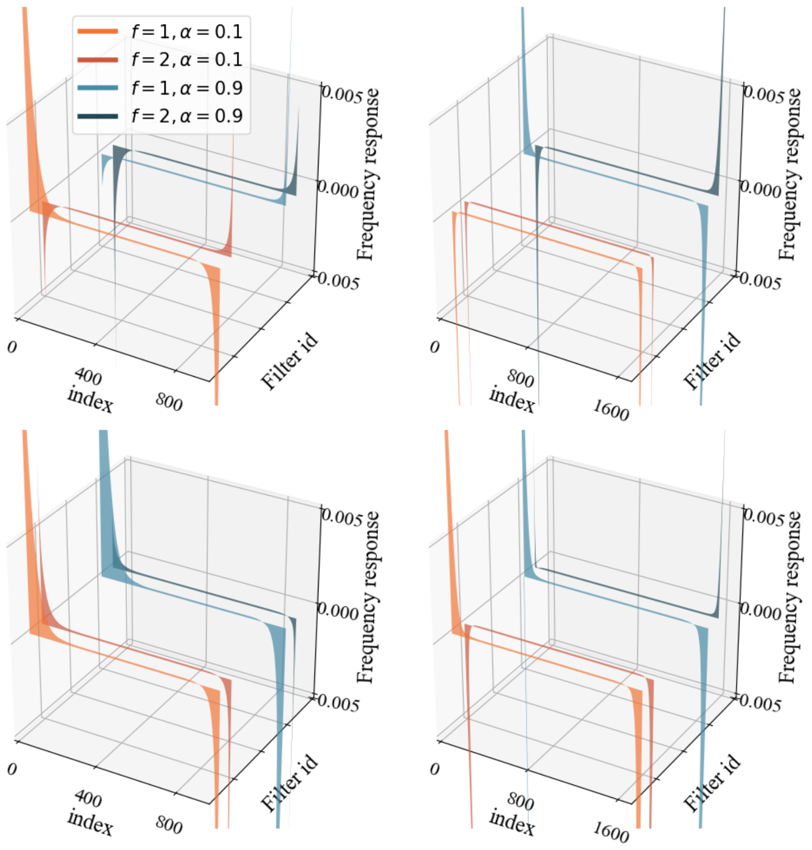


Figure 4.3: Frequency responses of the bank of $F = 2$ graph convolutional filters of a one layer GCNN [cf. (2.11)] run over a user similarity graph and an item dissimilarity graph. The z-axis is the frequency response which is cropped to improved visibility. The other two axis are the graph frequency index and the filter number. (Top) Filters designed w.r.t. the mean squared error criterion in (4.6). (Bottom) Filters designed w.r.t. the Bayesian personalized ranking criterion in (4.13). (Left) Filters on the user similarity graph. (Right) Filters on the item dissimilarity graph.

5 Numerical Experiments

This chapter presents the numerical experiments we carried out for our research. Section 5.1 discusses the experimental setting, with the metrics and baseline methods employed; Section 5.2 studies the optimal parameter for the similarity scenario; Section 5.3 presents the results of our model against state-of-the-art accuracy-oriented algorithms and methods proposing a different accuracy-diversity trade-off; Section 5.4 concludes the chapter.

5.1. Experimental Setting

We test our method on three datasets (MovieLens100k [33], Douban [55] and Flixster [41]) following the same experimental settings as in [61]: that is, for MovieLens100k we use 80% of the ratings as training and the rest for testing; for the other two datasets we use a submatrix of 3000 users and 3000 items, from which we consider 90% of the available ratings for training and the rest for testing. We carry out experiments extensively on Movielens100k to gain insights about our proposed method, while we use Douban and Flixster to corroborate the results on datasets with different sparsity. Table 5.1 shows detailed statistics about the datasets' structure.

We perform tests using the four combinations for $\mathscr{G}_s$ and $\mathscr{G}$ we showed in table 4.1. For each of the four combinations, we test the linear and the nonlinear model; finally, we learn parameters via ranking or rating optimization. Thus, we have 16 configurations of support, model and loss function to test on each dataset: e.g. UU linear optimized by rating, or IU nonlinear optimized by ranking. We considered a simple GCNN architecture composed of a single hidden layer with two parallel filters. We trained the GCNN using the ADAM optimizer with the default parameters [46] and sought different learning rates γ and fitting-regularizer parameter μ.

To avoid exponential growth of hyperparameter testing in different accuracy-diversity setups, we proceeded with the following rationale. First, we perform an extensive parameter analysis in the MovieLens100k data set, since this dataset is common in the two most similar graph convolutional works [37, 61]. We then use the remaining two data sets to

Table 5.1: Features of the considered datasets.

Data set	Users	Items	Ratings	Sparsity
MovieLens100k	943	1,682	100,000	6.3×10^{-2}
Douban	3,000	3,000	136,891	1.5×10^{-2}
Flixster	3,000	3,000	26,173	2.9×10^{-3}

corroborate the accuracy-diversity trade-off trends our joint model impose. Second, we chose the hyper-parameters of the similarity graph (number of nearest neighbor, filter order, and length of the recommendation list) from the linear graph convolutional filter optimized for rating [cf. (4.6)], which is the method proposed in [37]. Besides being a faster design method to seek for different parameters, this strategy allowed evaluating the accuracy-diversity trade-off by using information solely from the similarity graph. Finally, we kept fixed these parameters for the similarity graph and evaluated different combinations on the dissimilarity graph.

5.1.1. BASELINES

To further acquire knowledge about where in the spectrum of RS our method fits, we compare its performance against four methods which rely only on similarities to generate the recommendation:

- kNN: as shown in Sections (2.3.1) and (2.3.2), the kNN collaborative filtering can be seen as a graph convolutional filter operating on the similarity graph; we compare against user-based NN and item-based NN via graph convolution operating on $\mathcal{G}_s$.
- LR-MC [60]: low rank matrix completion fills the missing entries by seeking a low rank matrix which satisfies constraints imposed by the similarities between users and between items.
- MCGNN [61]: recommendation is seen as an iterative two steps procedure where similarity based features are extracted using a GNN and propagated using a recurrent neural network to complete the matrix.
- MF-BPR [67]: the standard matrix factorization [47] technique for recommender systems optimized for ranking via BPR [67].

5.1.2. METRICS

We measure the performance of our method with several metrics for ranking, rating and diversity to have a complete overview of the algorithm's behavior.[1] Denote the test set by $\mathcal{T}_s$ and the recommendation list of length k for user u.

Rating. For rating, we use the root mean square error (RMSE) between the mean centered true rating X_{ui} and mean centered predicted rating $\hat{X}_{ui} \in \mathcal{T}_s$ for the tuple (u, i).

[1]We have also evaluated the models with different metrics including: the mean absolute value (MAE), a surrogate of the RMSE for rating; precision and recall $@k$, which are ranking-oriented metrics for accuracy; and entropy diversity [3], which measures the ability of the model to recommend also items in the long-tail. We have observed these metrics respect in general the same accuracy-diversity trade-off trend and have omitted them for clarity of exposition.

The RMSE is defined as:

$$\text{RMSE} = \frac{\sqrt{\sum_{(u,i)\in\mathcal{T}_s} |\hat{X}_{ui} - X_{ui}|^2}}{|\mathcal{T}_s|}. \tag{5.1}$$

A lower value of RMSE indicates a better fit; hence, a better performance.

Ranking. For ranking we use the normalized discounted cumulative gain @ k (NDCG@k), which keeps into account item relevance. Denote by $\mathcal{I}_{uk} = \{i_{u1}, \cdots, i_{uk}\}$ the set of k items predicted with the highest rating for user u, i.e., $\hat{X}_{ui_{u1}} \geq \hat{X}_{ui_{u2}} \geq \ldots \geq \hat{X}_{ui_{uk}}$. To define the NDCG, we first first define the discounted cumulative gain (DCG) for which we consider the true ratings $X_{ui} := \text{rel}_i$ (called also the relevance for item i) of the items $i \in \mathcal{I}_{uk}$ ordered w.r.t. the predicted order in $\mathcal{I}_{uk}$, i.e., $\text{rel}_1 \geq \text{rel}_2 \geq \ldots \geq \text{rel}_k$. Then, the DCG for user u in a list of length k is defined as

$$\text{DCG}_u@k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}. \tag{5.2}$$

The $\text{DCG}_u@k$ accounts for the ordering of the true values in the predicted list $\mathcal{I}_{uk}$. This ordering can at most be the ideal one $X_{ui_{u1}} = \hat{X}_{ui_{u1}} \geq X_{ui_{u2}} = \hat{X}_{ui_{u2}} \geq \ldots \geq X_{ui_{uk}} = \hat{X}_{ui_{uk}}$, i.e., when the algorithm orders the items in the predicted list $\mathcal{I}_{uk}$ following the true order of preference for user u. In this instance, we refer to it as the ideal DCG for user u ($i\text{DCG}_u@k$). Then, the NDCG@k for the a list k over all users $\mathcal{U}$ is defined as

$$\text{NDCG}@k = \frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \frac{\text{DCG}_u@k}{i\text{DCG}_u@k}. \tag{5.3}$$

That is the sum over all users of the respective normalized DCGs_u w.r.t. the ideal ones. A high value of NDCG@k indicates a better recommendation in the list of order k; hence, a better performance [43].

Diversity. For diversity we use two metrics proposed in the literature: the aggregated diversity @ k (AD@k) [34], and the individual diversity @ k (ID@k) [97, 99]. The aggregated diversity AD@k measures the fraction of items $\mathcal{I}$ the algorithm includes in the union of all recommendation lists $\mathcal{I}_{uk}$. That is,

$$\text{AD}@k = \frac{1}{|\mathcal{I}|} \bigcup_{u=1}^{|\mathcal{U}|} \mathcal{I}_{uk}. \tag{5.4}$$

A higher aggregated diversity indicates the algorithm recommended a larger portion of the items present in the catalog, consequently, a better performance.

The individual diversity for a list of length k (ID@k) measures the average diversity within the recommendation lists of all users. That is, for $d(i,j)$ being a distance metric of two items i and j quantifying their dissimilarity, the individual diversity is computed as

$$\text{ID}@k = \frac{1}{|\mathcal{U}|} \sum_{u\in\mathcal{U}} \frac{2}{k(k-1)} \sum_{(i,j)\in\mathcal{I}_{uk}, i\neq j} d(i,j) \tag{5.5}$$

where the internal sum computes the individual diversity of the list $\mathcal{I}_{uk}$ of user u and the external sum averages across all users. A higher individual diversity indicates the average recommendation lists is more diverse. Notice the individual diversity requires computing a distance between items (often done based on their features). To use these metric also in featureless items, we followed [48] and computed the Euclidean distance based on first seven SVD latent features for items i and j.

We highlight a high individual diversity does not imply a high aggregated diversity and viceversa [3, 90]. For instance, an algorithm can recommend a set of k items diverse items to all users, implying a high individual diversity but a poor aggregated diversity. Or, an algorithm can cover all very alike items in the data set implying the high aggregated diversity, but a poor individual diversity.

5.2. PARAMETER ANALYSIS ON SIMILARITY GRAPH

We analyze the settings where the similarity graph is user-based and item-based [cf. Sections 2.3.1 and 2.3.2]. For each graph, we evaluate different nearest neighbors $n \in \{5, 10, \ldots, 40\}$, filter orders $K \in \{1, 2, 3\}$, and list length $k \in \{10, 20, \ldots, 100\}$.

5.2.1. NN AND SIMILAR FILTER ORDER

We first analyzed combinations between different NN and filter orders. We fixed the length of the list to $k = 10$ which is a common choice in the literature [3, 99, 90, 45, 63]. Figure 5.2 shows the RMSE, the AD@10, and the ID@10 for both scenarios. The NNs play a critical role in the accuracy-diversity trade-off. Increasing the number of NN reduces the RMSE (i.e., improves accuracy) but it degrades both diversity metrics. This is because each entity connects with more similar entities whose combined effect smoothness ratings. For almost all NN, there is always a $K > 1$ that improves both accuracy and diversity of the vanilla NN [cf.(2.5)-(2.6)]. The latter further highlights the importance of multi-hop neighbors. The behavior of filter order K corroborates also the observation in [37] that higher-order neighbors improve more in the user-based scenario than in the item-based one. From these results, we choose the combination that achieves the lowest RMSE. For the user-based scenario, we have: $30-$NN, $K = 3$, RMSE$=0.96$, AD@10 $= 0.15$, and ID@10 $= 0.02$. For the item-based scenario, we have: $35-$NN, $K = 2$, RMSE$= 0.96$, AD@10 $= 0.48$, and ID@10 $= 0.02$.

5.2.2. LENGTH RECOMMENDATION LIST

In Figure 5.2, we show the effect the recommendation list has on trade-off NDCG-AD, and NDCG-ID. A longer recommendation list improves diversity, but reduces accuracy. This is rather expected because in a longer list chances to include different items increase. At the same time, a longer list makes more challenging to identify the correct item order; hence, reducing the the NDCG.

Comparing the user NN with the item NN approach, we see little difference in terms of NDCG, while a larger one in terms of AD and ID.

- User NN achieves a lower AD but a higher ID. This implies the algorithm prioritises a few relevant items in the catalog but diversifies the list of each user, respectively. In our opinion, this is because the user NN has a narrow view of all items in the catalog as it explores user-similarities. The model fails to account for the broad range of items (each item is treated individually) and prioritises the popular

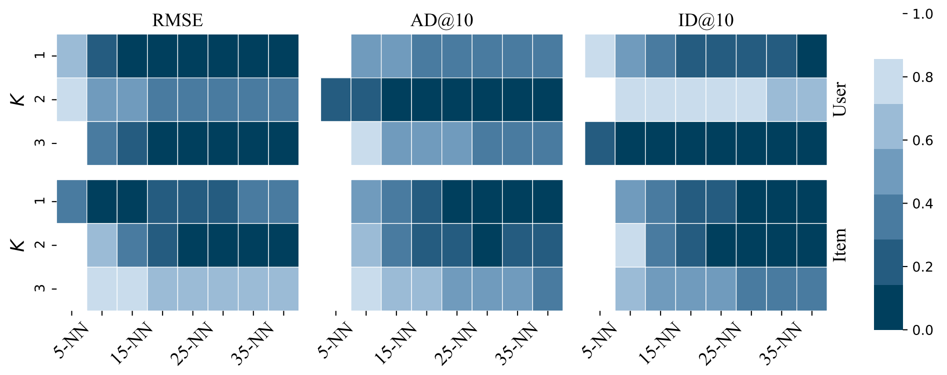


Figure 5.1: RMSE, AD@10, and ID@10 for different filter orders K (vertical axis) and nearest neighbors (horizontal axis). (Top) User-based graph convolutional filter [cf. (2.18)]. (Bottom) Item-based graph convolutional filter [cf. (2.21)]. Increasing the NN improves the RMSE but reduces diversity. The filter orders $K>1$ shows the vanilla NN $K-1$ [cf.(2.5)-(2.6)] can be improved by multi-hop neighbors. Results are scaled in the range $[0,1]$ to improve visibility.

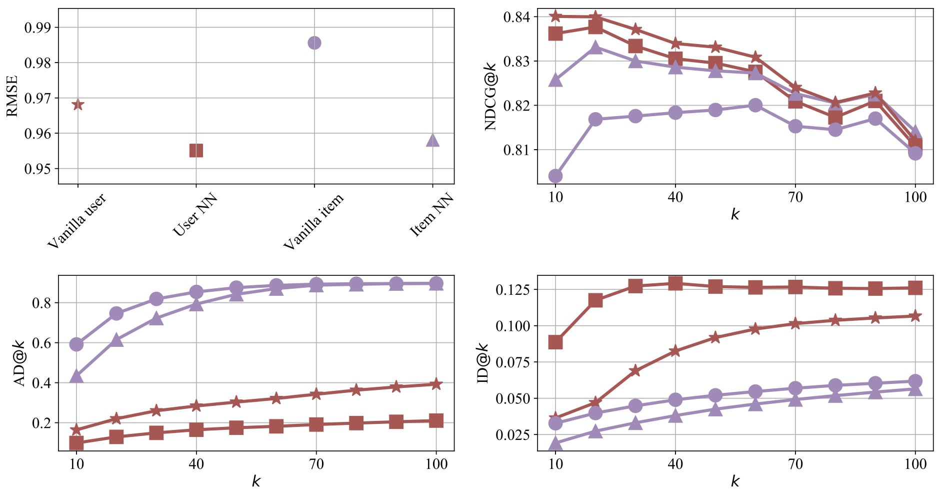


Figure 5.2: RMSE, NDCG@k, AD@k and ID@k as recommendation set's size k varies. We see the NDCGk reduces for lists more than 20 items while diversity increases.

choices, which are different between them. The plateau the user NN reaches for relatively low ID further corroborates the latter.

- Item NN achieves instead a higher AD but a lower ID. I.e., the model covers a larger portion of the catalog (recommends different items to different users) but to a specific user it recommends similar items. Item NN is less user-centric since it leverages item similarities and ignores the influence of other similar users. Consequently, the model has a broader view of the items to build recommendations; this explains an AD that is up to four times higher than the user NN. Nevertheless, since each user is treated individually less importance is given to diversifying items within the list. To some extent, we can see this model as as highly personalizing the user list because different items are recommended to different users but these items are highly similar within them.

Lastly, we see the user NN and the item NN, optimized for rating performs worse in terms of list-oriented metrics compared with their vanilla counterparts [cf. (2.5)-(2.6)]. This is because the RMSE treats equally the fitting to all entries in the user-item matrix, ultimately, ignoring the item priority in the recommendation lists. Therefore, we recommend learning the graph convolutional filter coefficients w.r.t. the RMSE only if the goal is to complete the matrix rather than building a short recommendation list. I.e., identifying both preferred and non-preferred items from users.

Based on these results, we set the list length to $k = 20$ since this value achieves the highest NDCG for both the user NN and the item NN. While a longer list can be an option to improve diversity, it is not user-satisfactory to search within it. The proposed joint model improves diversity without increasing the list length. We have seen this improvement also in a list of length $k = 10$.

As result of our analysis, for the NN graph, we set the parameters as: (i) user-scenario with 30−NN, filter order $K = 3$, list length $k = 10$; (ii) item-scenario with 30−NN, filter order $K = 2$, list length $k = 20$.

5.3. Accuracy-Diversity Trade-off Analysis

5.3.1. Accuracy-Diversity Trade-off for Rating

We first study the trade-off when the joint models are trained for rating [cf. Sec. 4.2]. For the NN module, we used the parameters derived in the previous Section. For the FN module, we fixed the number of neighbors to the arbitrary common value 40, evaluated different filter orders $K \in \{1,2,3\}$, and show the best results.

Figure 5.3 shows the results for the combinations in Table 4.1 as a function of $\alpha \in [0.1, 0.9]$. As we increase the influence of FNs ($\alpha \to 1$), the RMSE increases. The linear filters are more robust to α than the GCNN. We attribute the latter to the convexity of their design problem. Increasing α increases diversity, while the AD and ID exhibit opposite behavior. Values of α up to 0.5 offer a good trade-off as the RMSE remains unaffected but diversity increases substantially.

To further quantify the trade-off, we allow the RMSE to deteriorate by at most 3% w.r.t. the NN setup [cf. 5.2] and pick a value of α that respects such constraint. Table 5.2 compares the different models. For a user NN graph, the joint models (i.e., UU and UI) boost substantially one diversity metric. We believe this is because models build only on user-NN graphs are conservative to both diversity metrics [cf. Fig. 5.2], therefore, the margin for improvement is larger. Contrarily, for an item NN graph, the joint models (i.e., IU and II) are conservative and improve by little both diversity metrics. We also highlight the case of II-GCNN which improves the RMSE and AD while keeping the same ID.

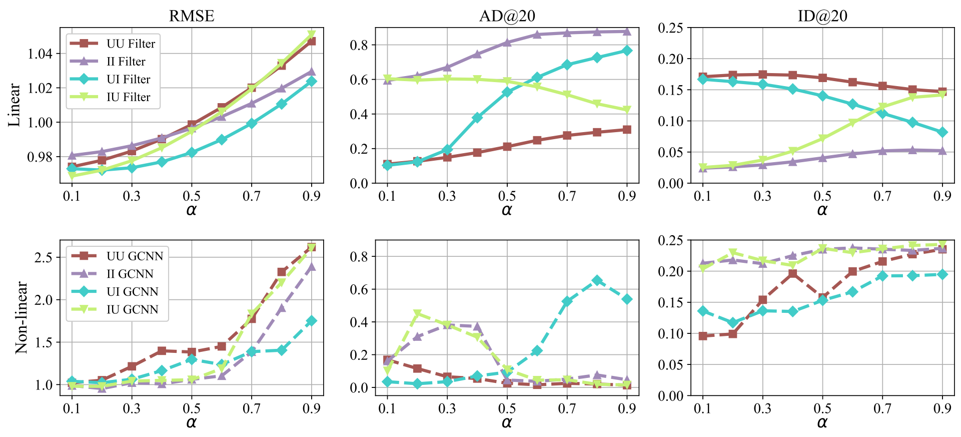


Figure 5.3: RMSE, AD@20, and ID@20 as a function of the accuracy-diversity parameter α for models optimized for rating. As more information from the dissimilar connections is included, the RMSE deteriorates but diversity improves. The RMSE of the GCNN is more sensitive to α as its hyperparamters are not tuned.

Table 5.2: RMSE, AD@20 and ID@20 for different models optimized for rating. In brackets we show the change in percentage of the proposed joint models w.r.t. the NN counterpart.

		RMSE	AD@20	ID@20
User Linear	User NN	0.96	0.19	0.02
	UU filter	0.98 (+2.1%)	0.15 (-21.5%)	0.17 (+**750%**)
	UI filter	0.98 (+2.1%)	0.53 (+**178%**)	0.14 (+**600%**)
User GCNN	User GCNN	1.03	0.02	0.15
	UU GCNN	1.05 (+1.9%)	0.12 (+**500%**)	0.10 (-33.3%)
	UI GCNN	1.06 (+2.9%)	0.04 (+100%)	0.14 (-6.7%)
Item Linear	Item NN	0.96	0.65	0.03
	II filter	0.98 (+2.1%)	0.62 (-4.6%)	0.03 (0%)
	IU filter	0.98 (+2.1%)	0.60 (-7.7%)	0.03 (0%)
Item GCNN	Item GCNN	0.97	0.29	0.22
	II GCNN	0.95 (−**2.1%**)	0.31 (+**6.9%**)	0.22 (0%)
	IU GCNN	0.98 (+1%)	0.45 (+55.2%)	0.23 (+4.6%)

5.3.2. Accuracy-Diversity Trade-off for Ranking

With the same setting of the last section, we now evaluate the trade-off when the joint models are optimized for ranking [cf. Sec 4.3]. These results are shown in Figure 5.4. A higher importance to FNs ($\alpha \rightarrow 1$) reduces the NDCG@20 but improves diversity. Both the filter and the GCNN are less sensitive to α when designed for ranking. While for the filter we may still attribute this robustness to the optimality of the design problem, the results for the GCNN suggest the BPR leverages better the information from FNs. Note also the filter on the UI combination pays little in NDCG but gains substantially in AD and ID.

To further quantify these results, in Table 5.3 we show the diversity gain when reducing the NDCG by at most 3%. We note that it is often sufficient to deteriorate the NDCG by 1% and gain substantially in diversity. Bigger diversity improvements are achieved when one of the two graphs is item-based. Lastly, we notice the joint GCNN models

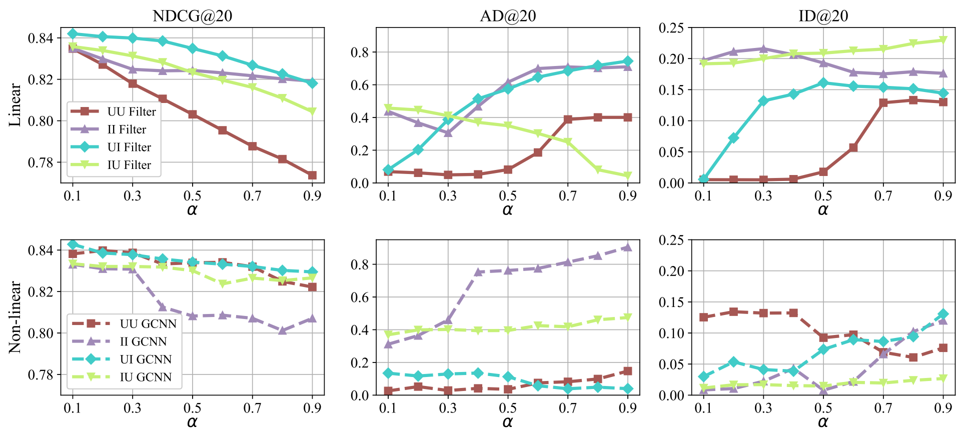


Figure 5.4: NDCG@20, AD@20 and ID@20 as a function of the accuracy-diversity parameter α for ranking-optimized when models. As more information from FNs is included, the NDCG@20 deteriorates but diversity improves. The NDCG is less sensitive to α compared with the RMSE for both the joint graph convolutional filter and GCNN.

Table 5.3: NDCG@20, AD@20 and ID@20 for the models working on the NN graph and for the joint models optimized for ranking. In brackets we show the change in percentage of the proposed joint model w.r.t. the NN graph counterpart.

		NDCG@20	AD@20	ID@20
User Linear	User NN	0.84	0.19	0.02
	UU filter	0.83 (-1.2%)	0.07 (-63.1%)	0.01 (-50%)
	UI filter	0.83 (-1.2%)	0.65 (+**242%**)	0.16 (+**700%**)
User GCNN	User GCNN	0.84	0.10	0.12
	UU GCNN	0.82 (-2.4%)	0.15 (+50%)	0.08 (-33.3%)
	UI GCNN	0.83 (-1.2%)	0.11 (+10%)	0.07 (-41.6%)
Item Linear	Item NN	0.83	0.65	0.03
	II filter	0.82 (-1.2%)	0.70 (+**7.7%**)	0.18 (+**500%**)
	IU filter	0.83 (0%)	0.41 (-36.9%)	0.20 (+566%)
Item GCNN	Item GCNN	0.83	0.40	0.02
	II GCNN	0.83 (0%)	0.46 (+15%)	0.02 (0%)
	IU GCNN	0.83 (0%)	0.47 (+**17.5%**)	0.03 (+**50%**)

gain less in diversity compared with linear filters. The GCNN can be further improved by tuning its parameters.

5.3.3. COMPARISONS WITH ACCURACY-ORIENTED MODELS

In this section, we analyze how the trade-offs of the joint models compare with those achieved by five accuracy-oriented alternatives including state-of-the-art user NN filter [cf. (2.18)], item NN filter [cf. (2.21)], and the multi-graph convolutional neural network (MGCNN) [41]; but also the conventional methods of low-rank matrix completion (LR-MC) [60] and matrix factorization optimized w.r.t. BPR (MF-BPR) [67]. Save the last, the first four are designed for rating. We first compare the models in MovieLens100k dataset and then in Douban and Flixster. We consider only the UI combination.

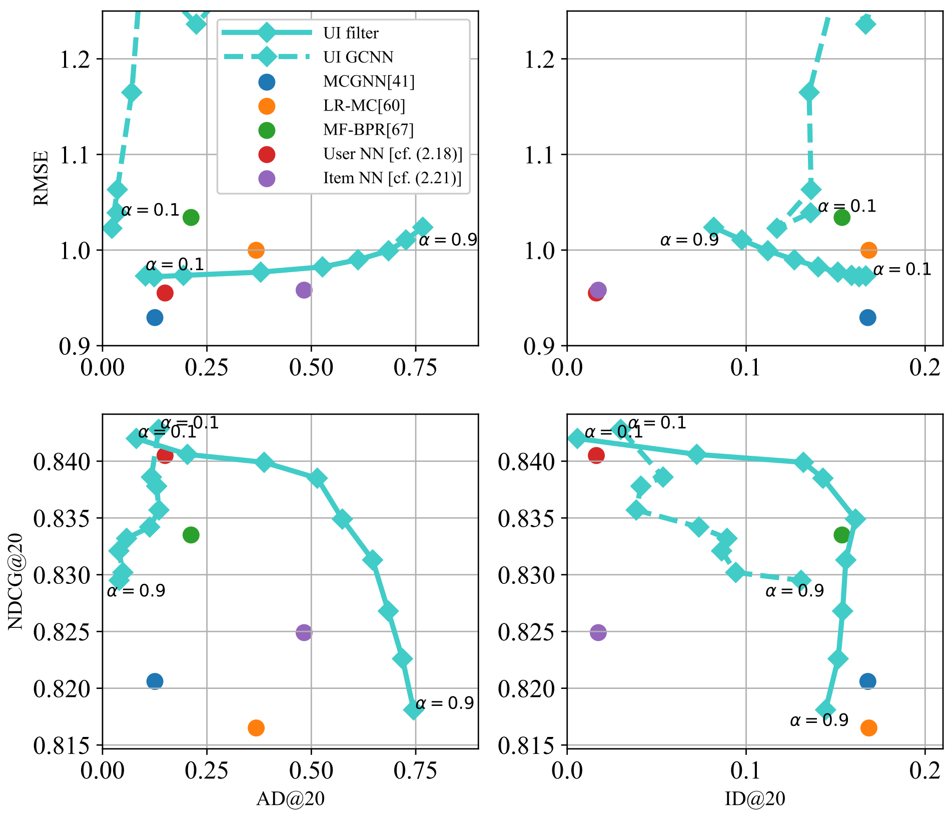


Figure 5.5: RMSE and NDCG@20 as a function of the AD@20 and ID@20 for the joint models on the UI graph combinations and for five baselines on the MovieLens100k. The joint models are optimized for ranking. By changing α the propsoed models impose a different trade-off than the baselines, often overcoming them.

MovieLens100k. Figure 5.5 contrasts the RMSE and NDCG@20 with the diversity metrics the AD@20 (left) and ID@20 (right) for $\alpha \in [0.1, 0.9]$. The accuracy of GCNN is more sensitive to α than the other models. The GCNN gives also more importance to diversity within the list (ID) rather than covering the catalog (AD). This indicates a few items are recommended by the GCNN but are different between them. Contrarily, the joint linear filters are more robust to accuracy losses, gain in AD, but pay in ID. Contrasting the proposed approaches with the other alternatives, we observe:

- Rating-optimzied models (MGCNN, user NN filter, item NN filter, and LR-MC) achieve a lower RMSE but face problems in AD. The item NN filter achieves a reasonable AD but its ID is very low. The MGCNN over-fits the RMSE by prioritising a few popular items to all users as shown by the low AD and high ID. The joint linear filter can substantially improve the AD by paying little in RMSE, while the GCNN requires additional tuning. The improved AD comes often at expenses of ID; yet values of $\alpha \approx 0.3$ offer a good balance between the two. We can further improve the ID with the IU combination [cf. Fig. 5.3].

- The ranking-optimized method (BPR-MF) achieves a high NDCG but still lower than the rating-design user NN filter. This high accuracy is again linked to filling the list with a small group of different items. The joint models optimized for ranking overcome this limitation by making the list slightly more similar (lowering ID) but increasing the catalog coverage (improving AD). This strategy keeps the NDCG high.

Overall, we conclude that a high accuracy from the NNs is tied with an increase of list diversity (ID) but also with a scarce catalog coverage (AD). The proposed joint models can keep a reasonable accuracy while contributing to a higher diversity.

Douban & Flixster. We now compare the different models in two datasets containing fewer interactions compared with MovieLens100k; see Table 5.1. The sparsity of these datasets brings additional challenges when evaluating the NDCG@k. For a list of length 20 there is only one test user for Fixster and none for Douban. To have statistically meaningful results, we measured the NDCG for a list of length $k = 5$, which leads 1,373 test users for Douban and 126 for Fixster. However, to have a unified diversity comparison with the MovieLens100k dataset, we computed the diversity for a list of length 20.

In Table 5.4, we show the performance for Douban and Fixster datasets, respectively. For our models we report the extreme values $\alpha = 0.1$ and $\alpha = 0.9$ and an hand-picked value of α. As α increases, the joint models lose in accuracy but gain in diversity. We see again the sensitivity of GCNNs to α for which the RMSE may also reach unacceptable values. The joint models optimized for ranking can always provide a better NDCG w.r.t. MF-BPR while offering a higher diversity. In general, the best trade-off by the joint models is achieved by the GCNN designed for rating and filters designed for ranking.

Table 5.4: Performance comparison in Douban and Flixster dataset. We show the RMSE for methods trained for rating and the NDCG@5 for methods trained for ranking. For the intermediate value of α, we show in brackets the difference in percentage compared with the best value of competing alternatives.

	Douban					**Flixster**				
	α	RMSE	NDCG@5	AD@20	ID@20	α	RMSE	NDCG@5	AD@20	ID@20
MCGNN	-	**0.80**	-	0.03	**0.12**	-	**0.93**	-	0.08	0.07
LR-MC	-	1.39	-	0.80	0.04	-	3.17	-	0.45	0.09
MF-BPR	-	-	**0.75**	0.78	0.05	-	-	**0.72**	0.50	**0.10**
User NN	-	0.76	-	0.52	0.04	-	1.04	-	**0.58**	0.06
Item NN	-	0.80	-	**0.99**	0.05	-	1.12	-	0.33	0.03
Rating linear	0.1	**0.76**	-	0.51	0.04	0.1	1.04	-	0.58	0.06
	0.6	0.84 (+5%)	-	0.75 (-24%)	0.05 (-58%)	0.6	1.05 (+13%)	-	0.69 (+19%)	0.06 (-33%)
	0.9	0.92	-	0.96	0.05	0.9	1.06	-	**0.71**	0.06
Rating GCNN	0.1	0.83	-	0.49	0.10	0.1	1.46	-	0.60	0.10
	0.3	0.85 (+6%)	-	0.44 (-56%)	0.12 (0%)	0.4	1.46 (+57%)	-	0.56 (-3%)	0.11 (+22%)
	0.9	3.31	-	0.63	0.11	0.9	2.84	-	0.48	0.12
Ranking linear	0.1	-	**0.80**	0.44	**0.15**	0.1	-	**0.75**	0.34	**0.14**
	0.4	-	**0.80 (+7%)**	0.48 (-38%)	0.12 (+140%)	0.3	-	**0.75 (+4%)**	0.35 (-30%)	0.13 (+30%)
	0.9	-	0.77	0.77	0.07	0.9	-	0.74	0.36	0.13
Ranking GCNN	0.1	-	**0.80**	0.47	0.13	0.1	-	0.74	0.30	0.13
	0.4	-	**0.80 (+7%)**	0.90 (+15%)	0.05 (0%)	0.6	-	0.74 (+3%)	0.35 (-30%)	0.13 (+30%)
	0.9	-	0.76	0.91	0.05	0.9	-	0.73	0.36	0.13

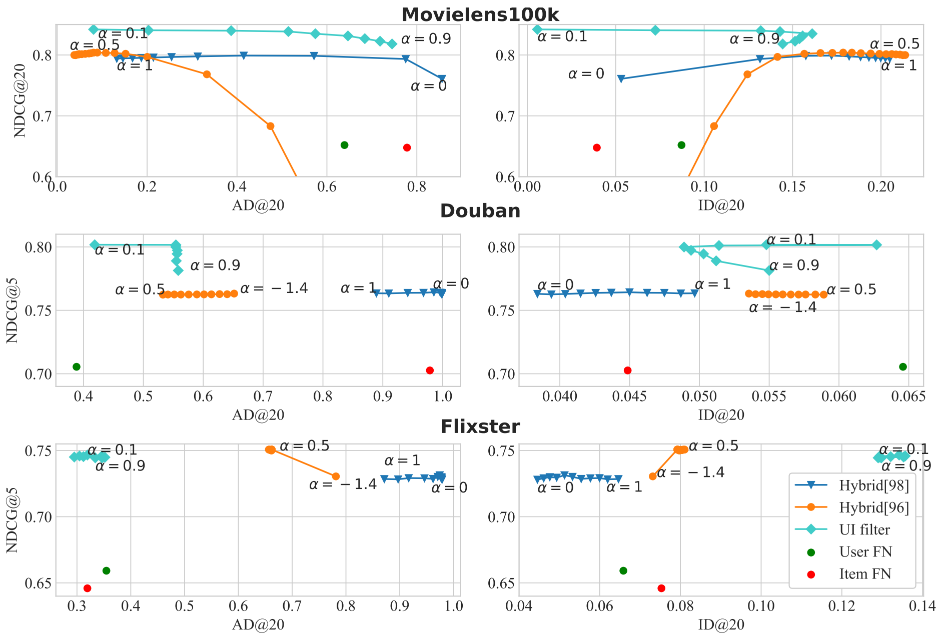


Figure 5.6: Comparison of joint UI filter optimized for ranking [cf. (4.15)] with the hybrid approaches from [96] and [98]. We also show the vanilla user-FN and item-FN for reference. The proposed approach pays the less in terms of NDCG while offering different diversity gains.

5.3.4. Comparison with Accuracy-Diversity Algorithms

In this final section, we compare the joint UI linear model with two hybrid alternatives that propose a similar trade-off [96, 98]. The hybrid approach in [96] mixes a user-based vanilla NN with a user-based vanilla FN. FNs are computed based on the number of items consumed separately and only FNs are multiplied by a single scalar $\alpha \in [-1.4, 0.5]$. The hybrid approach in [98] merges a heat diffusion with a random walk to balance accuracy with diversity over item-item graphs. This approach controls the influence of each model similarly to our method through a scalar $\alpha \in [0, 1]$. Both works predict the probability of an item being consumed by a user rather than the rating. Therefore, we compare the accuracy w.r.t. the NDCG.

In Figure 5.6, we show the trade-offs of the different methods for all three datasets. We also show two vanilla FN collaborative filters for reference. We see the proposed joint model achieves consistently the highest NDCG while offering a margin to improve accuracy. This behavior is better highlighted in MovieLens100K dataset for which the method hyperparameters have been chosen. We attribute the latter to the fact that the joint model learns its parameters to improve ranking accuracy rather than being a simple fusion of two separate entities. The hybrid strategy from [98] focuses entirely on the catalog coverage as can be seen by the high AD. This strategy heavily affects both the NDCG and ID for which this approach performs the worst. The hybrid strategy from [96] offers a trade-off in both diversity metrics but the role of the two graphs depends largely on the dataset sparsity. In Flixster, for instance, we see this strategy offers little trade-off as the performance for all values of α but -1.4 is the same. To some extent this trend is also present in our joint model, yet it has more control on diversity while retaining the highest NDCG.

5.4. Discussion

The accuracy-diversity trade-off represents a crucial factor for personalizing recommendations. However, balancing this trade-off is challenging because of the complex and irregular user-item relationships. Graphs have a proven history as core mathematical tools for representing such data and graph convolutional RecSys algorithms have reached state-of-the-art accuracy. This thesis used graph convolutions to establish an accuracy-diversity trade-off for RecSys. The proposed approach exploits jointly the information from the nearest neighbors and the furthest neighbors in both a learning-to-rate and learning-to-rank setting. Our findings proved that graph convolutions, in both the linear graph filtering or nonlinear graph convolutional neural network form, can effectively learn from the nearest and the furthest neighbors to establish two trade-offs: i) an accuracy-to-coverage trade-off, in which accuracy is traded to recommend niche items; and ii) an accuracy-to-individual diversity trade-off, in which accuracy is traded to improve the diversity in the list.

We formulated a joint learning problem that accounts for the accuracy-diversity trade-off. When the graph convolutional model is composed only of linear filters, we proved the learning problem is convex and provided solutions for it. The latter is in accordance with the learning-to-rate design performed only on nearest neighbors [37]. The convexity of the learning problem renders the linear model more robust to hyperparameter choice in different datasets, while the nonlinear model required careful tuning.

The performance of the joint graph convolutional method confirms our hypothesis that information from nearest neighbors aids recommendation accuracy while information from furthest neighbors aids diversity. This finding is also in line with the results obtained with the nearest neighbor collaborative filtering or the furthest neighbor collaborative filtering [96, 70]. The proposed method can trade accuracy to improve substantially one diversity criteria or improve both by a lesser amount. The latter is also a behavior observed in literature; i.e., the aggregated and individual diversity are opposed to each other [3].

Contrasting the joint model with state-of-the-art graph convolutional RecSys methods, we showed a diversity improvement by up to seven times while paying around 1% in accuracy. We also found the joint GCNN model combining an item nearest neighbor graph and an item furthest neighbor graph improved accuracy and both diversity metrics. Contrasting the joint model with other accuracy-diversity trade-off approaches, we showed the proposed models retains the highest accuracy while improving diversity.

6 CONCLUSION

This Chapter concludes the thesis. In Section 6.1 we give a summary of the thesis; in Section 6.2 we answer each research question discussed in Chapter 1; in Section 6.3 we indicate possible directions for future research. Finally, in Section 6.4 we highlight broad impacts of our research.

6.1. SUMMARY

In this thesis, we proposed a new accuracy-diversity trade-off framework for RS using graph convolutions.

In Chapter 1, we introduced the topic and prepared the ground with motivation and research questions. In Chapter 2, we reviewed background knowledge useful for the thesis. We started with RS and their rationale. Later, we discussed the basics of graphs signal processing, particularly about graph convolutions, GCNNs, and their graph spectral interpretation. Consequently, we saw how to perform recommendation employing graph convolutions. This is done by leveraging information only from positively correlated neighbors, resulting in an accuracy-oriented model. In Chapter 3, we reviewed the relevant literature regarding diversity and its trade-off with accuracy, together with relevant works employing graph convolutions for recommendation.

In Chapter 4, we presented our approach. Our framework balances accuracy and diversity in RS employing graph convolutions. We learn two parametric maps, one on a NN graph, regarded as accuracy source, and the other on a FN graph, regarded as diversity source. We exploited the flexibility of graph convolutions to: (i) design filters with a parameter α controlling the influence of each source; (ii) learn models optimized for rating or ranking; and (iii) work with linear and nonlinear models (GCNNs). We studied the graph frequency response of the filters involved in our models, to gain insights not directly visible in nodal domain.

In Chapter 5, we evaluated our proposed models with numerical experiments on three real world datasets. We measured the accuracy with rating (RMSE), ranking (NDCG@k), and diversity (AD@k and ID@k) metrics, to assess the accuracy-trade-off. We compared our models with state-of-the-art accuracy-oriented methods and with algorithms proposing solutions to the accuracy-diversity dilemma. Results evidenced exploiting negative correlations contributed to increasing diversity in one of its two forms (e.g. user NN and user FN) or in both AD and ID at the same time (e.g. user NN and item FN). Values of $\alpha \approx 0.3$ showed the most promising results for the trade-off, with an accuracy drop always lower than 3% and a gain in diversity up to 750% w.r.t. methods working only with NN.

6.2. ANSWER TO RESEARCH QUESTIONS

In this Section, we recall the research questions posed in Chapter 1 and answer each of them with the findings of this thesis.

(RQ1) *How can we employ graph convolutions to tweak the accuracy-diversity trade-off for Recommender Systems?*

(RQ2) *How can we learn the graph convolutional parameters to model jointly accuracy and diversity?*

To address these research questions, in Chapter 4 [cf. Section 4.1] we proposed a new graph convolution-based framework in RS for a novel accuracy-diversity tradeoff. Besides learning from a NN graph responsible for accuracy, we introduced a FN graph to account for diversity already at learning phase. We build a graph convolutional filter on top of each graph to predict missing ratings, while jointly leveraging information from both graphs. This requirement led to filter design choices to balance the two graphs via parameter α. Exploiting graph convolutional filters' properties, we designed parameter learning as optimization of a loss function [cf. Sections 4.2 and 4.3].

We discussed rating and ranking optimization, both in linear and nonlinear scenarios. In both rating and ranking optimization, we articulated about the optimality of the defined loss functions: (i) in rating, the loss function is a regularized-least squares convex problem, thus guaranteeing a closed-form solution; (ii) in ranking, the loss function is still convex but a closed-form solution is difficult to find, due to the logistic nature of the problem. We found the optimal parameters via stochastic gradient descent updates. In both rating and ranking optimization, the regularizer is α-aware, to learn parameters according to the importance of NN and FN graphs, respectively.

(RQ3) *How can we explain the accuracy-diversity trade-off achieved by the graph convolutional model in the spectral domain?*

To answer this research question, in Chapter 4 [cf. Section 4.4] we studied the graph frequency response of graph convolutional filters employed in our model. We chose the model based on user-NN and item-FN. We evidenced a bandstop behavior for filters based on similarities, corroborating the findings of [37]. The same behavior is present also in filters based on dissimilarities. This indicates the joint model focused on low frequency features, corresponding to features common to highly (dis)similar neighbors, and high frequency features, corresponding to features common to (dis)similar neighbors with discordant ratings. We also gave an analysis of the importance of each neighborhood by looking at the learned filter coefficients. Direct neighbors are most important in the NN graph, while in the FN graph more relevance is given to two-hops neighbors. We argued this is because direct neighbors in the NN graph are the most reliable for accuracy, and two-hops neighbors in the FN graphs provide a wider variety of choices for recommendation while keeping a reasonable accuracy.

(RQ4) *How does the proposed accuracy-diversity trade-off model behave in Recommender System datasets?*

To provide an answer to this research question, in Chapter 5 we tested the performance of our model on three real world datasets with increasing sparsity. Our model showed the dependency on α, i.e. when more weight is given to NNs (FNs), accuracy (diversity) is high while diversity (accuracy) is low. Numerical experiments highlighted our model can significantly boost either one of the two diversities, or increase both by a lesser amount. We compared our model against state-of-the-art accuracy-oriented models, showing diversity-agnostic models tend to overfit to popular items to achieve higher accuracy but paying in diversity. Our models spanned a wide range of choices, with the user NN item FN model being able to increase both AD and ID without affecting substantially accuracy. Finally, we tested user NN item FN model against two algorithms proposing a different accuracy-diversity trade-off. Our model achieved the highest ranking accuracy, as we learned the optimal parameters, while the two algorithms represent hybrid models working without optimization.

6.3. FUTURE WORK

To the best of our knowledge, this is the first work proposing graph convolutions to solve the accuracy-diversity trade-off in recommender systems. As such, there are aspects requiring additional analysis. In this section, we lay down possible directions for future research.

COMBINATION OF LINEAR AND NONLINEAR MODELS

In our analysis, we did not mix linear models with nonlinear models. Nevertheless, the flexibility offered by graph convolutions would allow to combine the two. For instance consider problems (4.3) and (4.13). We only accounted the parametric maps $\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}^i;\mathbf{B}_i;\mathscr{H}_{\mathrm{s}})$ and $\mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathscr{H}_{\mathrm{d}})$ to be either two graph convolutional filters or two GCNNs. One possibility is to employ a graph convolutional filter on similarity graph $\mathscr{G}_{\mathrm{s}}$ and a GCNN on dissimilarity graph $\mathscr{G}_{\mathrm{d}}$, or vice-versa. Such configuration might offer deeper insights into the impact of singular components in different scenarios. E.g., consider rating optimization; linear models perform better in AD than in ID, while the opposite is true for nonlinear models. Ideally, we could design a linear map on $\mathscr{G}_{\mathrm{s}}$ and a GCNN on $\mathscr{G}_{\mathrm{d}}$, and vice-versa, depending on which type of diversity is more important for the application.

NEAREST-FURTHEST COMBINATION

Properties of graph convolutions allowed us to work on two separate graphs (user and item graphs) at the same time. This contributed to four possible combinations, as explained in Section 4.1 [cf. Table4.1]. The current results are not exhaustive to decide which nearest-furthest graph combination is the most suitable for a specific accuracy and diversity criteria. The most promising models seem to be the ones working on user-NN and item-FN. A possible direction in this regard could be a study of the spectral properties of NN and FN graphs, similar to what we performed in Section 4.4. In detail, our model learned the parameter in a joint fashion by leveraging both graphs. However, it could be interesting to study the characteristics of NN and FN graphs in a standalone manner and analyze whether the behavior is the same in a joint model. In this way, we could draw conclusions on the role of users and items in the recommendation.

EXPLAINABILITY

In Section 4.4, we conducted a spectral analysis of the filters in linear and nonlinear models. The results of this analysis gave us interesting insights about recommendation when diversity is involved, e.g. the importance of direct and indirect neighbors in the two graphs, and the bandlimited nature of the filters. However, it is still needed to identify the link between the different spectral components and the items included in the recommendation lists. For instance, a direction for future research could be a categorical study of recommended items concerning the spectral frequencies. That is, analyze the categories of items in the recommendation lists to find a link between the spectral components and the category which are recommended the most. This could potentially shed light on the meaning behind the features our models focus on, i.e. what is the association between low/high frequency components in the spectral domain and the category of a recommended item. By doing so, we could design filters able to target a specific category of items while only looking at the frequency components.

6.4. BROAD IMPACTS

Graph convolutions are a powerful tool for learning over graphs. Their versatility allowed us to define a joint model combining information from nearest and furthest neighbors, to achieve a novel accuracy-diversity trade-off. Diversity boosting can be employed to escape the so-called *filter bubble* [51], the effect for which a user is exposed to the same category of information in a network, e.g. items in a catalog or news in a social network. Our model can be a viable solution to burst the filter bubble in social networks, where users tend to surround themselves with users sharing akin ideologies, resulting in a stagnation of ideas. By leveraging information jointly from similar and dissimilar users, social networks could adapt their feed providing a more diverse view to users.

More broadly, our research can be extended to the field of network information spreading. We highlighted the importance of indirect neighbors for accuracy, corroborating the result of [37]. At the same time, we showed furthest neighbors to be beneficial for the sake of diversity. With these results in place, we could rethink the way information spreads within a network. Graph convolutions can be employed for information spreading processes, to provide also a spectral analysis which could aid explainability of such phenomena. One example could be fake news, characterized by a trade-off between the accuracy of the news and the intent of the news provider [9]. Our research could pave the road for more involved scenarios in the fake news field.

BIBLIOGRAPHY

[1] F. Gama, **E. Isufi**, G. Leus and A. Ribeiro, "From Graph Filters to Graph Neural Networks", to appear in the IEEE Signal Processing Magazine; *Special Issue on Graph Signal Processing: Foundations and Emerging Directions.* (2020).

[2] Zeinab Abbassi and Vahab S Mirrokni. “A recommender system based on local random walks and spectral methods”. In: *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis.* 2007, pp. 102–108.

[3] Gediminas Adomavicius and YoungOk Kwon. “Improving aggregate recommendation diversity using ranking-based techniques”. In: *IEEE Transactions on Knowledge and Data Engineering* 24.5 (2011), pp. 896–911.

[4] Gediminas Adomavicius and YoungOk Kwon. “Maximizing aggregate recommendation diversity: A graph-theoretic approach”. In: *Proc. of the 1st International Workshop on Novelty and Diversity in Recommender Systems (DiveRS 2011).* Citeseer. 2011, pp. 3–10.

[5] Gediminas Adomavicius and YoungOk Kwon. “Optimization-based approaches for maximizing aggregate recommendation diversity”. In: *INFORMS Journal on Computing* 26.2 (2014), pp. 351–369.

[6] Gediminas Adomavicius and YoungOk Kwon. “Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach”. In: *Proceedings of WITS.* Vol. 8. Citeseer. 2008.

[7] Gediminas Adomavicius and YoungOk Kwon. “Toward more diverse recommendations: Item re-ranking methods for recommender systems”. In: *Workshop on Information Technologies and Systems.* Citeseer. 2009.

[8] Ajay Agarwal and Minakshi Chauhan. “Similarity measures used in recommender systems: a study”. In: *International Journal of Engineering Technology Science and Research IJETSR, ISSN* (2017), pp. 2394–3386.

[9] Hunt Allcott and Matthew Gentzkow. “Social media and fake news in the 2016 election”. In: *Journal of economic perspectives* 31.2 (2017), pp. 211–36.

[10] Tevfik Aytekin and Mahmut Özge Karakaya. “Clustering-based diversity improvement in top-N recommendation”. In: *Journal of Intelligent Information Systems* 42.1 (2014), pp. 1–18.

[11] Kirell Benzi et al. “Song recommendation with non-negative matrix factorization and graph total variation”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Ieee. 2016, pp. 2439–2443.

[12] Rianne van den Berg, Thomas N Kipf, and Max Welling. “Graph convolutional matrix completion”. In: *arXiv preprint arXiv:1706.02263* (2017).

[13] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Keith Bradley and Barry Smyth. "Improving recommendation diversity". In: *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*. Citeseer. 2001, pp. 85–94.

[15] Deng Cai et al. "Graph regularized nonnegative matrix factorization for data representation". In: *IEEE transactions on pattern analysis and machine intelligence* 33.8 (2010), pp. 1548–1560.

[16] Laurent Candillier, Frank Meyer, and Françoise Fessant. "Designing specific weighted similarity measures to improve collaborative filtering systems". In: *Industrial Conference on Data Mining*. Springer. 2008, pp. 242–255.

[17] Sylvain Castagnos, Armelle Brun, and Anne Boyer. "When Diversity Is Needed... But Not Expected!" In: 2013.

[18] Lei Chen et al. "Revisiting Graph based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach". In: *arXiv preprint arXiv:2001.10167* (2020).

[19] Siheng Chen et al. "PCT: Large-scale 3D point cloud representations via graph inception networks with applications to autonomous driving". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 4395–4399.

[20] Nitin Chiluka, Nazareno Andrade, and Johan Pouwelse. "A link prediction approach to recommendations in large-scale user-generated content systems". In: *European Conference on Information Retrieval*. Springer. 2011, pp. 189–200.

[21] Charles LA Clarke et al. "Novelty and diversity in information retrieval evaluation". In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 2008, pp. 659–666.

[22] Nima Dehmamy, Albert-László Barabási, and Rose Yu. "Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15387–15397.

[23] Michael D Ekstrand et al. "User perception of differences in recommender algorithms". In: *Proceedings of the 8th ACM Conference on Recommender systems*. 2014, pp. 161–168.

[24] Farzad Eskandanian and Bamshad Mobasher. "Using Stable Matching to Optimize the Balance between Accuracy and Diversity in Recommendation". In: *arXiv preprint arXiv:2006.03715* (2020).

[25] Daniel M Fleder and Kartik Hosanagar. "Recommender systems and their impact on sales diversity". In: *Proceedings of the 8th ACM conference on Electronic commerce*. 2007, pp. 192–199.

[26] Mingxin Gan and Rui Jiang. "Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities". In: *Decision Support Systems* 55.3 (2013), pp. 811–821.

[27] Adnan Gavili and Xiao-Ping Zhang. “On the shift operator, graph frequency, and optimal filtering in graph signal processing”. In: *IEEE Transactions on Signal Processing* 65.23 (2017), pp. 6303–6318.

[28] Anupriya Gogna and Angshul Majumdar. “Balancing accuracy and diversity in recommendations using matrix completion framework”. In: *Knowledge-Based Systems* 125 (2017), pp. 83–95.

[29] David Goldberg et al. “Using collaborative filtering to weave an information tapestry”. In: *Communications of the ACM* 35.12 (1992), pp. 61–70.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[31] Mario Haim, Andreas Graefe, and Hans-Bernd Brosius. “Burst of the filter bubble? Effects of personalization on the diversity of Google News”. In: *Digital journalism* 6.3 (2018), pp. 330–343.

[32] Elaheh Malekzadeh Hamedani and Marjan Kaedi. “Recommending the long tail items through personalized diversification”. In: *Knowledge-Based Systems* 164 (2019), pp. 348–357.

[33] F Maxwell Harper and Joseph A Konstan. “The movielens datasets: History and context”. In: *Acm transactions on interactive intelligent systems (tiis)* 5.4 (2015), pp. 1–19.

[34] Jonathan L Herlocker et al. “Evaluating collaborative filtering recommender systems”. In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53.

[35] Rong Hu and Pearl Pu. “Helping Users Perceive Recommendation Diversity.” In: *DiveRS@ RecSys*. 2011, pp. 43–50.

[36] Fei Hua et al. “Learning Combination of Graph Filters for Graph Signal Modeling”. In: *IEEE Signal Processing Letters* (2019).

[37] Weiyu Huang, Antonio G Marques, and Alejandro R Ribeiro. “Rating prediction via graph signal processing”. In: *IEEE Transactions on Signal Processing* 66.19 (2018), pp. 5066–5081.

[38] Neil J Hurley. “Personalised ranking with diversity”. In: *Proceedings of the 7th ACM conference on Recommender systems*. 2013, pp. 379–382.

[39] Neil Hurley and Mi Zhang. “Novelty and diversity in top-n recommendation–analysis and evaluation”. In: *ACM Transactions on Internet Technology (TOIT)* 10.4 (2011), pp. 1–30.

[40] Vassilis N Ioannidis, Antonio G Marques, and Georgios B Giannakis. “A recurrent graph neural network for multi-relational data”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 8157–8161.

[41] Mohsen Jamali and Martin Ester. “A matrix factorization technique with trust propagation for recommendation in social networks”. In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pp. 135–142.

[42] Amin Javari and Mahdi Jalili. “A probabilistic model to resolve diversity–accuracy challenge of recommendation systems”. In: *Knowledge and Information Systems* 44.3 (2015), pp. 609–627.

[43] R Jayashree and A Christy. “Improving the enhanced recommended system using Bayesian approximation method and normalized discounted cumulative gain”. In: *Procedia Computer Science* 50 (2015), pp. 216–222.

[44] Rong Jin and Luo Si. “A study of methods for normalizing user ratings in collaborative filtering”. In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. 2004, pp. 568–569.

[45] George Karypis. “Evaluation of item-based top-n recommendation algorithms”. In: *Proceedings of the tenth international conference on Information and knowledge management*. 2001, pp. 247–254.

[46] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[47] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

[48] Matevz Kunaver, Š Dobravec, and A Košir. “Using latent features to measure the diversity of recommendation lists”. In: *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2015, pp. 1230–1234.

[49] Matevž Kunaver and Tomaž Požrl. “Diversity in recommender systems–A survey”. In: *Knowledge-Based Systems* 123 (2017), pp. 154–162.

[50] Amaury L’Huillier, Sylvain Castagnos, and Anne Boyer. “Understanding usages by modeling diversity over time”. In: 2014.

[51] Kibeom Lee and Kyogu Lee. “Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items”. In: *Expert Systems with Applications* 42.10 (2015), pp. 4851–4858.

[52] Stan Lipovetsky. “Analytical closed-form solution for binary logit regression by categorical predictors”. In: *Journal of applied statistics* 42.1 (2015), pp. 37–49.

[53] Jian-Guo Liu, Kerui Shi, and Qiang Guo. “Solving the accuracy-diversity dilemma via directed random walks”. In: *Physical Review E* 85.1 (2012), p. 016118.

[54] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. “Content-based recommender systems: State of the art and trends”. In: *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[55] Hao Ma et al. “Recommender systems with social regularization”. In: *Proceedings of the fourth ACM international conference on Web search and data mining*. 2011, pp. 287–296.

[56] Jeremy Ma et al. “Diffusion filtering of graph signals and its use in recommendation systems”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4563–4567.

[57] Wenming Ma, Junfeng Shi, and Ruidong Zhao. "Normalizing item-based collaborative filter using context-aware Scaled baseline predictor". In: *Mathematical Problems in Engineering* 2017 (2017).

[58] Gonzalo Mateos et al. "Connecting the dots: Identifying network structure via graph signal processing". In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 16–43.

[59] Christian Matt et al. "Escaping from the filter bubble? The effects of novelty and serendipity on users' evaluations of online recommendations". In: (2014).

[60] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. "Spectral regularization algorithms for learning large incomplete matrices". In: *Journal of machine learning research* 11.Aug (2010), pp. 2287–2322.

[61] Federico Monti, Michael Bronstein, and Xavier Bresson. "Geometric matrix completion with recurrent multi-graph neural networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3697–3707.

[62] Tien T Nguyen et al. "Exploring the filter bubble: the effect of using recommender systems on content diversity". In: *Proceedings of the 23rd international conference on World wide web*. 2014, pp. 677–686.

[63] Katja Niemann and Martin Wolpers. "A new collaborative filtering approach for increasing the aggregate diversity of recommender systems". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 955–963.

[64] Athanasios N Nikolakopoulos et al. "Personalized diffusions for top-n recommendation". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. 2019, pp. 260–268.

[65] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[66] Umberto Panniello, Alexander Tuzhilin, and Michele Gorgoglione. "Comparing context-aware recommender systems in terms of accuracy and diversity". In: *User Modeling and User-Adapted Interaction* 24.1-2 (2014), pp. 35–65.

[67] Steffen Rendle et al. "BPR: Bayesian personalized ranking from implicit feedback". In: *arXiv preprint arXiv:1205.2618* (2012).

[68] Paul Resnick and Hal R Varian. "Recommender systems". In: *Communications of the ACM* 40.3 (1997), pp. 56–59.

[69] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook". In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[70] Alan Said et al. "Increasing diversity through furthest neighbor-based recommendation". In: *Proceedings of the WSDM* 12 (2012).

[71] Alan Said et al. "User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm". In: *Proceedings of the 2013 conference on Computer supported cooperative work*. 2013, pp. 1399–1408.

[72] Aliaksei Sandryhaila and Jose MF Moura. "Discrete signal processing on graphs: Frequency analysis". In: *IEEE Transactions on Signal Processing* 62.12 (2014), pp. 3042–3054.

[73] Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs". In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[74] Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. "On the graph Fourier transform for directed graphs". In: *IEEE Journal of Selected Topics in Signal Processing* 11.6 (2017), pp. 796–811.

[75] Badrul Sarwar et al. "Item-based collaborative filtering recommendation algorithms". In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 285–295.

[76] J Ben Schafer et al. "Collaborative filtering recommender systems". In: *The adaptive web*. Springer, 2007, pp. 291–324.

[77] Harry Sevi, Gabriel Rilling, and Pierre Borgnat. "Harmonic analysis on directed graphs and applications: from Fourier analysis to wavelets". In: *arXiv preprint arXiv:1811.1163* (2018).

[78] Bita Shams and Saman Haratizadeh. "Graph-based collaborative ranking". In: *Expert Systems with Applications* 67 (2017), pp. 59–70.

[79] Upendra Shardanand and Pattie Maes. "Social information filtering: algorithms for automating "word of mouth"". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995, pp. 210–217.

[80] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[81] Arlei Silva et al. "ProfileRank: finding relevant content and influential users based on information diffusion". In: *Proceedings of the 7th Workshop on Social Network Mining and Analysis*. 2013, pp. 1–9.

[82] Shivam Singh, Sujoy Bag, and Mamata Jenamani. "Relative similarity based approach for improving aggregate recommendation diversity". In: *2015 Annual IEEE India Conference (INDICON)*. IEEE. 2015, pp. 1–6.

[83] Jianing Sun et al. "Multi-Graph Convolution Collaborative Filtering". In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2019, pp. 1306–1311.

[84] Nava Tintarev, Matt Dennis, and Judith Masthoff. "Adapting recommendation diversity to openness to experience: A study of human behaviour". In: *International Conference on User Modeling, Adaptation, and Personalization*. Springer. 2013, pp. 190–202.

[85] Robin Van Meteren and Maarten Van Someren. "Using content-based filtering for recommendation". In: *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. 2000, pp. 47–56.

[86] Saúl Vargas. "Novelty and diversity enhancement and evaluation in recommender systems and information retrieval". In: *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 2014, pp. 1281–1281.

[87] Saúl Vargas and Pablo Castells. "Rank and relevance in novelty and diversity metrics for recommender systems". In: *Proceedings of the fifth ACM conference on Recommender systems*. ACM. 2011, pp. 109–116.

[88] Hongwei Wang et al. "Knowledge graph convolutional networks for recommender systems". In: *The world wide web conference*. 2019, pp. 3307–3313.

[89] Hongwei Wang et al. "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 417–426.

[90] Jing Wang and Jian Yin. "Combining user-based and item-based collaborative filtering techniques to improve recommendation diversity". In: *2013 6th International Conference on Biomedical Engineering and Informatics*. IEEE. 2013, pp. 661–665.

[91] Xiang Wang et al. "Kgat: Knowledge graph attention network for recommendation". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 950–958.

[92] Xiang Wang et al. "Neural graph collaborative filtering". In: *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 2019, pp. 165–174.

[93] Jacek Wasilewski and Neil Hurley. "Incorporating diversity in a learning to rank recommender system". In: *The twenty-ninth international flairs conference*. 2016.

[94] Liang Xiang et al. "Temporal recommendation on graphs via long-and short-term preference fusion". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 723–732.

[95] Rex Ying et al. "Graph convolutional neural networks for web-scale recommender systems". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 974–983.

[96] Wei Zeng et al. "Can dissimilar users contribute to accuracy and diversity of personalized recommendation?" In: *International Journal of Modern Physics C* 21.10 (2010), pp. 1217–1227.

[97] Mi Zhang and Neil Hurley. "Avoiding monotony: improving the diversity of recommendation lists". In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pp. 123–130.

[98] Tao Zhou et al. "Solving the apparent diversity-accuracy dilemma of recommender systems". In: *Proceedings of the National Academy of Sciences* 107.10 (2010), pp. 4511–4515.

[99] Cai-Nicolas Ziegler et al. "Improving recommendation lists through topic diversification". In: *Proceedings of the 14th international conference on World Wide Web*. ACM. 2005, pp. 22–32.

Accuracy-Diversity Trade-off in Recommender Systems via Graph Convolutions

Elvin Isufi[a], Matteo Pocchiari[b], Alan Hanjalic[c]

Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands

[a]*e.isufi-1@tudelft.nl*
[b]*m.pocchiari@student.tudelft.nl*
[c]*a.hanjalic@tudelft.nl*

Abstract

Graph convolutions, in both their linear and neural network forms, have reached state-of-the-art accuracy on recommender system (RecSys) benchmarks. However, recommendation accuracy is tied with diversity in a delicate trade-off and the potential of graph convolutions to improve the latter is unexplored. Here, we develop a model that learns joint convolutional representations from a nearest neighbor and a furthest neighbor graph to establish a novel accuracy-diversity trade-off for recommender systems. The nearest neighbor graph connects entities (users or items) based on their similarities and is responsible for improving accuracy, while the furthest neighbor graph connects entities based on their dissimilarities and is responsible for diversifying recommendations. The information between the two convolutional modules is balanced already in the training phase through a regularizer inspired by multi-kernel learning. We evaluate the joint convolutional model on three benchmark datasets with different degrees of sparsity. The proposed method can either trade accuracy to improve substantially the catalog coverage or the diversity within the list; or improve both by a lesser amount. Compared with accuracy-oriented graph convolutional approaches, the proposed model shows diversity gains up to seven times by trading as little as 1% in accuracy. Compared with alternative accuracy-diversity trade-off solutions, the joint graph convolutional model retains the highest accuracy while offering a handle to increase diversity. To our knowledge, this is the

first work proposing an accuracy-diversity trade-off with graph convolutions and opens the doors to learning over graphs approaches for improving such trade-off.
Keywords: Accuracy-diversity, collaborative filtering, graph filters, graph convolutional neural networks, graph signal processing.

1. Introduction

Despite accuracy is still the most dominant criterion guiding the design and evaluation of recommender systems (RecSys), numerous studies have shown that recommendation diversity –decreasing the similarity of the items in the recommended item list– significantly improves user satisfaction [1, 2, 3]. However, accuracy and diversity do not always go hand in hand and the development of a recommender system to consider both criteria typically requires dealing with an accuracy-diversity trade-off, a.k.a. balance or dilemma [4, 5].

The thin balance between accuracy and diversity is tied with the complexity and irregularity of the user-item relationships. Dealing with this complexity and irregularity has produced creative adaptations of existing RecSys paradigms, such as modifying accuracy-oriented algorithms into diversity-oriented counterparts [6, 7]. As an example, the nearest neighbor (NN) collaborative filtering connects entities (users, items) based on pairwise similarities and leverages these connections to interpolate the missing values from proximal entities. To secure accuracy for the target user, the system learns from the preferences of the most similar (nearest) neighboring entities. However, this typically leads to low recommendation diversity, as the NNs are too similar to the target user. In the search for a better accuracy-diversity trade-off, [7] proposed to look at the furthest neighbors (FNs) instead, i.e., a subset of k users that are most dissimilar to the target-user in terms of preferences. The assumption here is that recommending items FNs disliked most could bring more diversity while preserving an acceptable level of accuracy. Other RecSys algorithms focusing on improving diversity by connecting entitites based on their dissimilarity include [8, 9]. Alternative approaches aiming at trading accuracy with diversity include re-ranking [10, 11], leveraging side information [12, 13], or merging different models operating with different criteria [8, 9].

We believe that in order to obtain sufficient depth in understanding the accuracy-diversity trade-off, RecSys approaches are needed that can fully capture the abovementioned complex and irreguar user-item relationships. Graphs have proved themselves as excellent tools to develop such approaches [14], which made graph-based RecSys one of the most rapidly developing areas. Examples of graph-based RecSys approaches are diffusion-based recommendations [15], random walks [16], and graph neural network-based recommendations [17, 18, 19], to name a few.

In parallel to the increasing importance of graphs in the RecSys domain, the signal processing and machine learning communities have developed processing tools for data over graphs [14, 20]. The quintessential tool in these areas is the graph convolution. Graph convolutions extend to graphs the operation of convolution used to process temporal and spatial signals and serve as the building block for graph convolutional neural networks (GCNNs) [21]. Graph convolutions, both in their linear or GCNN form, have been successfully applied to RecSys reaching state-of-the-art accuracy [19, 22]. Despite the promise, graph convolutions have only been used to over-fit accuracy, leaving unexplored their ability to diversify recommendations and, ultimately, improve the accuracy-diversity trade-off.

In this work, we explore the potential of graph convolutions to improve the accuracy-diversity trade-off for recommender systems. We conduct this exploration by developing a novel model composed of two graph convolutional components, one providing accuracy-oriented recommendations from a NN graph, and one providing diversity-oriented recommendations from a FN graph. Differently from current works, we train a single joint model to fit the data, rather than using two separate models. Our specific contribution in this paper can be summarized as follows:

i) We propose a novel accuracy-diversity trade-off framework for RecSys via graph convolutions. The model operates on a NN graph to improve accuracy and on a FN graph to improve diversity. Each graph can capture user-user or item-item relationships, allowing to also include the hybrid settings, such as a user-NN and an item-FN graph. To the best of our knowledge, this is the first contribution providing an accuracy-diversity trade-off by using these hybrid setups. The proposed model relies only on the available ratings, which we find important since side information, such as metadata or context, can be unavailable or could bias the trade-off.

ii) We develop design strategies that estimate the joint model parameters in view of both accuracy and diversity. These design strategies are versatile to both rating and ranking frameworks. When the joint model is composed of linear graph convolutional filters, we analyze the optimality of the design problem and provide solutions for it.

iii) We analyze the joint model in the graph-spectral domain to provide an alternative interpretation of how the proposed approach balances accuracy with diversity. The joint model presents a band-stop behavior on both the NN and the FN graph, and builds recommendations by focusing on the extremely low and high graph frequencies.

iv) We evaluate two types of trade-offs: i) an accuracy-diversity trade-off w.r.t. catalog coverage (i.e., aggregated diversity), and ii) an accuracy-diversity trade-off w.r.t. list diversity (i.e., individual diversity). The first trade-off shows the models' ability to recommend niche items and personalize recommendations. The second trade-off shows the models' ability to diversify items in the list. The proposed models can either trade accuracy to boost substantially one diversity metric, or improve by a lesser amount.

The remainder of this paper is organized as follows. Section 2 places our contribution in the context of current literature. Section 3 reviews NN collaborative filtering from a graph convolutional learning perspective. Section 4 provides a high-level overview of the proposed approach. Sections 5 and 6 contain the design strategies for rating and ranking, respectively. Section 7 provides the graph-spectral analysis of the joint models, while Section 8 contains the numerical results. Section 9 discusses our findings.

2. Related Work

Accuracy-diversity trade-off. Along with the initial work [23], also [24] promotes the accuracy-diversity trade-off as a joint objective for effective RecSys. A popular direction to tweak this trade-off is by two-step approaches, in which re-ranking is applied to a retrieved list to boost diversity [10, 25]. The work in [11] re-ranks items based on rating variance in the neighborhood of a user, while [26] uses re-ranking to cover a larger portion of the catalog. Also [27, 28] diversify items to improve coverage in a user-personalized manner. The work in [27] optimizes the recommendation list to improve accuracy and diversity but reduce item popularity, while uses matching problems to improve coverage while minimizing the accuracy loss. Instead, [29] proposes a new metric to quantify diversity within a list and develops an optimization algorithm to improve it. Methods and algorithms in this category rely heavily on the initial recommendation list, which makes it difficult to attribute to which extent the improved trade-off is due to re-ranking or to the properties of the list.

Another category of approaches considers a single algorithm and leverage side information, such as metadata or context, to improve diversity. The work in [12] builds an item-item dissimilarity graph from features and uses this graph in a learning-to-rank framework. Also the work in [30] uses item features to provide a single method for matrix completion. Differently, [13] leverages context and evaluates different pre-filtering, post-filtering, and modeling schemes in terms of accuracy and diversity. Our approach, instead, balances accuracy with diversity without relying on side-information in both a learning-to-rate and learning-to-rank framework.

A third category of approaches modifies conventional accuracy-oriented algorithms to improve diversity. Authors in [31] build similarities by avoiding the influence of popular objects or high-degree users on the direction of random walk, which is shown heuristically to improve diversity. The work in [6] adjusts the calculation of user similarities in the classic NN approach to improve diversity. The work in [32] follows up on [12] and uses the regularizer in the learning-to-rank loss to improve diversity. In our view, the latter overloads the regularizer with an additional objective. Since the primary goal of the regularizer is to generalize the model to unseen data, leveraging it also to improve diversity leads to a triple accuracy-diversity-generalization trade-off, which is challenging to handle. Likely, one of the three objectives will be treated as a byproduct, which reduces the possibilities to steer the optimization of the accuracy-diversity trade-off. Differently, [7, 33] connect users based on their dissimilarities and propose the so-called furthest neighbor (FN) collaborative filtering –contrarily to the vanilla NN collaborative filtering. By using information from neighbors that a user disagrees with, this approach was shown to improve diversity by affecting accuracy by little. Yet, the degree to which FNs affect the accuracy-diversity trade-off remains insufficiently investigated. In our approach, we leverage both the NN and the FN in a joint convolutional model to better understand this influence.

While changing the inner-working mechanism of a single model can improve diversity, a single model often lacks the ability to capture the complex relationships contained in highly-sparse RecSys datasets. A fourth category of approaches overcomes this issue by working with an ensemble of models, also referred to as joint or hybrid models. These models have a higher descriptive power that can better balance accuracy with diversity at the expense of complexity, which is often of the same order of magnitude. Authors in [8] propose a joint collaborative filtering algorithm that leverages the influence of both similar and dissimilar users. The dissimilarity is computed by counting the items two users have consumed individually but not jointly. The predicted ratings from the similar and dissimilar users are merged into a final score and the influence of each group is controlled by a scalar. The way dissimilarity is computed ignores the fact that users may have consumed the same item, but rated it differently. Also, building dissimilarities from non-consumed items ignores the fact that a user may also like an item other users have consumed separately. To avoid the latter, we account for the ratings when building dissimilarities between entities. The authors of [9] follow a similar strategy as [8] and mix a heat spreading with a random walk to provide an accuracy-diversity trade-off. A probabilistic model to balance accuracy with diversity is further proposed in [34]. The latter considers the order in which items are consumed and proposes a joint model, which on one branch maximizes accuracy, while on the other, diversity. In contrast to this, we train the whole model jointly.

Graph convolutions in RecSys. Graph convolutions have been introduced to the RecSys domain only recently [35, 17, 36]. The approach proposed in [35], subsequently extended to [37], showed the NN collaborative filter is a non-parametric graph convolutional filter of order one. This work also showed that higher-orders parametric graph convolutional filters improve rating prediction. These graph convolutional filters are the basis to form GCNNs [21] and we will use them to balance accuracy with diversity. The work in [17] merges GCNNs with a recurrent neural network to complete the user-item matrix. Instead, [36] completes the matrix with a variational graph autoencoder, in which graph convolutions are performed by an order-one graph convolutional filter. The work in [38] uses the same graph convolution as [37], but uses it in a learning-to-rank setting. Although starting from different standpoints and naming the method differently, the two approaches are identical from a technical perspective. Taken together, [37] and [38] showed that linear graph convolutions may often suffice in highly sparse RecSys datasets. We shall corroborate this behavior also in the accuracy-diversity trade-off setting.

The authors in [18] deployed a GCNN with filters of order one on an augmented graph comprising the user-item bipartite interaction graph and also user-user and item-item proximal graphs. The work in [39] also learns from the user-item bipartite interactions through an order one GCNN, but augments the propagation rule to promote exchanges from similar items. Authors in [19] combine random walks with graph convolutions to perform recommendations in large-scale systems containing millions of nodes. Authors in [40] first build a user-specific knowledge graph and then apply graph neural networks to compute personalized recommendations. They also regularize the loss to enforce similar scores in adjacent items. Lastly, GCNNs have been used for location recommendation in [41]. Two GCNNs are run over two graphs, a point-of-interest graph and a social relationship graph to identify these points-of-interest for a user.

Altogether, these works show the potential of graph convolutions in changing the RecSys landscape. However, all approaches focus only on accuracy and ignore recommendation diversity. In this work, we consider graph convolutions to establish an accuracy-diversity trade-off in both a learning-to-rate and learning-to-rank setup.

3. Learning from Similar Nearest Neighbors

Consider a recommender system setting comprising a set of users $\mathcal{U} = \{1, \ldots, U\}$ and a set of items $\mathcal{I} = \{1, \ldots, I\}$. Ratings are collected in the user-item matrix $\mathbf{X} \in \mathbb{R}^{U \times I}$, in which entry X_{ui} contains the rating of user u to item i. Ratings are mean-centered[1], so that we can adopt the common convention $X_{ui} = 0$ if value (u, i) is missing. The objective is to populate matrix $\mathbf{X}$ by exploiting the relationships between users and items contained in the available ratings. We capture these relationships through a graph, which is built following the principles of NN collaborative filtering. This graph is used to predict ratings and the k items with the highest predicted rating form the recommendation list.

In user-based NNs, relationships are measured by the Pearson correlation coefficient. Consider a matrix $\mathbf{B} \in \mathbb{R}^{U \times U}$, in which entry B_{uv} measures the correlation between users u and v. Matrix $\mathbf{B}$ is symmetric and can be seen as the adjacency matrix of a *user-correlation* graph $\mathcal{G}_\text{u} = (\mathcal{U}, \mathcal{E}_\text{u})$. The vertex set of $\mathcal{G}_\text{u}$ is the user set $\mathcal{U}$ and the edge set $\mathcal{E}_\text{u}$ contains an edge $(u, v) \in \mathcal{E}_\text{u}$ only if $B_{uv} \neq 0$. Each item i is treated separately and user ratings are collected in

[1]Mean-centering can be done across users, items, or both. The proposed methods work with any choice.

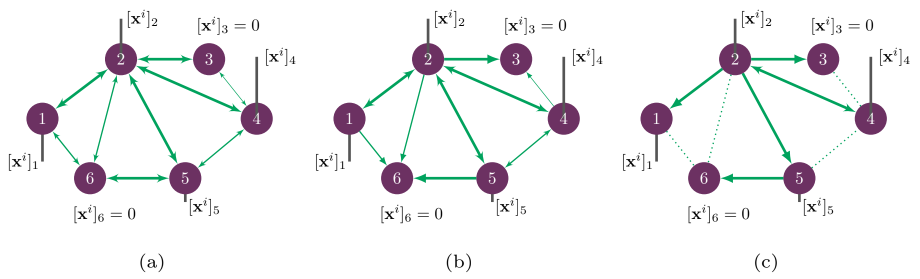


Figure 1: Building an item-specific graph $\mathbf{B}_i$ from the global correlation graph $\mathbf{B}$. (a) User graph $\mathbf{B}$. Nodes represent users and arrows correlations. Ratings to item i are a graph signal $\mathbf{x}_i$ shown by vertical bars. (b) Treat each undirected edge as two directed edges. Remove any directed edge starting from a user who did not rate item i. (c) Keep the $n = 1$ strongest incoming edge for each user representing the nearest neighbor. The adjacency matrix of this graph is $\mathbf{B}_i$.

vector $\mathbf{x}^i \in \mathbb{R}^U$ (corresponding to the ith column of $\mathbf{X}$). Vector $\mathbf{x}^i$ can be seen as a signal on the vertices of $\mathcal{G}_\mathrm{u}$, which uth entry $[\mathbf{x}^i]_u := X_{ui}$ is the rating of user u to item i, or zero otherwise [42]; see Figure 2 (a). Predicting ratings for item i translates into interpolating the missing values of graph signal $\mathbf{x}^i$. These values are estimated by shifting available ratings to neighboring users. First, we transform the global graph $\mathbf{B}$ to an item-specific graph $\mathbf{B}_i$ which contains only the top-n positively correlated edges per user and normalize their weights; see Figures 2 (b)-(c). The NN shifted ratings to immediate neighbors can be written as

$$\hat{\mathbf{x}}^i = \mathbf{B}_i \mathbf{x}^i \tag{1}$$

which holds true because matrix $\mathbf{B}_i$ respects the sparsity of the user-graph adapted to item i.

In item-based NNs, the procedure follows likewise. First, we construct an item-item correlation matrix $\mathbf{C} \in \mathbb{R}^{I \times I}$ in which entry C_{ij} is the Pearson correlation coefficient between items i and j. Matrix $\mathbf{C}$ is symmetric and it is the adjacency matrix of an item-correlation graph $\mathcal{G}_\mathrm{i} = (\mathcal{I}, \mathcal{E}_\mathrm{i})$. The vertex set of $\mathcal{G}_\mathrm{i}$ matches the item set $\mathcal{I}$ and the edge set $\mathcal{E}_\mathrm{i}$ contains an edge $(i,j) \in \mathcal{E}_\mathrm{i}$ only if $C_{ij} \neq 0$. Then, we consider the complementary scenario and treat each user

u separately. We collect the ratings of user u to all items in the graph signal $\mathbf{x}_u \in \mathbb{R}^I$ (corresponding to the uth row of $\mathbf{X}$). Finally, item-based NN interpolates the missing values in $\mathbf{x}_u$ through shifts to neighboring items. Building a user-specific graph $\mathbf{C}_u$ from $\mathbf{C}$, keeping only the top-n positively correlated edges per item, and normalizing the weights, we predict the ratings as

$$\hat{\mathbf{x}}_u = \mathbf{C}_u \mathbf{x}_u. \tag{2}$$

In either scenario, matrices $\mathbf{B}$, $\{\mathbf{B}_i\}_i$, $\mathbf{C}$, and $\{\mathbf{C}_u\}_u$ can be regarded as instances of a general graph adjacency matrix variable $\mathbf{S}$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ containing $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. We denote the available rating signal by $\mathbf{x}$ and the estimated rating signal by $\hat{\mathbf{x}}$ so that we write estimators (1) and (2) with the unified notation

$$\hat{\mathbf{x}} = \mathbf{S}\mathbf{x}. \tag{3}$$

As it follows from (3), NN estimators rely only on ratings present in the immediate surrounding of a node. But higher-order neighbors carry information that can improve prediction and their information should be accounted for accordingly to avoid destructive interference. Graph convolutional filters have proven themselves to be the tool for capturing effectively multi-resolution neighbor information when learning over graphs [14], including recent success in multi-resolution NN collaborative filtering [37, 17]. We detail in the sequel the graph convolutional filter and the respective GCNN extension.

3.1. Nearest Neighbor Graph Convolutional Filters

Estimator (3) accounts for the immediate neighbors to predict ratings. Similarly, we can account for the two-hop neighbors via the second-order shift $\mathbf{S}^2\mathbf{x}$. Writing $\mathbf{S}^2\mathbf{x} = \mathbf{S}(\mathbf{S}\mathbf{x})$ shows the second-order shift builds a NN estimator $\mathbf{S}(\cdot)$ on the previous one $\mathbf{S}\mathbf{x}$. We can also consider neighbors up to K-hops away as $\mathbf{S}^K\mathbf{x} = \mathbf{S}(\mathbf{S}^{K-1}\mathbf{x})$. To balance the information coming from the different resolutions $\mathbf{S}\mathbf{x}, \mathbf{S}^2\mathbf{x}, \ldots, \mathbf{S}^K\mathbf{x}$, we consider a set of parameters $\mathbf{h} = [h_0, \ldots, h_K]^\top$

and build the Kth order NN predictor

$$\hat{\mathbf{x}} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} := \mathbf{H}(\mathbf{S})\mathbf{x} \tag{4}$$

where $\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k$ is referred to as graph convolutional filter of order K [14, 42][2]. The ratings $\hat{\mathbf{x}}$ in (4) are built as a shift-and-sum of the available ratings $\mathbf{x}$. Particularizing $\mathcal{G}$ to $\mathcal{G}_{\mathrm{u}}$, (4) becomes a graph convolutional filter estimator over the user NN graph with estimated ratings for item i

$$\hat{\mathbf{x}}^i = \sum_{k=0}^{K} h_k \mathbf{B}_i^k \mathbf{x}^i := \mathbf{H}(\mathbf{B}_i)\mathbf{x}^i. \tag{5}$$

Particularizing $\mathcal{G}$ to $\mathcal{G}_{\mathrm{i}}$, (4) becomes a graph convolutional filter over the item NN graph with estimated ratings for user u

$$\hat{\mathbf{x}}_u = \sum_{k=0}^{K} h_k \mathbf{C}_u^k \mathbf{x}_u := \mathbf{H}(\mathbf{C}_u)\mathbf{x}_u. \tag{6}$$

Particularizing the order to $K = 1$ and the coefficients to $h_0 = 0$ and $h_1 = 1$, (5) and (6) become respectively the vanilla NN collaborative filters (1) and (2); i.e., vanilla NN collaborative filters are graph convolutional filters of order one.

Graph convolutional filters are defined by the $K+1$ parameters $\mathbf{h}$. Further, since the shift operator matrix $\mathbf{S}$ matches the NN structure, it is sparse, therefore, obtaining the output $\hat{\mathbf{x}}$ in (4) amounts to a complexity of order $\mathcal{O}(|\mathcal{E}|K)$. These properties are important to deal with scarcity of data and scalability.

3.2. *Nearest Neighbor Graph Convolutional Neural Networks*

Besides numerical efficiency, graph convolutional filters have high mathematical tractability and are the building block for graph convolutional neural networks [21]. To build a GCNN with the filter in (4), consider the composition

[2]The term $k = 0$, $\mathbf{S}^0\mathbf{x} = \mathbf{x}$, does not contribute to predicting ratings. We kept it in (4) since this term plays a role in GCNN in Section 3.2. Particularizing the graph to the directed cycle and the signal to a periodic discrete signal, operation (4) reduces to the temporal convolution operation.

of a set of L layers. The first layer $\ell = 1$ comprises a bank of F_1 filters $\mathbf{H}_1^f(\mathbf{S})$ each defined by coefficients $\{h_{1k}^f\}_k$. Each of these filters outputs graph signals $\mathbf{u}_1^f = \mathbf{H}_1^f(\mathbf{S})\mathbf{x}$, which are subsequently passed through a pointwise nonlinearity $\sigma(\cdot)$ to produce a collection of F_1 features $\mathbf{x}_1^f$ that constitute the output of layer $\ell = 1$, i.e.,

$$\mathbf{x}_1^f = \sigma\Big[\mathbf{u}_1^f\Big] = \sigma\Big[\mathbf{H}_1^f(\mathbf{S})\mathbf{x}\Big] = \sigma\Bigg[\sum_{k=0}^{K} h_{1k}^f \mathbf{S}^k \mathbf{x}\Bigg] \quad \text{for } f = 1, \ldots, F_1. \tag{7}$$

At subsequent intermediate layers $\ell = 2, \ldots, L-1$, the output features $\{\mathbf{x}_\ell^g\}_g$ of the previous layer $\ell - 1$ become inputs to a bank of $F_\ell F_{\ell-1}$ convolutional filters $\mathbf{H}_\ell^{fg}(\mathbf{S})$ each of which outputs the features $\mathbf{u}_\ell^{fg} = \mathbf{H}_\ell^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^g$. The filter outputs obtained from a common input $\mathbf{x}_{\ell-1}^g$ are aggregated and the result is passed through a nonlinearity $\sigma(\cdot)$ to produce the F_ℓ output features

$$\mathbf{x}_\ell^f = \sigma\Bigg[\sum_{g=1}^{F} \mathbf{u}_\ell^{fg}\Bigg] = \sigma\Bigg[\sum_{g=1}^{F} \mathbf{H}_\ell^{fg}(\mathbf{S})\mathbf{x}_{\ell-1}^g\Bigg] = \sigma\Bigg[\sum_{g=1}^{F}\sum_{k=0}^{K} h_{\ell k}^{fg}\mathbf{S}^k\mathbf{x}_{\ell-1}^g\Bigg] \quad \text{for } f = 1, \ldots, F_\ell. \tag{8}$$

Operation (8) is the propagation rule of a generic layer ℓ of the GCNN, which final outputs are the F_L features $\mathbf{x}_L^1, \ldots, \mathbf{x}_L^{F_L}$. These final convolutional features are passed through a shared multi-layer perceptron per node to map the F_L features per node n, $[x_{Ln}^1; \ldots; x_{Ln}^{F_L}]$, into the output estimate $\hat{x}_n$.

The GCNN can be seen as a map $\mathbf{\Phi}(\cdot)$ that takes as input a graph signal rating $\mathbf{x}$, an entity-specific graph $\mathbf{S}$, and a set of parameters $\mathcal{H} = \{h_{\ell k}^{fg}\}$ for all layers ℓ, orders k, and feature pairs (f, g). This map produces the estimate

$$\mathbf{\Phi}(\mathbf{x}; \mathbf{S}; \mathcal{H}) := \hat{\mathbf{x}}. \tag{9}$$

The GCNN leverages the coupling between the rating and the NN graph in the input layer to learn higher-order representations in the intermediate layers. This coupling is captured by the bank filters as per (4). Consequently, the GCNN inherits the numerical benefits of the graph convolutional filter. Denoting by $F = \max_\ell F_\ell$ the maximum number of features for all layers, the number of parameters defining the GCNN is of order $\mathcal{O}(F^2KL)$ while its computational complexity amounts to $\mathcal{O}(F^2KL|\mathcal{E}|)$. The latter are in the same order of magnitude as for the graph convolutional filter [cf. (4)], which is the particular GCNN map $\mathbf{\Phi}(\cdot)$ [cf. (9)] limited to the linear space. In the remainder of this paper, we will denote by $\mathbf{\Phi}(\cdot)$ both the filter and the GCNN and refer to them with the common terminology *graph convolutions*.

4. Accounting for *Di*similar Furthest Neighbors

We work with a NN similarity-graph $\mathcal{G}_{\mathrm{s}} = \{\mathcal{V}, \mathcal{E}_{\mathrm{s}}\}$ [cf. Sec. 3; Fig. 2] and a FN dissimilarity-graph $\mathcal{G}_{\mathrm{d}} = \{\mathcal{V}, \mathcal{E}_{\mathrm{d}}\}$. The dissimilarity graph is built by following the opposite principles of NNs, i.e., connecting each entity to its top-$\bar{n}$ most negatively related ones. To illustrate the latter, consider $\mathbf{C}$ captures item-item correlations while vector $\mathbf{x}_u$ contains the ratings of user u to all items. The user-specific dissimilarity graph has the adjacency matrix $\bar{\mathbf{C}}_u$ obtained from $\mathbf{C}$ by: i) removing any edge starting from an item not rated by user u; ii) keeping for each item i the $\bar{n}$ most negatively connections; iii) normalizing the resulting matrix to make $\bar{\mathbf{C}}_u$ right stochastic. In other words, defining $\overline{\mathcal{N}}_{iu}$ as the set containing the $\bar{n}$ most dissimilar items to i rated by user u, entry (i,j) of $\bar{\mathbf{C}}_u$ is

$$[\bar{\mathbf{C}}_u]_{ij} = \begin{cases} C_{ij} \big/ \sum_{j' \in \overline{\mathcal{N}}_{iu}} C_{ij'} & \text{if } j \in \overline{\mathcal{N}}_{iu} \\ 0 & \text{if } j \notin \overline{\mathcal{N}}_{iu} \end{cases} . \tag{10}$$

The normalization step is identical to the NN approach and ensures a similar magnitude of signal shifting [cf.(3)] on both NN and FN graphs. This normalization implies the entries of $\bar{\mathbf{C}}_u$ are positive, i.e., a larger value indicates a stronger dissimilarity. In the considered datasets, positive correlations go up to 1.0 while the negative correlations down to -0.2.

On each graph $\mathcal{G}_{\mathrm{s}}$ and $\mathcal{G}_{\mathrm{d}}$ we have a convolutional module $\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}; \mathbf{S}_{\mathrm{s}}; \mathcal{H}_{\mathrm{s}})$ and $\mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}; \mathbf{S}_{\mathrm{d}}; \mathcal{H}_{\mathrm{d}})$, outputting an estimate of the user-item matrix $\hat{\mathbf{X}}_{\mathrm{s}}$ and $\hat{\mathbf{X}}_{\mathrm{d}}$, respectively. We combine the two outputs in the joint estimate

$$\hat{\mathbf{X}} = (1-\alpha)\hat{\mathbf{X}}_{\mathrm{s}} + \alpha\hat{\mathbf{X}}_{\mathrm{d}} \tag{11}$$

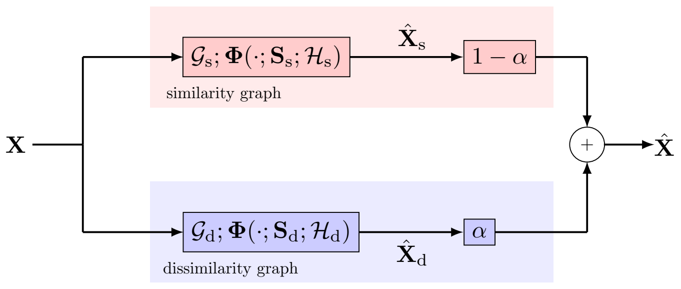

$\mathcal{G}_\mathrm{s}; \mathbf{\Phi}(\cdot; \mathbf{S}_\mathrm{s}; \mathcal{H}_\mathrm{s})$
$\hat{\mathbf{X}}_\mathrm{s}$
$1-\alpha$
similarity graph
$\mathbf{X}$
$+$
$\hat{\mathbf{X}}$
$\mathcal{G}_\mathrm{d}; \mathbf{\Phi}(\cdot; \mathbf{S}_\mathrm{d}; \mathcal{H}_\mathrm{d})$
$\hat{\mathbf{X}}_\mathrm{d}$
α
dissimilarity graph

Figure 2: Rating prediction with similar and dissimilar Graphs. We construct a NN graph $\mathcal{G}_\mathrm{s}$ capturing similarities between entities, and a FN graph $\mathcal{G}_\mathrm{d}$ capturing dissimilarities between entities. On each graph, we run a graph convolutional module $\mathbf{\Phi}(\cdot)$ with respective parameter set $\mathcal{H}_{(\cdot)}$ [cf. (4), (9)]. The estimated outputs are combined through a parameter α to obtain the final joint estimate $\hat{\mathbf{X}}$.

Table 1: Combinations between the similarity-graph $\mathcal{G}_\mathrm{s}$ and the dissimilarity-graph $\mathcal{G}_\mathrm{d}$.

		Dissimilar $\mathcal{G}_\mathrm{d}$	
Similar $\mathcal{G}_\mathrm{s}$		User	Item
	User	User-user (UU)	User-item (UI)
	Item	Item-user (IU)	Item-item (II)

where scalar $\alpha \in]0, 1[$ balances the influence of similar and dissimilar connections; Figure 2. Each graph $\mathcal{G}_\mathrm{s}$ or $\mathcal{G}_\mathrm{d}$ can be a user or an item graph and the graph convolutional modules $\mathbf{\Phi}(\cdot)$ can be linear [cf. (4)] or nonlinear [cf. (9)]. This framework yields eight combinations to investigate the trade-off. We limit ourselves to situations where the graph convolutional modules are the same on both graphs and focus on the four combinations in Table 1. To ease exposition, we shall discuss the theoretical methods with the hybrid combination user NN graph (i.e., $\mathcal{G}_\mathrm{s,u}$ with adjacency matrix $\mathbf{B}_i$ for item i) and item FN graph (i.e., $\mathcal{G}_\mathrm{d,i}$ with adjacency matrix $\bar{\mathbf{C}}_u$ for user u). This setting implies we predict rating $\hat{X}_{ui}$ by learning, on one side, from the coupling $(\mathcal{G}_\mathrm{s,u}, \mathbf{x}^i)$, and, on the other side, from the coupling $(\mathcal{G}_\mathrm{d,i}, \mathbf{x}_u)$.

Joint models like the one we consider are popular beyond the RecSys literature. The works in [43, 44] consider two different shift operators of the same graph to model signal diffusion with graph convolutional filters [cf. (4)]. This strategy is subsequently extended to GCNNs in [45]. Instead, [46, 47] exploit different relationships between data to build GCNNs. The common motivation in all these works is that a model based on a single graph (often capturing similarities between nodes [48]) or a single shift operator is insufficient to represent the underlying relationships. Therefore, we argue a joint model capturing different interactions helps representing better the data. A model based only on NNs fails giving importance to items that differ from the main trend. FNs account for this information and aid diversity. However, the information from FNs should be accounted for properly during training to keep the accuracy at the desired level. We detail this aspect in the upcoming two sections.

5. Learning for Rating

In this section, we estimate the joint model parameters w.r.t. the mean squared error (MSE) criterion. Analyzing the MSE quantifies also the trade-off for all items in the dataset (unbiasing the results from the user preferences in the list)[3]. The MSE also provides insights into the role played by the FNs. To this end, consider a training set of user-item pairs $\mathcal{T} = \{(u, i)\}$ for the available ratings in $\mathbf{X}$. Consider also the user-similarity graph $\mathcal{G}_{\mathrm{s,u}}$, the item-dissimilarity graph $\mathcal{G}_{\mathrm{d,i}}$, and their respective graph convolutions $\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}^i; \mathbf{B}_i; \mathcal{H}_{\mathrm{s}})$ and $\mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u; \bar{\mathbf{C}}_u; \mathcal{H}_{\mathrm{d}})$. We estimate parameters $\mathcal{H}_{\mathrm{s}}$ and $\mathcal{H}_{\mathrm{d}}$ by solving the regularized problem

$$\begin{aligned} \underset{\mathcal{H}_{\mathrm{s}}, \mathcal{H}_{\mathrm{d}}}{\text{minimize}} \quad & \frac{1}{\mu} \mathrm{MSE}_{(u,i)\in\mathcal{T}} \left(\mathbf{\Phi}_{\mathrm{s}}(\mathbf{x}^i; \mathbf{B}_i; \mathcal{H}_{\mathrm{s}}) + \mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u; \bar{\mathbf{C}}_u; \mathcal{H}_{\mathrm{d}}); \mathbf{X}_{\mathcal{T}} \right) + \frac{1}{2} \left(\frac{\|\mathcal{H}_{\mathrm{s}}\|_2^2}{1-\alpha} + \frac{\|\mathcal{H}_{\mathrm{d}}\|_2^2}{\alpha} \right) \\ \text{subject to} \quad & 0 < \alpha < 1 \end{aligned} \tag{12}$$

[3]We shall see that often methods trained with the MSE perform better in the list w.r.t. trained for ranking.

where $\text{MSE}_{(u,i)\in\mathcal{T}}(\cdot;\cdot)$ measures the fitting error w.r.t. the available ratings $\mathbf{X}_{\mathcal{T}}$, while the second term acts as an accuracy-diversity regularizer[4].

Scalar α controls the information flow from the NN and the FN graph. For $\alpha \to 0$, we have $\|\mathcal{H}_{\text{d}}\|_2^2/\alpha \gg \|\mathcal{H}_{\text{s}}\|_2^2/(1-\alpha)$, therefore, problem (12) forces parameters $\mathcal{H}_{\text{d}}$ to a smaller norm rather than using them to fit the data. Hence, this setting mainly leverages information from the similar graph $\mathbf{\Phi}_{\text{s}}(\cdot)$, ultimately, reducing the ensemble to a single NN graph convolutional model. For $\alpha \to 1$, we have the opposite case $\|\mathcal{H}_{\text{s}}\|_2^2/(1-\alpha) \gg \|\mathcal{H}_{\text{d}}\|_2^2/\alpha$, which implies the information from the similar graph plays little role in fitting since parameters $\mathcal{H}_{\text{s}}$ are forced towards zero. Hence, problem (12) mainly exploits information from FNs to reduce the MSE. Intermediate values of α closer to zero than to one lead to models where most information is leveraged from the NNs to keep the MSE low, while some information is taken from FNs to add diversity. We refer to α as the trade-off parameter. Scalar μ balances the fitting error with the overall regularization and allows generalizing the model to unseen data.

5.1. Graph convolutional filter

Recall the graph convolutional filter in (4) and consider graphs $\mathcal{G}_{\text{s,u}}$ and $\mathcal{G}_{\text{d,i}}$ can have different number of nodes. To account for this technicality in the design phase, we first transform the filters into a more manageable form. The filter output on the user-similarity graph $\mathbf{B}_i$ can be written as

$$\mathbf{\Phi}_{\text{s}}(\mathbf{x}^i;\mathbf{B}_i;\mathbf{h}_{\text{s}}) = \sum_{k=0}^{K} h_{s,k}\mathbf{B}_i^k\mathbf{x}^i := \mathbf{B}_{\text{s},i}\mathbf{h}_{\text{s}} \tag{13}$$

where the $U \times (K+1)$ matrix $\mathbf{B}_{\text{s},i} = [\mathbf{B}_i^0\mathbf{x}^i, \dots, \mathbf{B}_i^{K_s}\mathbf{x}^i]$ contains the shifted ratings of item i over graph $\mathbf{B}_i$ and vector $\mathbf{h}_{\text{s}} = [h_{s,0}, \dots, h_{s,K_s}]^\top$ the parameters. The uth row of $\mathbf{B}_{\text{s},i}$ is the $1 \times (K+1)$ vector $[\mathbf{B}_{\text{s},i}]_{u:}$ containing the shifted ratings of user u for item i. We then stack the $|\mathcal{T}|$ row vectors $[\mathbf{B}_{\text{s},i}]_{u:}$ for all

[4]In (12), we allowed ourselves a slight abuse of notation and indicated with $\|\mathcal{H}_{(\cdot)}\|_2^2$ the ℓ_2-norm squared of the vector containing the coefficients in set $\mathcal{H}_{(\cdot)}$.

pairs $(u,i) \in \mathcal{T}$ in

$$\mathbf{M}_{\mathrm{s}} = \big[\dots; [\mathbf{B}_{\mathrm{s},i}]_{u:}; [\mathbf{B}_{\mathrm{s},j}]_{u:}; \dots; [\mathbf{B}_{\mathrm{s},k}]_{v:}; [\mathbf{B}_{\mathrm{s},l}]_{v:}; \dots\big] \in \mathbb{R}^{|\mathcal{T}|\times(K+1)}.$$

The τth row of $\mathbf{M}_{\mathrm{s}}$ corresponds to the τth (u,i) tuple. Denoting by $\mathbf{x}_{\mathcal{T}} = \mathrm{vec}(\mathbf{X}_{\mathcal{T}})$ the $|\mathcal{T}| \times 1$ vector of available ratings, we can write the filter output for all training samples as $\hat{\mathbf{x}}_{\mathrm{s},\mathcal{T}} = \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}}$. Likewise, we can write the filter output over the item-dissimilarity graph $\bar{\mathbf{C}}_u$ as

$$\mathbf{\Phi}_{\mathrm{d}}(\mathbf{x}_u; \bar{\mathbf{C}}_u; \mathbf{h}_{\mathrm{d}}) = \sum_{k=0}^{K} h_{d,k}\bar{\mathbf{C}}_u^k \mathbf{x}_u := \bar{\mathbf{C}}_{\mathrm{d},u}\mathbf{h}_{\mathrm{d}} \tag{14}$$

where matrix $\bar{\mathbf{C}}_{\mathrm{d},u} = [\bar{\mathbf{C}}_u^0\mathbf{x}_u, \dots, \bar{\mathbf{C}}_u^{K_d}\mathbf{x}_u] \in \mathbb{R}^{I\times(K+1)}$ collects the shifted ratings of user u w.r.t. graph $\bar{\mathbf{C}}_u$ and vector $\mathbf{h}_{\mathrm{d}} = [h_{d,0}, \dots, h_{d,K}]^\top$ the filter parameters. Then, we construct the $|\mathcal{T}| \times (K+1)$ matrix $\mathbf{M}_{\mathrm{d}}$ by collecting the rows $[\bar{\mathbf{C}}_{\mathrm{d},u}]_{i:}$ for all $(u,i) \in \mathcal{T}$ so that to write $\hat{\mathbf{x}}_{\mathrm{d},\mathcal{T}} = \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}$.

With these in place, the design problem (12) particularizes to

$$\begin{aligned} \underset{\mathbf{h}_{\mathrm{s}},\mathbf{h}_{\mathrm{d}}}{\text{minimize}} \quad & \frac{1}{2\mu}\big\|\mathbf{x}_{\mathcal{T}} - \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}} - \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}\big\|_2^2 + \frac{1}{2}\bigg(\frac{\|\mathbf{h}_{\mathrm{s}}\|_2^2}{1-\alpha} + \frac{\|\mathbf{h}_{\mathrm{d}}\|_2^2}{\alpha}\bigg). \\ \text{subject to} \quad & 0 < \alpha < 1 \end{aligned} \tag{15}$$

which is a regularized-least squares problem in the filter coefficients $\mathbf{h}_{\mathrm{s}}$ and $\mathbf{h}_{\mathrm{d}}$. The closed-form solution for (15) can be found by setting the gradient to zero, i.e.,

$$-\frac{1}{\mu}\mathbf{M}_{\mathrm{s}}^\top\big(\mathbf{x}_{\mathcal{T}} - \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}} - \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}\big) + \frac{1}{1-\alpha}\mathbf{h}_{\mathrm{s}} = \mathbf{0} \tag{16a}$$

$$-\frac{1}{\mu}\mathbf{M}_{\mathrm{d}}^\top\big(\mathbf{x}_{\mathcal{T}} - \mathbf{M}_{\mathrm{s}}\mathbf{h}_{\mathrm{s}} - \mathbf{M}_{\mathrm{d}}\mathbf{h}_{\mathrm{d}}\big) + \frac{1}{\alpha}\mathbf{h}_{\mathrm{d}} = \mathbf{0} \tag{16b}$$

or equivalently solving the linear system of equations

$$\frac{1}{\mu}\begin{bmatrix}\mathbf{M}_{\mathrm{s}}^\top\mathbf{x}_{\mathcal{T}} \\ \mathbf{M}_{\mathrm{d}}^\top\mathbf{x}_{\mathcal{T}}\end{bmatrix} = \begin{bmatrix}\mathbf{M}_{\mathrm{s}}^\top\mathbf{M}_{\mathrm{s}} - \frac{1}{1-\alpha}\mathbf{I} & \mathbf{M}_{\mathrm{s}}^\top\mathbf{M}_{\mathrm{d}} \\ \mathbf{M}_{\mathrm{d}}^\top\mathbf{M}_{\mathrm{s}} & \mathbf{M}_{\mathrm{d}}^\top\mathbf{M}_{\mathrm{d}} - \frac{1}{\alpha}\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{h}_{\mathrm{s}} \\ \mathbf{h}_{\mathrm{d}}\end{bmatrix}. \tag{17}$$

If the matrix inversion in (17) is ill-conditioned, we can always solve (15) with of-the-shelf iterative methods. The above procedure leads to an optimal balance between the information coming from the NNs and the FNs.

5.2. *Graph convolutional neural network*

We now consider models $\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_{\rm s})$ and $\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_{\rm d})$ are GCNNs running respectively over graphs $\mathbf{B}_i$ and $\bar{\mathbf{C}}_u$. Particularizing (12) to this setting implies solving

$$\begin{aligned}&\underset{\mathcal{H}_{\rm s},\mathcal{H}_{\rm d}}{\text{minimize}} \quad \frac{1}{2\mu}\sum_{(u,i)\in\mathcal{T}}\left|X_{ui}-\Big[\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_{\rm s})\Big]_u-\Big[\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_{\rm d})\Big]_i\right|^2+\frac{1}{2}\left(\frac{\|\mathcal{H}_{\rm s}\|_2^2}{1-\alpha}+\frac{\|\mathcal{H}_{\rm d}\|_2^2}{\alpha}\right)\\&\text{subject to} \quad 0<\alpha<1\end{aligned} \tag{18}$$

where $[\mathbf{\Phi}_{\rm s}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_{\rm s})]_u$ is the user-similarity GCNN output for user u and $[\mathbf{\Phi}_{\rm d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_{\rm d})]_i$ is the item-dissimilarity GCNN output for item i. Problem (18) preserves the trade-offs of the general version (12), but it is non-convex and little can be said about its global optimality. However, because of the compositional form of the GCNN, we can estimate parameters $\mathcal{H}_{\rm s}$ and $\mathcal{H}_{\rm d}$ via standard backpropagation since the graph convolutional filters are linear operators in the respective parameters [49]. The following remark is in order.

Remark 1. In (18), we considered the accuracy-diversity parameter α only in the regularizer and not also in the fitting part as in (11). We found that including the latter to the MSE term leads to a more conservative solution towards diversity. We have consistently seen that keeping α only in the regularizer allows for a better trade-off. Furthermore, the regulariser in (18) does not need be rational in α, but can be in any form as long as it balances the NNs with the FNs. An alternative is $\Omega(\mathcal{H}_{\rm s};\mathcal{H}_{\rm d};\alpha)=\frac{1}{2}\big(\alpha\|\mathcal{H}_{\rm s}\|_2^2+(1-\alpha)\|\mathcal{H}_{\rm d}\|_2^2\big)$.

6. Learning for Ranking

This section designs the joint model for ranking. We considered the Bayesian personalized ranking (BPR), which is a state-of-the-art learn-to-ranking framework [50]. BPR considers the rating difference a user u has given to two items i and j. Let symbol $i\succ_u j$ indicate user u rated item i more than item j and augment the training set as $\mathcal{T}\subseteq\mathcal{U}\times\mathcal{I}\times\mathcal{I}$ to contain triples of the form $\mathcal{T}=\{(u,i,j)|i\succ_u j\}$. For each available tuple (u,i) we created four triples

$\{(u,i,j)\}_j$ such that $X_{ui} > X_{uj}$ following [50]. Subsequently, the estimated ratings for tuples (u,i) and (u,j) are respectively

$$\begin{aligned}\hat{X}_{ui}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) &= \left[\mathbf{\Phi}_\mathrm{s}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_\mathrm{s})\right]_u + \left[\mathbf{\Phi}_\mathrm{d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_\mathrm{d})\right]_i \\ \hat{X}_{uj}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) &= \left[\mathbf{\Phi}_\mathrm{s}(\mathbf{x}^j;\mathbf{B}_j;\mathcal{H}_\mathrm{s})\right]_u + \left[\mathbf{\Phi}_\mathrm{d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_\mathrm{d})\right]_j\end{aligned} \tag{19}$$

and the utility function is

$$\hat{X}_{uij}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) = \hat{X}_{ui}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) - \hat{X}_{uj}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) \tag{20}$$

which expresses the rating difference as a parametric relationship between user u, item i, and item j. The utility function is used to estimate parameters $\mathcal{H}_\mathrm{s}, \mathcal{H}_\mathrm{d}$ by maximizing the likelihood

$$p(i \succ_u j|\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) := \sigma\Big(\hat{X}_{uij}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d})\Big) = \Big(1 + e^{-\hat{X}_{uij}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d})}\Big)^{-1} \tag{21}$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the logistic sigmoid function [50]. By applying the natural logarithm (monotonic increasing) to (21) and regularizing it, we can estimate the joint convolutional model parameters by solving the regularized optimization problem

$$\begin{aligned}&\underset{\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}}{\text{minimize}} \quad -\frac{1}{\mu}\sum_{(u,i,j)\in\mathcal{T}} \ln\sigma\big(\hat{X}_{uij}\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}\big) + \alpha\|\mathcal{H}_\mathrm{s}\|_2^2 + (1-\alpha)\|\mathcal{H}_\mathrm{d}\|_2^2 \\ &\text{subject to} \quad 0 < \alpha < 1\end{aligned} \; . \tag{22}$$

Differently from (5), the regularizer in (22) is linear in α. We opted for this choice because the linear was more robust to μ. Nevertheless, the regulariser in (22) respects the same trend as that in (5): for $\alpha \to 0$, NNs are mainly used for fitting since $\alpha\|\mathcal{H}_\mathrm{s}\|_2^2 \to 0$; vice-versa, for $\alpha \to 1$ the FNs are mainly used for fitting since $(1-\alpha)\|\mathcal{H}_\mathrm{d}\|_2^2 \to 0$.

6.1. Graph convolutional filter

Particularizing the convolutional models to filters [cf. (4)], (19) becomes

$$\begin{aligned}\hat{X}_{ui}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) &= \left[\mathbf{B}_{\mathrm{s},i}\right]_{u:}\mathbf{h}_\mathrm{s} + \left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_{i:}\mathbf{h}_\mathrm{d} \\ \hat{X}_{uj}(\mathcal{H}_\mathrm{s},\mathcal{H}_\mathrm{d}) &= \left[\mathbf{B}_{\mathrm{s},j}\right]_{u:}\mathbf{h}_\mathrm{s} + \left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_{j:}\mathbf{h}_\mathrm{d}\end{aligned} \tag{23}$$

where $\left[\mathbf{B}_{\mathrm{s},i}\right]_{u:}$ is the uth row of the similar user-NN graph matrix $\mathbf{B}_{\mathrm{s},i}$ [cf. (13)] and $\left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_{i:}$ is the ith row of the dissimilar item-FN graph matrix $\bar{\mathbf{C}}_{\mathrm{d},u}$ [cf. (14)]. Substituting (23) into (22) leads to

$$\begin{aligned}
\underset{\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}}{\text{minimize}} \quad & -\frac{1}{\mu}\sum_{(u,i,j)\in\mathcal{T}} \ln\sigma\big(\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}\big) + \Big(\alpha\|\mathbf{h}_\mathrm{s}\|_2^2 + (1-\alpha)\|\mathbf{h}_\mathrm{d}\|_2^2\Big)\\
\text{subject to} \quad & \hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}) = \big(\left[\mathbf{B}_{\mathrm{s},i}\right]_{u:}\mathbf{h}_\mathrm{s} + \left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_{i:}\mathbf{h}_\mathrm{d}\big) - \big(\left[\mathbf{B}_{\mathrm{s},j}\right]_{u:}\mathbf{h}_\mathrm{s} + \left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_{j:}\mathbf{h}_\mathrm{d}\big)^{\cdot}\\
& 0<\alpha<1
\end{aligned} \tag{24}$$

Function $-\ln\sigma(\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d}))$ is convex since it involves a log-sum-exp of an affine function [51]. Consequently, problem (24) is convex in $\mathbf{h}_\mathrm{s}$ and $\mathbf{h}_\mathrm{d}$. Convexity guarantees we can find a minimizer for (24) but not a closed-form solution. In fact, finding an analytical solution for logistic fitting problems is notoriously difficult except for particular instances [52]. However, we can get the optimal parameters for (24) through the stochastic gradient descent updates

$$\mathbf{h}_\mathrm{s} \leftarrow \mathbf{h}_\mathrm{s} + \frac{\gamma}{\mu}\left[e^{-\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d})}\sigma\big(\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d})\big)\big(\left[\mathbf{B}_{\mathrm{s},i}\right]_u^\top - \left[\mathbf{B}_{\mathrm{s},j}\right]_u^\top\big) - 2\alpha\mathbf{h}_\mathrm{s}\right] \tag{25a}$$

$$\mathbf{h}_\mathrm{d} \leftarrow \mathbf{h}_\mathrm{d} + \frac{\gamma}{\mu}\left[e^{-\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d})}\sigma\big(\hat{X}_{uij}(\mathbf{h}_\mathrm{s},\mathbf{h}_\mathrm{d})\big)\big(\left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_i^\top - \left[\bar{\mathbf{C}}_{\mathrm{d},u}\right]_j^\top\big) - 2(1-\alpha)\mathbf{h}_\mathrm{d}\right] \tag{25b}$$

where γ is the stepsize. These optimal parameters guarantee the best learning-to-rank solution for any balance between the NNs and FNs (α) and between fitting and generalization (μ).

6.2. Graph convolutional neural network

When $\mathbf{\Phi}_\mathrm{s}(\mathbf{x}^i;\mathbf{B}_i;\mathcal{H}_\mathrm{s})$ and $\mathbf{\Phi}_\mathrm{d}(\mathbf{x}_u;\bar{\mathbf{C}}_u;\mathcal{H}_\mathrm{d})$ are GCNNs, the BPR optimization problem is that in (22). Because of the nonlinearity, it is difficult to establish if a global minimum exists and we should seek for a satisfactory local minimum. Since cost (22) is differentiable w.r.t. $\mathcal{H}_\mathrm{s}$ and $\mathcal{H}_\mathrm{d}$, we can achieve this local minimum through conventional backpropagation.

Either estimated for rating or ranking, the coefficients of the joint model dictate the filter behavior (either directly or within the GCNN layers) on the

NN and FN graphs. Besides analyzing the filter behavior in the node domain (as multi-hop rating aggregation) and in the respective cost functions (as accuracy-diversity trade-off), we can also get insight on the trade-off by analyzing the graph convolutional modules in the graph spectral domain [14]. We discuss this aspect next.

7. Spectral explanation

We conduct here a spectral analysis of graph convolutions to show they act as band-stop filters on both NN and FN graphs. First, we recall the concept of Fourier transform for signals on directed graphs [53]. Assuming the shift operator $\mathbf{S}$ is diagonalizable, we can write $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ with eigenvector matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ and complex eigenvalues $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$. The graph Fourier transform (GFT) of signal $\mathbf{x}$ is

$$\tilde{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}. \tag{26}$$

The ith GFT coefficient $\tilde{x}_i$ of $\tilde{\mathbf{x}}$ quantifies the contribution of the ith eigenvector $\mathbf{u}_i$ to expressing the variability of $\mathbf{x}$ over the graph. The latter is analogous to the discrete Fourier transform for temporal or spatial signals. In this analogy, the complex eigenvalues $\lambda_i \in \mathbf{\Lambda}$ are referred to as the graph frequencies [54, 42]. The inverse transform is $\mathbf{x} = \mathbf{U}\tilde{\mathbf{x}}$.

To measure the graph signal variability, we follow [53] and order the graph frequencies λ_i based on their distance from the maximum eigenvalue $\lambda_{\max}(\mathbf{S})$. This ordering is based on the notion of total variation (TV), which for the eigenpair $(\lambda_i, \mathbf{u}_i)$ is defined as

$$\text{TV}(\mathbf{u}_i) = \left|1 - \frac{\lambda_i}{\lambda_{\max}(\mathbf{S})}\right| \|\mathbf{u}_n\|_1 \tag{27}$$

where $\|\cdot\|_1$ is the ℓ_1-norm. The closer λ_i to the maximum eigenvalue $\lambda_{\max}(\mathbf{S})$, the smoother the corresponding eigenvector $\mathbf{u}_i$ over the graph (i.e., values on neighboring nodes are similar). If signal $\mathbf{x}$ changes little (e.g., similar users have similar ratings), the corresponding GFT $\tilde{\mathbf{x}}$ has nonzero entries mostly in entries

$\tilde{x}_i$ which index i corresponds to a low graph frequency $\lambda_i \to \lambda_{\max}(\mathbf{S})$ (low TV). Contrarily, if signal $\mathbf{x}$ varies substantially in connected nodes, the GFT $\tilde{\mathbf{x}}$ has nonzero values also in entries $\tilde{x}_i$ which index i corresponds to a high graph frequency $\lambda_i \gg \lambda_{\max}(\mathbf{S})$ (high TV); refer to [53, 14] for further detail.

With this analogy in place, we substitute the eigendecomposition $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ into the graph convolutional filter (4) and obtain the filter input-output relationship in the spectral domain

$$\tilde{\hat{\mathbf{x}}} = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k \tilde{\mathbf{x}} := \mathbf{H}(\mathbf{\Lambda})\tilde{\mathbf{x}} \tag{28}$$

where $\tilde{\hat{\mathbf{x}}} = \mathbf{U}^{-1}\hat{\mathbf{x}}$ is the GFT of the output and $\mathbf{H}(\mathbf{\Lambda}) = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k$ contains the filter frequency response on the main diagonal. Relation (28) shows in first place graph convolutional filters respect the convolutional theorem because they act as a pointwise multiplication between the filter transfer function $\mathbf{H}(\mathbf{\Lambda})$ and the input GFT $\tilde{\mathbf{x}}$. Therefore, analyzing $\mathbf{H}(\mathbf{\Lambda})$ shows how the filter processes the input ratings $\mathbf{x}$ to estimate $\tilde{\hat{\mathbf{x}}}$. We evaluate the frequency responses of the filter and respective GCNN when deployed on the similar NN and the dissimilar FN graphs for the MovieLens 100K dataset. The latter allows for a direct comparison with the vanilla NN and the graph convolutional NN filter [37].

7.1. Graph convolutional filters

Recall the joint model with user-NN filter $\mathbf{H}(\mathbf{B}_i) = \sum_{k=0}^{K} h_{s,k}\mathbf{B}_i^k\mathbf{x}^i$ [cf. (13)] and an item-FN filter $\mathbf{H}(\bar{\mathbf{C}}_u) = \sum_{k=0}^{K} h_{d,k}\bar{\mathbf{C}}_u^k\mathbf{x}_u$ [cf. (14)]. Substituting the eigendecompositions $\mathbf{B}_i = \mathbf{U}_{\text{s},i}\mathbf{\Lambda}_{\text{s},i}\mathbf{U}_{\text{s},i}^{-1}$ and $\bar{\mathbf{C}}_u = \bar{\mathbf{U}}_{\text{d},u}\bar{\mathbf{\Lambda}}_{\text{d},u}\bar{\mathbf{U}}_{\text{d},u}^{-1}$, we can write the outputs in the graph frequency domain respectively as

$$\tilde{\hat{\mathbf{x}}}^i = \sum_{k=0}^{K} h_{\text{s},k}\mathbf{\Lambda}_{\text{s},i}^k\tilde{\mathbf{x}}^i \quad \text{and} \quad \tilde{\hat{\mathbf{x}}}_u = \sum_{k=0}^{K} h_{\text{d},k}\bar{\mathbf{\Lambda}}_{\text{d},u}^k\tilde{\mathbf{x}}_u. \tag{29}$$

In (29), $\mathbf{H}(\mathbf{\Lambda}_{\text{s},i}) := \sum_{k=0}^{K} h_{\text{s},k}\mathbf{\Lambda}_{\text{s},i}^k$ and $\mathbf{H}(\bar{\mathbf{\Lambda}}_{\text{d},u}) := \sum_{k=0}^{K} h_{\text{d},k}\bar{\mathbf{\Lambda}}_{\text{d},u}^k$ are the responses of filters $\mathbf{H}(\mathbf{B}_i)$ and $\mathbf{H}(\bar{\mathbf{C}}_u)$, respectively. To estimate the responses, we first get the parameters from (15) for rating or (22) for ranking and order the eigenvalues $\lambda_{n,i}$ (resp. $\lambda_{n,u}$) of each $\mathbf{B}_i$ (resp. $\bar{\mathbf{C}}_u$) as per the total variation in

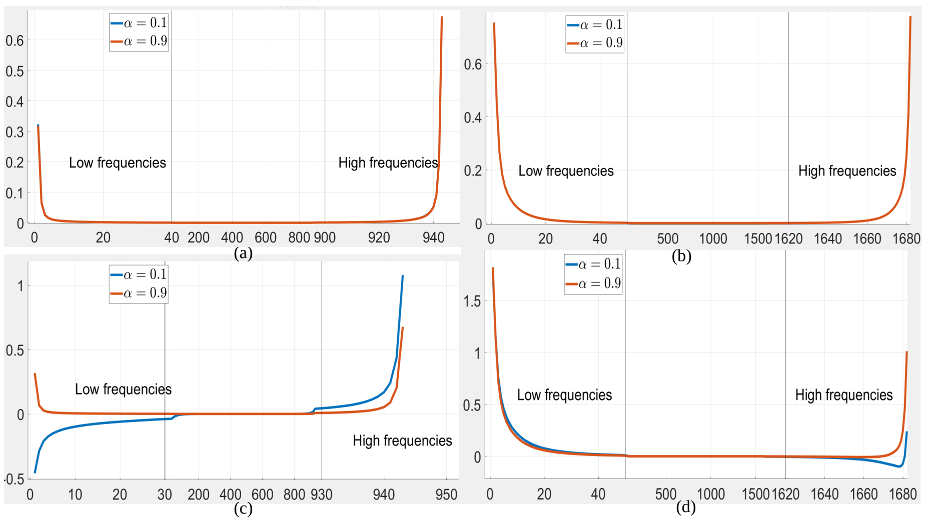


Figure 3: Frequency responses of the graph filters over a user-NN and an item-FN graph. The horizontal axis is the graph frequency index, while the vertical axis is the estimated frequency response. (Top) Filters designed w.r.t. the mean squared error criterion in (15). (Bottom) Filters designed w.r.t. the Bayesian personalized ranking criterion in (22). (a-c) Frequency response of the user-NN graph filter. (b-d) Frequency response of the item-FN graph filter.

(27). Subsequently, for each $\mathbf{B}_i$ (resp. $\bar{\mathbf{C}}_u$) we record the frequency responses $\{\mathbf{H}(\mathbf{\Lambda}_{\mathrm{s},i})\}_i$ (resp. $\{\mathbf{H}(\bar{\mathbf{\Lambda}}_{\mathrm{d},u})\}_u$) and average them across all items I (resp. users U) to get a single frequency response over the user-NN graph (resp. item-FN graph). The frequency responses are shown in Figure 3 for different values of α.

In all cases, we observe a band-stop behavior since more than 90% of the response in the middle frequencies is zero. The latter corroborates the behavior of the vanilla and graph convolutional NN filter [37]. Another behavior inherited from the NN/FN graphs is that filters preserve the extreme low and high graph frequencies. Low graph frequencies are signals with a small total variation [cf. (27)], while high graph frequencies are signals with a high total variation.

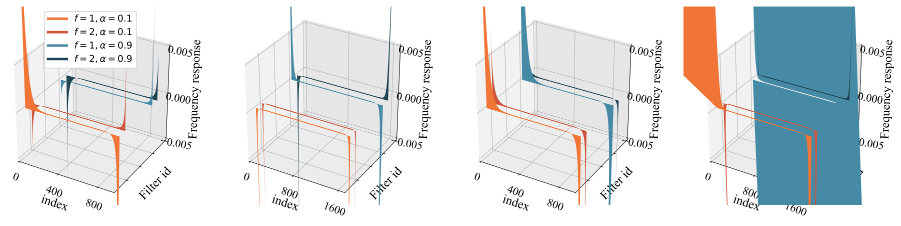
- *In the user-NN graph*, low frequencies represent signals where similar users give similar ratings. This part is the global trend of preferences among similar users, which is leveraged to predict ratings. High frequencies represent discordant ratings between similar users for a particular item and can be seen as a primitive source for diversity.

- *In the item-FN graph*, the spectral behavior is the same but implications are different. Low frequencies represent ratings with a small difference in *dissimilar* neighboring items; implying, a user u gave similar ratings to dissimilar items. These low frequencies may also be because users rate negatively a subset of dissimilar connected items and positively another subset of dissimilar connected items. The high pass components represent ratings changing significantly between neighboring dissimilar items; e.g., one of the two dissimilar items sharing an edge is rated positively while the other negatively. This part contributes towards keeping high the recommendation accuracy while relying on negative correlations between items.

These insights show the joint linear models eliminate irrelevant features (band-stop behavior), smooth out ratings (low frequencies), and preserve discriminative features to aid diversity (high frequencies). This phenomenon is observed for different values of α (importance on NNs vs. FNs) and design criteria (MSE [cf. (15)] vs. BPR [cf. (24)]). The frequency response changes less with α in the MSE design (lines differ by 10^{-3}) than in BPR. This might be because the MSE focuses on the average rating prediction for all items (preferred or not), while the BPR prioritizes a subset of most preferred items. In BPR, we also observed a stronger band-stop behavior for $\alpha \rightarrow 1$ meaning the joint model focuses even more on extreme frequencies to predict ratings. This suggests the model relies on the average trend on both graphs (lower frequencies) and on highly dissimilar values in adjacent entities (higher frequencies).

7.2. Graph convolutional neural networks

We now analyze the frequency response of the filters in the GCNN (8). Figure 4 illustrates the latter for a one-layer GCNN with $F = 2$ filters over each graph. We observe again the strong band-stop behavior. In the NN graph, the stopped band is narrower than in the FN graph, and it is narrower if the GCNN is learned for ranking than rating. The band-stop behavior and the



(a) (b) (c) (d)

Figure 4: Frequency responses of the bank of $F = 2$ graph convolutional filters of a one-layer GCNN [cf. (7)] over a user NN graph and an item FN graph. The z-axis is the frequency response that is cropped to improve visibility. The other two axis are the graph frequency index and the filter number. (a) and (b) Filters designed w.r.t. the mean squared error [cf. (15)]. (c) and (d) Filters designed w.r.t. the Bayesian personalized ranking [cf. (22)]. (a) and (c) Filters on the user NN graph. (b) and (d) Filters on the item FN graph.

Table 2: Features of the considered datasets.

Data set	Users	Items	Ratings	Sparsity
MovieLens100k	943	1,682	100,000	6.3×10^{-2}
Douban	3,000	3,000	136,891	1.5×10^{-2}
Flixster	3,000	3,000	26,173	2.9×10^{-3}

increased focus on the extremly low and high graph frequencies suggest the GCNN leverages the information in a similar way as the linear counterpart. We refer to the previous section to avoid repetition. Lastly, we remark the band-stop behavior is also observed in the vanilla NN [cf. (1)-(2)] and in the linear graph convolutional NN filter [37].

8. Numerical Experiments

This section corroborates the proposed schemes through experiments with three real datasets of different sparsity, namely, MovieLens100k [55], Douban [56] and Flixster [57]. Table 2 summarizes their features. We evaluate the trade-offs of the joint models for all combinations in Table 1. We considered both the linear [cf. (4)] and the nonlinear graph convolutional models [cf. (8)] designed for rating [cf. (12)] and ranking [cf. (22)], leading to 16 combinations. We

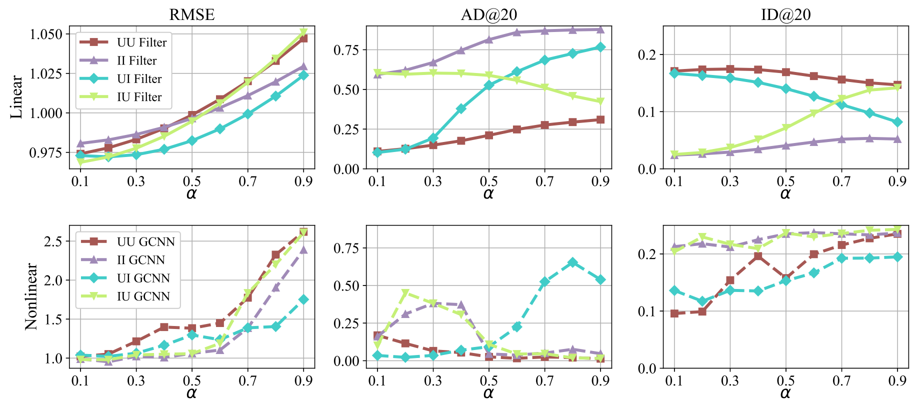


Figure 5: RMSE, AD@20 and ID@20 as a function of the accuracy-diversity parameter α for models optimized for rating. As more information from the dissimilar connections is included, the RMSE deteriorates but diversity improves. The RMSE of the GCNN is more sensitive to α as its hyperparameters are not tuned.

considered the same data pre-processing and train-test split as in [17].

We quantified accuracy through the root MSE (RMSE) –the lower the better– and the normalized discounted cumulative gain @k (NDCG@k) –the higher the better– and diversity through the aggregated diversity @k (AD@k) and individual diversity @k (ID@k) –both the higher the better [58, 59, 10]. The RMSE measures the fitting error for all ratings, while the NDCG@k accounts also for the item relevance in a list of length k. The AD@k is a global at-the-dataset metric and measures the fraction of all items included in *all* recommendation lists of length k. The ID@k is a local at-the-user metric and measures the average diversity in *each* recommendation list. A high ID@k does not imply a high AD@k and vice-versa [26, 60]. Appendix A provides further detail[5].

We considered a GCNN architecture composed of a single hidden layer with

[5]We have also evaluated the models with different metrics including: the mean absolute value (MAE), a surrogate of the RMSE for rating; precision and recall @k, which are ranking-oriented metrics for accuracy; and entropy diversity, which measures the models' ability to recommend items in the long-tail. We have observed these metrics respect the accuracy-diversity trade-off we report and have omitted them for conciseness.

two parallel filters. We trained the GCNN using the ADAM optimizer with the default parameters [61] and sought different learning rates γ and fitting-regularizer parameter μ. To limit the search of different hyperparameters, we proceeded with the following rationale. First, we performed an extensive parameter analysis in the MovieLens100k dataset, since this dataset is common in the two most similar graph convolutional works [17, 37] and in the accuracy-diversity trade-off works [8, 26, 25]. We then used the best performing setting in this dataset and corroborated the trade-offs in the remaining two. Second, we chose the hyperparameters of the similarity graph (number of nearest neighbor, filter order, length of the recommendation list) from the linear graph convolutional filter optimized for rating [cf. (15)] [37]. Besides being a faster design method to seek different parameters, this strategy allowed evaluating also the accuracy-diversity trade-off of the graph convolutional NN filter. Finally, we kept fixed these parameters for the NN graph and evaluated different combinations on the FN graph.

8.1. Accuracy-Diversity Trade-off for Rating

We first study the trade-off when the joint models are trained for rating [cf. Sec. 5]. For the NN module, we used the parameters derived in Appendix B. For the FN module, we fixed the number of neighbors to the arbitrary common value 40, evaluated different filter orders $K \in \{1, 2, 3\}$, and show the best results.

Figure 5 shows the results for the combinations in Table 1 as a function of $\alpha \in [0.1, 0.9]$. As we increase the influence of FNs ($\alpha \to 1$), the RMSE increases. The linear filters are more robust to α than the GCNN. We attribute the latter to the convexity of their design problem. Increasing α increases diversity, while the AD and ID exhibit opposite behavior. Values of α up to 0.5 offer a good trade-off as the RMSE remains unaffected but diversity increases substantially.

To further quantify the trade-off, we allow the RMSE to deteriorate by at most 3% w.r.t. the NN setup [cf. Appendix B] and pick a value of α that respects such constraint. Table 3 compares the different models. For a user NN graph, the joint models (i.e., UU and UI) boost substantially one diversity

Table 3: RMSE, AD@20 and ID@20 for different models optimized for rating. In brackets we show the change in percentage of the proposed joint models w.r.t. the NN counterpart.

		RMSE	AD@20	ID@20
User Linear	User NN	0.96	0.19	0.02
	UU filter	0.98 (+2.1%)	0.15 (-21.5%)	0.17 (**+750%**)
	UI filter	0.98 (+2.1%)	0.53 (**+178%**)	0.14 (**+600%**)
User GCNN	User GCNN	1.03	0.02	0.15
	UU GCNN	1.05 (+1.9%)	0.12 (**+500%**)	0.10 (-33.3%)
	UI GCNN	1.06 (+2.9%)	0.04 (+100%)	0.14 (-6.7%)
Item Linear	Item NN	0.96	0.65	0.03
	II filter	0.98 (+2.1%)	0.62 (-4.6%)	0.03 (0%)
	IU filter	0.98 (+2.1%)	0.60 (-7.7%)	0.03 (0%)
Item GCNN	Item GCNN	0.97	0.29	0.22
	II GCNN	0.95 (**−2.1%**)	0.31 (**+6.9%**)	0.22 (0%)
	IU GCNN	0.98 (+1%)	0.45 (+55.2%)	0.23 (+4.6%)

metric. We believe this is because models build only on user-NN graphs are conservative to both diversity metrics [cf. Fig. B.10], therefore, the margin for improvement is larger. Contrarily, for an item NN graph, the joint models (i.e., IU and II) are conservative and improve by little both diversity metrics. We also highlight the case of II-GCNN which improves the RMSE and AD while keeping the same ID.

8.2. Accuracy-Diversity Trade-off for Ranking

With the same setting of the last section, we now evaluate the trade-off when the joint models are optimized for ranking [cf. Sec 6]. These results are shown in Figure 6. A higher importance to FNs ($\alpha \rightarrow 1$) reduces the NDCG@20 but improves diversity. Both the filter and the GCNN are less sensitive to α when designed for ranking. While for the filter we may still attribute this robustness to the optimality of the design problem, the results for the GCNN suggest the BPR leverages better the information from FNs. Note also the filter on the UI combination pays little in NDCG but gains substantially in AD and ID.

To further quantify these results, in Table 4 we show the diversity gain

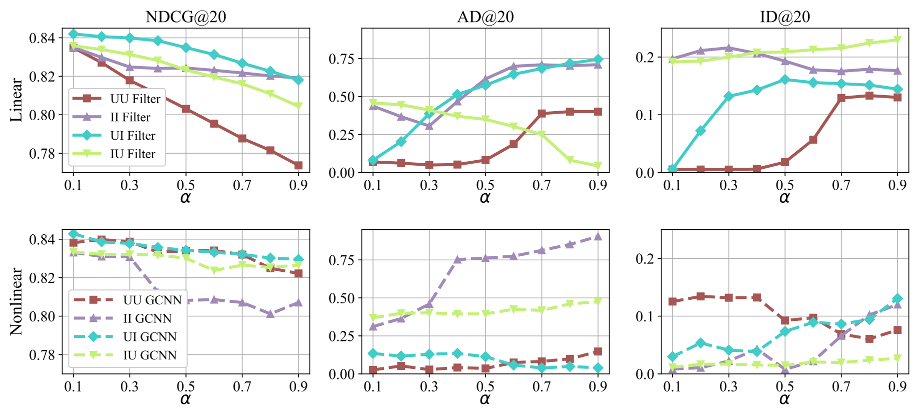


Figure 6: NDCG@20, AD@20 and ID@20 as a function of the accuracy-diversity parameter α for ranking-optimized models. As more information from FNs is included, the NDCG@20 deteriorates but diversity improves. The NDCG is less sensitive to α compared with the RMSE for both the joint graph convolutional filter and GCNN.

when reducing the NDCG by at most 3%. We note that it is often sufficient to deteriorate the NDCG by 1% and gain substantially in diversity. Bigger diversity improvements are achieved when one of the two graphs is item-based. Lastly, we notice the joint GCNN models gain less in diversity compared with linear filters. The GCNN can be further improved by tuning its parameters.

8.3. Comparisons with Accuracy-Oriented Models

In this section, we analyze how the trade-offs of the joint models compare with those achieved by five accuracy-oriented alternatives including state-of-the-art user NN filter [cf. (5)], item NN filter [cf. (6)], and the multi-graph convolutional neural network (MGCNN) [57]; but also the conventional methods of low-rank matrix completion (LR-MC) [62] and matrix factorization optimized w.r.t. BPR (MF-BPR) [50]. Save the last, the first four are designed for rating. We first compare the models in MovieLens100k dataset and then in Douban and Flixster. We consider only the UI combination.

MovieLens100k. Figure 7 contrasts the RMSE and NDCG@20 with the diversity metrics the AD@20 (left) and ID@20 (right) for $\alpha \in [0.1, 0.9]$. The

Table 4: NDCG@20, AD@20 and ID@20 for the models working on the NN graph and for the joint models optimized for ranking. In brackets we show the change in percentage of the proposed joint model w.r.t. the NN graph counterpart.

		NDCG@20	AD@20	ID@20
User Linear	User NN	0.84	0.19	0.02
	UU filter	0.83 (-1.2%)	0.07 (-63.1%)	0.01 (-50%)
	UI filter	0.83 (-1.2%)	0.65 (**+242%**)	0.16 (**+700%**)
User GCNN	User GCNN	0.84	0.10	0.12
	UU GCNN	0.82 (-2.4%)	0.15 (+50%)	0.08 (-33.3%)
	UI GCNN	0.83 (-1.2%)	0.11 (+10%)	0.07 (-41.6%)
Item Linear	Item NN	0.83	0.65	0.03
	II filter	0.82 (-1.2%)	0.70 (**+7.7%**)	0.18 (**+500%**)
	IU filter	0.83 (0%)	0.41 (-36.9%)	0.20 (+566%)
Item GCNN	Item GCNN	0.83	0.40	0.02
	II GCNN	0.83 (0%)	0.46 (+15%)	0.02 (0%)
	IU GCNN	0.83 (0%)	0.47 (**+17.5%**)	0.03 (**+50%**)

accuracy of GCNN is more sensitive to α than the other models. The GCNN gives also more importance to diversity within the list (ID) rather than covering the catalog (AD). This indicates a few items are recommended by the GCNN but are different between them. Contrarily, the joint linear filters are more robust to accuracy losses, gain in AD, but pay in ID. Contrasting the proposed approaches with the other alternatives, we observe:

- Rating-optimized models (MGCNN, user NN filter, item NN filter, and LR-MC) achieve a lower RMSE but face problems in AD. The item NN filter achieves a reasonable AD but its ID is very low. The MGCNN overfits the RMSE by prioritizing a few popular items to all users as shown by the low AD and high ID. The joint linear filter can substantially improve the AD by paying little in RMSE, while the GCNN requires additional tuning. The improved AD comes often at expenses of ID, yet values of $\alpha \approx 0.3$ offer a good balance between the two. We can further improve the ID with the IU combination [cf. Fig. 5].

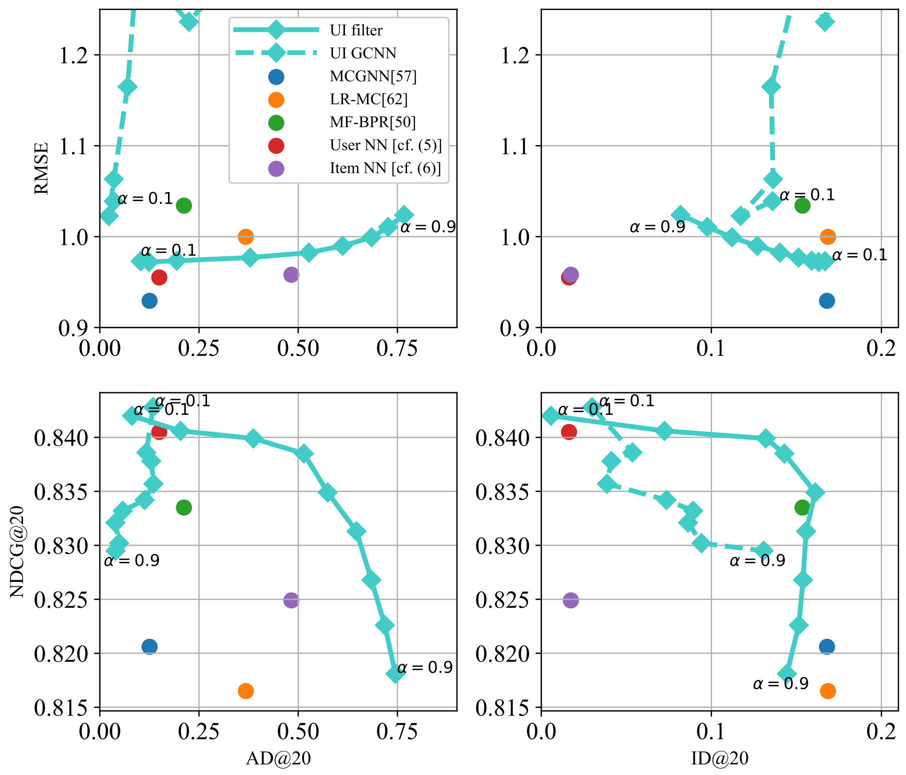


Figure 7: RMSE and NDCG@20 as a function of the AD@20 and ID@20 for the joint models on the UI graph combinations and for five baselines on the MovieLens100k. The joint models are optimized for ranking. By changing α the propsoed models impose a different trade-off than the baselines, often overcoming them.

495 - The ranking-optimized method (BPR-MF) achieves a high NDCG but still lower than the rating-design user NN filter. This high accuracy is again linked to filling the list with a small group of different items. The joint models optimized for ranking overcome this limitation by making the list slightly more similar (lowering ID) but increasing the catalog coverage
500 (improving AD). This strategy keeps the NDCG high.

Overall, we conclude that a high accuracy from the NNs is tied with an increase of list diversity (ID) but also with a scarce catalog coverage (AD). The proposed joint models can keep a reasonable accuracy while contributing to a higher diversity.

Douban & Flixster. We now compare the different models in two datasets containing fewer interactions compared with MovieLens100k; see Table 2. The sparsity of these datasets brings additional challenges when evaluating the NDCG@k. For a list of length 20 there is only one test user for Fixster and none for Douban. To have statistically meaningful results, we measured the NDCG for a list of length $k = 5$, which leads $1,373$ test users for Douban and 126 for Fixster. However, to have a unified diversity comparison with the MovieLens100k dataset, we computed the diversity for a list of length 20.

In Table 5, we show the performance for Douban and Fixster datasets, respectively. For our models, we report the extreme values $\alpha = 0.1$ and $\alpha = 0.9$ and a hand-picked value of α. As α increases, the joint models lose in accuracy but gain in diversity. We see again the sensitivity of GCNNs to α for which the RMSE may also reach unacceptable values. The joint models optimized for ranking can always provide a better NDCG w.r.t. MF-BPR while offering a higher diversity. In general, the best trade-off by the joint models is achieved by the GCNN designed for rating and filters designed for ranking.

Table 5: Performance comparison in Douban and Flixster dataset. We show the RMSE for methods trained for rating and the NDCG@5 for methods trained for ranking. For the intermediate value of α, we show in brackets the difference in percentage compared with the best value of competing alternatives.

	Douban					Flixster				
	α	RMSE	NDCG@5	AD@20	ID@20	α	RMSE	NDCG@5	AD@20	ID@20
MCGNN	-	**0.80**	-	0.03	**0.12**	-	**0.93**	-	0.08	0.07
LR-MC	-	1.39	-	0.80	0.04	-	3.17	-	0.45	0.09
MF-BPR	-	-	**0.75**	0.78	0.05	-	-	**0.72**	0.50	**0.10**
User NN	-	0.76	-	0.52	0.04	-	1.04	-	**0.58**	0.06
Item NN	-	0.80	-	**0.99**	0.05	-	1.12	-	0.33	0.03
Rating linear	0.1	**0.76**	-	0.51	0.04	0.1	1.04	-	0.58	0.06
	0.6	0.84 (+5%)	-	0.75 (-24%)	0.05 (-58%)	0.6	1.05 (+13%)	-	0.69 (+19%)	0.06 (-33%)
	0.9	0.92	-	0.96	0.05	0.9	1.06	-	**0.71**	0.06
Rating GCNN	0.1	0.83	-	0.49	0.10	0.1	1.46	-	0.60	0.10
	0.3	0.85 (+6%)	-	0.44 (-56%)	0.12 (0%)	0.4	1.46 (+57%)	-	0.56 (-3%)	0.11 (+22%)
	0.9	3.31	-	0.63	0.11	0.9	2.84	-	0.48	0.12
Ranking linear	0.1	-	**0.80**	0.44	**0.15**	0.1	-	**0.75**	0.34	**0.14**
	0.4	-	**0.80 (+7%)**	0.48 (-38%)	0.12 (+140%)	0.3	-	**0.75 (+4%)**	0.35 (-30%)	0.13 (+30%)
	0.9	-	0.77	0.77	0.07	0.9	-	0.74	0.36	0.13
Ranking GCNN	0.1	-	**0.80**	0.47	0.13	0.1	-	0.74	0.30	0.13
	0.4	-	**0.80 (+7%)**	0.90 (+15%)	0.05 (0%)	0.6	-	0.74 (+3%)	0.35 (-30%)	0.13 (+30%)
	0.9	-	0.76	0.91	0.05	0.9	-	0.73	0.36	0.13

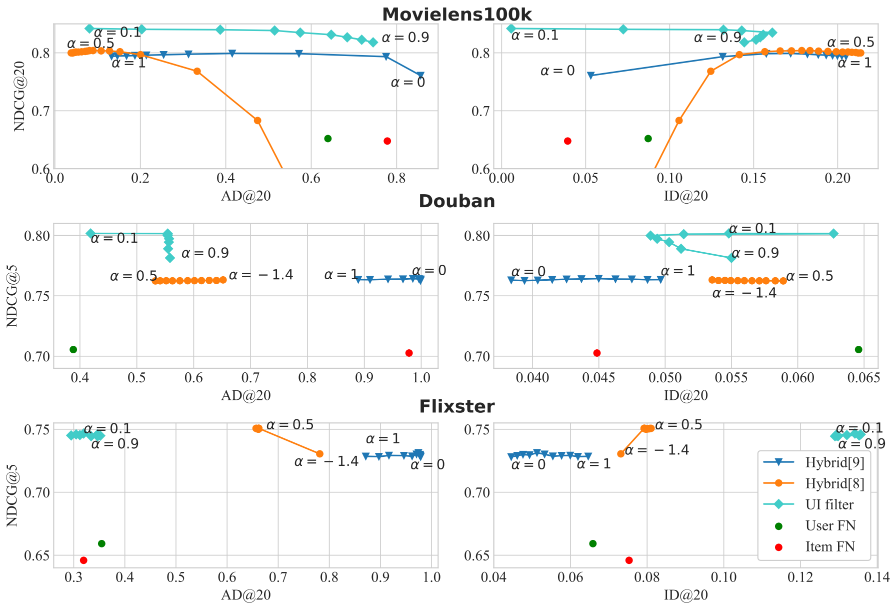


Figure 8: Comparison of joint UI filter optimized for ranking [cf. (24)] with the hybrid approaches from [8] and [9]. We also show the vanilla user-FN and item-FN for reference. The proposed approach pays the less in terms of NDCG while offering different diversity gains.

8.4. Comparison with Accuracy-Diversity Algorithms

In this final section, we compare the joint UI linear model with two hybrid alternatives that propose a similar trade-off [8, 9]. The hybrid approach in [8] mixes a user-based vanilla NN with a user-based vanilla FN. FNs are computed based on the number of items consumed separately and only FNs are multiplied by a single scalar $\alpha \in [-1.4, 0.5]$. The hybrid approach in [9] merges a heat diffusion with a random walk to balance accuracy with diversity over item-item graphs. This approach controls the influence of each model similarly to our method through a scalar $\alpha \in [0, 1]$. Both works predict the probability of an item being consumed by a user rather than the rating. Therefore, we compare the accuracy w.r.t. the NDCG.

In Figure 8, we show the trade-offs of the different methods for all three datasets. We also show two vanilla FN collaborative filters for reference. We see the proposed joint model achieves consistently the highest NDCG while offering a margin to improve accuracy. This behavior is better highlighted in MovieLens100K dataset for which the method hyperparameters have been chosen. We attribute the latter to the fact that the joint model learns its parameters to improve ranking accuracy rather than being a simple fusion of two separate entities. The hybrid strategy from [9] focuses entirely on the catalog coverage as can be seen by the high AD. This strategy heavily affects both the NDCG and ID for which this approach performs the worst. The hybrid strategy from [8] offers a trade-off in both diversity metrics but the role of the two graphs depends largely on the dataset sparsity. In Flixster, for instance, we see this strategy offers little trade-off as the performance for all values of α but -1.4 is the same. To some extent, this trend is also present in our joint model, yet it has more control over diversity while retaining the highest NDCG.

9. Discussion

The accuracy-diversity trade-off represents a crucial factor for improving user satisfaction when personalizing recommendations. However, achieving the 'right' trade-off is challenging, not only because of its subjective aspects that are difficult to quantify, but also because of the complex and irregular user-item relationships that influence both accuracy and diversity. This paper focused on the latter and investigated the potential of graphs that have a proven history as core mathematical tools for representing such data. More specifically, it focused on graph convolutions as the means of dealing with the data complexity and irregularity and to achieve an effective accuracy-diversity trade-off for recommender systems. The overall conclusion of this paper is that graph convolutions have large potential to learn an accuracy-diversity trade-off from the ratings in user-item matrix without relying on side information. Results in three datasets showed graph convolutions attained the highest accuracy while improving diversity compared with other alternatives operating in a similar setting.

The proposed approach relies on information from the nearest and the furthest neighbors in both a learning-to-rate and learning-to-rank setting. We analyzed how this information is leveraged during parameter design and from a graph spectral domain perspective. We formulated a learning problem that accounts for the trade-off through a regularizer term. When the graph convolutional model is composed only of linear filters, we proved the learning problem is convex and provided solutions for it. Convexity rendered the linear model more robust to hyperparameter choice, while the nonlinear model required careful tuning. In the graph spectral domain, we showed graph convolutions operate as bandstop filters in both the nearest neighbor and in the furthest neighbors graphs. This analysis concluded the joint model exploits the general agreement about preferring or not an item but also complete disagreements between connected nearest and furthest neighbors.

We developed an accuracy-to-coverage trade-off, in which accuracy is traded to recommend niche items; and an accuracy-to-individual diversity trade-off, in which accuracy is traded to improve the diversity in the list. The joint convolutional model offers a balance in each setting that is difficult to be achieved with a single model. Comparisons with the nearest neighbor accuracy oriented approaches –including state-of-the-art graph convolutional RecSys methods but also vanilla and graph convolutional nearest neighbor collaborative filtering– showed a diversity improvement by up to seven times while paying about 1% in accuracy. Comparisons with the vanilla furthest neighbor collaborative filtering showed consistently a higher accuracy because of the information from the nearest neighbors. The trend in these findings is in line with that in [8, 7]. Overall, we have seen graph convolutions can trade accuracy to improve substantially one diversity criteria or improve both by a lesser amount.

The current manuscript has also open aspects. One main question we left unaddressed is why nonlinear GCNN models do not outperform the linear counterparts. Although we have seen a GCNN case that improves accuracy and both diversity metrics, most of the results suggested the joint model should be less complex, the sparser the dataset. This is not entirely surprising as shown also in [37, 38]. Second, the current results are not exhaustive to decide which nearest-furthest neighbor graph combination is the most suitable for a specific accuracy and diversity criteria. This aspect is certainly relevant, but it is beyond the scope of this paper. Lastly, more research is needed toward explainability. The graph spectral analysis helps in this regard but research is still needed to identify the link between the different spectral components and the items included in the recommendation lists. Nonetheless, to the best of our knowledge this is the first work showcasing the potential of graph convolutions to establish an accuracy-diversity trade-off for recommender systems.

Appendix A. Metrics

Denote the test set by $\mathcal{T}_s$ and the length of recommendation list by k.

RMSE. For X_{ui} being the true value and $\hat{X}_{ui}$ the estimated rating for tuple $(u,i) \in \mathcal{T}_s$, the RMSE is defined as

$$\text{RMSE} = \frac{\sqrt{\sum_{(u,i)\in\mathcal{T}_s} |\hat{X}_{ui} - X_{ui}|^2}}{|\mathcal{T}_s|}. \tag{A.1}$$

A lower value of RMSE indicates a better fit; hence, a better performance.

NDCG@k. Denote by $\mathcal{I}_{uk} = \{i_{u1}, \cdots, i_{uk}\}$ the set of k items predicted with the highest ratings for user u, i.e., $\hat{X}_{ui_{u1}} \geq \hat{X}_{ui_{u2}} \geq \ldots \geq \hat{X}_{ui_{uk}}$. We first define the discounted cumulative gain (DCG) for which we consider the true ratings $X_{ui} := \text{rel}_i$ (called also the relevance for item i) for items $i \in \mathcal{I}_{uk}$ ordered w.r.t. the predicted order in $\mathcal{I}_{uk}$, i.e., $\text{rel}_1 \geq \text{rel}_2 \geq \ldots \geq \text{rel}_k$. The DCG for user u in a list of length k is defined as

$$\text{DCG}_u@k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}. \tag{A.2}$$

The $\text{DCG}_u@k$ accounts for the ordering of the true values in the predicted list $\mathcal{I}_{uk}$. This ordering can at most be the ideal one $X_{ui_{u1}} = \hat{X}_{ui_{u1}} \geq X_{ui_{u2}} = \hat{X}_{ui_{u2}} \geq \ldots \geq X_{ui_{uk}} = \hat{X}_{ui_{uk}}$, i.e., when the algorithm orders the items in the predicted list $\mathcal{I}_{uk}$ following the true order. In this instance, we refer to it as the ideal DCG for user u ($i\text{DCG}_u@k$). Then, the NDCG@k for the a list k over all users $\mathcal{U}$ is defined as

$$\mathrm{NDCG}@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\mathrm{DCG}_u@k}{i\mathrm{DCG}_u@k}. \tag{A.3}$$

A high value of NDCG@k indicates a better recommendation in the list of order k; hence, a better performance.

Aggregated diversity. The aggregated diversity measures the fraction of items $\mathcal{I}$ included in the union of all the recommendation lists $\mathcal{I}_{uk}$, i.e.,

$$\mathrm{AD}@k = \frac{1}{|\mathcal{I}|} \bigcup_{u=1}^{|\mathcal{U}|} \mathcal{I}_{uk}. \tag{A.4}$$

A higher aggregated diversity indicates the algorithm recommends a larger portion of the items present in the catalog, consequently, a better performance.

Individual diversity. The individual diversity for a list of length k (ID@k) measures the average diversity within the recommendation lists of all users. For $d(i,j)$ being a distance metric of two items i and j quantifying their dissimilarity, the individual diversity is computed as

$$\mathrm{ID}@k = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{2}{k(k-1)} \sum_{(i,j) \in \mathcal{I}_{uk}, i \neq j} d(i,j) \tag{A.5}$$

where the inner sum computes the individual diversity for the list $\mathcal{I}_{uk}$ of user u and the outer sum averages across all users. A higher ID indicates the average recommendation list is more diverse. Notice the ID requires computing a distance between items (often based on item features). To use this metric also in featureless items, we followed [63] and computed the Euclidean distance based on the first seven SVD latent features for items i and j.

Appendix B. Parameter Analysis on Similarity Graph

We here analyze the performance of the user-NN and item-NN graphs [cf. Section 3]. For each graph, we evaluated different nearest neighbors $n \in \{5, 10, \ldots, 40\}$, filter orders $K \in \{1, 2, 3\}$, and list length $k \in \{10, 20, \ldots, 100\}$.

NN and filter order. We first analyzed combinations between different NN and filter orders. We fixed the length of the list to $k = 10$ which is a common

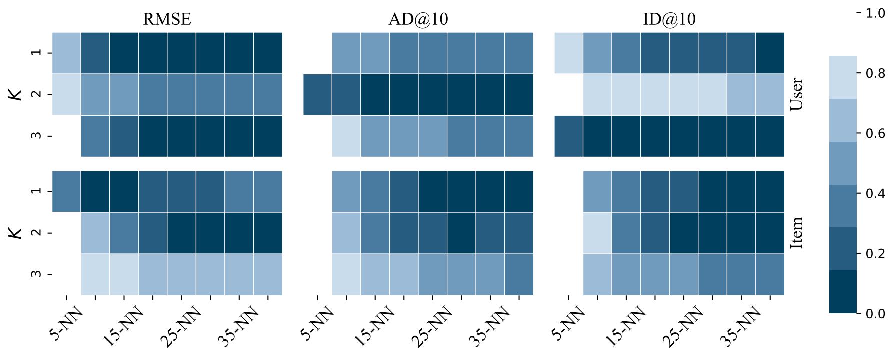


Figure B.9: RMSE, AD@10, and ID@10 for different filter orders K (vertical axis) and nearest neighbors (horizontal axis). (Top) User-based graph convolutional filter [cf. (5)]. (Bottom) Item-based graph convolutional filter [cf. (6)]. Increasing the NN improves the RMSE but reduces diversity. The filter orders $K > 1$ shows the vanilla NN $K - 1$ [cf.(1)-(2)] can be improved by multi-hop neighbors. Results are scaled in the range $[0, 1]$ to improve visibility.

choice in the literature [26, 59, 60, 64, 65]. Figure B.10 shows the RMSE, the AD@10, and the ID@10 for both scenarios. The number of NNs plays a role in the trade-off. More NNs reduce the RMSE but degrade both diversity metrics. This is because each entity gets connected with more similar entities whose combined effect smoothness ratings. For almost all NNs, there is always an order $K > 1$ that improves both accuracy and diversity of the vanilla NN [cf.(1)-(2)].

From these results, we choose the combination that achieves the lowest RMSE. For the user-based scenario, we have: $30-$NN, $K = 3$, RMSE$= 0.955$, AD@10 $= 0.150$, and ID@10 $= 0.017$. For the item-based scenario, we have: $35-$NN, $K = 2$, RMSE$= 0.958$, AD@10 $= 0.482$, and ID@10 $= 0.017$.

Length recommendation list. In Figure B.10, we show the effect the recommendation list has on trade-off NDCG@k-AD@k, and NDCG@k-ID@k. A longer list improves diversity, but reduces accuracy. This is rather expected because chances to include different items increase in a longer list. At the same time, a longer list makes more challenging identifying the correct order; hence, reducing the NDCG@k.

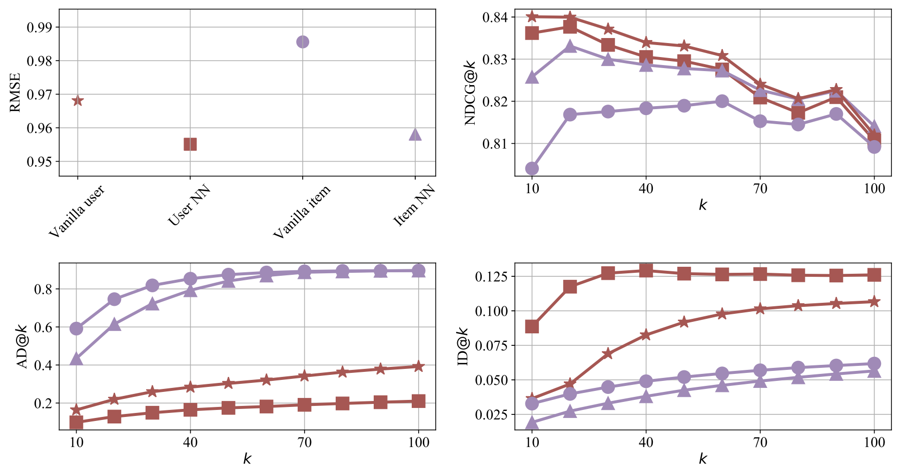


Figure B.10: RMSE, NDCG@k, AD@k and ID@k as recommendation set's size k varies. We see the NDCGk reduces for lists more than 20 items while diversity increases.

Comparing the user NN with the item NN, we see little difference in terms of NDCG, while there is more different in AD and ID.

- User NN achieves a lower AD but a higher ID. This implies the algorithm prioritizes a few relevant items in the catalog but diversifies the list of each user, respectively. In our opinion, this is because the user NN has a narrow view of all items in the catalog as it explores user-similarities. The model fails to account for the broad range of items (each item is treated individually) and prioritizes popular choices, which are different between them. The plateau the user NN reaches for relatively low ID further corroborates the latter.

- Item NN achieves a higher AD but a lower ID. I.e., the model covers a larger portion of the catalog (recommends different items to different users) but, to a specific user, it recommends similar items. Item NN is less user-centric since it leverages item similarities and ignores the influence of other similar users. Consequently, the model has a broader view on items to build recommendations; this explains an AD that is up to four times higher compared with the user NN. Nevertheless, since each user is treated individually less importance is given to diversifying items within the list. We can see this model as highly personalizing the user list because different items are recommended to different users but these items are highly similar.

Based on these results, we set the list length to $k = 20$ since this value achieves the highest NDCG for both the user NN and the item NN. While a longer list can be an option to improve diversity, it is not user-satisfactory to search within it.

References

[1] C. C. Aggarwal, et al., Recommender systems, Vol. 1, Springer, 2016.

[2] K. Bradley, B. Smyth, Improving recommendation diversity, in: Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland, Citeseer, 2001, pp. 85–94.

[3] M. Kaminskas, D. Bridge, Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems, ACM Transactions on Interactive Intelligent Systems (TiiS) 7 (1) (2016) 1–42.

[4] M. Kunaver, T. Požrl, Diversity in recommender systems–a survey, Knowledge-Based Systems 123 (2017) 154–162.

[5] Q. Wu, Y. Liu, C. Miao, Y. Zhao, L. Guan, H. Tang, Recent advances in diversified recommendation, arXiv preprint arXiv:1905.06589.

[6] M. Gan, R. Jiang, Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities, Decision Support Systems 55 (3) (2013) 811–821.

[7] A. Said, B. Kille, B. J. Jain, S. Albayrak, Increasing diversity through furthest neighbor-based recommendation, Proceedings of the WSDM 12.

[8] W. Zeng, M.-S. Shang, Q.-M. Zhang, L. Lü, T. Zhou, Can dissimilar users contribute to accuracy and diversity of personalized recommendation?, International Journal of Modern Physics C 21 (10) (2010) 1217–1227.

[9] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, Y.-C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems, Proceedings of the National Academy of Sciences 107 (10) (2010) 4511–4515.

[10] M. Zhang, N. Hurley, Avoiding monotony: improving the diversity of recommendation lists, in: Proceedings of the 2008 ACM conference on Recommender systems, 2008, pp. 123–130.

[11] G. Adomavicius, Y. Kwon, Overcoming accuracy-diversity tradeoff in recommender systems: A variance-based approach, in: Proceedings of WITS, Vol. 8, Citeseer, 2008.

[12] N. J. Hurley, Personalised ranking with diversity, in: Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 379–382.

[13] U. Panniello, A. Tuzhilin, M. Gorgoglione, Comparing context-aware recommender systems in terms of accuracy and diversity, User Modeling and User-Adapted Interaction 24 (1-2) (2014) 35–65.

[14] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proceedings of the IEEE 106 (5) (2018) 808–828.

[15] A. N. Nikolakopoulos, D. Berberidis, G. Karypis, G. B. Giannakis, Personalized diffusions for top-n recommendation, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 260–268.

[16] Z. Abbassi, V. S. Mirrokni, A recommender system based on local random walks and spectral methods, in: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, 2007, pp. 102–108.

[17] F. Monti, M. Bronstein, X. Bresson, Geometric matrix completion with recurrent multi-graph neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 3697–3707.

[18] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, X. He, Multi-graph convolution collaborative filtering, in: 2019 IEEE International Conference on Data Mining (ICDM), IEEE, 2019, pp. 1306–1311.

[19] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems.

[21] F. Gama, E. Isufi, G. Leus, A. Ribeiro, Graphs, convolutions, and neural networks, arXiv preprint arXiv:2003.03777.

[22] H. Yang, Aligraph: A comprehensive graph neural network platform, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 3165–3166.

[23] B. Smyth, P. McClave, Similarity vs. diversity, in: International conference on case-based reasoning, Springer, 2001, pp. 347–361.

[24] D. Bridge, J. P. Kelly, Ways of computing diverse collaborative recommendations, in: International conference on adaptive hypermedia and adaptive web-based systems, Springer, 2006, pp. 41–50.

[25] G. Adomavicius, Y. Kwon, Toward more diverse recommendations: Item re-ranking methods for recommender systems, in: Workshop on Information Technologies and Systems, Citeseer, 2009.

[26] G. Adomavicius, Y. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, IEEE Transactions on Knowledge and Data Engineering 24 (5) (2011) 896–911.

[27] E. M. Hamedani, M. Kaedi, Recommending the long tail items through personalized diversification, Knowledge-Based Systems 164 (2019) 348–357.

[28] F. Eskandanian, B. Mobasher, Using stable matching to optimize the balance between accuracy and diversity in recommendation, in: Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization, UMAP '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 71–79. doi:10.1145/3340631.3394858.

[29] N. Hurley, M. Zhang, Novelty and diversity in top-n recommendation–analysis and evaluation, ACM Transactions on Internet Technology (TOIT) 10 (4) (2011) 1–30.

[30] A. Gogna, A. Majumdar, Balancing accuracy and diversity in recommendations using matrix completion framework, Knowledge-Based Systems 125 (2017) 83–95.

[31] J.-G. Liu, K. Shi, Q. Guo, Solving the accuracy-diversity dilemma via directed random walks, Physical Review E 85 (1) (2012) 016118.

[32] J. Wasilewski, N. Hurley, Incorporating diversity in a learning to rank recommender system, in: The twenty-ninth international flairs conference, 2016.

[33] A. Said, B. Fields, B. J. Jain, S. Albayrak, User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm, in: Proceedings of the 2013 conference on Computer supported cooperative work, 2013, pp. 1399–1408.

[34] A. Javari, M. Jalili, A probabilistic model to resolve diversity–accuracy challenge of recommendation systems, Knowledge and Information Systems 44 (3) (2015) 609–627.

[35] W. Huang, A. G. Marques, A. Ribeiro, Collaborative filtering via graph signal processing, in: 2017 25th European Signal Processing Conference (EUSIPCO), IEEE, 2017, pp. 1094–1098.

[36] R. v. d. Berg, T. N. Kipf, M. Welling, Graph convolutional matrix completion, arXiv preprint arXiv:1706.02263.

[37] W. Huang, A. G. Marques, A. R. Ribeiro, Rating prediction via graph signal processing, IEEE Transactions on Signal Processing 66 (19) (2018) 5066–5081.

[38] L. Chen, L. Wu, R. Hong, K. Zhang, M. Wang, Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 27–34.

[39] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019, pp. 165–174.

[40] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, Z. Wang, Knowledge-aware graph neural networks with label smoothness regularization for recommender systems, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 968–977.

[41] T. Zhong, S. Zhang, F. Zhou, K. Zhang, G. Trajcevski, J. Wu, Hybrid graph convolutional networks with multi-head attention for location recommendation, World Wide Web (2020) 1–27.

[42] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE signal processing magazine 30 (3) (2013) 83–98.

[43] F. Hua, C. Richard, J. Chen, H. Wang, P. Borgnat, P. Goncalves, Learning combination of graph filters for graph signal modeling, IEEE Signal Processing Letters.

[44] H. Sevi, G. Rilling, P. Borgnat, Harmonic analysis on directed graphs and applications: from fourier analysis to wavelets, arXiv preprint arXiv:1811.11636.

[45] N. Dehmamy, A.-L. Barabási, R. Yu, Understanding the representation power of graph neural networks in learning graph topology, in: Advances in Neural Information Processing Systems, 2019, pp. 15387–15397.

[46] S. Chen, S. Niu, T. Lan, B. Liu, Pct: Large-scale 3d point cloud representations via graph inception networks with applications to autonomous driving, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 4395–4399.

[47] V. N. Ioannidis, A. G. Marques, G. B. Giannakis, A recurrent graph neural network for multi-relational data, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 8157–8161.

[48] G. Mateos, S. Segarra, A. G. Marques, A. Ribeiro, Connecting the dots: Identifying network structure via graph signal processing, IEEE Signal Processing Magazine 36 (3) (2019) 16–43.

[49] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[50] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, AUAI Press, Arlington, Virginia, USA, 2009, p. 452–461.

[51] S. Boyd, S. P. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[52] S. Lipovetsky, Analytical closed-form solution for binary logit regression by categorical predictors, Journal of applied statistics 42 (1) (2015) 37–49.

[53] A. Sandryhaila, J. M. Moura, Discrete signal processing on graphs: Frequency analysis, IEEE Transactions on Signal Processing 62 (12) (2014) 3042–3054.

[54] A. Sandryhaila, J. M. Moura, Discrete signal processing on graphs, IEEE transactions on signal processing 61 (7) (2013) 1644–1656.

[55] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, Acm transactions on interactive intelligent systems (tiis) 5 (4) (2015) 1–19.

[56] H. Ma, D. Zhou, C. Liu, M. R. Lyu, I. King, Recommender systems with social regularization, in: Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 287–296.

[57] M. Jamali, M. Ester, A matrix factorization technique with trust propagation for recommendation in social networks, in: Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 135–142.

[58] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems (TOIS) 22 (1) (2004) 5–53.

[59] C.-N. Ziegler, S. M. McNee, J. A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th international conference on World Wide Web, ACM, 2005, pp. 22–32.

[60] J. Wang, J. Yin, Combining user-based and item-based collaborative filtering techniques to improve recommendation diversity, in: 2013 6th International Conference on Biomedical Engineering and Informatics, IEEE, 2013, pp. 661–665.

[61] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[62] R. Mazumder, T. Hastie, R. Tibshirani, Spectral regularization algorithms for learning large incomplete matrices, Journal of machine learning research 11 (Aug) (2010) 2287–2322.

[63] M. Kunaver, Š. Dobravec, A. Košir, Using latent features to measure the diversity of recommendation lists, in: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2015, pp. 1230–1234.

[64] G. Karypis, Evaluation of item-based top-n recommendation algorithms, in: Proceedings of the tenth international conference on Information and knowledge management, 2001, pp. 247–254.

[65] K. Niemann, M. Wolpers, A new collaborative filtering approach for increasing the aggregate diversity of recommender systems, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 955–963.