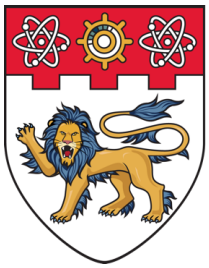


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Representation Learning for Recommender Systems

by

Lucas Vinh Tran

School of Computer Science and Engineering
Nanyang Technological University (NTU)

Institute for Infocomm Research
Agency for Science, Technology and Research (A*STAR)

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

May 2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

11 May 2020

.....

Date

.....

Lucas Vinh Tran

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

11 May 2020

.....

Date

.....

Prof. Gao Cong

Authorship Attribution Statement

The main content of this dissertation contains material from 2 papers published in the peer-reviewed conferences and 2 papers are under reviewed, in which I am listed as a first author. The full list of publications can be referred to Appendix A.

The material in **Chapter 3** is taken from:

(1) **Lucas Vinh Tran**, Thanh Tran, Shanshan Feng, Gao Cong, Xiaoli Li. Wasserstein Distance based Metric Learning Chain for Recommender Systems. *Under Review.*

The contributions of the co-authors are as follows:

- I came up with the key idea, conducted the experiments including designed and implemented all of the main source code. I prepared the manuscript drafts.
- Thanh Tran provided the experimental results for the sequential models as baselines for comparison.
- The final versions were revised and edited by Prof. Gao Cong, Dr. Xiaoli Li, and Dr. Shanshan Feng.

The material in **Chapter 4** is taken from:

(2) **Lucas Vinh Tran**, Yi Tay, Shuai Zhang, Gao Cong, Xiaoli Li. HyperML: A Boosting Metric Learning Approach for Recommender Systems. *Proceedings of the 13th International Conference on Web Search and Data Mining* (**WSDM 2020**), Pages 609-617, Houston, TX, USA. (**Best Paper Award Runner-Up**)

The contributions of the co-authors are as follows:

- I came up with the key idea. Prof. Gao Cong provided discussions and follow-up ideas on exploring metric learning based methods for recommendation.
- I conducted the experiments including designed and implemented all of the main source code. I prepared the very first manuscript drafts with the help from Prof. Gao Cong and Dr. Xiaoli Li.

- The final versions were revised and edited by Prof. Gao Cong, Dr. Xiaoli Li, Dr. Yi Tay, and Dr. Shuai Zhang.

The material **Chapter 5** is taken from:

(3) **Lucas Vinh Tran**, Gao Cong, Xiaoli Li. DropRec: On Adaptive Architecture Dropout for Personalized Recommendation. *Under Review.*

The contributions of the co-authors are as follows:

- I came up with the key idea, conducted all the experiments including designed and implemented all of the main source code. I prepared the manuscript drafts.
- The final versions were revised and edited by Prof. Gao Cong, and Dr. Xiaoli Li.

The material in **Chapter 7** is taken from:

(4) **Lucas Vinh Tran**, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong and Xiaoli Li. Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (**SIGIR 2019**). Pages 255-264, Paris, France.

The contributions of the co-authors are as follows:

- Prof. Gao Cong and Dr. Tuan-Anh Nguyen Pham provided initial directions and discussed about several model designs at the early stage.
- I came up with the key idea, conducted the experiments including designed and implemented all of the main source code. I prepared the manuscript drafts.
- Dr. Yi Tay provided feedback and helped to edit on several first draft versions.
- The final versions were revised and edited by Prof. Gao Cong, Dr. Xiaoli Li, and Dr. Yiding Liu.

11 May 2020 .....

Date

.....

Lucas Vinh Tran

Acknowledgments

I would like to express my sincere gratitude towards my supervisors, Prof. Gao Cong and Dr. Xiaoli Li, for providing great guidance and encouragement throughout my PhD journey. They have taught me many valuable lessons which helped to enhance my knowledge. They also suggested me how to discover new research problems and how to find good answers. To my mind, they are not only my supervisors but also my friends, who always be there to support me both technically and mentally. It is my pleasure to work under their mentorship.

I am also indebted to Prof. See Kiong Ng, who trusted me at the beginning of my postgraduate study and made it possible for me to obtain the full scholarship grant, which allowed me to pursue my PhD dream.

In the same way, I want to acknowledge Agency for Science, Technology and Research (A*STAR) for providing me with full scholarship funding for my doctoral program, and all the staffs at A*STAR who provided me various supports during the period.

Moreover, I would like to thank Prof. Guosheng Lin for his helps and comments, as well as his patience for me at the early stage. I would also like to thank my Thesis Advisory Committee members for their meaningful feedbacks and advices.

Besides, I am thankful to all of my collaborators and co-authors who have contributed significantly to my research through various great discussions. In particular, I want to express my grateful to Dr. Tuan-Anh Nguyen Pham for his supports and motivations, especially during the most difficult times of my study.

Furthermore, I wish to give my thanks to my colleagues, whom I have learned so much from and shared many memories with during the past few years. I feel very lucky to have them as a part of my PhD journey. Thanks to all of you.

Additionally, I would like to express my thankfulness to my family for all their care and encouragement, for always being a great family. Specially to my grandmother who had always looked after me since I was a child, I hope this little achievement would make her proud in the heaven.

And most importantly, I express my deepest gratitude to my beloved one, a beautiful and charming woman, Dong, for her unconditional love and countless sacrifices that help me to get to this point. Without her, I would not be in the position of writing this thesis today. I dedicate this dissertation to her.

Summary

Recommender systems are widely used in many big companies such as Facebook, Google, Twitter, LinkedIn, Amazon and Netflix. They help to deal with the problem of information overload by filtering the important information fragments efficiently according to users' preferences and interests. However, it is challenging to develop highly effective models for handling various recommendation problems, from individual-level to group-level tasks. To be specific, the standard recommendation problem is personalized with respect to a single user, where it aims to identify the most interesting and relevant items to make personalized recommendation. In contrast, group level recommendation deals with groups of users instead of individuals, in which fine-grained, intricate group dynamics and preferences have to be considered.

With the success of deep learning, many neural architectures have been proposed for learning-to-rank recommendation. Indeed, recent studies have shown their effectiveness and efficiency in providing better recommendations in terms of user and group-of-users satisfaction with the involvement of deep learning. The key intuition behind many successful end-to-end neural architectures for recommendation is to design appropriate frameworks that are not only able to learn rich user and item representations, but also well capture the implicit and hidden relationships behind users and items. Therefore, the objective of designing such effective neural architectures remains a challenge, especally in the context of different recommendation tasks such as general recommendation, next-item recommendation, shopping-basket recommendation, etc.

This dissertation focuses on designing neural architectures for personalized and group recommendation. More specifically, we explore representation learning techniques, both Euclidean and non-Euclidean representation, for learning-to-rank user/group-of-users and item pairs. The key contributions of this dissertation are listed below.

Personalized Recommendation. Our contributions are summarized as follows:

- **Wasserstein based Metric Learning Representation for Recommendation**. We introduce a novel Wasserstein distance-based Metric Learning Chain (W-MLC) model. Our W-MLC model employs a series of metric learning, together with a

Wasserstein distance to constraint on the user/item projection transformations, allowing us to encode user-item interactions better through a *deeper* chain. In addition, we propose a hinge loss function with personalized adaptive margins for different users. Extensive experiments on **eight** datasets of **three** different recommendation tasks reveal the effectiveness of our proposed model over **eight** strong state-of-the-art baselines.

- **Going Beyond Euclidean: Hyperbolic Representation for Recommendation**. We investigate the notion of learning user and item representations in non-Euclidean space. Specifically, we study the connection between metric learning in hyperbolic space and collaborative filtering by exploring Möbius gyrovector spaces where the formalism of the spaces could be utilized to generalize the most common Euclidean vector operations. Overall, this work aims to bridge the gap between Euclidean and hyperbolic geometry in recommender systems through metric learning approach. We propose HyperML (Hyperbolic Metric Learning), a conceptually simple but highly effective model for boosting the performance. Via a series of extensive experiments, we show that our proposed HyperML not only outperforms their Euclidean counterparts, but also achieves state-of-the-art performance on multiple benchmark datasets, demonstrating the effectiveness of personalized recommendation in hyperbolic geometry.

- **Neural Architecture Dropout Representation for Recommendation**. We introduce a new regularization effect on personalized recommendation ranking task by proposing a simple informative neural architecture to enhance the performance. Specifically, our proposed architecture, DropRec (Dropout for Recommendation), investigates the notion of adaptive architecture dropout between layers by leveraging the usage of attention mechanisms. Unlike standard approaches, we explore attention mechanisms as a method to avoid redundancy and activate only specific parts of the network for each user-item pair. In the end, we propose two variants of DropRec: Co-Attention based DropRec (C-DropRec) and Self-Attention based DropRec (S-DropRec), achieving significantly competitive performance on six widely adopted benchmark datasets over existing strong stacked multi-layered baselines, demonstrating the effectiveness of attention modules in focusing on different aspects of the architecture.

- **One-Off Comparison between Personalized Recommenders**. We put a separate chapter in order to further introduce one additional contribution to this dissertation by proposing a one-off comparison between all the proposed architectures and baselines, across all the benchmark datasets. The datasets, evaluation protocol, metrics, and baselines will be introduced in the later section. In fact, this very extensive

experiment could also be considered as a small survey of the proposed architectures on the implicit feedback problem. This chapter is designed to give the readers a general overview of the proposed representation learning techniques, as well as the performance of the well-known baselines. In addition, we also provide discussions and observations on this one-off comparison. Lastly, interesting conclusions and opinions will also be provided in this chapter.

Group Recommendation. Our contributions are described as follows:

- **Learning Representation for Group Recommendation**. We propose *Medley of Sub-Attention Networks* (MoSAN), a new novel neural architecture for the *group* recommendation task. Our proposed approach hinges upon the key intuition that the decision making process (in groups) is generally dynamic, i.e., a user's decision is highly dependent on the other group members. All in all, our key motivation manifests in a form of an attentive neural model that captures fine-grained interactions between group members. In our MoSAN model, each sub-attention module is representative of a single member, which models a user's preference with respect to all other group members. Subsequently, a Medley of Sub-Attention modules is then used to collectively make the group's final decision. Overall, our proposed model is both expressive and effective. We show that MoSAN not only achieves state-of-the-art performance but also improves standard baselines by a considerable margin.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivations

Given the growth of data in recent years, the role of recommender systems becomes more and more important. After all, interaction data (clicks, purchases, etc.) lives at the heart of many applications in different domains such as content streaming sites (e.g., music, podcast), e-commerce and so on. To this end, recommender systems not only serve as a great mitigation strategy, but also create an overall better user experience. Moreover, the advent of social networking services has brought about a significant ease in not only joining but also organizing communal/group activities. The importance and relevance of such communal activities cannot be overstated, as they are after all, deeply involved in enhancing the social fabric of communities. Thus, the need to cater to group preferences is both imperative and intuitive, especially given its crucial role in modern society. Its importance notwithstanding, many personalization algorithms are still generally tailored to the individual, i.e., by definition *'person-alized'* in the sense they tailor to only one person. Therefore, in this dissertation, we are also concerned with designing highly effective recommender systems that are targetted at modeling group preferences as opposed to individual preferences.

Pertaining to the collaborative ranking problem (i.e., interaction-only setting) in recommender systems for personalized recommendation, the major concern is to deal with the complex and non-linearity relationships between users and items, where conventional methods, such as the popular matrix factorization [KBV09], fail to capture due to the assumption in linearity. Thus, building effective recommendation algorithms with the ability to learn useful representations from input data becomes a challenge. Therefore, it is a natural choice to explore representation learning techniques to extract and encode favorable information for recommendation. Morever, while most recommendation techniques have been focusing on individual recommendation, in many cases, the recommended products or services are consumed by a group of users. Application scenarios of

group recommendation include, for example, having dinners with colleagues, watching movie with spouses, and going picnic with friends. Since traditional recommendation methods targeting individuals cannot be efficiently applied in group recommendation, it comes with the demand for more advanced group recommendation techniques.

In this dissertation, we focus on investigating deep neural networks for representation learning for better personalized and group recommendation, in which we refer to those approaches as deep representation learning methods. The advantages of using deep representation learning based recommendation are in two-folds: 1) effectively reducing the efforts of feature engineering, and 2) easily incorporating additional heterogeneous information such as text (e.g., reviews of a user), image (e.g., picture of a food) or location (e.g., longtitude and latitude of a restaurant). We will introduce a comprehensive literature review later in Chapter 2, where we will go deeper in reviewing deep learning techniques for recommender systems, especially in the context of representation learning. Notably, throughout this dissertation, we would like to emphasize that we only focus on **interaction-only** data. It means that we only consider user-item interactions for personalized recommendation problem, and group-item interactions for group recommendation problem.

In the scope of deep representation learning for personalized recommendation, a diverse plethora of machine learning models solves the personalized ranking problem in recommender systems via building matching functions [RFGS09, Ren10, SM07, HLZ+17]. Across the literature, a variety of matching functions have been traditionally adopted, such as inner product [RFGS09], metric learning [TATH18, HYC+17] and/or neural networks [HLZ+17, HDW+18]. In this dissertation, we particularly pay attention to the emerging of metric learning in recommender systems, in which metric learning has attracted extensive research across different domains [WDWK11, CHL05, XNJR02, KTW+12, CHL05, HYC+17, TATH18, WBS05]. In recommender systems, metric learning, with the notation of distance as a matching function between users and items, has been widely adopted mainly due to its crucial triangle inequality property, compared to inner product [RFGS09] or neural network [HLZ+17, HDW+18]. The basic idea of metric learning is to produce a distance metric between two objects to represent for their similarity, in which the two objects are more similar if their distance is small.

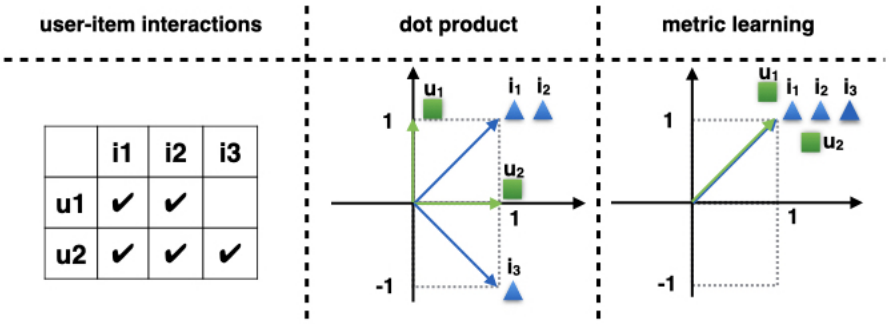


Figure 1.1: Learning with dot product vs. metric learning.

Particularly, we assume user u_1 and u_2 interacted with two similar items i_1, i_2 as illustrated in the toy example in Figure 1.1, in which user u_2 additionally interacted with item i_3 (the interaction table in the left figure). Thus, learning with dot product can lead to the result shown in the middle figure and we end up not recommend item i_3 to user u_1. In contrast, by learning with metric learning, the distance $d(u_1, i_3)$ between user u_1 and items i_3 are smaller. As an effect of the triangle inequality, the distance $d(i_1, i_3)$ between two items i_1 and i_3 satisfies $d(i_1, i_3) < d(u_1, i_1) + d(u_1, i_3)$. As $d(u_1, i_1)$ and $d(u_1, i_3)$ are small, $d(i_1, i_3)$ is small as well, showing that i_1 and i_3 are correctly portrayed. We can derive similar observation for the distance $d(i_2, i_3)$, which leads to the results in the right figure. In a same manner, when two similar users prefer a same item, as an effect of the triangle inequality, the distance between the two similar users is small as well. Therefore, inspired by this motivation, we focus more on the evolution of metric learning in this dissertation.

Although there exisits several metric learning-based recommender systems have been proposed [TATH18, HYC+17, KS10, CMTJ12, PKXY18], most of them represent each user and item embedding by a single shallow level of pull and push mechanism [HYC+17, CMTJ12, TSL19]. In this way, the formation of the pull and push mechanism indeed could be considered as deterministic as each user and item is presented as only a single point in a specific low-dimensional vector space. Moreover, although recent strong metric learning models (e.g., Collaborative Metric Learning (CML) [HYC+17] and Latent Relational Metric Learning (LRML) [TATH18]) demonstrate reasonable empirical success for collaborative ranking with implicit feedback, those matching functions only covered the scope of Euclidean space. Thus, in this dissertation, we also introduce an advanced method that explores metric learning in hyperbolic space as hyperbolic representation learning approach. Due to the exponentially expansion property of hyperbolic space,[1] we discovered that metric learning with the pull-push mechanism in hyperbolic space could boost the performance significantly: moving a point to a certain distance will require

[1] http://hyperbolicdeeplearning.com/

a much smaller force in hyperbolic space than in Euclidean space. Therefore, in this dissertation, we develop two novel deep metric learning representation techniques: one in Euclidean and one in hyperbolic space, to overcome the limitations of the previous proposals.

Additionally, with the insurmountable progress of deep learning in the recent years, almost, if not all, recommendation tasks are dominated by neural architectures [RFGS09, HLZ+17, ZYST19, XYH+17, HC17, XHZ+19, TLLK19, HYC+17, ZYST19, ZYST19, DCJ19]. Many recommendation models have been proposed using deep learning for learning user/item representations; thus, a variety of architecture designs have been nominated [RFGS09, HLZ+17, ZYST19, XYH+17, HC17, XHZ+19, TLLK19, HYC+17]. Interestingly, many of the proposed neural recommenders consider the shallow pyramid multi-layered perceptrons (MLP) structure as an essential part of the model [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. For example, [HLZ+17] combined matrix factorization (MF) and MLP for dual space embedding, [ZACC17] expressed the non-linearities by passing the element-wise product of user and item representations into a pyramid MLP, [HC17, XHZ+19] stacked an MLP on top of the architecture for prediction, or [TLLK19] proposed a square-distance MLP and combined with metric learning. As a result, these models mostly pay attention to explore the notion of stacking multiple layers to capture the non-linearities of user-item interactions. Different from existing methods, in this dissertation, we consider that an MLP of stacking multiple hidden layers may carry overlapping information across layers, in which parts of a neural network architecture could be removed to avoid redundancy. Therefore, we also further propose a new neural architecture that is able to demonstrate similar regularization effect as dropout [HSK+12], where we name it as a method of neural architecture dropout for recommendation.

On the other hand, in the scope of deep representation learning for group recommendation, learning representation for a group of users is a challenging task. Intuitively, a group is composed of a medley of individuals. While recommendation techniques targeting individuals are extensively studied, research into group recommendation has been limited. In fact, group preferences are not straightforward to model, given the inherent complexity of group dynamics. To tackle this problem, this dissertation aims to propose a new method that can exploit the interactions between group members in order to drive the model towards highly effective group-level recommendations. Moreover, it has been observed that the collective decisions of each member in a group are usually dynamic, i.e., a user's preference may be highly influenced by the other members in the group. Indeed, the final decision of a group tends to require a consensus amongst group members, in which this consensus heavily depends on the roles and expertise of each member. Therefore, it is crucial to model the interactions among group members. However, existing proposals for group recommendation fail to model the interactions of group members effectively [ARC+09, CM13, GLRW13, LTYL12, YLL12, YCL14, KBV09, BMR10, OCKR01,

MSC+06]. Most of existing solutions belong to memory-based methods that are based on either preference aggregation [KBV09] or score aggregation strategy [BMR10, OCKR01] and do not consider the interactions of group members. These strategies overlook the interactions between members in a group by simply using trivial methods to aggregate its members' preferences. Some existing solutions are model-based approaches and try to exploit user interactions for group recommendation. However, they cannot fully utilize the user interactions. And more importantly, modelling group representation via an end-to-end representation learning remains challenging. Therefore, this dissertation also deals with designing highly effective recommender systems that are targeted at modeling group preferences as opposed to individual preferences, where groups are ad-hoc (any combination of individuals) rather than pre-defined.

To this end, this dissertation is concerned with the task of developing and designing effective neural architectures using representation learning techniques for both personalized (or collaborative) and group recommendation ranking, in which a ranked list of prospective candidate items is served to each user or each group. More specifically, in this dissertation, we tackle the overall recommendation scenario with two different approaches: one is for personalized recommendation and one is for group recommendation. We introduce, in total, four novel representation learning techniques for both recommendation tasks.

1.2 Research Problems and Methodologies

1.2.1 Wasserstein based Metric Learning Representation for Recommendation

In the context of representation learning, metric learning has attracted extensive research in the past few years across different domains [WDWK11, CHL05, XNJR02, KTW+12, CHL05, HYC+17, TATH18, WBS05]. The basic idea of metric learning is to produce a distance metric between two objects to represent for their similarity. The two objects are more similar if their distance is small and vice versa. In representation learning for personalized recommendation, metric learning is adopted as a matching function between users and items. Recently, it has been widely utilized in recommendation tasks due to its crucial triangle inequality property, which is omitted in inner product based recommenders [RFGS09, HLZ+17, HDW+18].

To this end, several metric learning-based recommender systems have been proposed [TATH18, HYC+17, KS10, CMTJ12, PKXY18]. However, most of existing metric learning methods represent each user and item embedding by a single shallow level of pull and push mechanism [HYC+17, CMTJ12, TSL19]. In this way, the formation of the pull

and push mechanism indeed could be considered as deterministic as each user and item is presented as only a single point in a specific low-dimensional vector space.

Approaches and Methodology. For the first time, we propose a *Wasserstein distance-based Metric Learning Chain* (*W-MLC*) model, which utilizes the notion of metric learning chain, and allows a series of metric learning computations between user and item latent representations. Instead of learning user and item in a standard metric learning approach with one shallow layer, *W-MLC* dives into deeper levels with multiple pairwise objectives of metric learning. There are two novel aspects in *W-MLC*: (i) restriction and (ii) relaxation. Specifically, we project user/item embeddings into different spaces (i.e. creating a chain) to perform metric learning on multiple levels (restriction), whereas we introduce an adaptive margin for each user at different chain (relaxation). Different from standard metric learning methods, the metric learning chain allows us to encode richer semantic relations, providing more freedom to cluster similar items together and keep dissimilar items far apart. To this end, we preserve the pull and push mechanism of metric learning at different levels, while simultaneously enhancing the performance and reducing biases through our personalized and adaptive margin hinge loss.

1.2.2 Going Beyond Euclidean: Hyperbolic Representation for Recommendation

As discussed, a diverse plethora of machine learning models solves the personalized ranking problem in recommender systems via building matching functions [RFGS09, Ren10, SM07, HLZ+17]. Across the literature, a variety of matching functions have been traditionally adopted, such as inner product [RFGS09], metric learning [TATH18, HYC+17] and/or neural networks [HLZ+17, HDW+18]. Among those approaches, metric learning models (e.g., Collaborative Metric Learning (CML) [HYC+17] and Latent Relational Metric Learning (LRML) [TATH18]) are primarily focused on designing distance functions over objects (i.e., between users and items), demonstrating reasonable empirical success for collaborative ranking with implicit feedback. Nevertheless, those matching functions only covered the scope of Euclidean space.

Approaches and Methodology. For the first time, our work explores the notion of learning user-item representations in terms of metric learning in hyperbolic space, in which hyperbolic representation learning has recently demonstrated great potential across a diverse range of applications such as learning entity hierarchies [NK17] and/or natural language processing [TTH18, DSN+18]. Due to the exponentially expansion property of hyperbolic space, we discovered that metric learning with the pull-push mechanism in hyperbolic space could boost the performance significantly: moving a point to a certain

distance will require a much smaller force in hyperbolic space than in Euclidean space. To this end, in order to perform metric learning in hyperbolic space, we employ Möbius gyrovector spaces to generally formalize most common Euclidean operations such as addition, multiplication, exponential map and logarithmic map [GBH18b, Ung09].

1.2.3 Neural Architecture Dropout Representation for Recommendation

Across the literature, many recommendation models have been proposed using deep learning for user/item representations [RFGS09, HLZ+17, ZYST19, XYH+17, HC17, XHZ+19, TLLK19, HYC+17]. A variety of architecture designs have been nominated, in which they consider the shallow pyramid multi-layered perceptrons (MLP) structure as an essential part of the model [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. For example, [HLZ+17] proposed a fusion of matrix factorization (MF) and MLP by concatenating their last hidden layer to combine the two models, [ZACC17] jointly learned user and item representations and passed their element-wise product into a pyramid MLP, [HC17, XHZ+19] integrated an MLP on top of the architecture before the prediction layer, or [TLLK19] squared the last hidden layer of an MLP to make a square-distance MLP for metric learning. As a result, these models mostly pay attention to explore the notion of stacking multiple layers to capture the non-linearities of user-item interactions. Different from existing methods, in this thesis, we consider that an MLP of stacking multiple hidden layers may carry overlapping information across layers, in which parts of a neural network architecture could be removed to avoid redundancy.

Recent years have witnessed a success of attention mechanism with a wide range of applications in recommendation domain [XYH+17, TLH18, TLLK19, TSL19, CZJC18, TZLH18]. Several attention mechanisms with either vanilla attention or co-/self-attention have been applied to recommendation such as general attention [XYH+17, CZJC18], multi-head attention [TLH18] or metric learning attention [TLLK19, TSL19]. However, to the best of our knowledge, existing attention-based recommendation models only focused on handling user-item relations or extracting important features of side information. In this thesis, we introduce a new approach of attention mechanism on activating only specific parts of the neural architecture, achieving similar effect as dropout [HSK+12]. Indeed, we perform attention mechanism on hidden layers directly to capture the most informative layers to build more accurate and better representations, which is also the key difference of our work.

Approaches and Methodology. For the first time, we introduce a new concept of exploring attention mechanism as a regularization effect in the application of personalized recommendation. We propose a conceptually simple informative and effective model, in

which it is capable of not only achieving highly competitive results, but also providing explainable representations due to the impact of attention mechanism. The proposed paradigm reinforces the main idea of learning user-item representations, while maintaining the simplicity and effectiveness of the previous proposed models.

1.2.4 Learning Representation for Group Recommendation

People often participate in activities in groups, e.g., having dinners with colleagues, watching movies with partners, and shopping with friends. This calls for effective techniques for group recommendation. Unfortunately, existing recommendation algorithms designed for individuals are not effective for group recommendation. With the emerging of group event data, the research community further develops the research interest on making effective recommendations for a group of users [ARC+09, CM13, GLRW13, LTYL12, YLL12, YCL14, KBV09, BMR10, OCKR01, MSC+06] which facilitates groups making decisions, and helps social network services improve user engagement. This dissertation is concerned with designing highly effective recommender systems that are targeted at modeling group preferences as opposed to individual preferences, where groups are ad-hoc (any combination of individuals) rather than pre-defined.

Group preferences are not straightforward to model, given the inherent complexity of group dynamics. To this end, this work aims to exploit the interactions between group members in order to drive the model towards highly effective group-level recommendations. Moreover, it is natural that collective decisions have a tendency to be dynamic, i.e., a user's preference may be highly influenced by the other members in the group. Group-level agreement tends to require a *consensus* amongst group members, in which this consensus largely depends on each member's roles and expertise.[2] As such, it is crucial to model the *interactions* among group members. However, existing proposals for group recommendation fail to model the interactions of group members well. Most of existing solutions belong to memory based methods that are based on either preference aggregation [KBV09] or score aggregation strategy [BMR10, OCKR01] and do not consider the interactions of group members. These strategies neglect the interactions between members in a group, and use simple methods to aggregate its members' preferences. Some existing solutions are model-based approaches and try to exploit user interactions for group recommendation. However, they cannot fully utilize the user interactions.

Approaches and Methodology. To tackle the aforementioned challenges, we propose a new neural architecture for group recommendation. Specifically, our architecture is a new variant of the attention mechanism in which each group member is represented

[2]While this is not explicitly captured with any semantic information or meta-data, we hypothesize that this can be implicitly captured with simply interaction data.

with a single sub-attention network. Subsequently, a group of users is then represented as a *'medley'* of sub-attention networks that is responsible for making the overall recommendation. The role of each sub-attention network is to capture the preference of its representative group member, *with respect to* all other members in the group. As such, our proposed model leverages user-user interactions for making group recommendation decisions, which is not only well-aligned with the fundamental intuition of group-level dynamics but also expressive in the sense that it considers the user-user interactions. In fact, our experiments demonstrate that a simple attentive aggregation of user representations is insufficient and has roughly identical performance to that of an average pooling matrix factorization (MF) baseline. On the other hand, our experiments show that our model is significantly better than seven state-of-the-art baselines. All in all, our core intuition serves as an inductive bias for our model, providing more effective group recommender performance on multiple benchmark datasets.

1.3 Datasets, Evaluation Protocol, Metrics, and Baselines

1.3.1 Datasets

We use multiple widely-adopted benchmark and real-world datasets with diverse domains and densities to conduct extensive experiments on both personalized and group recommendation task. In particular, the descriptions of our datasets used in this dissertation are listed as bellows:

- **MovieLens**: A widely adopted benchmark dataset in the application domain of recommending movies to users provided by GroupLens research[3]. In this dissertation, we adopt four available configurations: MovieLens100k, MovieLens1M, HetRec, and MovieLens20M. Additionally, we also introduce another two generated datasets for group recommendation task, namely MovieLens-Simi and MovieLens-Rand, which will be presented in details in Chapter 7.

- **Epinions**: A consumer review dataset that was crawled on Epinions website and originally introduced in [MA07].[4]

- **Yelp**: A business directory and crowd-sourced review platform. We adopt the dataset from the Yelp Dataset Challenge 2018.[5]

[3]https://grouplens.org/datasets/movielens/
[4]http://www.trustlet.org/downloaded_epinions.html
[5]https://www.yelp.com/dataset/challenge

- **Tafeng**[6]: A grocery transaction dataset, which contains four months of transactions from November 2000 to February 2001 by *T-Feng* supermarket.

- **Tmall**[7]: A retail platform named *Tmall* that contains shopping logs of users. In this dissertation, we randomly pick 100k transactions for evaluation since the original dataset is extremely large.

- **Goodbooks**: A large book recommendation dataset contains six million ratings for ten thousand most popular (with most ratings) books.[8]

- **Meetup**: An event-based social network dataset. We use the dataset includes event-user pairs from NYC that was provided by [PLCZ16].

- **Amazon Reviews**: The amazon review datasets that was introduced in [HM16a]. In this dissertation, we choose the subsets[9] that are selected based on promoting diversity based on dataset size and domain. The details of each subcategories (i.e., subsets) will be introduced in the experimental section of each chapter.

- **Ciao**: A rating and review platform dataset were introduced in [GZTY14]. We use the entire crawled DVDs category dataset in the period from July 2000 to November 2013.[10]

- **Plancast**: An event-based social network dataset that is similar to Meetup where it allows users to directly follow others' event calendars. The dataset was originally introduced in [LHT+12].

For personalized recommendation task, we use a subset of the listed datasets for the different methods presented in this dissertation. Notably, we are aware that for the first three chapters that focusing on building personalized recommenders, there exists the different datasets used in the experiments. Although it may be acceptable to exploit a subset of the datasets to introduce new architectures, we understand that there still exists several ways to strengthen the experiments such as conducting more experiments across all the datasets for all the models. Therefore, we further introduce Chapter 6 as an extension of Chapter 3, 4 and 5; in which we conduct a very extensive experiment called a one-off comparison between all the introduced architectures across all the benchmark

[6] https://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset
[7] https://tianchi.aliyun.com/datalab/dataSet.htm?id=1
[8] https://github.com/zygmuntz/goodbooks-10k
[9] Datasets are obtained from http://jmcauley.ucsd.edu/data/amazon/ using the 5-core setting with the domain names truncated in the interest of space.
[10] https://www.ciao.co.uk/

datasets. Indeed, interesting insights and observations of the comparison will also be presented later in the chapter.

For group recommendation task, we particularly use MovieLens, Meetup and Plancast datasets, as they are the benchmark datasets that commonly use for the group-level recommendation. All the statistics of the datasets will be separately introduced in each chapter. In addition, we will also present the prerocessing steps of each dataset in details in each chapter for reproducibility.

Moreover, we particularly focus on the implicit feedback problem for recommendation in this dissertation. In other words, we focus on the problems of user-item recommendations and group-item recommendations without the usage of any side information such as item attributes, user profiles, etc. As such, we binarize the explicit data by keeping existing ratings as implicit feedback. In other words, all the rated entries of the interaction matrix are converted to ones, while all the missing entries are filled with zeros.

1.3.2 Evaluation Methodology

Throughout this thesis, for personalized recommendation task, we adopt the widely-used *leave-one-out* protocol for the model evaluation [HLZ+17, RFGS09, TLLK19]. Particularly, for datasets with available interaction timestamp (e.g. *Movielens* and *Yelp* datasets), the lastest interacted item of each user is held-out as the test set, the penultimate is used for the development set, and the rest of interactions are for the training set. For datasets without interaction timestamp (e.g. *Epinions*, *30Music* and *8tracks* datasets), for each user, we randomly pick one interacted item for testing set, another interacted item for development set, and the rest of interactions for training set. Moreover, we randomly select 100 negative samples which the user have not interacted with and rank the ground truth amongst these negative samples [HLZ+17, TATH18]. During training, we report the test scores of the model based on the best validation scores. All models are evaluated on the validation set at every 50 epochs.

On a side note, for group recommendation task, we randomly split each dataset into training, tuning and testing data with the ratio of 70%, 10% and 20%, respectively. Note that the groups in our setting are ad-hoc, i.e., it is possible that a group appears only in test data, but not in training data.

1.3.3 Performance Metrics

In this dissertation, we focus on the accuracy of top-K recommendation task where K is the number of recommendations. Specifically, we measure the performance by considering the number of times the test item apprears in the top-K recommended items for a target user. In particular, we evaluate model performance using four metrics: *precision*

(prec@K), *recall* (rec@K), *Normalized Discounted Cumulative Gain* (NDCG@K) and *Hit Ratio* (HR@K), which are well-established ranking metrics for recommendation tasks. Notably, the choice of the K value indeed depends on the task, where we usually set $K = 10$ for personalized recommendation task, and evaluate recommendation accuracy for group recommendation with $K = \{5, 10, 20\}$. In particular, prec@K is the fraction of top-K recommendations selected by a user (or a group), rec@K is the fraction of relevant items (true items) that have been retrieved in the top-K relevant items, NDCG@K measures how well a method can evaluate the rankings of true items in the recommendation list, and HR@K simply measures whether the test item is in the top-K recommended list or not.

Furthermore, when comparing our proposed approaches to the competing baselines, we observe some performance differences. As such, we need to evaluate how significant the improvement of our approaches to make sure that the results are reliable. To this end, we perform paired t-tests to report the performance difference so that our approaches are indeed statistically significant. In some experiments, we run N times and access the average results of the methods, where we usually set $N = \{5, 10\}$.

1.3.4 Comparison Algorithms

Personalized Recommendation. In this dissertation, we compare our proposed methods against other competing top-K recommendation algorithms. In particular, the baselines used in the personalized recommendation task are listed below, in which we categorized into four groups: 1) traditional state-of-the-art approaches (MF-BPR [RFGS09], PMF [SM07]), 2) general deep representation learning approaches (MLP, JRL [ZACC17], NeuMF [HLZ+17], CMN [ESF18]), 3) metric learning based approaches (CML [HYC+17], LRML [TATH18]), and 4) sequential approaches (PRME [FLZ+15], TransRec [HKM18], Caser [TW18]).

- **Matrix Factorization with Bayesian Personalized Ranking (MF-BPR)** [RFGS09]: It is a state-of-the-art matrix factorization model for implicit feedback recommendation, where the user-item interactions are modeled using the inner product.
- **Probabilistic Matrix Factorization (PMF)** [SM07] is a strong pointwise baseline based on matrix factorization which is built upon user-item pairs. It explores a probabilistic approach on matrix factorization for recommendation.
- **Multi-layered Perceptron (MLP)** [HLZ+17] is a feedforward neural network that applies multiple layers of non-linearities to capture the relationship between users and items. We choose the number of MLP layers from $\{3, 4, 5\}$. In this dissertation, we use two versions, one with pyramid structure [HLZ+17] and the other one has hidden layers with equal dimension, which will be introduced in the later chapters.

- **Joint Representation Learning (JRL)** [ZACC17] is a strong baseline that explores the stack hidden layers by passing the element-wise product of user and item latent vectors into a pyramid multi-layered perceptron for prediction. We also propose and compare with another variant of JRL.

- **Neural Collaborative Filtering (NeuMF/NCF)** [HLZ+17]: a strong neural network based recommendation model which models nonlinear user-item interactions. The key idea of NeuMF is to fuse the last hidden representation of MF and MLP into a joint model. We pre-trained two components of NeuMF to obtain its best results, denoted as NeuMF++.

- **Collaborative Memory Network (CMN)** [ESF18]: It is a strong recommender system based on memory network. The framework exploits user neighborhood-based CF and additionally integrates a memory network to learn attentive weights for similar users. We also use a pre-trained version of CMN, in which we denote as CMN++.

- **Collaborative Metric Learning (CML)** [HYC+17]: It is a strong metric learning baseline that learns user-item similarities using the Euclidean distance.

- **Latent Relational Metric Learning (LRML)** [TATH18]: It is also a strong metric learning baseline that learns adaptive relation vectors between user and item pairs to find its single optimal translation.

- **Personalized Ranking Metric Embedding (PRME)** [FLZ+15]: It is the personalized ranking-based metric embedding approach, which models the user preferences and the *first-order* Markov sequential transitions in two separate latent Euclidean spaces.

- **Translation-based Recommendation (TransRec)** [HKM18]: It is also a very strong sequential recommendation methods. *TransRec* utilizes the Euclidean distance to model the third order relationships between the user, the previous item, and the next item.

- **Convolutional Sequence Embedding Recommendation Caser)** [TW18]: It is another strong sequential model that explores convolutional neural network (CNN) with vertical and horizontal kernels to capture the complex relationships between items.

Note that for personalized recommendation task, our target is to develop new effective recommendation architectures and the baselines are chosen based on the different objectives in each chapter. In particular, we have:

- Chapter 3 focuses on developing an advanced metric learning architecture to overcome limitations of previous metric learning representation studies. Thus, we choose the existing metric learning based approaches as our core baselines. In addition, we also compare our proposed model with some key popular sequential models to demonstrate the generality of our framework.

- Chapter 4 further investigates hyperbolic metric learning architecture in hyperbolic space for recommendation. Thus, we also consider metric learning approaches in Euclidean space as our main competitors to illustrate the potential of using hyperbolic space over Euclidean space for metric learning.

- Chapter 5 proposes a general deep learning module that could fit into any MLP-like architectures to enhance the performance by introducing a regularization effect. Therefore, the core baselines in this chapter would be the models that were built upon MLPs.

Moreover, we also compare our proposed architectures with their variants and conduct extensive ablation studies to demonstrate the effectiveness and generalization of our methods. The details behind each proposed method will be described in Chapter 3, 4, and 5 later. On a side note, we notice that we could also further consider all the baselines for one-off comparison with all the proposed models in the personalized recommendation task. By doing this, we could create the overview performance comparison of different types of general recommenders. Therefore, we introduce another chapter with a very extensive experiment on providing comparison between all general introduced recommenders across all datasets presented above:

- Chapter 6 provides a very extensive one-off comparison between all the introduced architectures in the previous chapters across all the benchmark datasets. Thus, we could also consider this comparison as a small survey on implicit feedback recommendation problem, providing interesting insights and observations on the existing approaches compared to proposed representation learning techniques. Moreover, we also provide findings and opinions for the readers to have the general view on this implicit feedback direction in the personalized recommendation field.

Group Recommendation. For the group recommendation task, we compare with seven state-of-the-art baselines, in which we categorized into three groups: 1) traditional approaches (CF-AVG, CF-LM, CF-RD [ARC+09]), 2) probabilistic approachs (PIT [LTYL12], COM [YCL14]), and 3) neural network approaches (MF-AVG, ATT-AVG).

- **User-based CF with averaging strategy (CF-AVG)**: CF-AVG applies user-based CF to calculate a preference score for each user with respect to a candidate item i, and then averages the preference scores across all users to obtain the group recommendation score of item i.

- **User-based CF with least-misery strategy (CF-LM)**: Similar to CF-AVG, CF-LM first applies user-based CF to calculate a score for each user on item i. However, the recommendation score of item i is taken as the item's lowest preference score across all users.

- **User-based CF with relevance and disagreement strategy (CF-RD)** [ARC+09]: CF-RD uses CF-AVG or CF-LM to compute the group relevance score. It also considers a disagreement score across the group members, such as using the average pair-wise relevance difference (the average pair-wise disagreement method), or the mathematical variance of the relevance (the disagreement variance method).

- **Personal impact topic model (PIT)** [LTYL12]: PIT is an author-topic model. Assuming that each user has an impact weight that represents the influence of the user to the final decision of the group, PIT chooses a user with a relatively large impact score as the group's representative. The selected user then chooses a topic based on her preference, and then the topic generates a recommended item for the group.

- **Consensus model (COM)** [YCL14]: COM relies on two assumptions: (i) the personal impacts are topic-dependent, and (ii) both the group's topic preferences and individuals' preferences influence the final group decision.

- **Average Matrix Factorization (MF-AVG)**: This baseline is a simplified version of MoSAN and considers the average embedding of all users in the group. All users are weighted equally. We represent a group as $g = \sum_i w_i u_i$ where $w_i = \frac{1}{n}$, and we optimize the BPR objective to predict group recommendation scores.

- **Attentive Aggregation (ATT-AVG)**: This baseline is also a simplified version of MoSAN, and represents a group embedding using a vanilla attentive aggregation over all the user embeddings in the group. The user embeddings are optimized with the BPR objective function.

For both personalized and group recommendation task, the hyper-parameter settings of each competing baseline will also be presented in details in each relevant chapter.

1.4 Summary of Contributions

In this dissertation, we design and develop effective neural architectures for various recommendation tasks via different representation learning techniques. The contributions are summarized as follows:

- We propose a W-MLC (*Wasserstein distance-based Metric Learning Chain*) model that investigates the notion of metric learning chain for recommender systems via Wasserstein distance. To the best of our knowledge, this is the first work that explores a series of metric learning in different transformation spaces with the combination of Wasserstein distance in recommendation domain. Additionally, we also propose multiple loss functions and incorporate them into an end-to-end framework for adaptive model training. Specifically, unlike standard metric learning models, we learn the adaptive and personalized margin for each user under the pull and push mechanism. Moreover, we introduce the similarity measure between transformation matrices via Wasserstein distance, in which the 2-Wasserstein distance also satisfies the crucial triangle inequality property for metric learning. We conduct extensive experiments on eight widely-used benchmark datasets in three well-known recommendation tasks to demonstrate the effectiveness of our proposed model against eight state-of-the-art baseline recommendation methods.

- We investigate the notion of training recommender systems in hyperbolic space as opposed to Euclidean space by exploring Möbius gyrovector spaces with the Riemannian geometry of the Poincaré model. To the best of our knowledge, this is the first work that explores the use of hyperbolic space for metric learning in the recommender systems domain. We devise a new method HyperML (*Hyperbolic Metric Learning*), a strong competitive metric learning model for one-class collaborative filtering (i.e., personalized ranking). Unlike previous metric learning models, we incorporate a penalty term called *distortion* to control and balance between accuracy and preservation of distances. We conduct a series of extensive experiments delving into the inner workings of our proposed HyperML on **ten** public benchmark datasets. Our model demonstrates the effectiveness of hyperbolic geometry, outperforming not only its Euclidean counterparts but also a suite of competitive baselines. Notably, HyperML outperforms the state-of-the-art CML and LRML models, which are also metric learning models in Euclidean space across all benchmarks. We achieve a boosting performance gain over competitors, pulling ahead by up to 32.32% performance in terms of standard ranking metrics.

- We introduce a new kind of regularization effect, DropRec (*Dropout for Recommendation*), by exploring attention mechanism directly on hidden layers to avoid redundancy in the context of personalized recommendation. We propose two effective neural architectures, C-DropRec and S-DropRec, for collaborative filtering with implicit feedback. In this work, we design simple but informative and effective architectures, which enable incorporating attention mechanisms into hidden layers. We conduct extensive experiments on six widely adopted benchmark datasets for recommender system to demonstrate the effectiveness of employing attention mechanisms on hidden layers, consistently outperforming several recent strong competitive baselines with only stacked multi-layered perceptrons. Moreover, we also investigate the explainable insights of our two architectures to further exploit the effectiveness of attention mechanisms on providing regularization.

- We provide an extension experiment for the personalization task by comparing all the introduced representation learning techniques to all the baselines, across all the introduced widely-adopted benchmark datasets. Specifically, this one-off experiment includes 13 architectures with 12 datasets in terms of two evaluation metrics. We not only show that our proposed representation learning techniques are significant, but also providing interesting insights and observations, such as the performance of metric learning based recommenders compared to dot product based recommenders. Moreover, we also include our analyses and opinions on the results of the comparison. This excited experiment could be considered as a general benchmark for future models for comparison if they would like to conduct similar experimental settings.

- We propose MoSAN (*Medley of Sub-Attention Networks*), a novel deep learning architecture for the group recommendation problem. Our model distinguishes itself from all prior work in group recommendation based on the fact that it considers user-user interactions using sub-attention networks. To the best of our knowledge, this is the first neural model that explores the usage of user-user interactions for the group recommendation task. We conduct extensive experiments on four publicly available benchmark datasets. Our experimental results demonstrate that MoSAN achieves state-of-the-art performance, outperforming a myriad of strong competitors in the task at hand. In addition to comparison against well-studied baselines, we conduct ablation studies against two baselines AVG-MF (Average Matrix Factorization) and ATT-AVG (Attentive Aggregation) and observe that our approach significantly outperforms both approaches. This shows that our proposed model provides a more useful inductive bias for the task at hand. We show that the attention weights of MoSAN are interpretable, i.e., it is able to discover the different weights of each user across groups, highlighting the impact of each user in different groups.

1.5 Thesis Organization

Chapter 1 presents the research background, scope and overview of this dissertation.

Chapter 2 provides a detailed literature survey of deep representation learning techniques for personalized and group recommendation.

Chapter 3 introduces W-MLC (*Wasserstein distance-based Metric Learning Chain*), a metric learning chain representation architecture for recommendation. This model utilizes the notion of metric learning chain, and allows a series of metric learning computations between user and item latent representations. We show that our proposed model outperforms various recent strong metric learning baselines, indicating the effectiveness and the need of our *deep* metric learning-based model.

Chapter 4 introduces HyperML (*Hyperbolic Metric Learning*), hyperbolic representation learning for recommendation. The key idea is to model user-item pairs in hyperbolic space, while maintaining the simplicity and effectiveness of the metric learning paradigm. We show that the proposed framework achieves very competitive results and outperforms recent advanced Euclidean metric learning models on multiple personalized ranking benchmarks.

Chapter 5 introduces DropRec (*Dropout for Recommendation*), a new kind of regularization effect by exploring attention mechanism directly on hidden layers to avoid redundancy in the context of personalized recommendation. We show the remarkable recommendation results of our proposed model on the variety of datasets and the advantage of attention mechanisms in providing regularization effect to enhance the performance.

Chapter 6 introduces a one-off experiment of proposed architectures and baselines in the previous chapters in the context of personalized recommendation. We show the interesting recommendation results and insights of the experiment. We additionally provides our analyses and opinions on the results.

Chapter 7 introduces MoSAN (*Medley of Sub-Attention Networks*), a novel deep representation learning architecture for the group recommendation problem. Our model distinguishes itself from all prior work based on the fact that it considers user-user interactions using sub-attention networks. We show that MoSAN achieves state-of-the-art performance, outperforming a myriad of strong competitors, and is able to discover the different weights of each user across groups, highlighting the impact of each user in different groups.

Chapter 8 conclues this thesis, provides discussions, and introduces some new challenges and potential directions for future work.

Chapter 2

Literature Review

This dissertation is related to designing effective neural architectures for recommender systems, especially exploring different deep representation learning techniques in various recommendation tasks. In this chapter, we first give a general overview of recommender systems and deep learning in Section 2.1. Then, we break the gap between recommender systems and deep learning by reviewing deep learning based recommendation models in terms of representation learning in Section 2.2. After that, we dive into details by providing the overview of learning representation for personalized recommendation in Section 2.3. Specifically, we recall deep learning representation for recommendation in Section 2.3.1, introduce metric learning representation in Section 2.3.2, and then present hyperbolic representation in Section 2.3.3. Finally, in Section 2.4, we present detailed literature reviews for learning representation in group recommendation.

2.1 Review on Recommender Systems and Deep Learning

In this section, we first give an overview of recommender systems, and then provide theoretical foundations of deep learning by introducing a few popular deep learning frameworks that are usually considered as the foundations and widely-applied in recommendation domain.

2.1.1 Overview of Recommender Systems

Nowadays, recommender systems live at the heart of many popular online services platforms across different domains such as e-commerce (e.g., Amazon product recommendation), social networks (e.g., Facebook newsfeed recommendation), entertainment (e.g., Netflix movie recommendation, Spotify music recommendation). In general, these recommender systems have been designed to provide recommendations based on the

preferences of users. In fact, the systems estimate users' preference on items, generate candidate items that users might like, and recommend those items to users accordingly.

In the conventional setting of recommender systems, i.e., interaction-only setting, we have a system with a set of users U, a set of items I, and users' historical interactions information which usually represent as an interaction matrix H. Each entry $h_{i,j}$ of the matrix H refers to the interaction of user i and item j. Notably, in the context of group recommendation, we would also have a set of groups of users G as input. Moreover, the interactions (i.e., the entries of the matrix) can be *explicit feedback* such as movie ratings from 1 to 5 scale, or *implicit feedback* such as click, purchase, watch, listen and so on. In real-world applications, explicit feedback information is hard to collect as it is expensive and requires users to put more efforts in giving feedback to the systems. As such, most of the time, we usually deal with implicit feedback setting since the information is easier to gather and more abundant in forms. Therefore, in this dissertation, we focus on solving various recommendation tasks with the implicit feedback setting. On a side note, in the experiments, we also consider the sequence of the interactions by adding the timestamp to replicate the real-world setting.

In addition, performance metric is also an important factor to evaluate a model. In this dissertation, we mainly focus on the accuracy of the model, although there could be other metrics to consider in real-world systems such as privacy, interpretability, or diversity. Since we concern with the implicit feedback setting, we pay attention to the ranking metrics. Specifically, we use *Precision* (Prec), *Recall* (Rec), *Normalized Discounted Cumulative Gain* (NDCG), and *Hit Ratio* (HR). We will give details of those metrics in the later chapters.

2.1.2 Foundations of Deep Learning

In light of the increasing success of deep learning, recent studies have proved the benefits of using deep learning in various recommendation tasks. Recommendation architectures have been utilizing deep learning in order to overcome limitations of traditional recommendation techniques. In this section, we briefly introduce a few basic neural networks that have been widely used in many recommendation models. In the later section, we will introduce how these models become the foundations for building neural architectures for recommendations in terms of representation learning.

2.1.2.1 Matrix Factorization

Matrix Factorization (MF) [KBV09] is a traditional and popular technique for developing recommender systems. The objective of MF is to learn the low-dimensional representations of users and items and perform the inner product to these representations to disclose the interests of users on different items. In other words, MF decomposes

the rating matrix into low-dimensional user/item latent representations. Particularly, given an interaction user-item matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$ where M is the number of users and N is the number of items, MF decomposes $\mathbf{R}$ into a user representation matrix $\mathbf{U} \in \mathbb{R}^{M \times K}$ and a item representation matrix $\mathbf{I} \in \mathbb{R}^{N \times K}$ with K is the latent representation (i.e., embedding size). In formal, the objective function is defined as:

$$\min_{\mathbf{U},\mathbf{I}} ||\mathbf{R} - \mathbf{U}\mathbf{I}^\top||_F^2 + \lambda_1 ||\mathbf{U}||_F^2 + \lambda_2 ||\mathbf{I}||_F^2, \tag{2.1}$$

where $||\cdot||_F$ is the Frobenius norm; λ_1 and λ_2 are regularization parameters.

Notably, MF has become the foundations of many neural recommenders in recent years. In fact, various variants of MF have been proposed in which many of them becomes the very strong baselines for different recommendation tasks such as WMF (*Weighted Matrix Factorization*) [HKV08, PZC+08] or MF-BPR (*Bayesian Personalized Ranking*) [RFGS09]. Recently, a very well-known neural architecture called NeuMF [HLZ+17] also leverages MF as part of the model to learn duel embeddings for downstream tasks.

2.1.2.2 Multi-layered Perceptron

Multi-layerd Perceptrons (MLP) is also a traditional and popular technique which usually consists of multiple fully connected layers with one input layer, few hidden layers, and one output layer. The most important highlight of MLP is the capability to capture the non-linearities through the information transformations across layers. Specifically, an MLP with L layers is defined as follows:

$$\begin{aligned} h_1(x) &= f_1(W_1 x + b_1) \\ h_2(x) &= f_2(W_2 h_1 + b_1) \\ &\cdots \\ h_L(x) &= o(W_L h_{L-1} + b_L), \end{aligned} \tag{2.2}$$

where W_* and b_* are the weights and biases; f_* is the non-linear activation function, in which the common choice of f are usually rectifier function (ReLU): $f(x) = max(0, x)$, hyperbolic tangent function (tanh): $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$ or sigmoid function (sigmoid): $f(x) = 1/(1 + e^{-x})$; and $o(x)$ is the output activation function that is customized based on the tasks.

In recommender systems, MLP has become an essential part for many neural recommenders [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. In particular, we empirically observe that the optimal setting of MLP in recommendation domain is often set to $\{3, 4, 5\}$ layers, with the activation function of ReLU or LeakyReLU [MHN13]. Notably, an MLP commonly has a pyramid structure where the latent dimension of the

subsequent layer equals to half of the dimension of the previous layer, i.e., $d_k = \frac{d_{k-1}}{2}$ with d_k is the dimension of layer k. However, we will show that it is not always the case, where we further introduce a new type of MLP with hidden layers of equal dimension in the later chapters.

2.1.2.3 Autoencoder

Different from MF and MLP, Autoencoder (AE) is an unsupervised neural network, where the framework is typically and originally used for dimensionality reduction. Typically, AE consists of three layers: one input layer, one hidden layer (i.e., the bottleneck layer), and one output layer. In this framework, the input and output layer have the same latent dimension, whereas the dimension of the middle layer is usually much smaller and considered as the low-dimensional representation. In addition, we refer to the process from input to hidden layer as *encode*, and the process from hidden to output layer as *decode*. Specifically, given the input x with the latent dimension of d, we have:

$$z = f_1(W_1 x + b_1), \tag{2.3}$$

$$x' = f_2(W_2 z + b_2), \tag{2.4}$$

where $z \in \mathbb{R}^k$ is the low-dimensional representation (i.e., the hidden layer) with $k << d$; and x' is the resconstructed representation from the input x.

Moreover, AE learns the parameters by minimizing the reconstruction error as follows:

$$\mathcal{L}(x, x') = \|x - x'\|^2. \tag{2.5}$$

In general, AE can be applied into recommender systems with two approaches: 1) learn low-dimensional feature reprentation of users/items at the bottleneck layer, and 2) fill the missing entries of the interaction matrix directly in the reconstruction layer. To some extent, AE can also be stacked to formulate deep neural networks. There are many variants of AE such as variational AE [Doe16], constrative AE [RVM+11], and marginalized AE [CXWS12], and almost all of them can be applied into recommendation tasks. Additionally, there are some favourite AE-based recommendation models that attract recent attention from the community such as CDAE (*Collaborative Denoising Auto-Encoder*) [WDZE16] or Multi-VAE/DAE [LKHJ18]. Interestingly, Multi-VAE/DAE is one of the very few popular models that achieves superb performance in terms of accuracy [DCJ19].

2.1.2.4 Recurrent Neural Network

Different from previous deep neural networks that assume all inputs to be independent, Recurrent Neural Network (RNN) [GMH13] is a class of deep neural networks dealing with the dependencies patterns of sequential information. Indeed, RNN allows operation on sequential/time series data to capture the dependencies. There are two popular variants of RNN were proposed: Long Short-Term Memory (LSTM) [HS97] and Gated Recurrent Unit (GRU) [CvMG+14]. The two variants are introduced as belows.

Long Short-Term Memory (LSTM) [HS97] is proposed to tackle the problem of gradient vanishing and exploding. Specifically, the framework consists of three functions act as gate keepers when processing the sequence at each time step. Particularly, at time step t, the non-linear operations of LSTM is defined as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.6}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.7}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{2.8}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.9}$$
$$h_t = o_t \odot tanh(c_t), \tag{2.10}$$

where i_t, f_t and o_t are the input, forget and output gates; W_*, b_* are the weights and biases; $[\cdot, \cdot]$ denotes vector concatenation; σ represents a sigmoid function and $\odot$ represents the Hadamard product.

Gated Recurrent Unit (GRU) [CvMG+14] also adopts the gate mechanisms to control the previous memory and current inputs of the sequence, which is also known as a simplified version of LSTM. Moreover, GRU additionally introduces an update gate and a reset gate at each time step. Mathematically, GRU can be expressed as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \tag{2.11}$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \tag{2.12}$$
$$\tilde{h}_t = tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h), \tag{2.13}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{2.14}$$

where the notations are similar to LSTM. Notably, h_t represents the hidden state that encodes sequential information at time step t.

With the ability to handle sequential information, RNNs become the natural choice for dealing with session-based recommendation problems such as next-item recommendation [ZTY+19], next-POI recommendation [FLZ+15] or shopping-basked recommendation [TLLK19]. For example, GRU4Rec [HKBT16] is one of the most famous RNN-based models for recommendation that shows promising results in both academia and industry.

2.1.2.5 Convolutional Neural Network

Convolutional Neural Network (CNN) [KSH12] is a class of deep neural networks with convolution layers and pooling operations that handles grid-like data such as image and video. A typical CNN usually consists of multiple convolutional layers, each of which contains a set of kernels, denoted as K, that extract features from the local areas of its input by performing convolution. We give an example of performing convolution by the first convolutional layer on the input X as follows:

$$F_{i,j} = \sum_m \sum_n X_{i-m,j-n} K_{m,n}, \tag{2.15}$$

where m and n are the position indexes of a filter; i and j represent a position on the grids X and F. The output F of each convolutional layer is a feature map and considered as the input for the next convolutional layer.

In recommendation domain, most CNN-based recommenders typically utilized CNN architectures to extract useful features. For instance, [WWT+17] uses CNN to extract visual features of images, [ZNY17] adopts CNN for modeling user behaviors and dealing with review texts, or [vdODS13] proposes CNN to extract music signals for song/playlist recommendation. In addition, there exists direct approaches of using CNNs for collaborative filtering and graph-based models [HDW+18, TW18, YHC+18, vdBKW17], where these models also shows comparable and promising results.

2.1.2.6 Word2Vec

Word2Vec [MSC+13] is a popular technique that consists of two-layer neural networks, in which the framework was developed to generate low-dimensional word embeddings. Indeed, the word representations learned by Word2Vec have rich and semantic representations, with the ability to provides interpretability to the model. Moreover, the architecture design of Word2Vec also enables the model to capture the relationship between a word and its surroundings. In terms of implementation, the skip-gram architecture is usually adopted, in which the objective is to predict the surrounding context words w_c given a current word w_i (i.e., the target word). The objective function is described as:

$$\arg\max \sum_{w_c \in C(i)} \log p(w_c|w_i), \quad \text{where} \quad p(w_c|w_i) = \frac{\exp(w_c^\top w_i)}{\sum_{w \in \mathcal{W}} \exp(w^\top w_i)}, \tag{2.16}$$

where $C(i)$ represents the set of context words of w_i; and $\mathcal{W}$ represents all the possible words in the corpus.

On a side note, Word2Vec also employs two additional techniques, i.e., hierarchical softmax [MB05] and negative sampling [GH12], to reduce the training time cost. The obtained word representations can then be used for different downstream tasks.

Similar to previous methods, various recommendation models also employ the Word2Vec framework to solve various tasks. Among of which, the most two popular models are Item2Vec [BK16] that learns item embeddings for collaborative filtering problem, and POI2Vec [FCAC17] that jointly models the user preference and POI sequential transition influence for predicting potential visitors to a given POI.

2.2 Representation Learning Techniques for Recommender Systems

With the proliferation of deep learning, deep learning techniques have been extensively applied in recommender systems thanks to their state-of-the-art performances and high-quality recommendations [CZH+17, CKH+16, CAS16, HLZ+17, ZYST19]. Deep learning is able to capture non-linear and non-trivial relationships between users and items, which provides better understanding of user demands and item characteristics, as well as the interactions between them. The success of deep learning in recommender systems has shown promising results across different recommendation tasks [ZYST19]. Indeed, the advantage of deep learning based recommendation models is the ability in learning powerful representations and easily incorporating side information of users and items such as user reviews or item categories. However, there are still very limited works that exploit deep learning techniques into group recommendation. We will give a comprehensive review later in Section 2.4.

Deep learning, to some extent, is also considered as a representation methodology to learn rich and meaningful representations from original inputs [GBC16]. As such, deep representation learning techniques become more and more favourable. Given the abundant auxiliary information such as user demographics, item descriptions and locations, incorporating these information become much more simple and straightforward, since we can easily get the final representations for downstream recommendation tasks by simply concatenating all the side information vectors or feeding those vectors into a one-layered neural network. Moreover, deep representation learning also makes the process of learning user and item representations becomes viable due to the capability of composing the whole neural framework into a single differentiable function for end-to-end training.

On a side note, deep representation learning techniques also have the strength to enhance the conventional recommendation models. For instance, integrating non-linear activation functions (i.e., sigmoid, tanh, ReLU) to the conventional models such as Matrix Factorization (MF) or Factorization Machine (FM) model could possibly lead to the

improvement of the performance since the enhanced models could then capture the non-linearities and complex structures of the data. In fact, there are several works achieve the improvement by generalizing these methods with deep learning, such as neural matrix factorization [HLZ+17, DR15] and deep factorization machine [HC17].

In this section, we review various representation techniques in recommender systems with deep learning based models, among of which are investigated and leveraged in our studies, either as our compared baselines or parts of our proposed architectures.

2.2.1 Representation Learning via Multi-layered Perceptron

MLP is a basic and powerful neural network that commonly-used in a wide range of appications across different recommendation tasks due to the ability of capturing non-linearities as introduced. Moreover, MLP is also known as a technique for feature representation learning due to its effectiveness and scalability [CAS16, CKH+16, ESH15, BK16]. Indeed, many real-world applications employ MLP as a tool for learning feature representations of users and items such as YouTube video recommendation [CAS16], Google Play Store Apps recommendation [CKH+16], or Microsoft TV & Apps recommendation [ESH15, BK16].

Representation learning via MLP has been fascinated recently, in which most of the proposed architectures consist of an MLP structure as an essential part of the model [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. For instance, [HLZ+17] blended MF and MLP for dual space embedding, [ZACC17] passed the element-wise product of user and item representations into an MLP to capture the non-linearity relationships, [HC17, XHZ+19] stacked an MLP on top of the overall architecture for prediction, or [TLLK19] proposed a square-distance MLP for metric learning.

On the other hand, representation learning via MLP also shows its effectiveness and efficiency in numerous industrial applications. For example, YouTube recommendation architecture [CAS16] consists of two components: candidate generation component and ranking component, in which each component explores an MLP to learn users and features representations. Another example of representation learning via MLP in industry-level is the Wide & Deep model [CKH+16], which is already deployed in the Google Play Store. Similar to [CAS16], the proposed architecture also consists of two components: the wide component and the deep component. The wide component is the conventional generalized linear model (GLM), whereas the deep component is simply an MLP. The design of the architecture is to balance between generalization and memorization of the model. Specifically, the wide component is used to memorize important features and the deep component produces the generalization.

All in all, we observe that MLP takes a critical role in feature engineering for industrial recommender systems, which illustrates the practical usage of representation learning techniques via MLP in both academia and industry.

2.2.2 Representation Learning via Autoencoder

As introduced, representation learning via AE mainly focuses on modeling non-linearities with the reconstruction layer due to the encoder and decoder structure. As such, various proposed models leverage the representation strength of AE by intergrating it into conventional recommendation models, especially for learning side information. For instance, CDL (Collaborative Deep Learning) [WWY15] incorporates AE to PMF (Probabilistic Matrix Factorization) model [SM07], CVAE (Collaborative Variational Autoencoder) [LS17] explores variational AE to enhance [WWY15], or mDA (Marginalized Denoising Autoencoder) [LKF15] employs marginalized AE to extract user and item features. Notably, most of the AE-based frameworks utilize representation learning techniques with AE to extract side information with different variants. One special example of using feature representation learning is AutoRec [SMSX15]. Instead of applying AE to extract side information, AutoRec directly adopts the rows and columns of the interaction matrix as inputs.

To this end, we observe the most powerful aspect of representation learning techniques via AE is that we can always easily extend the inputs to incorporate any additional features to make the hybrid models [SGM16, ZYX$^+$17]. Thus, representation learning via AE becomes the most promising direction. In fact, Multi-VAE/DAE [LKHJ18], one of the AE-based recommendation models, illustrates their remarkable performance in terms of accuracy across recent proposed deep representation learning techniques [DCJ19].

2.2.3 Representation Learning via Recurrent Neural Network

Many conventional recommendation models usually neglect the temporal dynamics in their architecture. Thus, these proposed models cannot effectively model the sequential patterns of user activities. Therefore, it is essential to explore representation learning techniques via RNN to process the sequential information for recommendation. Indeed, we next review representation learning techniques via RNN with the two popular sequential recommendation tasks: Session-based Recommendation [WCW19] and Sequence-aware Recommendation [QCJ18].

RNN for Session-based Representation Learning. Session-based recommendation is the task of predicting the next action based on the historical actions of a user in a given time frame. Thus, candidate items are usually generated based on the most recent interactions. Notably, user identifications are not always presented in this task. One of the very first neural architectures for session-based recommendation that ultilizes RNN is the GRU4Rec [HKBT16]. In particular, given T sessions where each session consists of a sequence of items, GRU4Rec takes the current target item with its session as input, and then generates the probability of being the next item for all the items in that session

by employing RNN. Indeed, GRU4Rec is considered as the foundation for many later proposed models that also explore RNN to build session-based recommender systems. For example, [HQKT16, Twa16] proposes to use another RNN to incorporate item features, [JL17] employs session-based neighborhood to enhance GRU4Rec, [TXL16] uses data augmentation and combine with GRU4Rec, or [LRC+17] explores attention mechanism [VSP+17] within the sessions to improve the performance. On a side note, when there exists user identifications, we can incorporate those information to generate personalized recommendations for each user in the context of session-based [QKHC17, RSL17].

RNN for Sequence-aware Representation Learning. Sequence-aware recommendation is different from the previous task as: 1) the input only has the sequential information, but not sessions, and 2) both user identifications and timestamp are utilized in this task for personalized recommendation. By considering the timestamp, we would be able to derive the historical interaction sequences of users with items, in which the sequence of interactions (i.e., timestamps) is usually ignored in conventional collaborative filtering task. Given a target user with his sequential interactions, the recommender aims to predict a list of items that the user will likely interact in the future. Similar to session-based recommendation, representation learning techniques via RNN are also adopted for this task due to their ability to handle sequential information. For example, [DLZ17] proposes three variants of GRU to integrate user features into the architecture, or [LWW+16] explores RNN to incorporate contextual information, e.g., location, weather and time, into the sequential historical interactions of users for recommendation. On a side note, shopping-basket recommendation task could also be considered as a special case of sequence-aware recommedation, where the key difference is that user's historical interactions are organized in baskets instead of only in sequences [YLW+16, WLW+15]. As such, representation learning techniques via RNN are also obviously employed for this task. Typical examples for this problem are the e-commerce platforms such as Amazon, in which users tend to put items in a basket whey they go shopping before making payment.

2.2.4 Representation Learning via Convolutional Neural Network

CNN architecture is designed to deal with grid-like topology data sources, with the ability to extract local and global feature representations for downstream tasks. Representation learning techniques via CNN are usually employed to process textual, visual, audio and video information. We next review several works that exploit deep representation learning via CNN to extract features in recommendation domain.

CNN for Text Representation Learning. In recommender systems, text representation usually refers to the reviews of users for particular items. To this end, several works employ CNN to extract meaningful representations for textual information, as CNN is known to be suitable for document modeling due to its convolution and pooling operations that can capture the contextual information globally and locally [ZNY17, CC17, KPO+16]. For example, DeepCoNN (Deep Cooperative Neural Network) [ZNY17] designs a couple of CNNs to jointly modeling both user and item reviews. To enhance the performance, [CC17] further extend the DeepCoNN architecture by adding an extra layer to represent the target user-item pair.

CNN for Image Representation Learning. As originally designed for grid-like data, CNN has understandably achieved amazing success on image tasks. As such, there exists several works that leverage representation learning via CNN to apply image processing in recommender systems. For example, VBPR [HM16b] proposes a framework that incorporate image to the well-known BPR [RFGS09] for image ranking task. Specifically, VBPR adopts CNN to extract visual features and simply incorporate into the BPR framework. In addition, recent works also utilize CNN to extract visual features for different recommendation tasks such as fashion-aware recommendation [HM16a], pairwise image recommendation [NCL18], POI recommendation [WWT+17] and restaurant recommendation [CT17]. Notably, popular deep representation learning architectures via CNN such as VGG16 [SZ15] becomes a prefer choice to extract visual features recently due to its incredible performance.

CNN for Audio Representation Learning. In addition to the previous applications, CNN is also known to learn representations for audio feature extraction. In recommender systems, CNN is commonly applied to extract audio features for music recommendation task.[1] For instance, [vdODS13] proposes using CNN to predict latent factors from music audio, where the CNN component extracts features from music audio signals with respect to different timescales for prediction.

2.2.5 Representation Learning via Word2Vec

Word2Vec [MSC+13] is a special and popular technique for representation learning compared to the previous introduced models. While previous conventional recommendation architectures focus on jointly learning the representation for users and items, it is not always feasible in terms of real-word applications. Firstly, in real-world systems, the

[1]Note that this music recommendation task refers to dealing with music signals for recommendation. It is different from the music recommendation task that only considers interactions between users and songs.

number of users and items are usually very high, from millions to billions, which makes the representation learning process becomes much more expensive. Secondly, the interactions between users and items are not always available in some cases. For example, some e-commerce platforms allow users to purchase and make transactions even without providing their information. To this end, previous representation learning techniques may not be appropriate in industry.

As introduced, Word2Vec [MSC+13] is a two-layer neural network that is used to learn rich and distributed low-dimensional word embeddings. Word2Vec adopts the skip-gram architecture to capture the relationship between a target word and its surroundings, in which the surrounding words are considered as the context words. Inspired by this architecture, Item2Vec [BK16] treats a sequence of items similar to a sentence to apply the Word2Vec framework. Specifically, Item2Vec aims to learn item representations and model the item-item relations by leveraging the framework. In experimental setting, the sequences of items can be extracted by considering the timestamp. In real-world applications such as e-commerce systems, the sequences of items can also be generated from historical logs such as shopping baskets or click sequences. Notably, Item2Vec neither incorporate spatial formation nor consider sliding window in the architecture as compared to Word2Vec. However, its usage of representation learning via Word2Vec demonstrates promising direction to apply Word2Vec for learning user and item representations in various recommendation tasks such as POI recommendation (POI2Vec) [FCAC17].

2.2.6 Representation Learning via Graph Convolutional Network

Graph Convolutional Network (GCN) [KW17, DBV16, ZCZ+18b, ZCZ18a, WPC+19] has been proven as a strong method and catch the attentions of the research community recently. A GCN based methods often design to model the local structural information of a node in a graph structure. Then, the graph convolutional operation is applied to update the representation of that node based on the signal passing of its neighborhoods. With the remarkable performance of GCN, representation learning via GCN is an emerging direction recently for recommendations, in which the proposed methods usually leverage the user-item graph structure or multi-hop neighbors for learning representations [WHW+19, HDW+20, ZLJ+18, YHC+18]. Specifically, these representation learning techniques are usually developed based on collaborative signals and user-item interaction data. The collaborative signals could imply the behavior similarity of users, whereas user-item interactions could be constructed as a bipartile graph to employ GCN. In fact, these techniques mainly focus on discover the complex relationships of users/items via graphs through high-order connectivities. On a side note, representation learning via

GCN could also be explored for social recommendations in which one can, for example, additionally build user-user social graph to help filter information of users.

Althought GCN-based recommenders have only been explored recently, there already exists many notable works on representation learning via GCN for recommender systems. For example, [vdBKW17] views matrix completion as link prediction on user-item interaction graph to perform rating prediction, or [ZLJ+18] discovers the hidden connectivity in the spectral domain with collaborative filtering. Moreover, [WHW+19, HDW+20] proposes representation learning techniques that are not only able to learn high-order connectivity, but also providing fast neighborhood aggregation and representation update for capturing the collaborative signals. In terms of large-scale recommendations, [YHC+18] introduces a new large-scale deep recommendation engine which was actually developed and deployed at Pinterest for production. In addition, [FML+19, WSF+19, WZG+19] utilize social information to construct user-user bipartile graph or item-item bipartile graph to apply representation learning techniques via GCN for social recommendations. A more comprehensive review on various representation learning techniques on graphs could be found at [TD19].

2.3 Representation Learning for Personalized Recommendation

As introduced in previous sections, deep representation learning techniques are extremely efficacious in learning latent representations for users and items, as well as any other contextual/side information. In this section, we recall deep representation learning techniques via MLP as they are most relevant to this thesis, and introduce two representation techniques that catch the attention of the community recently due to its effectiveness with state-of-the-art performance, namely metric learning representation and hyperbolic representation. Notably, we focus on personalized recommendation in this section, while group recommendation will be reviewed in the next section.

2.3.1 Neural Representation

Neural Recommender Systems. Neural networks have been extensively applied in recommender systems thanks to their high-quality recommendations [CKH+16, CAS16, OTOT17, HLZ+17, CZH+17, TLH18]. Particularly, deep learning is able to capture non-linear and non-trivial relationships between users and items, which provides in-depth understanding of user demands and item characteristics, as well as the interactions between them. A tremendous part of literature has focused on integrating deep learning into recommender systems to perform various personalized recommendation tasks, where a comprehensive review can be found at [ZYST19].

To this end, many recommender systems are designed to tackle the problem of collaborative filtering with implicit feedback [TATH18, SM07, RFGS09, HZKC16, Kor08, HYC+17, SM07, KBV09]. With a shift from traditional to neural networks approaches, many recent works mainly focused on designing architectures, in which they consider the shallow pyramid MLP structure as an essential part of the model [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. Specifically, [HLZ+17] combined MLP with MF for a dual embedding, [ZACC17] fed an element-wise product of users and items into MLP, [HC17, XHZ+19] assigned an MLP on top of the architecture before the prediction layer, or [TLLK19] proposed a shallow square-distance MLP as part of the architecture. As a result, these models explore the notion of stacking multiple layers to capture the non-linearities of user-item interactions but neglect the informative signal (i.e., implicit relationship) among the hidden layers of the network. However, there is still very little work that exploits neural techniques into group recommendation. As such, in this dissertation, we pay attention to representation learning techniques via MLP for personalized recommendation. In particular, we introduce a new effect on building neural architecure, which will be presented in Chapter 5. We will review the representation techniques for group recommendation in the later section.

Neural Attention. Among various representation learning architectures, attention mechanism is one of the most exciting recent advancements in deep learning [DCLT18, VSP+17, VTBE15, BCB15, CBS+15]. The usage of neural attention in recommender systems has also gained considerable interest, with many works that exploit this recent advance for standard recommendation tasks [TLH18, XYH+17, CZJC18].

Notably, recent years have witnessed a success of attention mechanism with a wide range of applications in recommendation domain [XYH+17, CZH+17, TLLK19, TLH18, TSL19]. For instance, [XYH+17] applied general attention on factorization machines, [CZH+17] used attention to extract image/video features, [TLH18] introduced multi-head attention with co-attention mechanism for reviews, or [TSL19, TLLK19] employed attention mechanism on distances as metric learning attention. Nevertheless, these approaches explored attention mechanism on the user/item embeddings instead of hidden layers, which is also the key difference from our work. On a side note, many recent works have proposed complex deep architectures for personalized recommendation [WHW+19, TLLK19, ESF18, ZWC19, CWTY19]. However, these methods require incorporating additional attributes to learn the representations. In contrast, this dissertation introduces a new regularization effect on personalized recommendation ranking task by proposing a simple informative neural architecture to enhance the performance. Moreover, we also leverage the strength of attention mechanism on building representation learning technique for group-level recommendation task, which will be introduced in Chapter 7.

2.3.2 Metric Learning Representation

Across the rich history of recommender systems research, a myriad of machine learning models have been proposed using matching functions to define similarity scores [RFGS09, Ren10, SM07, RFGS09, HZKC16, Kor08, HLZ+17, HYC+17]. Traditionally, a large number of the recommendation systems are mainly focused on factorizing the interaction matrix, combining the user-item embeddings using the inner product as a matching function to account for similarity scores [RFGS09, Ren10, SM07, RFGS09, HZKC16, Kor08, HLZ+17, KNK13, HHS+18, KBV09]. Some early works can be considered as shallow models where users and items are encoded by latent factors and the inner product is used to encode user-item interactions as user-item affinity scores. In fact, these works are limited in modeling complex user-item relationships due to the linear nature of inner product operation. As a result, these proposed models are still sub-optimal as dot product is not a metric learning and does not convey the crucial triangle inequality property. Notably, some works have addressed this issue using neural network architectures. For instance, He *et al.* [HLZ+17] proposed a neural network based model that combined a generalized matrix factorization via component and a non-linear user-item modeling component via an MLP architecture. Some works substituted the MLP architecture with the auto-encoder architecture [LKF15, LKHJ18, WDZE16].

On the other hand, many approaches in personalized recommender system based on the distance/similarity metric between two points using Euclidean distance have shown their strong competency in improving the model accuracy in different domains [WBS05, WDWK11, CHL05, KTW+12, XNJR02, TLLK19, TSL19]. To this end, [HYC+17] argued that using inner product formulation lacks expressiveness due to its violation of the triangle inequality. As a result, the authors proposed a Collaborative Metric Learning (CML) method, a strong recommendation baseline where user-item interactions are encoded using Euclidean distance. On a side note, there also exists other works that adopted Euclidean distance to model user-item affinity and item-item transitions in separated spaces [FLZ+15], or item-item transitions where users play as translators [HKM18].

2.3.3 Hyperbolic Representation

Hyperbolic representation learning has recently demonstrated great potential across a diverse range of applications [NK17, CDPB19, NK18, GBH18a, SSGR18, DFC+18, TBG19, GSGR19, LLSZ19]. For instance, [TTH18] proposed training a question answering system in hyperbolic space. [DSN+18] proposed learning word embeddings using a hyperbolic neural network. [GDM+19] proposed a hyperbolic variation of self-attention and the transformer network, and applied it to tasks such as visual question answering and neural machine translation. [GBH18b] proposed recurrent neural networks in hyperbolic space, [CCD17] proposed a method of embedding graphs in hyperbolic space. In the

scope of recommender systems, [CHW+19] is the most similar work to ours that embeds bipartite user-item graphs in hyperbolic space, but it does not learn the embeddings with metric learning manner. While the advantages of hyperbolic space seem eminent in the wide variety of application domains, there is no work that investigates this embedding space within the context of metric learning in recommender systems. Thus, hyperbolic representation is a promising direction that could possibly be explored in the future. In this thesis, we propose one advanced neural architecture that adopts hyperbolic representation learning for recommender systems by performing metric learning in hyperbolic space.

2.4 Representation Learning for Group Recommendation

In the previous reviews, most of the introduced representation learning techniques mainly focus on personalized recommendation. Those recommendation methods targeting individuals, however, cannot be efficiently applied in group recommendation task. In this section, we first review tranditional approaches for group recommendation. Then, we review recent advanced techniques such as probabilistic models and deep learning for group recommender systems. In this dissertation, we also introduce a new deep representation learning technique for group recommendation, in which we utilize the powerful of deep learning to model the group decision making process. We will present the details of the proposed method in Chapter 7.

Conventional Memory-based and Model-based Approaches. Group recommendation methods can be characteristically dichotomized into memory-based and model-based approaches, where the memory-based approach can be further divided into the preference aggregation and the score aggregation [ARC+09]. The preference aggregation makes recommendations based on a group profile that combines all user preferences [YZHG06], while the score aggregation computes a score of an item for each user, and then aggregates the scores across users to derive a group recommendation score of the item [BMR10, OCKR01]. The two most popular strategies for score aggregation are the average (AVG) and the least misery (LM) strategies. The AVG strategy takes the average score across individuals in the group as the final recommendation score, thereby maximizing overall group satisfaction [YZHG06]. Alternatively, the LM strategy pleases everyone by choosing the lowest among all individuals' scores as the final score [BMR10]. Both score aggregation methods have major drawbacks. The AVG strategy may return items that are favorable to some members but not to the others, while the LM strategy may end up recommending mediocre items that no one either loves or hates. [BMR10] pointed

out that the performance of either strategy depends on group size and inner-group similarity. [ARC+09] proposed the concepts of relevance and disagreement. Arguing that preference disagreements on each item among group members are inevitable, the authors experimentally show that taking into account disagreement significantly improves the recommendation quality of AVG and LM strategies.

Model-based approaches [KBV09, WB11, SK09, AC10] for group recommendation are also notable. [SYMM11] proposed a model that incorporates item categories into recommendation, arguing that item categories influence the group's decision and items of different categories are not strictly comparable. The method, however, only applies to pre-defined groups such as couples, which can be treated as pseudo-users and apply single user recommendation techniques, while real-life groups are often ad-hoc and formed just for one-off or few activities [LTYL12, QRT16]. Applying game theory in group recommendation, [CM13] considered each group event as a non-cooperative game, or a game with competition among members in the group, and suggested that the recommendation goal should be the game's Nash equilibrium. However, since a Nash equilibrium can be a set of items, the game theory approach may fail to recommend one specific item.

Probabilistic Approaches. Probabilistic models have also been applied to solve group recommendation. [LTYL12] proposed a personal impact topic (PIT) model for group recommendation, assuming that the most influential user should represent the group and have big impact on the group's decisions. However, such an assumption does not reflect the reality that a user's influence only contributes to the group's final decision if he/she is an expert in the field. [YCL14] proposed a consensus model (COM) for group recommendation. The model assumes (i) that a user's influence depends on the topic of decision, and (ii) that the group decision making process is subject to both the topic of the group's preferences and each user's personal preferences. Despite such assumptions, COM suffers from a drawback similar to that of PIT: COM assumes that a user has the same probability to follow the group's decisions across different groups. Additionally, [GLRW13] assumed that the score of a candidate item depends not only on its relevance to each member in a group but also its relevance to the whole group. They develop an information-matching based model for group recommendation, but the model suffers from high time complexity, taking several days to run on several datasets as reported by [YCL14]. Owing to its computational prohibitivity, we do not compare with the method in our experiments.

Neural Approaches. Recently, [HCX+14] proposed a deep-architecture model called DLGR that learns high-level comprehensive features of group preferences to avoid the vulnerability of the data. Similar to the work [SYMM11], DLGR only focuses on pre-defined groups instead of ad-hoc groups, and thus cannot be applied to our setting.

Additionally, a recent work exploits neural attention for a group recommendation setup called AGREE [CHM+18]. However, AGREE also focuses on pre-defined groups where it requires additional group preference representation information as one component to learn the final representation of a group.

Chapter 3

Wasserstein based Metric Learning Representation for Recommendation

3.1 Introduction

Metric learning has attracted extensive research in the past few years across different domains [WDWK11, CHL05, XNJR02, KTW+12, CHL05, HYC+17, TATH18, WBS05, LL16]. The basic idea of metric learning is to produce a distance metric between two objects to represent for their similarity. The two objects are more similar if their distance is small and vice versa. In recommender systems, metric learning is adopted as a matching function between users and items. Recently, it has been widely utilized in recommendation tasks due to its crucial triangle inequality property, which is usually ignored in inner product based recommenders [RFGS09, HLZ+17, HDW+18].

Particularly, assume a user u interacted with two *similar* items i_1, i_2. Learning with dot product can lead to the following 2-dimensional results: u=(1, 1); i_1=(1, 0); i_2=(0, 1) because $u^T i_1$=1, and $u^T i_2$=1. However, the dot product between i_1 and i_2 is $i_1^T i_2$=0, indicating that i_1 and i_2 are not similar. In contrast, by learning with metric learning, the distance $d(u, i_1)$ between user u and item i_1 and the distance $d(u, i_2)$ between user u and item i_2 are smaller. As an effect of the triangle inequality, the distance $d(i_1, i_2)$ between two items i_1 and i_2 satisfies $d(i_1, i_2) < d(u, i_1) + d(u, i_2)$. As $d(u, i_1)$ and $d(u, i_2)$ are small, $d(i_1, i_2)$ is small as well, showing that i_1 and i_2 are correctly portrayed as similar. In a same manner, when two similar users prefer a same item, as an effect of the triangle inequality, the distance between the two similar users is small as well.

To this end, several metric learning-based recommender systems have been proposed to preserve the triangle inequality property [LL16, TATH18, HYC+17, KS10, CMTJ12, PKXY18]. However, most of existing metric learning methods represent each user and item embedding by a single shallow level of pull and push mechanism [HYC+17, CMTJ12, TSL19]. In this way, the formation of the pull and push mechanism indeed could be

considered as deterministic as each user and item is presented as only a single point in a specific low-dimensional vector space.

For the first time, we propose a *Wasserstein distance-based Metric Learning Chain* (*W-MLC*) model, which utilizes the notion of metric learning chain, and allows a series of metric learning computations between user and item latent representations. Instead of learning user and item in a standard metric learning approach with one shallow layer, *W-MLC* dives into deeper levels with multiple pairwise objectives of metric learning. There are two novel aspects in *W-MLC*: (i) restriction and (ii) relaxation. Specifically, we project user/item embeddings into different spaces sequentially (i.e. creating a chain) to perform metric learning on multiple levels (restriction), whereas we introduce an adaptive margin for each user at different chain (relaxation). For a clear explanation, after obtaining the first embeddings of users and items as usual, we project these embeddings to another space with smaller latent dimension. This process creates a chain of length 1 with one projection. Therefore, we could form a chain of length c if we apply the same process continuously c times. Different from standard metric learning methods, the metric learning chain allows us to encode richer semantic relations and provides more freedom to cluster similar items together and keep dissimilar items far apart. To this end, we preserve the pull and push mechanism of metric learning at different levels, while simultaneously enhancing the performance and reducing biases through our personalized and adaptive margin hinge loss.

Moreover, the key challenge is to effectively preserve the semantic embeddings and the triangle inequality when we execute multiple projections continuously to perform metric learning. Thus, we introduce the Wasserstein distance [GS84, ZCWZ18, PC19a] to maintain good user/item latent representations when controlling the projection matrices, which are usually generated based on prior distributions. Specifically, we employ a 2-Wasserstein distance to measure the similarity between the prior distributions due to two reasons: 1) the 2-Wasserstein distance is a real metric that satisfies the triangle inequality property (i.e., suitable for metric learning); and 2) exploring a single projection matrix for both user and item could possibly lead to stricter constraint, resulting in fitting user-item pair into one single point in vector space [TATH18]. Note that literature often uses Kullback-Leibler (KL) divergence to account for difference between two distributions [LKHJ18]. We argue that the KL divergence is asymmetric and does not satisfy the triangle inequality [KL51]. Hence, we do not employ the KL divergence in our work.

Intuitively, our proposed architecture is considered as an advanced metric learning method compared to previous approaches. In fact, we observe that existing standard metric learning approaches often only present each user and item as a single point in a specific low-dimensional vector space. Thus, these methods are considered as deterministic. To overcome this limitation, the metric learning chain is proposed to provide the flexibility in learning representations, while maintaining the strong pull and push

mechanism of conventional metric learning. To this end, we project user and item embeddings into different spaces to perform metric learning on multiple levels. With that in mind, we need to maintain good user and item latent representations when we execute multiple projections continuously. Therefore, we control the projection matrices based on the fact that they are usually generated with prior distributions/initializations. In addition, personalized adaptive margin is also introduced to adapt to the transformation of the representations.

To this end, we propose a multi-objective loss function that composes of three components. The first component is the pull and push loss function with the *personalized* safety margin, which performs the pull and push mechanism as introduced in [HYC+17, LZZ+20, CMTJ12]; the second component is the adaptive margin loss, which acts as the penalty term to maximize the personalized margin for each user; and the third component is the transformation loss, which controls the transformed embeddings from a space to another. In general, we incorporate the multi-objective loss function into an end-to-end *W-MLC* model. We show that our proposed *W-MLC* model outperforms various recent strong metric learning baselines, indicating the effectiveness and the need of our *deep* metric learning-based model.

The key contributions of our work are summarized as follows:

- We propose a *Wasserstein distance-based Metric Learning Chain* (W-MLC) model that investigates the notion of metric learning chain for recommender systems via Wasserstein distance. To the best of our knowledge, this is the first work that explores a series of metric learning in different transformation spaces with the combination of Wasserstein distance in recommendation domain.

- We propose a multi-objective loss function that consists of three components. Specifically, unlike standard metric learning models, we learn the adaptive and personalized margin for each user under the pull and push mechanism. Moreover, we introduce the similarity measure between transformation matrices via Wasserstein distance, in which the 2-Wasserstein distance also satisfies the crucial triangle inequality property for metric learning.

- We conduct extensive experiments on eight widely-used benchmark datasets in three well-known recommendation tasks to demonstrate the effectiveness of our proposed model against eight state-of-the-art baseline recommendation methods.

3.2 Preliminaries

In this work, we denote $\mathcal{X}$ and $\mathcal{Y}$ as two bounded subsets of $\mathbb{R}^d$, and $\mathcal{M}_+^1(\mathcal{X})$ as a set of probability distributions with positive Radon measures of unit mass on $\mathcal{X}$. We

also denote the upper cases X, Y as the random variables in these spaces. We use these notations to describe the preliminaries on the optimal transport, Wasserstein distance and Sinkhorn divergences as follows:

3.2.1 Optimal Transport

We consider genetic optimal transport (OT) metrics on general spaces $\mathcal{X}$. Specifically, we consider two probability measures $\alpha \in \mathcal{M}_+^1(\mathcal{X})$ and $\beta \in \mathcal{M}_+^1(\mathcal{Y})$. Following [GCB+19, GPC18], the Kantorovich formulation of OT between α and β is defined as:

$$W_c(\alpha, \beta) \overset{def.}{=} \min_{\pi \in \Pi(\alpha,\beta)} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) d\pi(x, y), \tag{3.1}$$

in which $\Pi(\cdot, \cdot)$ is composed of probability distributions over the product space $\mathcal{X} \times \mathcal{Y}$ with the fixed marginals α and β:

$$\Pi(\alpha, \beta) \overset{def.}{=} \{\pi \in \mathcal{M}_+^1(\mathcal{X} \times \mathcal{Y}); P_{1\sharp}\pi = \alpha, P_{2\sharp}\pi = \beta\}, \tag{3.2}$$

Here, $P_{1\sharp}\pi$ and $P_{2\sharp}\pi$ are the marginal distribution of π for the first and second variable, respectively; $P_1(x, y) = x$ and $P_2(x, y) = y$ are the simple projection maps; and $\sharp$ is the push-forward operator.

In Eqn. (3.1), we present $c(x, y)$ as the cost function, which is also known as *ground cost* or *ground metric*. $c(x, y)$ represents the cost of moving one unit mass from x to y. When $\mathcal{X} = \mathcal{Y}$ and $\mathcal{X}$ is equipped with a distance $d_{\mathcal{X}}$, choosing $c(x, y) = d_{\mathcal{X}}(x, y)^p$ where $p \geq 1$ introduces the p-Wasserstein distance between the two probability measures. Notably, other ground distances (i.e. cost functions) such as Euclidean, cosine or sparse L1-norm distance can also be chosen depended on different tasks.

3.2.2 Wasserstein Distance

The p-Wasserstein distance between probability distributions α and β over the metric space $\mathcal{X}$ is defined as:

$$\begin{aligned} W_{c,p}(\alpha, \beta) &\overset{def.}{=} \left(\min_{\pi \in \Pi(\alpha,\beta)} \int_{\mathcal{X}\times\mathcal{X}} c(x_1, x_2) d\pi(x_1, x_2) \right)^{\frac{1}{p}} \\ &= \left(\min_{\pi \in \Pi(\alpha,\beta)} \int_{\mathcal{X}\times\mathcal{X}} d_{\mathcal{X}}(x_1, x_2)^p d\pi(x_1, x_2) \right)^{\frac{1}{p}}, \end{aligned} \tag{3.3}$$

where $W_{c,p}(\alpha, \beta) = W_c(\alpha, \beta)^{\frac{1}{p}}$. In this work, we consider the regularized optimal transport problem by exploring an additional entropic regularization as introduced in [Cut13, GPC18] as follows:

$$W_c^{\epsilon}(\alpha, \beta) \stackrel{def.}{=} \min_{\pi \in \Pi(\alpha, \beta)} \int_{\mathcal{X} \times \mathcal{Y}} c(x, y) d\pi(x, y) + \epsilon H(\pi | \alpha \otimes \beta), \tag{3.4}$$

where $H(\cdot)$ is the relative entropy with respect to the product measure $\alpha \otimes \beta$, which is defined as:

$$H(\pi | \alpha \otimes \beta) \stackrel{def.}{=} \int_{\mathcal{X} \times \mathcal{Y}} \log \left(\frac{d\pi(x, y)}{d\alpha(x) d\beta(y)} \right) d\pi(x, y), \tag{3.5}$$

All in all, the regularized p-Wasserstein distance associated with cost c and regularization ϵ is then denoted as $W_{c,p}^{\epsilon}(\alpha, \beta)$.

3.2.3 Sinkhorn Divergences

To ensure $W_c^{\epsilon}(\alpha, \alpha) \neq 0$, [GPC18, GCB+19] further propose Sinkhorn divergences. The Sinkhorn loss between two measures is defined as:

$$\bar{W}_c^{\epsilon}(\alpha, \beta) = 2W_c^{\epsilon}(\alpha, \beta) - W_c^{\epsilon}(\alpha, \alpha) - W_c^{\epsilon}(\beta, \beta). \tag{3.6}$$

The Eqn. (3.6) ensures that $\bar{W}_c^{\epsilon}(\alpha, \beta) \neq 0$. Moreover, we observe two points that: (i) $\bar{W}_c^{\epsilon}(\alpha, \beta) \to 2W_c^{\epsilon}(\alpha, \beta)$ as $\epsilon \to 0$, which recover the unregularized OT problem; and (ii) $\bar{W}_c^{\epsilon}(\alpha, \beta) \to MMD_k(\alpha, \beta)$ as $\epsilon \to +\infty$, in which the associated kernel k is set to $-c$ and MMD is the Maximum Mean Discrepancy [GBR+06].

Notably, as computing $W_{c,p}$ exactly is expensive, [Cut13, GPC18] propose a Sinkhorn algorithm to compute Sinkhorn divergences faster and easier in terms of implementation. The Sinkhorn algorithm is also known as to provide efficient computation by a fixed-point iteration, and suitable for gradient-based optimization [GPC18, FMS19]. Indeed, the Sinkhorn algorithm can be developed efficiently on parallel architectures [Cut13]. As a result, the Sinkhorn algorithm is capable of incorporating into an end-to-end training of a neural framework.

3.3 Metric Learning Chain

In this section, we first introduce the background of metric learning in collaborative filtering. Then, we describe our Wasserstein distance based Metric Learning Chain (W-MLC) method as follows:

3.3.1 Background

3.3.1.1 Metric Learning for k-NN

The objective of metric learning is to learn a distance metric that pulls all similar user-item pairs together while pushing dissimilar user-item pairs away, which is called as the pull and push mechanism [HYC+17, WBS05]. Specifically, to exploit the learned metric for k-NN classification, [WBS05] shows that it is sufficient to just learn a metric that makes all the class labels of the k-NN similar to the class of the considered object. As a result, [WBS05] defines a pull loss that reflects the distances between similar objects as:

$$\mathcal{L}_{pull} = \sum_{j \to i} d(x_i, x_j)^2, \tag{3.7}$$

where $j \to i$ shows that the object j is closely connected to object i.

In addition, [WBS05] also defines another push loss to push the dissimilar objects away to maintain a safety margin m around the k-NN decision boundaries:

$$\mathcal{L}_{push} = \sum_{i,j \to j} \sum_{k} (1 - y_{ik})[1 + d(x_i, x_j)^2 - d(x_i, x_k)^2]_+, \tag{3.8}$$

in which $[z]_+ = max(0, z)$ is the hinge loss function; and $(1 - y_{ik})$ is the control term with:

$$y_{ik} = \begin{cases} 1, & \text{if } i \text{ and } k \text{ have the same class} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Thus, the overall objective function is the weighted combination of $\mathcal{L}_{pull}$ and $\mathcal{L}_{push}$.

3.3.1.2 Collaborative Metric Learning

The recent Collaborative Metric Learning [HYC+17] (CML) method adopted Euclidean distance to model user-item interactions for the implicit feedback recommendation task. Specifically, the pull and push losses of CML pull positive items closer to a given user while pushing negative items apart from that user. Moreover, due to the triangle inequality, CML also creates the effect of clustering: (i) the users that prefer the same items, and (ii) the items that are preferred by the same users [HYC+17]. CML aims to minimize the following loss function:

$$\mathcal{L}_m = \sum_{(i,j) \in \mathcal{S}} \sum_{(i,k) \notin \mathcal{S}} w_{ij}[m + d(i, j)^2 - d(i, k)^2]_+, \tag{3.10}$$

where j is a positive/interacted item and k is the negative/non-interacted item of user i; w_{ij} is the weighted ranking loss; and m is the safety margin.

The Euclidean distance between users and items in CML satisfies the triangle inequality. However, as mentioned in [TATH18], CML tends to fit a pair of a user and his/her positive item into a single point in the vector space, which is sub-optimal. Hence, in this work, we not only consider the Euclidean distance between objects, but also the Wasserstein distance between distributions to measure the similarity between transformation matrices (see details in Section 3.3.3), which is served as a relaxation mechanism. With this measurement, we are allowed to perform the series of metric learning in our framework after multiple projections. We then introduce the similarity measure between two distributions in the next section.

3.3.2 Similarity Measure

To examine the distance between two distributions, we prefer to explore the Wasserstein distance. The choice of Wasserstein distance is due to two reasons: 1) effectively measuring the similarity between two distributions, and 2) preserving the triangle inequality property. Thus, it is intuitive to explore Wasserstein distance for metric learning chain, where the chain not only requires to compute similarity measurement between distributions of projection matrices, but also satisfies triangle inequality at the same time.

For simplicity, the p-Wasserstein distance between two probability measures α and β is denoted as:

$$W_{c,p}^{\epsilon}(\alpha, \beta) = \inf \mathbb{E}[d(X, Y)^p], \tag{3.11}$$

where $\mathbb{E}[Z]$ is the expected value of random variable Z. For convenience, we use $W_p(\alpha, \beta)$ to replace $W_{c,p}^{\epsilon}(\alpha, \beta)$ from now on.

We note that [AGS05] proves that for all $p \geq 1$, the p-Wasserstein distance preserves all the properties of a metric including the symmetry and triangle inequality property, which explains our appropriate choice of exploring the Wasserstein distance.

Following previous works [GS84, ZCWZ18, PC19a], we use the 2-Wasserstein distance, i.e., $W_p(\alpha, \beta)$, to speed up the calculation process. For instance, in the case of minimizing the 2-Wasserstein distance between two Gaussian distributions, we would get:

$$\begin{aligned} dist^2 &= W_2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2))^2 \\ &= \|\mu_1 - \mu_2\|_2^2 + \mathrm{TR}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}}))^{\frac{1}{2}}. \end{aligned} \tag{3.12}$$

Notably, we find that the Wasserstein distance is able to measure the similarity between two distributions while simultaneously satisfying the triangle inequality. Therefore, this explains our appropriate choice of exploring Wasserstein distance in our proposed model.

3.3.3 Our Model Formulation

The input of our *Wasserstein distance-based Metric Learning Chain* (W-MLC) model includes a user i with an embedding $\mathbf{u}_i$, a positive item j with an embedding $\mathbf{v}_j$, and a negative item k with an embedding $\mathbf{v}_k$. Generally, we learn their Euclidean distance between a user i and an item j as follows:

$$d(i,j) = \|\mathbf{u}_i - \mathbf{v}_j\|_2^2, \tag{3.13}$$

where the learned distance represents the preference of user i towards item j, in which the items that the user liked will be embedded closer than the ones that she did not like in the latent space. Thus, it is expected that the positive item j is closer to user i than the negative item k. Specifically, our learned metric pulls the positive user-item pairs closer and pushes the negative items away from the user. This pull and push mechanism in metric learning also provides the clustering effect where the users that liked the same items and the items that are liked by the same users will be likely clustered together [HYC+17].

Moreover, we proposes a metric learning chain by simultaneously performing pairwise hinge losses on different transformed spaces. Specifically, we explore the normal pull and push mechanism for all the transformed user and item embeddings, we further introduce another loss for the transformation matrices.

Figure 3.1 summarizes our proposed metric learning chain for recommendation. We provide the pairwise input of user, positive item, and negative item. Given the first k-dimensional user and item embeddings, we continuously transfer them into other spaces using randomly generated matrices. For each transformed embedding and matrix, we perform pairwise objective training by introducing multiple appropriate loss functions. We describe them as follows:

Pull and Push Loss. Firstly, we define our pull and push mechanism, i.e., loss function as:

$$\mathcal{L}_P = \sum_{(i,j)\in\mathbb{S}} \sum_{(i,k)\notin\mathbb{S}} [m_i + d(i,j)^2 - d(i,k)^2]_+, \tag{3.14}$$

where j is a positive item and k is a negative item; $\mathbb{S}$ contains all the positive user-item pairs, i.e., the observed implicit feedback; $[z]_+ = \max(0, z)$ is the standard hinge loss; and m_i is the safety margin of user i. We note that there are two important differences of our proposed loss function as compared to [HYC+17]: 1) our safety margin m_i is automatically learned and customized for each user instead of using a fixed and pre-defined value for all users in [HYC+17], and 2) we do not apply the ranking loss weight w_{ij}. Thus, we are able to provide an adaptive margin for individual user based on his personalized preference.

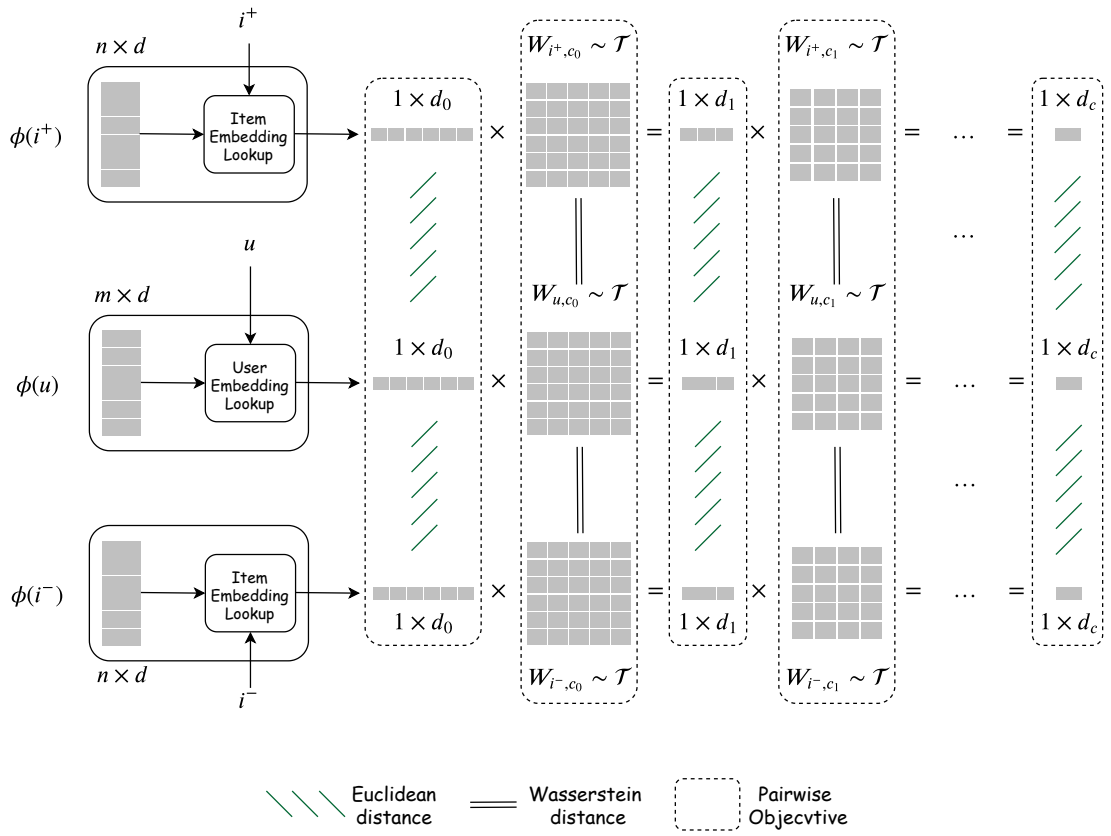


Figure 3.1: Illustration of our proposed metric learning chain. W-MLC is trained using a chain of multiple pairwise distances with metric learning. In particular, the component $\Phi(u)$, $\Phi(i^+)$, $\Phi(i^-)$ learn the (first) k-dimensional latent representation for user u, positive item i^+, and negative item i^- (i.e., the vectors with dimension of $1 \times d_0$), respectively. Then, those learnt representations are transformed into subsequent representations (i.e., the $1 \times d_*$ vectors) through transformation matrices (i.e., W_{*,c_i}). For each triplet, we perform pairwise objective training (i.e., the dashed rounded rectangles) by introducing multiple appropriate loss functions with *distance*, in which the distance is either Euclidean (for latent representations) or Wasserstein (for transformation matrices).

Adaptive Margin Loss. Since we use the adaptive margin for each user, we also provide another margin loss to reduce the variance as:

$$\mathcal{L}_M = -\frac{1}{|U|}\sum_i m_i, \tag{3.15}$$

where $|U|$ denotes the total number of user. Indeed, Eqn. (3.15) acts as a penalty term when we learn an adaptive margin for each user.

Transformation Loss. When performing the transformation, it is essential that the transformation preserves the distances of the converted embeddings in the new latent space. To this end, we introduce multiple variants of loss for preservation. Notably, the transformation matrices are randomly generated following given distributions. Our transformation loss function is then defined as:

$$\mathcal{L}_T = \sum_{(i,j)\in\mathbb{S}} \sum_{(i,k)\notin\mathbb{S}} (E_{ij}^2 + exp(-E_{ik})), \tag{3.16}$$

where E_{ij} is the energy between two transformation matrices i (of user i) and j (of item j); and $exp(z) = e^z$. Here, we denote $E_{ij} = W_2(M_i, M_j)$ with M_i is the transformation matrix of user i. Intuitively, our transformation loss penalizes the ranking errors with the energy-based approach. Specifically, we expect the energy of the positive user-item pairs will be larger than the negative ones as similar to [LCH+06, ZCWZ18].

On a side note, if we explore Sinkhorn divergence for approximation instead of Wasserstein distance, we derive the second variant of our transformation loss with the energy of $E_{ij} = S(M_i, M_j)$. In addition, this loss function could be turned off if we consider each triplet has the same transformation (i.e., sharing the same matrix M). Thus, we consider it as the *no-energy* version. Moreover, we consider the direct *distance* between two matrices using Frobenius norm as $E_{ij} = d_2(M_i, M_j) = \sqrt{\sum_p \sum_q (m_{i,pq} - m_{j,pq})^2}$, in which we simply name it as our Euclidean version. Therefore, we derive the four variants of our proposed model, in which they are in turn denoted as N- (No), E- (Eucldean), W- (Wasserstein), and S- (Sinkhorn).

Multi-Objective Learning. We integrate all the proposed loss functions into a unified end-to-end multi-objective learning framework, i.e., the pull and push loss $\mathcal{L}_P$, the adaptive margin loss $\mathcal{L}_M$ and the transformation loss $\mathcal{L}_T$. The overall objective function is defined as:

$$\min_{\Theta} \mathcal{L} = \sum_{i=1}^{c} (\mathcal{L}_{P,i} + \gamma_i \mathcal{L}_{M,i} + \eta_i \mathcal{L}_{T,i}), \tag{3.17}$$

where Θ denotes all parameters of our model; c is the number of chain; γ_i and η_i are the adaptive margin learning weight and the transformation learning weight at the chain i, respectively.

Notably, γ and η play important roles in controlling the balance trade-off of different parts. Indeed, we also examine the performance of our proposed model by observing different adaptive margin learning weight γ and transformation learning weight η in our experiment in Section 3.4.

Dataset	# users/ # baskets/ # playlists	# items	# interactions	Density (%)
General Recommendation Task (RT1)				
MovieLens 100k	943	1,682	100,000	6.31%
MovieLens 1M	6,040	3,706	1,000,209	4.47%
Epinions	23,253	137,289	631,189	0.02%
Yelp	22,040	18,447	1,043,823	0.26%
Next-Item Shopping Basket Recommendation Task (RT2)				
Tafeng	56,410	22,291	523,653	0.042%
Tmall	51,891	42,061	383,780	0.018%
Automatic Playlist Continuation Recommendation Task (RT3)				
30Music	32,140	258,388	648,534	0.008%
8Tracks	25,309	92,915	219,442	0.009%

Table 3.1: Statistics of all datasets used in our experimental evaluation for three different Research Tasks RT1, RT2, and RT3.

3.4 Experiments

To show the effectiveness of our proposed models, we evaluate the performance of our metric learning chain on three different well-known recommendation tasks (**RT**) as follows:

- **RT1: General Recommendation**. It aims to recommend items to a user based on all his/her previous interactions.
- **RT2: Next-Item Shopping Basket Recommendation**. The goal of the task is to recommend products for a user to purchase next based on the items in the current basket of this user.
- **RT3: Automatic Playlist Continuation Recommendation**. It aims to recommend next songs for a user playlist based on the existing songs in the playlist.

RT1, RT2 and **RT3** are well described in the literatures [HLZ+17, HKBT16, TSL19]. Thus, we omit their description due to space limitation. Next, we describe the datasets we use in each recommendation task as follows:

3.4.1 Datasets

For the first research task **RT1**, we use four widely adopted datasets as follows:

- **MovieLens**: A widely adopted benchmark dataset in the application domain of recommending movies to users provided by GroupLens research[1]. We use two config-

[1]https://grouplens.org/datasets/movielens/

urations, namely MovieLens100k and MovieLens1M. The two datasets are already filtered with 20-core setting.

- **Epinions**: A consumer review dataset that was crawled on Epinions website and originally introduced in [MA07].[2]
- **Yelp**: A business directory and crowd-sourced review platform. We adopt the dataset from the Yelp Dataset Challenge 2018.[3]

For the second research task **RT2**, we exploit two real-world transaction datasets as below:

- **Tafeng**[4]: A grocery transaction dataset of *T-Feng* supermarket, which contains four months of transactions from November 2000 to February 2001.
- **Tmall**[5]: It contains shopping logs of users from *Tmall* retail platform. In this work, we randomly pick 100k transactions for evaluation since the original dataset is extremely large.

For the third research task **RT3**, we exploit two real-world music playlist datasets as below:

- **30Music**: A collection of playlists retrieved from *Last.fm*[6], which was originally introduced in [TQC+15].
- **8Tracks**: It is a collection of playlists extracted from *8tracks* platform[7], and is available for the research purpose[8].

Preprocessing Preparation. For preprocessing, we remove duplicated items in all the dataset. We adopt the popular *k-core* preprocessing step [HLZ+17, HM16a] (with *k-core*=5), filtering inactive users/baskets/playlists with less than 5 interactions. Since we focus on implicit feedback, we binarize the explicit data by transforming all observed ratings as positive and the remaining as negative. In both *Tmall* and *Tafeng* datasets, each user behavior is recorded with four types of actions: *click*, *add-to-favourite*, *add-to-cart*, and *purchase*. In general, we consider all these four types as the click action. The statistics of the preprocessed datasets are presented in Table 3.1.

[2]http://www.trustlet.org/downloaded_epinions.html
[3]https://www.yelp.com/dataset/challenge
[4]https://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset
[5]https://tianchi.aliyun.com/datalab/dataSet.htm?id=1
[6]https://www.last.fm
[7]https://8tracks.com/
[8]https://github.com/danielgondim/mpsd

3.4.2 Baselines

We compare our models against five well-known and **strong** general recommenders, in which we further divide them into two groups: *dot product* based recommenders (**Group 1**) (MF-BPR, NeuMF++, CMN++), and *metric learning* based recommenders (**Group 2**) (CML, LRML). We describe them as follows:

- **Matrix Factorization with Bayesian Personalized Ranking (MF-BPR)** [RFGS09]: It is a state-of-the-art matrix factorization model for implicit feedback recommendation, where the user-item interactions are modeled using the inner product.
- **Neural Collaborative Filtering (NeuMF++)** [HLZ+17]: It is a strong neural network based recommendation model which models nonlinear user-item interactions. We pre-trained two components of NeuMF to obtain its best results, denoted as NeuMF++.
- **Collaborative Memory Network (CMN++)** [ESF18]: It is a strong recommender system based on memory network. The framework exploits user neighborhood-based CF and additionally integrates a memory network to learn attentive weights for similar users. We also use a pre-trained version of CMN, in which we denote as CMN++.
- **Collaborative Metric Learning (CML)** [HYC+17]: It is a strong metric learning baseline that learns user-item similarities using the Euclidean distance. CML can be considered as a key ablative baseline in our experiments.
- **Latent Relational Metric Learning (LRML)** [TATH18]: It is also a strong metric learning baseline that learns adaptive relation vectors between user and item pairs to find its optimal translation.

In addition, recent works have shown outstanding performance of sequential recommenders over general recommenders, where historical items are modeled in an ascending order by their interacting timestamps [TW18, HKM18]. In this work, although we do not model the consuming order of items based on user's historical purchase logs, we still compare our models with some popular sequential models, where we refer as **Group 3**. These models are introduced as follows:

- **Personalized Ranking Metric Embedding (PRME)** [FLZ+15]: It is the personalized ranking-based metric embedding approach, which models the user preferences and the *first-order* Markov sequential transitions in two separate latent Euclidean spaces.

- **Translation-based Recommendation (TransRec)** [HKM18]: It is one of the strong sequential recommendation methods. *TransRec* utilizes the Euclidean distance to model the third order relationships between the user, the previous item, and the next item.

- **Convolutional Sequence Embedding Recommendation Caser)** [TW18]: It is another strong sequential model that explores convolutional neural network (CNN) with vertical and horizontal kernels to capture the complex relationships between items.

To summarize, we compare our proposed methods with three types of baselines, i.e., dot product based recommenders, metric learning recommenders, and sequential recommenders, to demonstrate the effectiveness of our proposed architectures across various recommendation tasks. Moreover, we compare the above baselines with two main variants of our proposed metric learning chain, namely *W-MLC* and *S-MLC*. The two proposed methods are introduced as below:

- **Wasserstein Metric Learning Chain (W-MLC)**: It is our proposed method, which models the Wasserstein distance to calculate the distance between the two linked matrices.

- **Sinkhorn Metric Learning Chain (S-MLC)**: This is another version of our proposed method that uses the Sinkhorn iteration to approximate the Wasserstein distance [GPC18, FMS19].

3.4.3 Experimental Settings

Evaluation Protocol and Metrics. We adopt NDCG@10 (Normalized Discounted Cumulative Gain) and HR@10 (Hit Ratio) evaluation metrics, which are well-established ranking metrics for recommendation tasks. Following [HLZ+17, TATH18], we adopt the widely-used *leave-one-out* protocol for the model evaluation. For datasets with available interaction timestamp (i.e. *Movielens 100k*, *Movielens 1M*, *Yelp*, *Tafeng*, and *Tmall* datasets), the lastest interacted item of each user/basket/playlist is held-out as the test set, the penultimate is used for the development set, and the rest of interactions are for the training set. For datasets without interaction timestamp (i.e. *Epinions*, *30Music* and *8tracks* datasets), for each user/playlist, we randomly pick one interacted item for testing set, another interacted item for development set, and the rest of interactions for training set. Note that for non-timestamped datasets, the order of interactions are kept as similar as the original datasets to give advantages for sequential recommenders. Next,

	Model	MovieLens 100k		MovieLens 1M		Epinions		Yelp	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Group 1	MF-BPR	0.733	0.471	0.716	0.432	0.460	0.316	0.737	0.452
	NeuMF++	0.760	0.501	0.735	0.452	0.486	0.322	0.792	0.502
	CMN++	0.756	0.492	0.727	0.448	0.466	0.300	0.800	0.508
Group 2	CML	0.760	0.493	0.712	0.438	0.480	0.316	0.783	0.488
	LRML	0.763	0.482	0.734	0.456	0.472	0.308	0.776	0.507
Group 3	PRME	0.760	0.501	<u>0.834</u>	<u>0.624</u>	<u>0.556</u>	<u>0.417</u>	0.786	0.495
	TransRec	<u>0.771</u>	<u>0.505</u>	0.825	0.610	0.547	0.410	<u>0.804</u>	<u>0.509</u>
	Caser	0.671	0.398	0.827	0.596	0.416	0.267	0.712	0.414
Ours	S-MLC	**0.782**	**0.530**	**0.850**	**0.659**	**0.622**	**0.473**	**0.832**	**0.589**
	W-MLC	**0.789**	**0.548**	**0.852**	**0.686**	**0.645**	**0.496**	**0.834**	**0.598**
Compared with Group 1	Imprv. of S-MLC	+2.89%	+5.79%	+15.65%	+45.80%	+27.98%	+46.89%	+4.00%	+15.94%
	Imprv. of W-MLC	+3.82%	+9.38%	+15.92%	+51.77%	+32.72%	+54.04%	+4.25%	+17.72%
Compared with Group 2	Imprv. of S-MLC	+2.49%	+7.51%	+15.80%	+44.52%	+29.58%	+49.68%	+6.26%	+16.17%
	Imprv. of W-MLC	+3.41%	+11.16%	+16.08%	+50.44%	+34.38%	+56.96%	+6.51%	+17.95%
Compared with Group 3	Imprv. of S-MLC	+1.43%	+4.95%	+1.92%	+5.61%	+11.87%	+13.43%	+3.48%	+15.72%
	Imprv. of W-MLC	+2.33%	+8.51%	+2.16%	+9.94%	+16.01%	+18.94%	+3.73%	+17.49%

Table 3.2: RT1: Overall performance of all models on four RT1 datasets. Last six lines show the relative improvement of S-MLC and W-MLC over the best baseline in each of three baseline groups. Best performances are in *bold* and best baseline results are <u>underlined</u>.

we follow [HLZ+17, ESH15] that randomly select 100 negative items which the user has not interacted with and rank the test item among the 100 negative items. During the training, we report the testing performance of the model based on the best development performance.

Hyper-parameter Settings and Implementation Details. We implement all models in Tensorflow. All models are trained with the Adam [KB15] or AdaGrad [DHS11] optimizer with learning rates selected from $\{0.01, 0.001, 0.0001, 0.00001\}$. For NeuMF++, the number of MLP layers are chosen from $\{1, 2, 3\}$. For CMN++, the number of hops are chosen from $\{1, 2, 3, 4\}$. For Caser, the Markov order L is chosen from $\{4, 5, 6, 7, 8, 9, 10\}$. The number of negative samples for each positive instance is set to 4. The embedding size d of all models is chosen from $\{8, 16, 32, 64, 128\}$ and the batch size is tuned amongst $\{128, 256, 512\}$. In our models, the multi-objective learning weight γ and η are empirically chosen from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, and the number of chains c is selected from {1, 2, 3, 4}. For models that optimize the hinge loss with a fixed margin, the margin m is selected from $\{0.1, 0.2, 0.5\}$. Note that for our model, the margin m is learned and personalized for each user. All the embeddings and parameters are randomly initialized using the random uniform initializer $\mathcal{U}(-\alpha, \alpha)$ with $\alpha = 0.01$. For metric learning models, we empirically set $\alpha = (\frac{3\beta^2}{2d})^{\frac{1}{3}}$ with $\beta = 0.01$ so that the embedding parameters arbitrarily close to the origin of the balls. Note that one can also explore other distributions for initializations such as normal distribution $\mathcal{N}(0, \alpha)$ in this case. We also empirically

Model	Tafeng		Tmall	
	HR@10	NDCG@10	HR@10	NDCG@10
NeuMF++	0.622	0.388	0.525	0.391
LRML	0.635	0.403	0.558	0.431
PRME	0.618	0.395	0.589	0.445
TransRec	0.638	0.419	0.641	0.498
S-MLC	**0.672**	**0.467**	**0.723**	**0.515**
W-MLC	**0.699**	**0.502**	**0.781**	**0.534**
Imprv. of S-MLC	+5.33%	+11.46%	+12.79%	+3.41%
Imprv. of W-MLC	+9.56%	+19.81%	+21.84%	+7.23%

Table 3.3: Next-Item Shopping Basket Recommendation: Overall performance of the baselines and our proposed models.

Model	30Music		8Tracks	
	HR@10	NDCG@10	HR@10	NDCG@10
NeuMF++	0.490	0.306	0.379	0.229
LRML	0.492	0.337	0.471	0.310
PRME	0.411	0.258	0.441	0.280
TransRec	0.474	0.312	0.455	0.293
S-MLC	**0.533**	**0.354**	**0.511**	**0.332**
W-MLC	**0.551**	**0.400**	**0.635**	**0.420**
Imprv. of S-MLC	+8.33%	+5.04%	+8.49%	+7.10%
Imprv. of W-MLC	+11.99%	+18.69%	+34.82%	+35.48%

Table 3.4: Automatic Playlist Continuation Recommendation: Overall performance of the baselines and our proposed models.

set the dropout rate $\rho = 0.5$ to prevent overfitting. For each dataset, we repeat each experiment for 5 runs and assess the average results. All the hyper-parameters are tuned using the development set.

3.4.4 Experimental Results

For an easy reference, we group MF-BPR, NeuMF++, and CMN++ into *Group 1*, which are dot product-based general recommenders. *Group 2* includes CML and LRML, which are metric learning-based general recommenders. *Group 3* contains PRME, TransRec and Caser, which are state-of-the-art sequential models. Table 3.2 shows the overall performances of our proposals and all the baselines on the **RT1**. Due to the space limitation, we only show results of NeuMF++ (i.e. best baseline in *Group* 1), LRML (i.e.

best baseline in *Group 2*), and *PRME* and *TransRec* (i.e. two best baselines in *Group 3*) in **RT2** and **RT3**. Table 3.3 shows the performance of our proposals and *top* baselines for shopping basket-based datasets. Table 3.4 shows the performance of our proposals and *top* baselines for playlist continuation datasets. We summarize our key findings as follows:

How do our proposals perform compared to all baselines? Table 3.2 shows that our proposals outperformed all the compared baselines in task **RT1**. Particularly, on average, S-MLC improved *HR*@10 by 12.63% and *NDCG*@10 by 28.61% compared to the best baseline in *Group 1*, boosted *HR*@10 by 13.53% and *NDCG*@10 by 29.47% compared to the best baseline in *Group 2*, and enhanced *HR*@10 by 4.67% and *NDCG*@10 by 9.93% compared to the best baseline in *Group 3*. Our W-MLC model also outperformed all the compared baselines with a large margin. On average, W-MLC improved *HR*@10 by 14.18% and *NDCG*@10 by 33.23% compared to the best baseline in *Group 1*, boosted *HR*@10 by 15.09% and *NDCG*@10 by 34.13% compared to the best baseline in *Group 2*, and enhanced *HR*@10 by 6.06% and *NDCG*@10 by 13.72% compared to the best baseline in *Group 3*. The improvement of our proposals over the compared methods is significant under the Directional Wilcoxon signed-rank test (*p-value* ¡ 0.015). We note that *Group 3* baselines utilized consumed items in order to model item-item transitions and performed better than *Group 1* and *Group 2* baselines, which did not exploit this information. In this sense, our proposals are similar to *Group 1* and *Group 2*. Since our proposals much improved compared to the *Group 3* baselines, modeling item-item transitions with our proposed metric learning chain can potentially enhance our reported performances. We leave this investigation for future works.

Our results in tasks **RT2** and **RT3** are also consistent with the task **RT1**. Table 3.3 shows performance of our proposals and *top 4* baselines in two large-scale shopping basket-based datasets. On average, our S-MLC model improved 8.25% and our W-MLC model enhanced 14.61% compared to the *top 4* baselines (*p-value* ¡ 0.015). Table 3.4 presents performance of our proposals and *top 4* baselines in two large-scale playlist continuation-based datasets. On average, our S-MLC and W-MLC models improved corresponding 11.96% and 30.89% compared to the best compared method (*p-value* ¡ 0.015).

All of these results show the effectiveness of our proposals. We also observe that on average, W-MLC improves S-MLC by 2.37% over four datasets in **RT1**, 5.81% over the two datasets in **RT2** and 16.79% over the two datasets in **RT3**. This is due to the fact that W-MLC is calculated using the *exact* 2-Wasserstein distances, whereas S-MLC explores Sinkhorn divergences/distances to speed-up the calculation process. As such, we observe the trade-off between the accuracy and fast computation on W-MLC and S-MLC, as consistent to similar observations in [CD14, GPC18, PC19b].

Number of Chain c	Yelp			Tafeng		
	CML	W-MLC	Δ%	TransRec	W-MLC	Δ%
$c = 1$	0.488	0.491	0.61%	0.419	0.397	-5.25%
$c = 2$	0.488	0.552	13.11%	0.419	0.411	10.48%
$c = 3$	0.488	0.576	18.03%	0.419	0.482	29.57%
$c = 4$	0.488	0.598	22.54%	0.419	0.502	34.95%

Table 3.5: Effects of the number of chain c on Yelp and Tafeng datasets w.r.t NDCG@10.

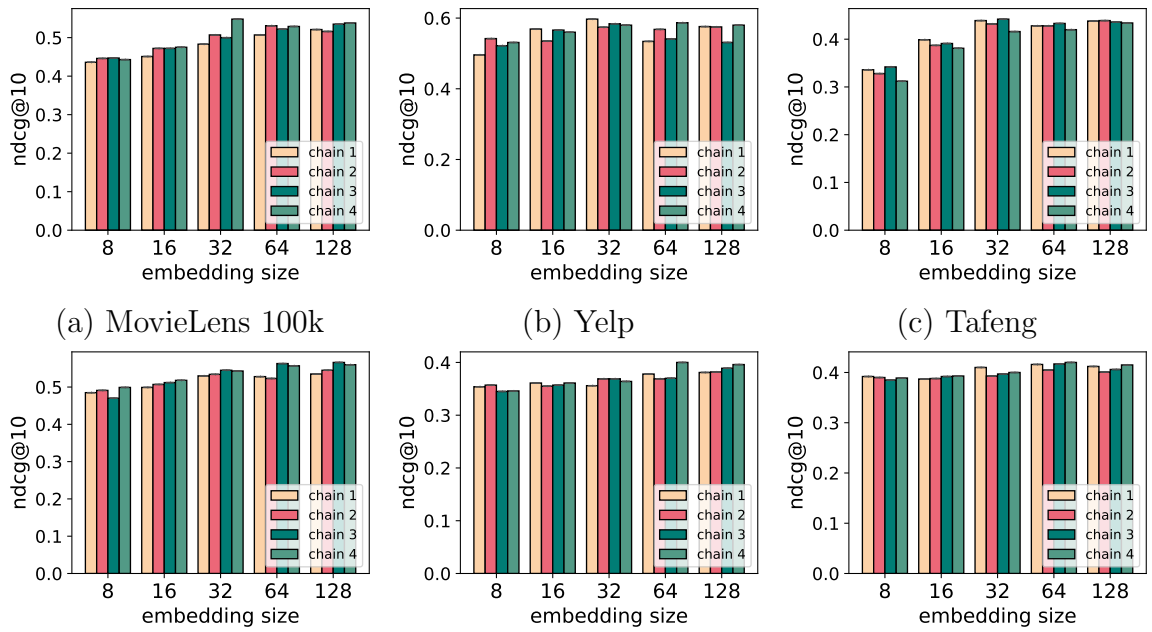
Is using metric learning chain with Wasserstein distance helpful? Note that our W-MLC model generalizes CML when the number of chains c=1 and the transformation matrices are identity matrices. In Table 3.2, our W-MLC model outperformed CML. On average, W-MLC improved CML by a large margin of 26.46%, showing the effectiveness of our metric learning chain with Wasserstein distance.

To further confirm the effectiveness of our metric learning chain proposal, we compare W-MLC when varying the number of chains c from {1, 2, 3, 4} with CML and the best baseline TransRec on Yelp and Tafeng datasets. We report *NDCG*@10 results of these models in Table 3.5. We observe that W-MLC tends to gain better performances with a larger number of metric learning chain c (i.e. going *deeper*). W-MLC achieves its best performance when $c = 4$ on Yelp and Tafeng datasets. Table 3.5 also shows that W-MLC outperformed CML with all values of c from {1, 2, 3, 4}, indicating the effectiveness and the need of the metric learning chain and the Wasserstein distance in our proposals. Additionally, W-MLC outperforms TransRec in Tafeng dataset with $c = \{2, 3, 4\}$. This again shows the effectiveness of using a deeper metric learning chain in our design.

How to select c? Figure 3.2 shows the *NDCG*@10 performance of our W-MLC model when varying the number of chains c from {1, 2, 3, 4} *w.r.t* different embedding sizes d from {8, 16, 32, 64, 128}. Overall, as c increases, W-MLC tends to gain additional improvement in all six datasets. In MovieLens 100k and Tafeng datasets, W-MLC got highest results with $c = 4$, $d = 32$ and $c = 3$, $d = 32$, respectively. In 30Music and 8tracks datasets, W-MLC achieved best results with $c = 4$ and $d = 64$. Overall, the selection of c depends on the dataset complexity and W-MLC generally performs well with $c = 3$ or $c = 4$.

3.4.5 Comparison of Multiple Variants

In this section, we compare the four variants of our proposed model: N-MLC, E-MLC, W-MLC, and S-MLC. Besides W-MLC and S-MLC, we further introduce another two variants: N-MLC (No distance: sharing the same matrix for projection) and E-MLC



(a) MovieLens 100k (b) Yelp (c) Tafeng

(d) Tmall (e) 30Music (f) 8Tracks

Figure 3.2: Comparison of varying the number of chains w.r.t different embeddings sizes in six datasets: MovieLens 100K and Yelp in the task RT1, Tafeng and Tmall in the task RT2, and 30Music and 8Tracks in the task RT3 (*Best viewed in color*).

(Euclidean distance: exploring Frobenius norm to calculate *distance* between matrices). At first, we notice that all our four variants achieve competitive results, which are reasonable due to the strength of the metric learning chain. Table 3.6 reports the performance of the variants on two datasets, MovieLens 100k and Epinions, in terms of HR@10 and NDCG@10. In general, we observe that W-MLC and S-MLC achieve remarkable performance, which demonstrates the effectiveness of our designed framework.

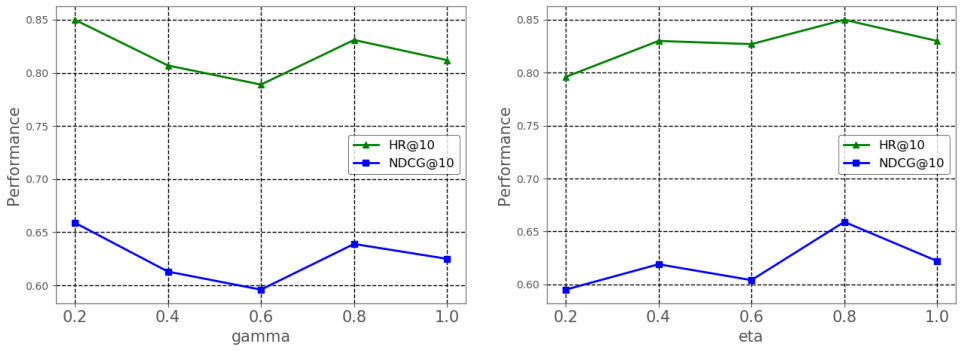
Notably, S-MLC is the approximation version of W-MLC with faster computation (1.25 times faster based on our calculation). Thus, the two models could be identified as comparable versions, in which S-MLC slightly sacrifices the recommendation accuracy for better computation and convergence. Moreover, we observe a competitive performance of the other variants: N-MLC and E-MLC. Although N-MLC and E-MLC leverage simple idea on creating the chain of metric learning, our models are still able to achieve good performance, which further explain the ability of our proposed model in boosting the performance.

Model	MovieLens 100k		Epinions	
	HR@10	NDCG@10	HR@10	NDCG@10
N-MLC	0.762	0.511	0.493	0.357
E-MLC	0.768	0.503	0.495	0.378
W-MLC	0.789	0.548	0.645	0.496
S-MLC	0.782	0.530	0.622	0.473

Table 3.6: Performance comparison between four different variants of MLC: N-MLC, E-MLC, W-MLC and S-MLC.

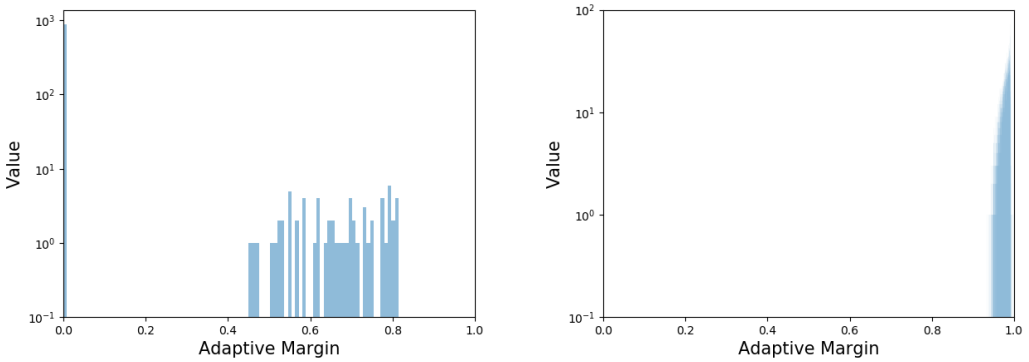
3.4.6 Different Multi-Objective Learning Weights for Accuracy Trade-off

We report the effect of different multi-objective learning weight γ and η on our proposed W-MLC with only *one* chain for illustration purpose. Due to limited space, we only report the values of γ and η on MovieLens 1M dataset. We also observe similar oscillated performance on other datasets with respect to different learning weights. In this section, we fixed the value of one learning weight and change the value of the other. Figure 3.3 represents the performance of W-MLC in terms of HR@10 and NDCG@10 when we control different values of the multi-objective learning weight γ and η. We observe that the two values are indeed dependent of each other. It is reasonable because when we relax the value of η, i.e., relax the constraint when embedding into another space, we need to control the accuracy trade-off by adaptively handling the safety margin of each user in the new space. Thus, we conclude that the multi-objective learning weight γ and η are important parameters on controlling the embedding space, while boosting the performance at the same time.

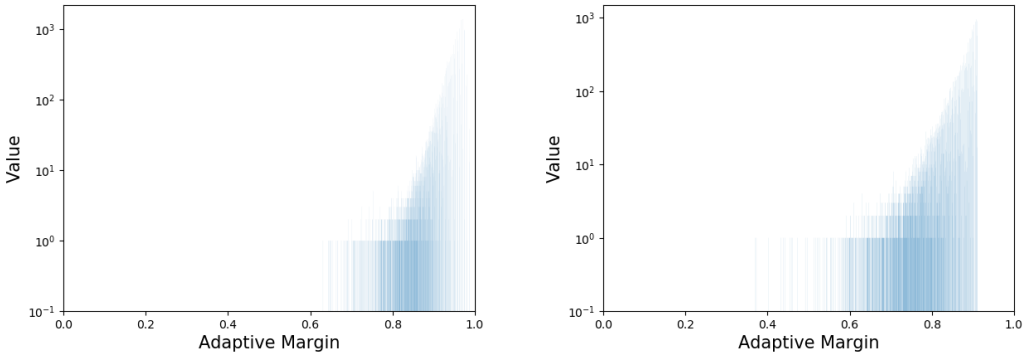


(a) Fixed γ (b) Fixed η

Figure 3.3: Performance of different Multi-Objective learning weight γ and η on MovieLens 1M dataset.



(a) MovieLens 100k (b) MovieLens 1M



(c) Epinions (d) Yelp

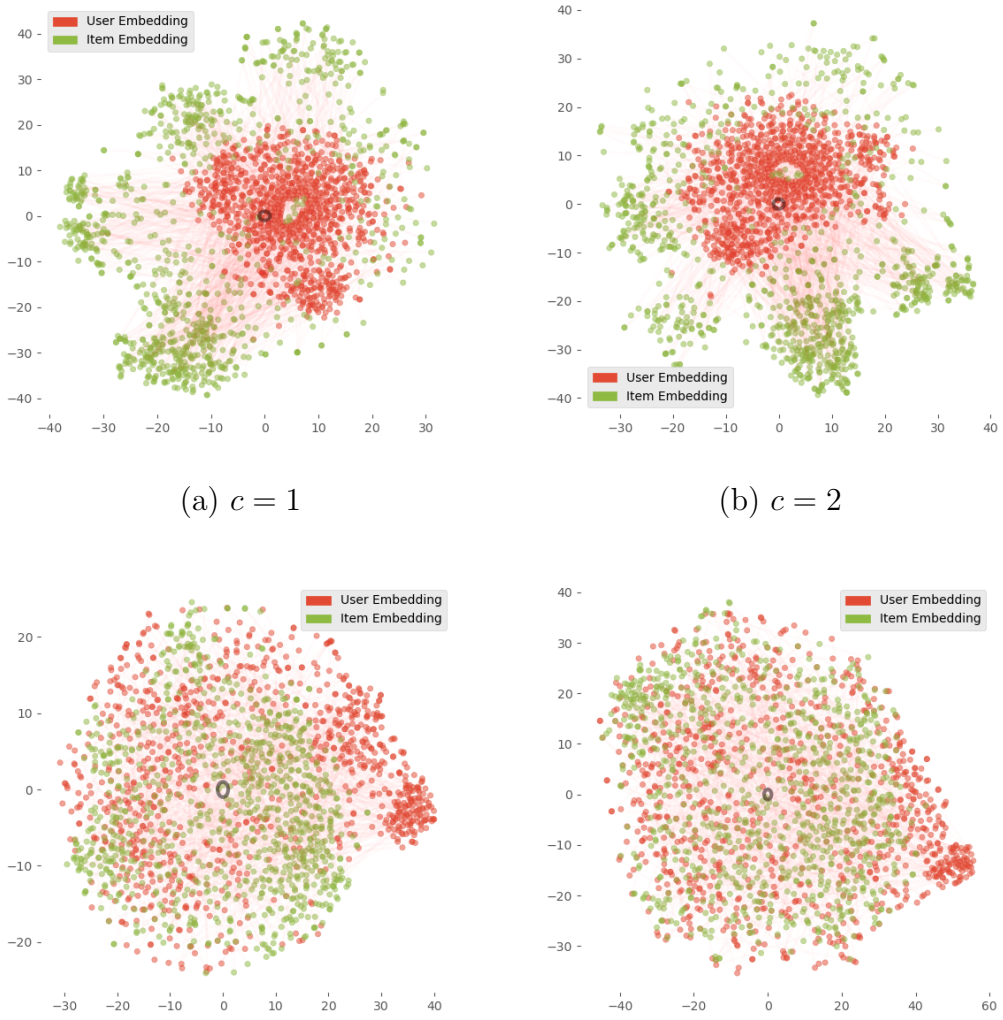
Figure 3.4: Visualization of the adaptive margins of W-MLC on MovieLens 100k, MovieLens 1M, Epinions, and Yelp dataset.

3.4.7 Insight on why our model performs better

There are two key factors leading to the superior performances of our proposals compared to the baselines: (i) our proposed adaptive margin loss, and (ii) our proposed metric learning chain. To additionally provide more insight on why our models outperformed the compared baselines, we show and analyse the *Adaptive Margin Visualization* and *Metric Learning Chain Visualization* as follows:

Adaptive Margin Visualization. To illustrate effectiveness of the automatically adaptive margin in our proposals, we visualize the adaptive margin of W-MLC on four datasets in the task **RT1** in Figure 3.4. We observe that the margins are in the range of 0.45 to 0.8 for MovieLens 100k dataset, whereas the range is from 0.4 to 0.9 for Yelp dataset. Importantly, Figure 3.4 shows that there are huge margin biases for different users on different datasets. Thus, we argue that the range of the adaptive margins could not only

provide personalization for different users, but also eliminate biases appearing in standard metric learning approaches where the safety margin is fixed as a input hyper-parameter.



(a) $c = 1$ (b) $c = 2$

(c) $c = 3$ (d) $c = 4$

Figure 3.5: Visualization of W-MLC's user and item embeddings on MovieLens 100k dataset. From left to right, user (*red*) and item (*green*) embeddings with the number of chains $c = 1, 2, 3, 4$, respectively (*Best viewed in color*).

Metric Learning Chain Visualization. We visualize the metric learning embeddings in different spaces with the number of chain $c = 4$ on MovieLens 100k dataset. Figure 3.5 presents our proposed W-MLC embeddings in four spaces at convergence. We discover that with the control over the constraints when projecting to different spaces, the metric

learning embeddings show different convergence at 4 levels. Note that at $c = 1, 2$, our chain demonstrates similar pull and push mechanism due to the triangle inequality, where the embeddings are spread around the origin of the Euclidean unit ball. However, we observe the shift in the visualization when we increase the chain to 3 and 4, where the similar items and users tend to be clustered but do not place around the origin due to the personalized adaptive margins. Thus, the visualizations support our claim that the *deep* level of metric learning with the control on the projection matrices could provide more relaxation on the constraint, while simultaneously boost the performance and inherit the pull and push mechanism of metric learning.

Chapter 4

Going Beyond Euclidean: Hyperbolic Representation for Recommendation

4.1 Introduction

A diverse plethora of machine learning models solves the personalized ranking problem in recommender systems via building matching functions [RFGS09, Ren10, SM07, HLZ+17]. Across the literature, a variety of matching functions have been traditionally adopted, such as inner product [RFGS09], metric learning [TATH18, HYC+17] and/or neural networks [HLZ+17, HDW+18]. Among those approaches, metric learning models (e.g., Collaborative Metric Learning (CML) [HYC+17] and Latent Relational Metric Learning (LRML) [TATH18]) are primarily focused on designing distance functions over objects (i.e., between users and items), demonstrating reasonable empirical success for collaborative ranking with implicit feedback. Nevertheless, those matching functions only covered the scope of Euclidean space.

For the first time, our work explores the notion of learning user-item representations in terms of metric learning in hyperbolic space, in which hyperbolic representation learning has recently demonstrated great potential across a diverse range of applications such as learning entity hierarchies [NK17] and/or natural language processing [TTH18, DSN+18]. Due to the exponentially expansion property of hyperbolic space, we discovered that metric learning with the pull-push mechanism in hyperbolic space could boost the performance significantly: moving a point to a certain distance will require a much smaller

This chapter is published as ***Lucas Vinh Tran***, *Yi Tay, Shuai Zhang, Gao Cong, Xiaoli Li. HyperML: A Boosting Metric Learning Approach for Recommender Systems. Proceedings of the 13th International Conference on Web Search and Data Mining (**WSDM 2020**), Pages 609-617, Houston, TX, USA.* [VTTZ+20].

force in hyperbolic space than in Euclidean space. To this end, in order to perform metric learning in hyperbolic space, we employ Möbius gyrovector spaces to generally formalize most common Euclidean operations such as addition, multiplication, exponential map and logarithmic map [GBH18b, Ung09].

Moreover, the ultimate goal when embedding a space into another is to preserve distances and more complex relationships. Thus, our work also introduces the definition of distortion to maintain good representations in hyperbolic space both locally and globally, while controlling the performance through the multi-objective learning framework. This reinforces the key idea of modeling user-item pairs in hyperbolic space, while maintaining the simplicity and effectiveness of the metric learning paradigm.

We show that a conceptually simple hyperbolic adaptation in terms of metric learning is capable of not only achieving very competitive results, but also outperforming recent advanced Euclidean metric learning models on multiple personalized ranking benchmarks.

Our Contributions. The key contributions of our work are summarized as follows:

- We investigate the notion of training recommender systems in hyperbolic space as opposed to Euclidean space by exploring Möbius gyrovector spaces with the Riemannian geometry of the Poincaré model. To the best of our knowledge, this is the first work that explores the use of hyperbolic space for metric learning in the recommender systems domain.

- We devise a new method HyperML (*Hyperbolic Metric Learning*), a strong competitive metric learning model for one-class collaborative filtering (i.e., personalized ranking). Unlike previous metric learning models, we incorporate a penalty term called *distortion* to control and balance between accuracy and preservation of distances.

- We conduct a series of extensive experiments delving into the inner workings of our proposed HyperML on **ten** public benchmark datasets. Our model demonstrates the effectiveness of hyperbolic geometry, outperforming not only its Euclidean counterparts but also a suite of competitive baselines. Notably, HyperML outperforms the state-of-the-art CML and LRML models, which are also metric learning models in Euclidean space across all benchmarks. We achieve a boosting performance gain over competitors, pulling ahead by up to 32.32% performance in terms of standard ranking metrics.

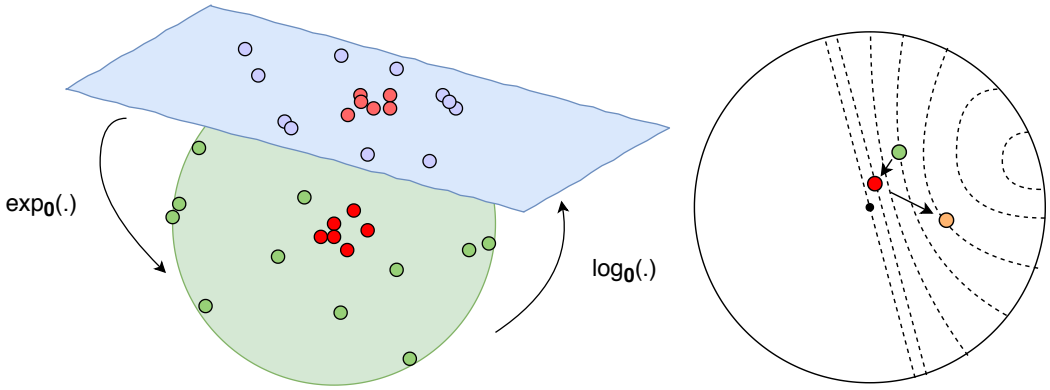


Figure 4.1: Illustration of our proposed HyperML. The *left* figure with the greenish ball represents the hyperbolic unit ball and the pale blue parallelogram illustrates its tangent space; red and vermeil circles represent user embeddings; green and purple circles represent item embeddings. The *right* figure illustrates an example of a triplet embedding of user (red circle), positive item (green circle) and negative item (orange circle), in which it demonstrates a small tree of one root and two children which is embedded into hyperbolic space with the exponentially expansion property (*Best viewed in color*).

4.2 Hyperbolic Metric Learning

This section provides the overall background and outlines the formulation of our proposed model. The key motivation behind our proposed model is to embed the two user-item pairs into hyperbolic space, creating the gradients of pulling the distance between the positive user-item pair close and pushing the negative user-item pair away.

Figure 4.1 depicts our overall proposed HyperML model. The figures illustrate our two approaches: 1) optimizing the embeddings within the unit ball and 2) transferring the points to the tangent space via the exponential and logarithmic maps for optimization. In the experiments, we also compare the mentioned variants of HyperML where both approaches achieve competitive results compared to Euclidean metric learning models.

4.2.1 Hyperbolic Geometry & Poincaré Embeddings

The hyperbolic space $\mathbb{D}$ is uniquely defined as a complete and simply connected Riemannian manifold with constant negative curvature [KPK+10]. In fact, there are three types of the Riemannian manifolds of constant curvature, which are Euclidean geometry (constant vanishing sectional curvature), spherical geometry (constant positive sectional curvature) and hyperbolic geometry (constant negative sectional curvature). In this work, we focus on Euclidean space and hyperbolic space due to the key difference in their space expansion. Indeed, hyperbolic spaces expand faster (exponentially) than Euclidean

spaces (polynomially). Specifically, for instance, in the two-dimensional hyperbolic space $\mathbb{D}_\epsilon^2$ of constant curvature $K = -\epsilon^2 < 0$, $\epsilon > 0$ with the hyperbolic radius of r, we have:

$$L(r) = 2\pi \sinh(\epsilon r), \tag{4.1}$$

$$A(r) = 2\pi(\cosh(\epsilon r) - 1), \tag{4.2}$$

in which $L(r)$ is the length of the circle and $A(r)$ is the area of the disk. Hence, both equations illustrate the exponentially expansion of the hyperbolic space $\mathbb{H}_\epsilon^2$ with respect to the radius r.

Although hyperbolic space cannot be isometrically embedded into Euclidean space, there exists multiple models of hyperbolic geometry that can be formulated as a subset of Euclidean space and are very insightful to work with, depending on different tasks. In this work, we prefer the Poincaré ball model due to its conformality (i.e., angles are preserved between hyperbolic and Euclidean space) and convenient parameterization [NK17].

The Poincaré ball model is the Riemannian manifold $\mathcal{P}^n = (\mathbb{D}^n, g_p)$, in which $\mathbb{D}^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$ is the *open* n-dimensional unit ball that is equipped with the metric:

$$g_p(\mathbf{x}) = \left(\frac{2}{1 - \|\mathbf{x}\|^2}\right)^2 g_e, \tag{4.3}$$

where $\mathbf{x} \in \mathbb{D}^n$; $\|\cdot\|$ denotes the Euclidean norm; and g_e is the Euclidean metric tensor with components $\mathbf{I}_n$ of $\mathbb{R}^n$.

The induced distance between two points on $\mathbb{D}^n$ is given by:

$$d_{\mathbb{D}}(\mathbf{x}, \mathbf{y}) = \cosh^{-1}\left(1 + 2\frac{\|\mathbf{x} - \mathbf{y}\|^2}{(1 - \|\mathbf{x}\|^2)(1 - \|\mathbf{y}\|^2)}\right). \tag{4.4}$$

In fact, if we adopt the hyperbolic distance function as a matching function to model the relationships between users and items, the hyperbolic distance $d_{\mathbb{D}}(\mathbf{u}, \mathbf{v})$ between user u and item v could be calculated based on Eqn. (4.4).

On a side note, let $\mathbf{v}_j$ and $\mathbf{v}_k$ represent the items user i liked and did not like with $d_{\mathbb{D}}(\mathbf{u}_i, \mathbf{v}_j)$ and $d_{\mathbb{D}}(\mathbf{u}_i, \mathbf{v}_k)$ are their distances to the user i on hyperbolic space, respectively. Our goal is to pull $\mathbf{v}_j$ close to $\mathbf{u}_i$ while pushing $\mathbf{v}_k$ away from $\mathbf{u}_i$. If we consider the triplet as a tree with two children $\mathbf{v}_j$, $\mathbf{v}_k$ of parent $\mathbf{u}_i$ and place $\mathbf{u}_i$ relatively close to the origin, the graph distance of $\mathbf{v}_j$ and $\mathbf{v}_k$ is obviously calculated as $d(\mathbf{v}_j, \mathbf{v}_k) = d(\mathbf{u}_i, \mathbf{v}_j) + d(\mathbf{u}_i, \mathbf{v}_k)$, or we will obtain the ratio $\frac{d(\mathbf{v}_j, \mathbf{v}_k)}{d(\mathbf{u}_i, \mathbf{v}_j) + d(\mathbf{u}_i, \mathbf{v}_k)} = 1$. If we embed the triplet in Euclidean space, the ratio $\frac{d_{\mathbb{E}}(\mathbf{v}_j, \mathbf{v}_k)}{d_{\mathbb{E}}(\mathbf{u}_i, \mathbf{v}_j) + d_{\mathbb{E}}(\mathbf{u}_i, \mathbf{v}_k)}$ is constant, which seems not to capture the mentioned graph-like structure. However, in hyperbolic space, the ratio $\frac{d_{\mathbb{D}}(\mathbf{v}_j, \mathbf{v}_k)}{d_{\mathbb{D}}(\mathbf{u}_i, \mathbf{v}_j) + d_{\mathbb{D}}(\mathbf{u}_i, \mathbf{v}_k)}$ approaches 1 as the edges are long enough, which makes the distances nearly preserved [SSGR18].

Thus, it is worth mentioning our key idea is that for a given triplet, we aim to embed the root (the user) arbitrarily close to the origin and space the children (positive and negative items) around a sphere centered at the parent. Notably, the distances between points grows exponentially as the norm of the vectors approaches 1. Geometrically, if we place the root node of a tree at the origin of $\mathbb{D}^n$, the children nodes spread out exponentially with their distances to the root towards the boundary of the ball due to the above mentioned property.

4.2.2 Gyrovector spaces

In this section, we make use of Möbius gyrovector spaces operations [GBH18b] to generally design the distance of user-item pairs for further extension.

Specifically, for $c \geq 0$, we denote $\mathbb{D}_c^n = \{x \in \mathbb{R}^n : c\|x\|^2 < 1\}$, which is considered as the open ball of radius $\frac{1}{\sqrt{c}}$. Note that if $c = 0$, we get $\mathbb{D}_c^n = \mathbb{R}^n$; and if $c = 1$, we retrieve the usual unit ball as $\mathbb{D}_c^n = \mathbb{D}^n$.

Some widely used Möbius operations of gyrovector spaces are introduced as follows:

Möbius addition: The Möbius addition of x and y in $\mathbb{D}_c^n$ is defined:

$$x \oplus_c y = \frac{(1 + 2c\langle x, y\rangle + c\|y\|^2)x + (1 - c\|x\|^2)y}{1 + 2c\langle x, y\rangle + c\|x\|^2\|y\|^2}. \tag{4.5}$$

Möbius scalar multiplication: For $c > 0$, the Möbius scalar multiplication of $x \in \mathbb{D}_c^n \setminus \{\mathbf{0}\}$ with $r \in \mathbb{R}$ is defined:

$$r \otimes_c x = \frac{1}{\sqrt{c}} \tanh(r \tanh^{-1}(\sqrt{c}\|x\|)) \frac{x}{\|x\|}, \tag{4.6}$$

and $r \otimes_c \mathbf{0} = \mathbf{0}$. Note that when $c \to 0$, we recover the Euclidean addition and scalar multiplication. The Möbius subtraction can also be obtained as $x \ominus_c y = x \oplus_c (-y)$.

Möbius exponential and logarithmic maps: For any $x \in \mathbb{D}_c^n$, the Möbius exponential map and logarithmic map, given $v \neq \mathbf{0}$ and $y \neq x$, are defined:

$$\exp_x^c(v) = x \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_x^c\|v\|}{2}\right) \frac{v}{\sqrt{c}\|v\|} \right), \tag{4.7}$$

$$\log_x^c(y) = \frac{2}{\lambda_x^c\sqrt{c}} \tanh^{-1}(\sqrt{c}\|(-x) \oplus_c y\|) \frac{(-x) \oplus_c y}{\|(-x) \oplus_c y\|}, \tag{4.8}$$

where $\lambda_x^c = \frac{2}{1-c\|x\|^2}$ is the conformal factor of $(\mathbb{D}_c^n, g^c)$ in which g^c is the generalized hyperbolic metric tensor. We also recover the Euclidean exponential map and logarithmic

map as $c \to 0$. Readers can refer to [GBH18b, Ung09] for the detailed introduction to Gyrovector spaces.

We then obtain the generalized distance in Gyrovector spaces:

$$d_c(x, y) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\|(-x) \oplus_c y\|). \tag{4.9}$$

When $c \to 0$, we recover the Euclidean distance since we have $\lim_{c\to 0} d_c(x, y) = 2\|x - y\|$. When $c = 1$, we retrieve the Eqn. (4.4). In other words, hyperbolic space resembles Euclidean as it gets closer to the origin, which motivates us to design our loss function in the multi-objective learning framework.

4.2.3 Model Formulation

Our proposed model takes a user (denoted as $\mathbf{u}_i$), a positive (observed) item (denoted as $\mathbf{v}_j$) and a negative (unobserved) item (denoted as $\mathbf{v}_k$) as an input. Each user and item is represented as a one-hot vector which map onto a dense low-dimensional vector by indexing onto an user/item embedding matrix. We learn these vectors with the generalized distance:

$$d_c(\mathbf{u}, \mathbf{v}) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\|(-\mathbf{u}) \oplus_c \mathbf{v}\|), \tag{4.10}$$

in which an item j that user i liked (positive) is expected to be closer to the user than the ones he did not like (negative).

In fact, we would like to learn the user-item joint metric to encode the observed positive feedback. Specifically, the learned metric pulls the positive pairs closer and pushes the other pairs further apart.

Notably, this process will also cluster the users who like the same items together, and the items that are liked by the same users together, due to the triangle inequalities. Similar to [HYC+17], for a given user, the nearest neighborhood items are: 1) the items liked by this user previously, and 2) the items liked by other users with similar interests to this user. In other words, we are also able to indirectly observe the relationships between user-user pairs and item-item pairs through the pull-push mechanism of metric learning.

Pull-and-Push Optimization. To formulate such constraint, we define our pull-and-push loss function as:

$$\mathcal{L}_P = \sum_{(i,j)\in\mathbb{S}} \sum_{(i,k)\notin\mathbb{S}} [m + d^2_{\mathbb{D}}(i, j) - d^2_{\mathbb{D}}(i, k)]_+, \tag{4.11}$$

where j is an item user i liked and k is the one he did not like; $\mathbb{S}$ contains all the observed implicit feedback, i.e. positive item-user pairs; $[z]_+ = \max(0, z)$ is the standard hinge loss; and $m > 0$ is the safety margin size. Notably, our loss function does not adopt the ranking loss weight compared to [HYC+17].

Distortion Optimization. The ultimate goal when embedding a space into another is to preserve distances while maintaining complex structures/relationships [SSGR18]. Thus, it becomes a challenge when embedding user-item pairs to hyperbolic space with the needs of preserving good structure quality for metric learning. To this end, we consider the two factors of good representations namely *local* and *global* factor. Locally, the children items must be spread out on the sphere around the parent user as described, with pull and push forces created by the gradients. Globally, the learned triplets should be separated reasonably from each other. While pull-and-push optimization satisfies the local requirement, we define the distortion optimization function to meet the global requirement as:

$$\mathcal{L}_D = \sum_{(i,j)\in\mathbb{S}} \Big[\frac{|d_{\mathbb{D}}(f(i), f(j)) - d_{\mathbb{E}}(i,j)|}{d_{\mathbb{E}}(i,j)}\Big]_+ + \sum_{(i,k)\notin\mathbb{S}} \Big[\frac{|d_{\mathbb{D}}(f(i), f(k)) - d_{\mathbb{E}}(i,k)|}{d_{\mathbb{E}}(i,k)}\Big]_+, \quad (4.12)$$

where $|\cdot|$ defines the absolute value; and $f(\cdot)$ is a mapping function $f : \mathbb{E} \to \mathbb{D}$ from Euclidean space $\mathbb{E}$ to hyperbolic space $\mathbb{D}$. In this work, we take $f(\cdot)$ as an identity function.

We recall that the further two points move toward the edge of the hyperbolic space, the closer their distance is to the path through the origin. Thus, the Eqn. (4.12) is designed to place them far enough to make the distortion as low as possible. Therefore, the idea of this equation is to act as a penalty term, in which we penalize the difference in the distance when embed to hyperbolic space to preserve the graph distance ratio as discussed. In short, we would obtain the separate sub-tree embeddings (i.e., the tripets) if we could put those points further enough towards the boundary [Sar11, SSGR18].

We aim to preserve the distances by minimizing $\mathcal{L}_D$ for the global factor. Ideally, the lower the distortion, the better the preservation.

Multi-Task Learning. We then integrate the pull-and-push part (i.e., $\mathcal{L}_P$) and the distortion part (i.e., $\mathcal{L}_D$) into an end-to-end fashion through a multi-task learning framework. The objective function is defined as:

$$\min_{\Theta} \mathcal{L} = \mathcal{L}_P + \gamma \mathcal{L}_D, \quad (4.13)$$

where Θ is the total parameter space, including all embeddings and variables of the networks; and γ is the multi-task learning weight.

There is an unavoidable trade-off between the precision (learned from the pull-push mechanism) and the distortion as similar to [SSGR18]. Thus, jointly training $\mathcal{L}_P$ and $\mathcal{L}_D$ can help to boost the model performance while providing good representations. Indeed, we examine the performance of HyperML with and without the distortion by varying different multi-task learning weight γ in our experiment in Section 4.3.

Dataset	Interactions	# Users	# Items	% Density
Movie20M	16M	53K	27K	1.15
Movie1M	1M	6K	4K	4.52
Goodbooks	6M	53K	10K	1.14
Yelp	1M	22K	18K	0.26
Meetup	248K	47K	17K	0.03
Clothing	358K	39K	23K	0.04
Sports & Outdoors	368K	36K	18K	0.06
Cell Phones	250K	28K	10K	0.09
Toys & Games	206K	19K	12K	0.09
Automotive	26K	3K	2K	0.49

Table 4.1: Statistics of all datasets used in our experimental evaluation

Gradient Conversion. The parameters of our model are learned by projected Riemannian stochastic gradient descent (RSGD) [NK17] with the form:

$$\boldsymbol{\theta}_{t+1} = \mathfrak{R}_{\theta_t}(-\eta_t \nabla_R \mathcal{L}(\boldsymbol{\theta}_t)), \tag{4.14}$$

where $\mathfrak{R}_{\theta_t}$ denotes a retraction onto $\mathbb{D}$ at $\boldsymbol{\theta}$ and η_t is the learning rate at time t.

The Riemannian gradient ∇_R is then calculated from the Euclidean gradient by rescaling ∇_E with the inverse of the Poincaré ball metric tensor as:

$$\nabla_R = \frac{(1 - \|\boldsymbol{\theta}_t\|^2)^2}{4} \nabla_E, \tag{4.15}$$

in which this scaling factor depends on the Euclidean distance of the point at time t from the origin [NK17, TTH18]. Notably, one could also exploit full RSGD for optimization to perform the updates instead of using first-order approximation to the exponential map [Bon13, GBH18a, WL18, BG19].

4.3 Experiments

4.3.1 Experimental Setup

Datasets. To evaluate our experiments, we use a wide spectrum of datasets with diverse domains and densities. The statistics of the datasets are reported in Table 4.1.

- **MovieLens**: A widely adopted benchmark dataset in the application domain of recommending movies to users provided by GroupLens research[1]. We use two configurations, namely MovieLens20M and MovieLens1M. Similar to [TATH18], the MovieLens20M datasets are filtered with 100-core setting.

[1] https://grouplens.org/datasets/movielens/

	MovieLens20M		MovieLens1M		Goodbooks		Yelp		Meetup	
	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10	HR@10
MF-BPR	63.462	82.206	55.173	74.057	49.559	71.033	56.443	77.926	48.359	62.468
MLP	62.500	84.380	54.851	73.812	48.597	70.226	52.777	75.784	43.310	55.616
NCF	59.485	81.859	55.503	74.127	50.823	72.014	53.078	72.757	52.334	62.210
CML	62.664	85.571	55.737	74.528	49.010	72.556	54.996	77.122	51.453	60.589
LRML	63.775	81.327	54.057	73.358	50.392	71.424	54.719	76.764	50.208	61.347
HyperML	**64.042**	**87.363**	**56.197**	**75.629**	**51.088**	**74.152**	**59.543**	**81.392**	**54.633**	**67.304**
Improvement	+0.42%	+2.09%	+0.83%	+1.48%	+0.52%	+2.20%	+5.49%	+4.45%	+4.39%	+7.74%
	Clothing		Sports		Cell phones		Games		Automotive	
	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10	HR@10	nDCG@10	HR@10
MF-BPR	13.189	20.509	26.130	38.553	26.483	37.434	22.156	34.877	20.433	31.707
MLP	13.947	22.726	24.431	37.015	25.732	37.677	21.074	32.251	16.789	27.685
NCF	16.809	26.470	20.268	30.232	22.496	32.697	20.959	30.871	17.340	28.441
CML	16.623	26.371	19.211	30.197	19.320	29.746	21.579	32.524	17.556	27.985
LRML	16.643	26.421	22.938	33.667	20.177	30.999	20.747	31.695	16.492	26.124
HyperML	**17.150**	**27.899**	**34.576**	**48.262**	**29.325**	**42.921**	**23.164**	**35.995**	**24.736**	**37.324**
Improvement	+2.03%	+5.40%	+32.32%	+25.18%	+10.73%	+13.92%	+4.55%	+3.21%	+21.06%	+17.72%

Table 4.2: Experimental results (nDCG@10 and HR@10) on ten public benchmark datasets. Best result is in bold face and second best is underlined. Our proposed HyperML achieves very competitive results, outperforming strong recent advanced metric learning baselines such as CML and LRML.

- **Goodbooks**: A large book recommendation dataset contains six million ratings for ten thousand most popular (with most ratings) books.[2]
- **Yelp**: A crowd-sourced platform for local businesses such as restaurants, bars, etc. We use the dataset from the 2018 edition of the Yelp Dataset Challenge.[3]
- **Meetup**: An event-based social network dataset. We use the dataset includes event-user pairs from NYC that was provided by [PLCZ16].
- **Amazon Reviews**: The amazon review datasets that was introduced in [HM16a]. The subsets[4] are selected based on promoting diversity based on dataset size and domain.

Evaluation Protocol and Metrics. We experiment on the one-class collaborative filtering setup. We adopt nDCG@10 (normalized discounted cumulative gain) and HR@10 (Hit Ratio) evaluation metrics, which are well-established ranking metrics for recommendation task. Following [HLZ+17, TATH18], we randomly select 100 negative samples which the user have not interacted with and rank the ground truth amongst these negative samples. For all datasets, the last item the user has interacted with is withheld

[2]https://github.com/zygmuntz/goodbooks-10k

[3]https://www.yelp.com/dataset/challenge

[4]Datasets are obtained from http://jmcauley.ucsd.edu/data/amazon/ using the 5-core setting with the domain names truncated in the interest of space.

as the test set while the penultimate serves as the validation set. During training, we report the test scores of the model based on the best validation scores. All models are evaluated on the validation set at every 50 epochs.

Compared Baselines. In our experiments, we compare with five well-established and competitive baselines which in turn employ different matching functions: inner product (MF-BPR), neural networks (MLP, NCF) and metric learning (CML, LRML).

- **Matrix Factorization with Bayesian Personalized Ranking (MF-BPR)** [RFGS09] is the standard and strong collaborative filtering (CF) baseline for recommender systems. It models the user-item representation using the inner product and explores the triplet objective to rank items.
- **Multi-layered Perceptron (MLP)** is a feedforward neural network that applies multiple layers of nonlinearities to capture the relationship between users and items. We select the best number of MLP layers from $\{3, 4, 5\}$.
- **Neural Collaborative Filtering (NCF)** [HLZ+17] is a neural network based method for collaborative filtering which models nonlinear user-item interaction. The key idea of NCF is to fuse the last hidden representation of MF and MLP into a joint model. Following [HLZ+17], we use a three layered MLP with a pyramid structure.
- **Collaborative Metric Learning (CML)** [HYC+17] is a strong metric learning baseline that learns user-item similarity using the Euclidean distance. CML can be considered a key ablative baseline in our experiments, signifying the difference between Hyperbolic and Euclidean metric spaces.
- **Latent Relational Metric Learning (LRML)** [TATH18] is also a strong metric learning baseline that learns adaptive relation vectors between user and item interactions to find a single optimal translation vector between each user-item pair.

Implementation Details. We implement all models in Tensorflow. All models are trained with the Adam [KB15] or AdaGrad [DHS11] optimizer with learning rates from $\{0.01, 0.001, 0.0001, 0.00001\}$. The embedding size d of all models is tuned amongst $\{32, 64, 128\}$ and the batch size B is tuned amongst $\{128, 256, 512\}$. The multi-task learning weight γ is empirically chosen from $\{0, 0.1, 0.25, 0.5, 0.75, 1.0\}$. For models that optimize the hinge loss, the margin λ is selected from $\{0.1, 0.2, 0.5\}$. For NCF, we use a pre-trained model as reported in [HLZ+17] to achieve its best performance. All the embeddings and parameters are randomly initialized using the random uniform initializer $\mathcal{U}(-\alpha, \alpha)$. For non-metric learning baselines, we set $\alpha = 0.01$. For metric learning

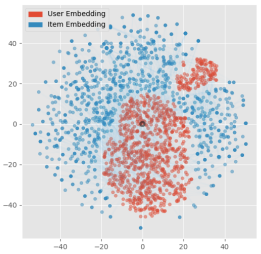
models, we empirically set $\alpha = (\frac{3\beta^2}{2d})^{\frac{1}{3}}$, in which we choose $\beta = 0.01$. The reason is that we would like all the embeddings of the metric learning models to be initialized arbitrarily close to the origin of the balls[5] for a fair comparison. For most datasets and baselines, we empirically set the embedding size of 64 and the batch size is 512. We also empirically set the dropout rate $\rho = 0.5$ to prevent overfitting. For each dataset, the optimal parameters are established by repeating each experiment for N runs and assessing the average results. We have used $N = 5$ for our experiment.

4.3.2 Experimental Results

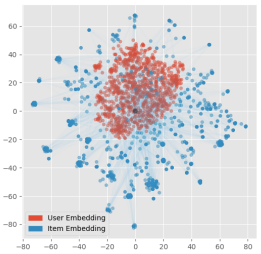
This section presents our experimental results on all datasets. For all obtained results, the best result is in boldface whereas the second best is underlined. As reported in Table 4.2, our proposed model consistently outperforms all the baselines on both HR@10 and nDCG@10 metrics across all benchmark datasets.

Pertaining to the baselines, we observe that there is no obvious winner among the baseline solutions. In addition, we also observe that the performance of MF-BPR and CML is extremely competitive, i.e. both MF-BPR and CML consistently achieve good results across the datasets. Notably, the performance of MF-BPR is much better than CML on datasets with less number of interactions. For datasets with larger size (i.e., MovieLens20M, MovieLens1M and Goodbooks), the performance of metric learning models perform better in which the gain of CML and LRML on the non-metric learning baselines across large datasets is approximately +0.39% and +0.91% respectively in terms of nDCG. One possible reason is that for small datasets with low interactions (e.g., Automotive with 26K interactions of 0.49% density), a simple model such as MF-BPR should be considered as a priority choice. In addition, the performance of a careful pre-trained NCF also often achieves competitive results with large datasets but not small ones in most cases. The explanation is because using the dual embedding spaces (since NCF combines MLP and MF) could possibly lead to overfitting if the dataset is not large enough [TATH18].

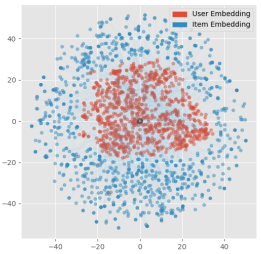
[5]The balls are referred as hyperbolic ball for HyperML model and Euclidean ball for CML and LRML model.



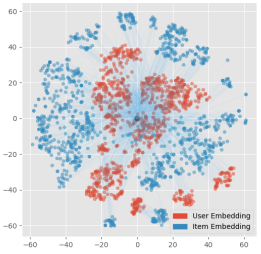
MovieLens20M



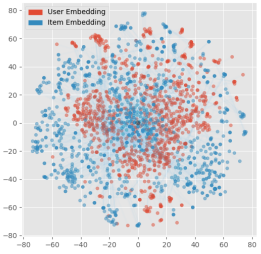
MovieLens1M



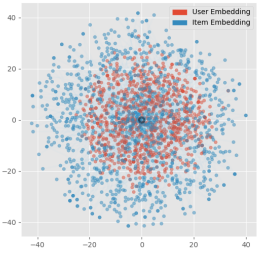
Goodbooks



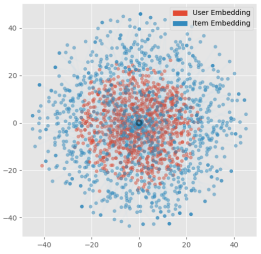
Yelp



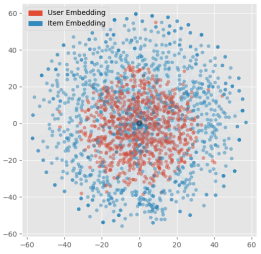
Meetup



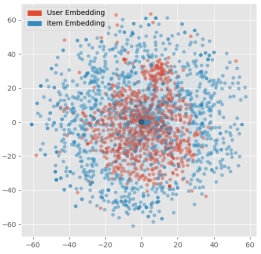
Clothing, Shoes, and Jewelry



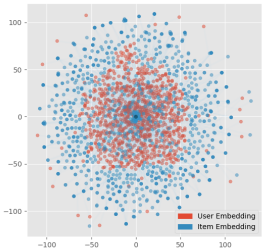
Sports and Outdoors



Cell phones and Accessories



Toys and Games



Automotive

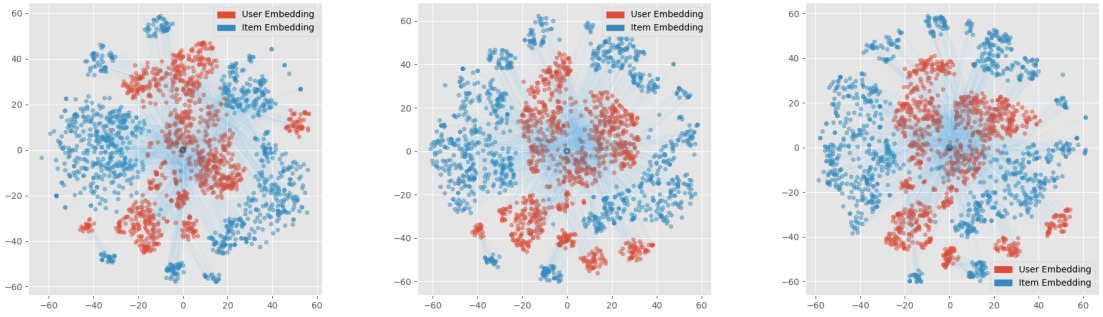
Figure 4.2: Two-dimensional hyperbolic embedding of ten benchmark datasets in the Poincaré disk using t-SNE. The images illustrate the embedding of user and item pairs after the convergence (*Best viewed in color*).

Remarkably, our proposed model HyperML demonstrates highly competitive results and consistently outperforms the best baseline method. The percentage improvements in terms of nDCG on ten datasets (in the same order as reported in Table 4.2) are +0.42%, +0.83%, +0.52%, +5.49%, +4.39%, +2.03%, +32.32%, +10.71%, +4.55% and +21.06% respectively. We also observe similar high performance gains on the hit ratio (HR@10). Note that the hyperbolic spaces expand faster, i.e. exponentially, than Euclidean spaces, in which the forces are generated by the rescaled gradients, pulling and pushing the points with more reasonable distances compared to Euclidean. Therefore, it enables us to achieve very competitive results of our proposed HyperML in the hyperbolic space over other strong Euclidean baselines. Our experimental evidence shows the remarkable recommendation results of our proposed HyperML model on the variety of datasets and the advantage of hyperbolic space over Euclidean space in boosting the performance in metric learning framework.

4.3.3 Embeddings after Convergence

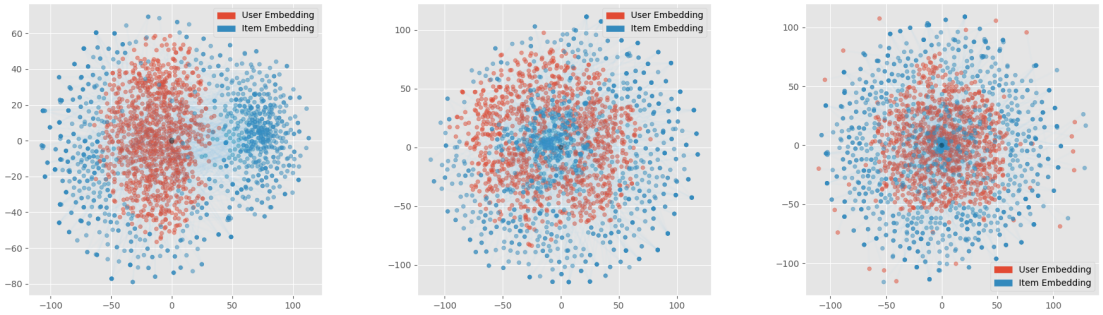
This section investigates the model convergence of our proposed HyperML to understand the behavior of the embeddings in hyperbolic space. Specifically, we study the final stage of the embeddings after convergence, then perform embeddings comparison between HyperML and other metric learning methods.

Hyperbolic Convergence. Figure 4.2 visualizes the two-dimensional hyperbolic embedding using t-SNE on the test set of ten benchmark datasets after the convergence. We observe that item embeddings form a sphere over the user embeddings. Moreover, since we conduct the analysis on the **test** set, the visualization of the user/item embeddings in Figure 4.2 demonstrates the ability of HyperML to self-organize and automatically spread out the item embeddings on the sphere around user embeddings, as mentioned in [TATH18, SSGR18, TBG19]. Moreover, the clustering characteristic of observing the user-user and item-item relationships discussed in Section 4.2 is also captured in Figure 4.2. It could be seen especially clearly for the MovieLens and Yelp dataset.



LRML CML HyperML

(a). Embeddings comparison on Yelp dataset.



LRML CML HyperML

(b). Embeddings comparison on Automotive dataset.

Figure 4.3: Comparison between two-dimensional Poincaré embedding and Euclidean embedding on Yelp and Automotive dataset. The images illustrate the embeddings of LRML, CML and HyperML after convergence (*Best viewed in color*).

Convergence Comparison. Figure 4.3 illustrates the comparison between the two-dimensional Poincaré embedding (HyperML) and Euclidean embedding (CML, LRML) on the Yelp and Amazon dataset. For the Euclidean embedding, we clip the norm (i.e., the norm of the embeddings is constrained to 1) and initialize all the embeddings very close to the origin, for an analogous comparison.

At first glance, we notice the difference between the three types of embedding by observing the distribution of user and item embeddings in the spaces after the convergence. While HyperML and CML have the item embeddings gradually assemble to a sphere structure surround user embeddings, the item embeddings of LRML have the opposite movement. The reason is because the motivation behind both CML and HyperML is to

	Meetup		Clothing	
	nDCG@10	HR@10	nDCG@10	HR@10
HyperML	54.633	67.304	17.150	27.899
HyperTS	54.612	67.277	17.190	27.959
	Sports		Cell phones	
	nDCG@10	HR@10	nDCG@10	HR@10
HyperML	34.576	48.262	29.325	42.921
HyperTS	31.896	45.272	29.933	43.532

Table 4.3: Performance comparison between HyperML and HyperTS.

create the learned metric through the pull-push mechanism, whereas the motivation of LRML is to additionally learn the translation vector, which establishes the main cause of different visualizations.

It is worth mentioning that since we initialize all the embeddings very close to the origin, we observe the difference between hyperbolic and Euclidean space that leads to the difference in the convergence of HyperML and CML. While both models form a sphere shape over the user embeddings equally, we observe that the user embeddings of HyperML tend to be located closer to the origin than CML while we get similar spread out observation of items. The explanation is that even with similar forces created by the gradients in the same direction, the different expansion property of the two spaces produces the distances between the triplets more separable, which leads to the boosting performance of the proposed model.

4.3.4 Comparison of Hyperbolic Variants

In this section, we study the variants of our proposed model: HyperML and HyperTS (applied optimization after mapping the user and item embeddings to the tangent space at $\mathbf{0}$ using the $\log_{\mathbf{0}}$ map). Notably, HyperTS is viable because the tangent space at the origin of the Poincaré ball resembles Euclidean space. Table 4.3 represents the performance of the variants on the datasets in terms of nDCG@10 and HR@10. In general, we observe both HyperML and HyperTS achieve highly competitive results, boosting the performance over Euclidean metric learning models across Meetup, Clothing, Sports, and Cell phones datasets.

4.3.5 Effect of Scaling Variable

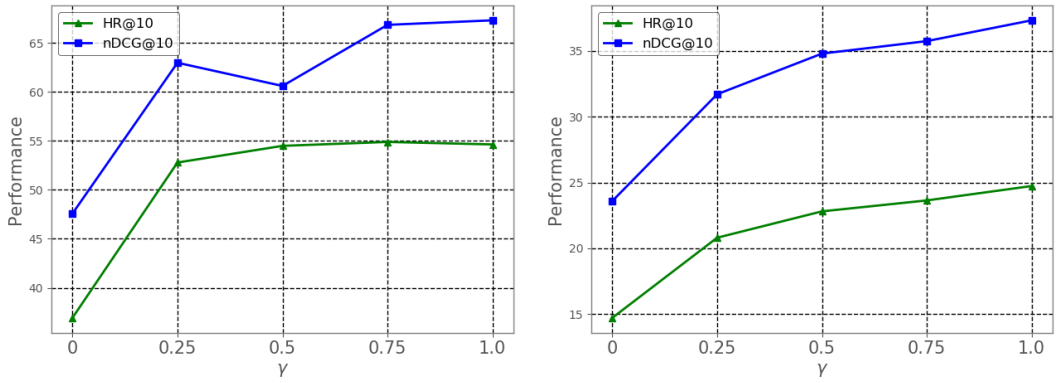
In this section, we study the effect of the variable c on our proposed HyperML and the CML baseline model. Table 4.4 represents the performance of HyperML regarding the different value of the scaling variable c comparing to CML in terms of nDCG@10. We

Scaling	Goodbooks			Games		
Variable c	HyperML	CML	$\Delta(\%)$	HyperML	CML	$\Delta(\%)$
$c = 0.5$	51.396	49.010	+4.87%	37.134	32.524	+14.17%
$c = 1.0$	51.088	49.010	+4.24%	35.995	32.524	+10.67%
$c = 2.0$	49.631	49.010	+1.27%	38.578	32.524	+18.61%
$c = 4.0$	46.017	49.010	−6.11%	35.466	32.524	+9.05%
$c = 8.0$	40.488	49.010	−17.39%	28.390	32.524	−12.71%

Table 4.4: Effects of the scaling variable c on Goodbooks and Games datasets in terms of nDCG@10.

observe that HyperML achieves best performance when $c = 0.5$ on Goodbooks dataset, but $c = 2.0$ on Games dataset. For other values of c, we notice the oscillated performance of HyperML.

As introduced, for $c > 0$, our ball shrinks to the radius of $\frac{1}{\sqrt{c}}$. Without loss of generality, the case of $c > 0$ can be reduced to $c = 1$ (the usual unit ball). However, we observe that different scaling variable c would effect the performance differently in practice, which should be set carefully for each dataset. In fact, with c carries values from 0.5 to 8.0, the percentage gain/loss of HyperML over CML varies from $+4.87\%/-17.39\%$ to $+14.17\%/-12.71\%$ on Goodbooks and Games dataset, respectively.



(a) Meetup (b) Automotive

Figure 4.4: Performance on Different Multi-objective Learning Weight γ.

4.3.6 Accuracy Trade-off with Different Multi-objective Learning Weight

In this section, we study the effect of different multi-objective learning weight γ on our proposed HyperML model on Meetup and Automotive dataset. Figure 4.4 represents

the performance of HyperML when changing the value of the multi-objective learning weight γ in terms of HR@10 and nDCG@10. We observe the obvious boost of the performance when γ increases from 0 to positive values on both two datasets. While for Meetup dataset, HyperML achieves best performance when $\gamma = 0.75$, we observe the performance of HyperML achieves its best result on Automotive dataset when $\gamma = 1.0$. For other values of γ, we also observe the oscillated performance of HyperML due to the trade-off. On a side note, when $\gamma = 0$, i.e. removing the distortion, we notice the decreasing performance of HyperML compared to CML by -21.48% and -15.48% in terms of nDCG@10 on Meetup and Automotive dataset, respectively.

Thus, we conclude that the multi-objective learning weight γ as well as the distortion play important roles on boosting the performance, in which the weight γ causes the trade-off between minimizing the distortion and the model's accuracy.

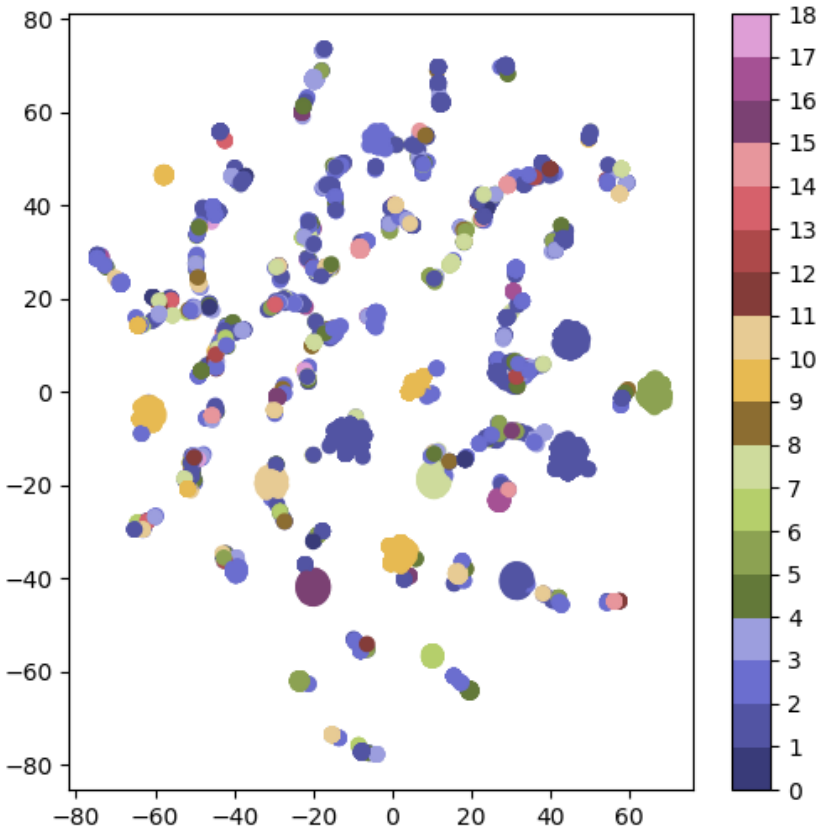


Figure 4.5: 2D t-SNE item embeddings visualization of hyperbolic metric in MovieLens1M dataset (*Best viewed in color*).

4.3.7 Metric Learning Visualization

In this section, we study the clustering effect of HyperML. Figure 4.5 represents the clustering effect in which the 18 colors represent 18 movie genres from the MovieLens1M dataset[6]. From the figure, we empirically discover that despite being only trained on implicit interactions, explicit rating information is surprisingly being discovered in HyperML. Within the hyperbolic space, the metric learning shows cluster structures of items with same genres induced by users, providing insight and achieving similar effect as [HM16b, HYC+17]. The visualization supports our previous claim that the nearest neighborhood items tend to be liked by the same users with similar interests. Notably, the t-SNE visualization also illustrates the sphere structure embeddings as introduced.

[6]The colors were assigned to the movie genres in the same order as reported in http://files.grouplens.org/datasets/movielens/ml-1m-README.txt

Chapter 5

Neural Architecture Dropout Representation for Recommendation

5.1 Introduction

Across the literature, many recommendation models have been proposed using deep learning for user/item representations [RFGS09, HLZ+17, ZYST19, XYH+17, HC17, XHZ+19, TLLK19, HYC+17, DCJ19]. A variety of architecture designs have been nominated, in which they consider the shallow pyramid multi-layered perceptrons (MLP) structure as an essential part of the model [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. For instance, [HLZ+17] combined matrix factorization (MF) and MLP for dual space embedding, [ZACC17] expressed the non-linearities by passing the element-wise product of user and item representations into a pyramid MLP, [HC17, XHZ+19] stacked an MLP on top of the architecture for prediction, or [TLLK19] proposed a square-distance MLP for metric learning. As a result, these models mostly pay attention to explore the notion of stacking multiple layers to capture the non-linearities of user-item interactions. Different perspective from existing methods, in this chapter, we consider that an MLP of stacking multiple hidden layers may carry overlapping information across layers, in which parts of a neural network architecture could be removed to avoid redundancy.

Recent years have witnessed a success of attention mechanism with a wide range of applications in recommendation domain [XYH+17, TLH18, TLLK19, TSL19, CZJC18, TZLH18]. Several attention mechanisms with either vanilla attention or co-/self-attention have been applied to recommendation such as general attention [XYH+17, CZJC18], multi-head attention [TLH18] or metric learning attention [TLLK19, TSL19]. However, to the best of our knowledge, existing attention-based recommendation models only focused on handling user-item relations or extracting important features of side information. In this chapter, we introduce a new approach of attention mechanism on activating only

specific parts of the neural architecture, achieving similar effect as dropout [HSK+12]. Indeed, we perform attention mechanism on hidden layers directly to capture the most informative layers to build more accurate and better representations, which is also the key difference of our work.

For the first time, we introduce a new concept of exploring attention mechanism as a regularization effect in the application of personalized recommendation. We propose a conceptually simple informative and effective model, in which it is capable of not only achieving highly competitive results, but also providing explainable representations due to the impact of attention mechanism. The proposed paradigm reinforces the main idea of learning user-item representations, while maintaining the simplicity and effectiveness of the previous proposed models.

Our Contributions. The overall contributions of this chapter are summarized as follows:

- We introduce a new kind of regularization effect by exploring attention mechanism directly on hidden layers to avoid redundancy in the context of personalized recommendation.
- We propose two effective neural architectures, C-DropRec and S-DropRec, for collaborative filtering with implicit feedback. We design simple but informative and effective architectures, which enable to incorporate attention mechanisms into hidden layers.
- We conduct extensive experiments on six widely adopted benchmark datasets for recommender system to demonstrate the effectiveness of employing attention mechanisms on hidden layers, consistently outperforming several recent strong competitive baselines with only stacked multi-layered perceptrons.
- We investigate the explainable insights of our two architectures to further exploit the effectiveness of attention mechanisms on providing regularization.

5.2 Understanding Dropout

Dropout [HSK+12] is a popular method that has been widely used to prevent overfitting in neural networks. In general, the high-level idea behind dropout is to combine an approximately exponential number of models to predict the output. Concretely, each node in the deep network is originally assigned a probability, in which we name it a *keep* probability and denote it as p_k. During the training process, dropout randomly sets the node value to 0 with the probability of $1 - p_k$. And during the inference process, all the weights in the layer will be multiplied by p_k. In addition, it is worth pointing out that

applying dropout during the inference process is actually equivalent to applying dropout during the training process with $p_k = 1$. In practice, there are multiple ways to apply dropout in a neural network, in which there is no one single solution. Typically, we usually apply dropout after the non-linear activation function. However, in some cases such as using rectified linear units (ReLUs), it might be better to put dropout before the activation function for computational efficiency [BS13, SHK+14].

In the scope of recommender systems, we easily observe that almost all the proposed model architectures apply dropout automatically as part of their hyper-parameter [HLZ+17, TW18, CZH+17, TLH18, VSP+17]. Those papers often varry the ratio of dropout usually from 0.2 to 0.8. However, in recommendation domain, many user-item based recommendation problems do not really require deep networks. In fact, the performance even deteriorates if many layers are used. Thus, despite of the fact that dropout has been very successful generally, it is difficult to explain if a new proposed recommendation model indeed yields good performance itself or that is the result of the combination between complicated architecture design and dropout. This could possibly explain why in many user-item based problems, authors usually integrate MLP (with dropout) as an approach to boost the performance.

Thus, inspired by this observation, our main motivation of this work is to answer the following question:

- Is it possible to design a general module that can: 1) incorporate into any deep recommendation architectures, 2) keep the good performance similar to dropout, and 3) explainable?

In the subsequent sections, we will introduce our proposed framework, performance comparision, and analyses to answer the question above.

5.3 Our Proposed Framework

We present a novel, simple and adaptive architecture dropout model for general recommendation, namely DropRec. Firstly, DropRec stacks each pair of user and item embedding into two l-layer MLPs to incorporate non-linearities to the recommendation model. Then, those $2l$ hidden layers are fed into co/self-attention mechanism(s) to output the transformed representation of user and item embedding. Finally, the transformed representation is then used for prediction by leveraging pairwise learning objective. Based on this approach, we propose our two variants, namely, C-DropRec and S-DropRec. Figure 5.1 illustrates the two variants of DropRec, in which the left figure represents the C-DropRec architecture and the right figure represents the S-DropRec architecture.

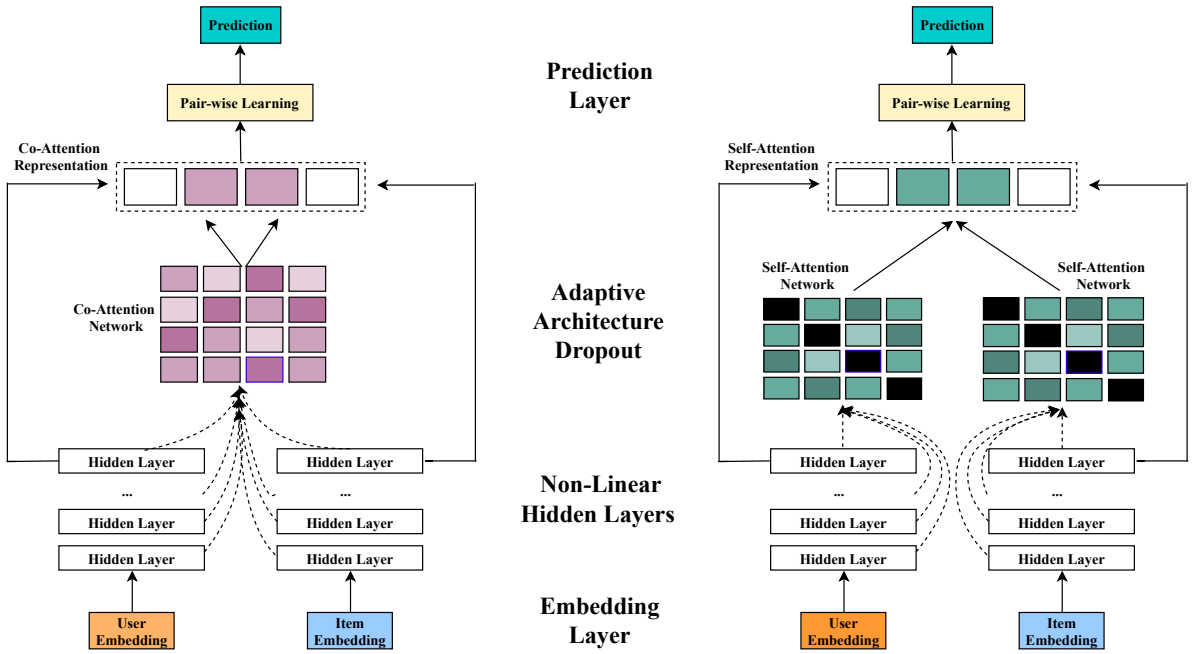


Figure 5.1: An illustration of Co-Attention based DropRec (C-DropRec) (*left*) and Self-Attention based DropRec (S-DropRec) (*right*). The curved dashed arrows represent the inputs of hidden layers into attention mechanisms (*Best viewed in color*).

Embedding Layer. Our proposed model takes a user (denoted as $\mathbf{u}_i$), a positive (observed) item (denoted as $\mathbf{v}_j$) and a negative (unobserved) item (denoted as $\mathbf{v}_k$) as an input. The user and item embedding matrices are parameterized by $\mathbf{P}_U \in \mathbb{R}^{d \times |U|}$ and $\mathbf{P}_V \in \mathbb{R}^{d \times |V|}$, where d is the dimension of the user and item embeddings (i.e., embedding size), while $|U|$ and $|V|$ are the total number of users and items respectively. The output of this layer is a pair of user and item embeddings in which we denote as $\mathbf{u} \in \mathbb{R}^d$ for user embedding and $\mathbf{v} \in \mathbb{R}^d$ for item embedding.

User/Item Hidden Layers. Each user and item embedding is then passed into a l-layers core MLP separately. We design each MLP consists of the same number of l hidden layers, where each layer performs a non-linear transformation over the output of the previous layer. Formally, the formula is defined as:

$$\mathbf{H}_k(x) = ReLU(\mathbf{W}_k x + \mathbf{b}_k), \tag{5.1}$$

where $\mathbf{W}_k \in \mathbb{R}^{n_{(k-1)} \times n_k}$ and $\mathbf{b}_k \in \mathbb{R}^{d_k}$, in which d_k is the specified dimension of layer n_k, and $n_{(k-1)}$ is the output dimension of the $(k-1)^{th}$ layer. We set the dimension $d_k = d$, which equals to the dimension of the user and item embedding. Notably, while a tower MLP structure is usually adopted by many recent works, i.e. $n_k = \frac{n_{(k-1)}}{2}$, we employ a fixed size of $n_k = n_{(k-1)}$ in our proposed frameworks.

5.3.1 Co-Attention based DropRec (C-DropRec)

From each l-layer MLP constructed from user and item embedding, we obtain l intermediate hidden outputs. Thus, we represent C-DropRec in which it computes the affinity matrix to attend/learn the similarity scores between those $2l$ hidden layers. In other words, from the two l-layer sequences, this co-attention network aims to select the most informative layers from the $2l$ hidden layers of the user-item pair.

Affinity Matrix. Given l user hidden layers (denote as $\mathbf{h}_i$) and l item hidden layers (denote as $\mathbf{r}_j$), $\forall i, j = 1, 2, \ldots, l$, a similarity (affinity) matrix $\mathbf{A} \in \mathbb{R}^{l \times l}$ is calculated as:

$$A_{ij} = \mathbf{h}_i^T \mathbf{r}_j, \tag{5.2}$$

where A_{ij} is an element of the affinity matrix $\mathbf{A} \in \mathbb{R}^{l \times l}$, representing the similarity between i-th user hidden layer and j-th item hidden layer.

On a side note, the computation of the affinity matrix requires all hidden layers of the MLPs to have equal length as similar to word embeddings in a sentence, which explains the inappropriate of the MLP tower structure in our proposed frameworks.

Pooling Operation. Next, we utilize the standard co-attention mechanism by taking the normalization of the row and column wise of the affinity matrix A and using them to weight the hidden layers of the two MLPs. The pooling operation is defined as:

$$\mathbf{h}'_i = \sum_{j=1}^{l} \frac{exp(A_{ij})}{\sum_{k=1}^{l} exp(A_{ik})} \mathbf{h}_j, \tag{5.3}$$

$$\mathbf{r}'_i = \sum_{j=1}^{l} \frac{exp(A^T_{ij})}{\sum_{k=1}^{l} exp(A^T_{ik})} \mathbf{r}_j, \tag{5.4}$$

where $\mathbf{h}'_i$ and $\mathbf{r}'_i$ are the local co-attention representations of $\mathbf{h}_i$ and $\mathbf{r}_i$, respectively; and $exp(z) = e^z$. The output vectors $\mathbf{h}'$ and $\mathbf{r}'$ are inferred as the local co-attention representations.

We learn a single co-attention representation for each output by summing over all local representations as:

$$\mathbf{h}_A = \sum_{i=1}^{l} \mathbf{h}'_i \quad \text{and} \quad \mathbf{r}_A = \sum_{i=1}^{l} \mathbf{r}'_i, \tag{5.5}$$

We then attain the final representation of C-DropRec as $\mathbf{p}_C = [\mathbf{h}_l; \mathbf{h}_A; \mathbf{r}_A; \mathbf{r}_l]$ by concatenating the final co-attention representations with the l-th layer of the MLPs to form a feature vector which contains rich and informative representation of the hidden layers.

5.3.2 Self-Attention based DropRec (S-DropRec)

Likewise, we represent S-DropRec that utilizing self-attention mechanisms. Intuitively, while the obtained hidden layers capture the non-linearities transformation of user and item representation, S-DropRec provides richer and larger extents of matching interfaces, observing information passing among each l hidden layers before learning representation.

Attentive Matrix. Similar to the affinity matrix, we define attentive matrices as $\mathbf{B} \in \mathbb{R}^{l \times l}$ and $\mathbf{B}' \in \mathbb{R}^{l \times l}$, which is formed by computing the matching scores between hidden states i, j of the l hidden layers of user and the l hidden layers of item, denoted as $\mathbf{f}_i$ and $\mathbf{g}_i$ respectively, $\forall i, j = 1, 2, \ldots, l$:

$$B_{ij} = \mathbf{f}_i^T \mathbf{f}_j \quad \text{and} \quad B'_{ij} = \mathbf{g}_i^T \mathbf{g}_j, \tag{5.6}$$

where B_{ij} and B'_{ij} are an element of the attentive matrix $\mathbf{B} \in \mathbb{R}^{l \times l}$ and $\mathbf{B}' \in \mathbb{R}^{l \times l}$, representing self-matching between layer i and layer j of each l-layered MLP.

Pooling Operation. We next normalize the attention matrix $\mathbf{B}$ and $\mathbf{B}'$ as:

$$\mathbf{f}'_i = \sum_{j=1}^{l} \frac{exp(B_{ij})}{\sum_{k=1}^{l} exp(B_{ik})} \mathbf{f}_j, \tag{5.7}$$

$$\mathbf{g}'_i = \sum_{j=1}^{l} \frac{exp(B'_{ij})}{\sum_{k=1}^{l} exp(B'_{ik})} \mathbf{g}_j, \tag{5.8}$$

where $\mathbf{f}'_i$ and $\mathbf{g}'_i$ are the local self-attention representations of $\mathbf{f}_i$ and $\mathbf{g}_i$, respectively; and $exp(z) = e^z$.

Thus, a single self-attention representation for each output is defined as:

$$\mathbf{f}_B = \sum_{i=1}^{l} \mathbf{f}'_i \quad \text{and} \quad \mathbf{g}_{B'} = \sum_{i=1}^{l} \mathbf{g}'_i, \tag{5.9}$$

In this architecture of S-DropRec, the two self-matching operation are the two self-attention mechanisms, in which each operation is related to l hidden layers of user and item. As a result, we obtain the final representation of S-DropRec from the two local self-attention representations as $\mathbf{p}_S = [\mathbf{f}_l; \mathbf{f}_B; \mathbf{g}_B; \mathbf{g}_l]$.

5.3.3 Model Optimization

This section illustrates the optimization and learning process of DropRec. After attaining the final representation $\mathbf{p}$, i.e. $\mathbf{p} = \mathbf{p}_C$ for C-DropRec and $\mathbf{p} = \mathbf{p}_S$ for S-DropRec, we feed this obtained vector into a single linear layer as:

$$M(\mathbf{u}, \mathbf{v}) = \mathbf{W}_m^T \mathbf{p} + b_m, \tag{5.10}$$

where $M(\cdot)$ is the scoring function for each user-item pair, $\mathbf{W}_m \in \mathbb{R}^{4d \times 1}$ and $b_m \in \mathbb{R}$.

Notably, the learned representation $\mathbf{p}$ of our C-DropRec and S-DropRec also integrates the last layer of each MLP (i.e., the l-th layer) to make the final representations. The reason is because the last layer of each MLP (i.e., one for user and one for item) contains the rich and semantic information after the information passing process. As such, concatenating these last layers could lead to the improvement in the performance due to the strong fusion of representations. Also, C-DropRec and S-DropRec are proposed to guide the network focusing on specific parts but not removing the other parts; thus, feeding the final layers for final representations is obviously understandable. Moreover, with the design of the pooling operations in C-DropRec and S-DropRec, we indeed observe that when the network produces very similar outputs for multiple hidden layers, a shallower network without attention mechanisms could also achieve similar results.

Finally, in order to train our proposed C-DropRec and S-DropRec, we leverage Bayesian Personalized Ranking (BPR) pairwise learning [RFGS09] to minimize the pairwise ranking loss between the positive and negative items, in which the objective function is defined as follows:

$$\arg\min_{\Theta} \sum_{(i,j,k) \in \mathcal{D}} - \ln \sigma\{M(\mathbf{u}_i, \mathbf{v}_j) - M(\mathbf{u}_i, \mathbf{v}_k)\} + \lambda(\|\Theta\|^2), \tag{5.11}$$

where (i, j, k) is the triplet that belongs to the set $\mathcal{D}$ that contains all pairs of positive item j and negative item k for each user i; σ is the sigmoid function; Θ represents the model parameters; and λ is the regularization parameter.

5.4 Empirical Evaluation

In this section, we aim to answer the following three key research questions (**RQ**):

- **RQ1:** Do C-DropRec and S-Droprec outperform existing stacked multi-layered models?
- **RQ2:** What is the regularization effect of DropRec? Are we able to derive any explainable insights?
- **RQ3:** How does DropRec perform with respect to different key hyperparameters?

Dataset	# Interaction	# User	# Item	Density (%)
MovieLens HetRec	859,824	2,113	10,109	4.03
Epinions	677,695	40,163	139,738	0.01
Ciao	54,942	17,615	16,121	0.02
Office Products	63,068	4,905	2,420	0.53
Sports & Outdoors	367,533	35,598	18,357	0.06
Automotive	26,329	2,928	1,835	0.49

Table 5.1: Statistics of the datasets used in our experiments.

5.4.1 Experimental Settings

Datasets. For our experimental evaluation, we adopt six benchmark datasets. We carefully select the datasets based on the diverse domains and densities. The statistics of the datasets are reported in Table 5.1.

- **MovieLens**: A widely adopted benchmark dataset in the application domain of recommending movies to users provided by GroupLens research.[1] In this work, we evaluate on the HetRec version.
- **Epinions**: A consumer review dataset that was crawled on Epinions website and originally introduced in [MA07].[2]
- **Ciao**: A rating and review platform dataset were introduced in [GZTY14]. We use the entire crawled DVDs category dataset in the period from July 2000 to November 2013.[3]
- **Amazon Reviews**: The amazon product rating and review datasets [HM16a]. We use the subsets[4] with 5-core setting. Particularly, we adopted three subcategories: Office Products, Sports & Outdoors, and Automotive dataset.

Evaluation Protocol and Metrics. Our evaluation protocol follows [HLZ+17]. For preprocessing preparation, we binarize the explicit data by keeping existing ratings as implicit feedback, and apply a popular 5-core setting to filter inactive users [TLLK19, HM16a]. We adopt two well-established ranking evaluation metrics for recommendation task: NDCG@10 (Normalized Discounted Cumulative Gain) and HR@10 (Hit Ratio). Following [HLZ+17], we randomly select 100 negative samples which the user have not

[1] https://grouplens.org/datasets/movielens/
[2] http://www.trustlet.org/downloaded_epinions.html
[3] https://www.ciao.co.uk/
[4] http://jmcauley.ucsd.edu/data/amazon/

Method Type	Method	MovieLens HetRec		Epinions		Ciao	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Standard Recommendation Method (Group 1)	MF-BPR	64.153	46.553	47.469	34.992	54.682	40.111
	PMF	65.058	46.764	48.466	35.956	55.725	41.077
Stacking Layers Method (Group 2)	MLP	62.693	44.042	47.219	34.357	54.764	38.512
	MLP++	62.500	43.666	46.566	33.827	55.204	39.460
	JRL	64.672	47.136	41.686	30.910	45.276	32.990
	JRL++	64.527	45.274	38.076	28.792	44.836	32.012
	NCF	65.154	47.256	41.783	30.689	51.882	36.848
	NCF++	64.141	46.883	40.889	30.314	50.961	37.532
Ours	C-DropRec	66.996*	48.700*	48.665*	38.010*	**60.128***	**47.265***
	S-DropRec	**69.305***	**53.236***	**51.153***	**44.562***	59.728*	41.556*
Improvement over Group 1/2	$\Delta(\%)$	+6.37	+12.66	+5.55	+23.94	+7.90	+15.06

Method Type	Method	Office Products		Sports & Outdoors		Automotive	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Standard Recommendation Method (Group 1)	MF-BPR	24.139	14.512	36.296	24.173	30.653	20.682
	PMF	23.335	13.673	35.268	24.070	30.525	19.429
Stacking Layers Method (Group 2)	MLP	22.867	13.207	36.715	24.086	26.966	16.841
	MLP++	22.512	12.987	36.108	23.525	29.901	18.298
	JRL	19.883	11.832	27.396	17.953	18.223	11.037
	JRL++	19.069	11.358	30.604	19.970	17.924	11.381
	NCF	22.366	13.031	29.585	19.490	27.458	17.589
	NCF++	23.847	13.466	29.746	19.760	27.809	18.929
Ours	C-DropRec	24.235*	15.160*	37.138*	24.262*	32.058*	22.536*
	S-DropRec	**29.689***	**19.361***	**41.834***	**29.848***	**32.514***	**22.626***
Improvement over Group 1/2	$\Delta(\%)$	+22.99	+33.41	+15.26	+23.47	+6.07	+9.40

Table 5.2: Experimental results (NDCG@10 and HR@10) on six public benchmark datasets. Best result is in bold face and second best is underlined. Our proposed DropRec achieves significantly competitive results, outperforming all the stacked multi-layered models. ++ denotes models with hidden layers of equal dimension. * denotes statistically significant improvements ($p < 0.01$) compared to all the baselines.

interacted with and rank the ground truth amongst these negative samples. For all datasets, we adopt the leave-one-out evaluation protocol [HLZ+17]. Our models are evaluated on the validation set at every 50 epochs. We report the test scores for different models based on the obtained best validation scores. For each dataset, we repeat each experiment for 10 runs and assess the average results.

Compared Baselines. In our experiments, we introduce five well-established and competitive baselines with their variants for comparison against our proposed DropRec. Specifically, we investigate two groups of baselines: standard recommendation methods (**Group 1**) (*with regularizers*) and diverse stacking hidden layers methods (**Group 2**) (*with dropout*) to study the efficient of our proposed adaptive architecture dropout for representation. Notably, we do not adopt recent complex attention-based deep architectures as baselines, in which they additionally require extra information such as user/item neighborhood interactions and attributes to incorporate into their architectures [ESF18, TLLK19, XHZ+19, TSL19]. The baselines in Group 1 and Group 2 are introduced as below:

- **Bayesian Personalized Ranking (MF-BPR)** [RFGS09] is a state-of-the-art pairwise matrix factorization method for recommender systems.
- **Probabilistic Matrix Factorization (PMF)** [SM07] is a strong pointwise baseline based on matrix factorization which is built upon user-item pairs.
- **Multi-layered Perceptron (MLP)** [HLZ+17] is a feedforward neural network that applies multiple layers to capture the non-linearities relationship between users and items. We use two versions, one with pyramid structure [HLZ+17], and the other one has hidden layers with equal dimension.
- **Joint Representation Learning (JRL)** [ZACC17] is a strong baseline that explores the stack hidden layers by passing the element-wise product of user and item latent vectors into a pyramid multi-layered perceptron for prediction. We also propose and compare with another variant of JRL.
- **Neural Collaborative Filtering (NCF)** [HLZ+17] is another strong and well-known method for collaborative filtering. The key idea of NCF is to fuse the last hidden representations of MF and MLP together into a joint model. We use the pre-trained version of NCF to achieve its best performance. We also compare with two versions of NCF as similar to JRL and MLP baselines.

Implementation Details. We implement all models in Tensorflow and perform grid search for model tuning. All models are trained using Adam [KB15] with a learning rate is tuned amongst $\{0.01, 0.001, 0.0001\}$. The embedding size d of all models is selected from $\{32, 64, 128\}$. The batch size B is tuned amongst $\{64, 128, 512\}$. All baselines are optimized with the regularization term λ is tuned amongst $\{0, 0.1, 0.01, 0.001, 0.0001\}$. For NCF, we use a pre-trained model as reported in [HLZ+17] to achieve the best performance for the comparison. All the embeddings and parameters are randomly initialized using the default xavier initializer. For most datasets and baselines, we empirically set the hyperparameters with the learning rate of 0.001, the batch size is 512 and the embedding size of 64. For baselines, we empirically set the dropout rate $\rho = 0.5$ to prevent overfitting. Early stopping is also applied to the baselines as a form of regularization. For all models, we vary l depths in $\{3, 4, 5\}$ and select the one that achieves the best performance, following [TLLK19, HLZ+17].

5.4.2 Do C-DropRec and S-DropRec outperform existing stacked multi-layered models?

This section experimentally presents our results on all six datasets (**RQ1**). For all obtained results, the best result is in boldface whereas the second best is underlined. As

reported in Table 5.2, our proposed C-DropRec and S-DropRec consistently outperform all the baselines in terms of HR@10 and NDCG@10 metrics across six datasets.

Remarkably, our proposed models significantly outperform the best baseline method. The percentage relative improvements in terms of NDCG@10 on six datasets (in the same order as reported in Table 5.2) are +12.66%, +23.94%, +15.06%, +33.41%, +23.47% and +9.40% respectively. We also observe similar high performance gains on the hit ratio (HR@10). Notably, although there is no obvious winner among the baselines, we observe that MF and NCF consistently achieve highly competitive results across datasets. Notably, the strong performance of MF could be foreseeable, in which [RZK19] pointed out that a careful tuned MF-BPR could even achieve very competitive results.

Our experimental evidence shows the remarkable recommendation results of our proposed DropRec models on the variety of datasets and the advantage of attention mechanisms in providing regularization effect to enhance the performance.

On a side note, the results shown in Table 5.2 indicates that S-DropRec has better performance on most of the datasets than C-DropRec. The results may be caused by the simple integrated co-attention mechanism. In fact, there exists a few other variants of co-attention mechanism that have shown remarkable results compared to self-attention mechanism such as coarse-grained co-attention mechanism and fine-grained co-attention mechanism [GLT19]. To optimize the performance of co-attention mechanism, we further compare our C-DropRec with two coarse-grained co-attention models: HieVQA [LYBP16] and IAN [MLZW17]; and two fine-grained co-attention mechanism: AP [dSTXZ16] and MAC [HM18]. Notably, they are all well-known architectures in the class of co-attention mechanisms.

Co-Attention Type	**Method**	**MovieLens HetRec**		**Epinions**		**Ciao**	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Coarse-grained co-attention	HieVQA	66.230	47.987	47.998	38.992	59.961	47.343
	IAN	67.113	48.404	48.656	39.112	60.751	47.537
Fine-grained co-attention	AP	67.803	49.011	48.334	39.236	60.875	48.601
	MAC	66.009	48.824	48.776	38.271	59.151	47.750
Ours	C-DropRec	66.996	48.700	48.665	38.010	60.128	47.265
Co-Attention Type	**Method**	**Office Products**		**Sports & Outdoors**		**Automotive**	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Coarse-grained co-attention	HieVQA	23.603	14.997	37.003	24.315	32.030	21.928
	IAN	24.747	15.900	37.882	24.156	32.431	22.290
Fine-grained co-attention	AP	24.316	15.163	36.772	24.578	31.979	22.150
	MAC	24.336	15.227	37.049	24.443	32.055	22.380
Ours	C-DropRec	24.235	15.160	37.138	24.262	32.058	22.536

Table 5.3: Experimental comparison between co-attention mechanisms in terms of NDCG@10 and HR@10 on all six datasets.

From Table 5.3, we observe that the performance of the co-attention mechanisms are indeed very comparable. As mentioned earlier in Section 5.2, one possible explaination

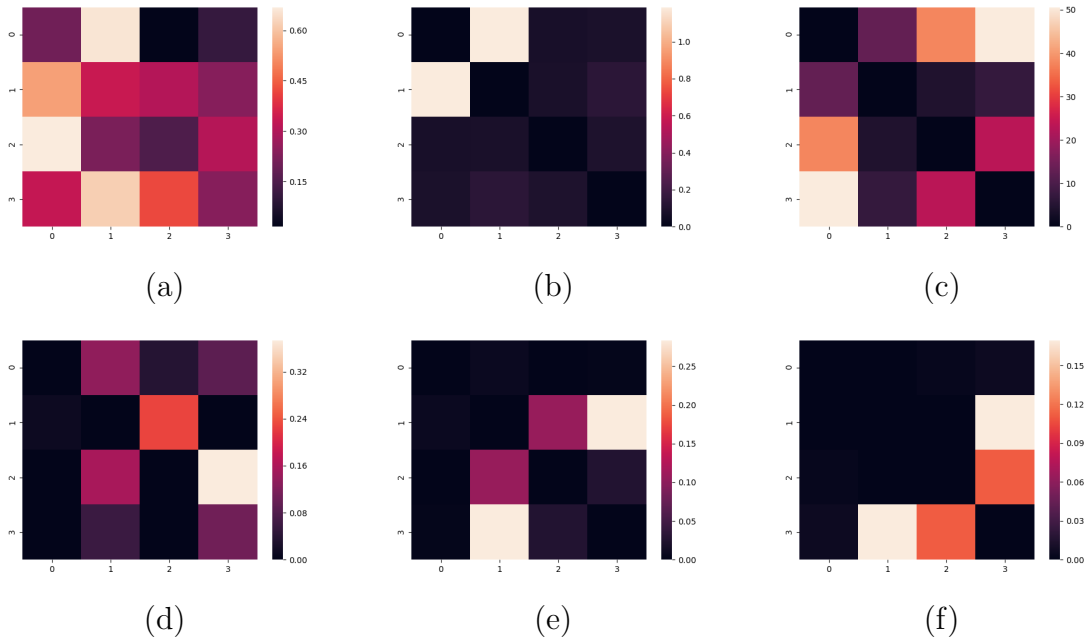


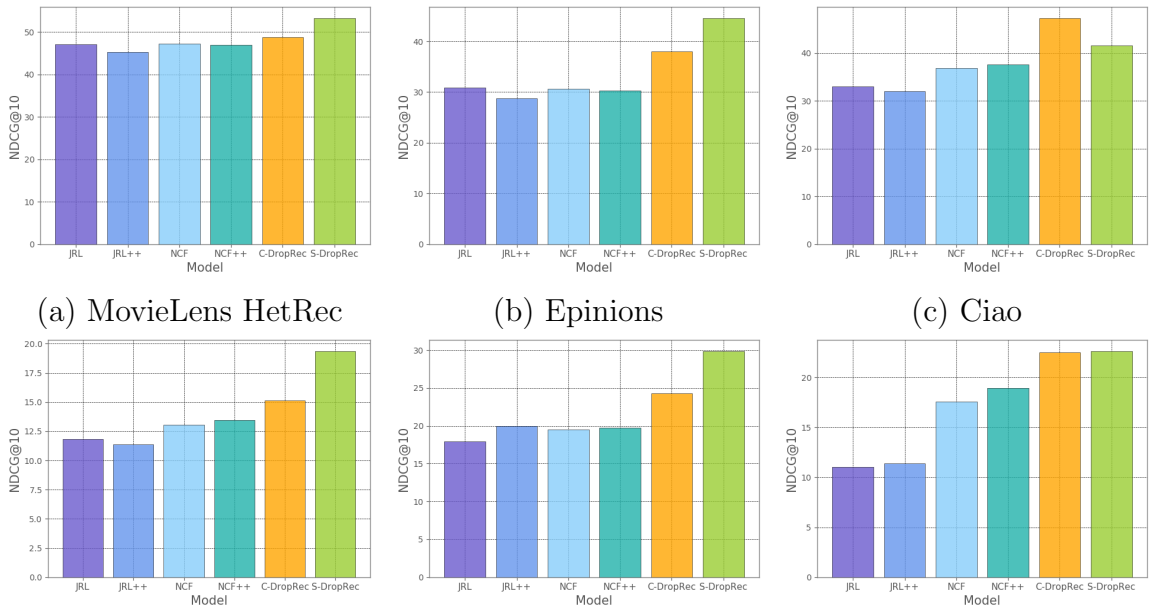
Figure 5.2: Visualization of attention weights learned by DropRec on MovieLens HetRec ((a)/(b),(c): co-/self-attention) and Automotive ((d)/(e),(f): co-/self-attention) with $l = 4$. The darker the color, the greater the correlation (*Best viewed in color*).

is that as the user-item based recommendation problems usually do not require very deep and complex networks, using complicated architectures often lead to non-increasing in the performance, or even causing the overfitting to the test set in the worse scenarios. Moreover, applying advanced attention mechanisms also contradict to our original purpose of developing a general simple and effective module to integrate into general recommenders. Thus, it is worth mentioning that using the simple co-attention architecture as C-DropRec is the best choice in our case.

5.4.3 What is the regularization effect of DropRec? Are we able to derive any explainable insights?

This section investigates the regularization effect of the co-attention network and self-attention network in our proposed C-DropRec and S-DropRec model (**RQ2**). Particularly, we study the performance gains of the two variants over recent strong stacking layers methods: JRL, JRL++, NCF and NCF++, with regularizers and dropout applied.

Firstly, Figure 5.2 illustrates the affinity and attentive matrices of DropRec on MovieLens and Automotive dataset in the case of $l = 4$. From the visualization, we notice different matching patterns across layers, in which the correlation amongst layers is also



(a) MovieLens HetRec (b) Epinions (c) Ciao

(d) Office Products (e) Sports & Outdoors (f) Automotive

Figure 5.3: Performance comparison between JRL (slate blue), JRL++ (corn flower blue), NCF (light skyblue), NCF++ (light seagreen) and C-DropRec (orange), S-DropRec (yellow green) across datasets in terms of NDCG@10 (*Best viewed in color*).

observed to be dataset dependent. Specifically, Figure 5.2(a) and 5.2(d) represent the contribution of co-attention module in C-DropRec to capture information passing of two stacked MLPs; Figure 5.2(b), 5.2(c), 5.2(e) and 5.2(f) visualize the inner workings of self-attention module in S-DropRec. The heat maps show imperfectly correlation between layers, which also explains the impact of attention mechanisms on controlling the importance of hidden layers in the architecture to achieve better performance.

Moreover, we observe from Table 5.4 that additionally exploring dropout ρ on DropRec even makes the performance decreases, which further demonstrates the regularization effect of our proposed model. Specifically, Table 5.4 represents the performance of C-DropRec and S-DropRec regarding the different values of the dropout ratio ρ in terms of NDCG@10. As noticed, integrating ρ into our proposed models does not help to improve the performance. Thus, we observe that DropRec attains similar regularization effect as dropout. On a side note, C-DropRec and S-DropRec are trained with regularization parameter $\lambda = 0$, which also strengthen our claim.

In addition, Figure 5.3 represents the relative improvement in terms of NDCG@10 of the two proposed models against the recent stacking layers baselines on all six datasets to demonstrate the effect of our adaptive architecture dropout. Different from Table 5.2, we further vary the values of λ and ρ from the range of $[0, 1]$ to search for the best

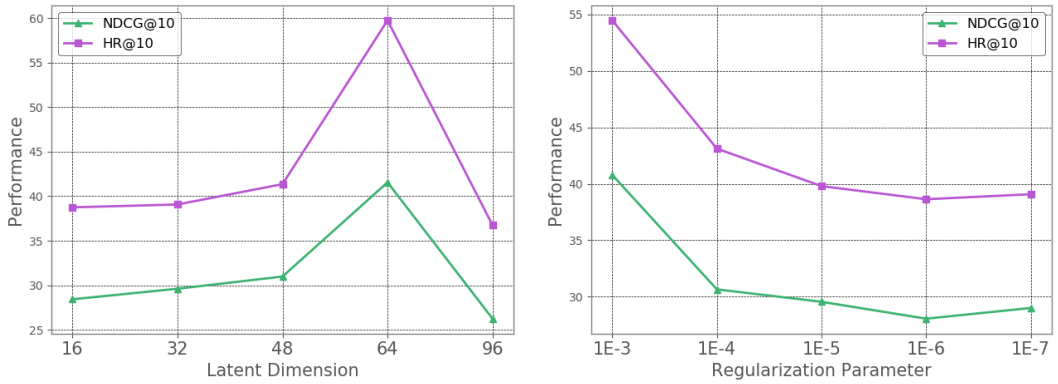
Dropout ρ	**Office Products**		
	C-DropRec	S-DropRec	Δ(%)
$\rho = 0$	15.160	19.361	-
$\rho = 0.2$	14.891	18.377	-3.43
$\rho = 0.4$	14.710	16.534	-8.79
$\rho = 0.8$	13.618	16.512	-12.44
Dropout ρ	**Sports & Outdoors**		
	C-DropRec	S-DropRec	Δ(%)
$\rho = 0$	24.262	29.848	-
$\rho = 0.2$	24.256	25.510	-7.28
$\rho = 0.4$	24.074	22.632	-12.48
$\rho = 0.8$	23.797	24.028	-10.71

Table 5.4: Regularization Effect of DropRec on Office Products and Sports & Outdoors dataset. Across two datasets, DropRec demonstrates the similar effect as dropout in terms of NDCG@10.

performance of the baselines. Note that in this experiment, we keep other parameters fixed to only observe the best performance with respect to λ and ρ. From the comparison, we observe that our adaptive architectures constantly perform better than the compared recent strong baselines across all six datasets.

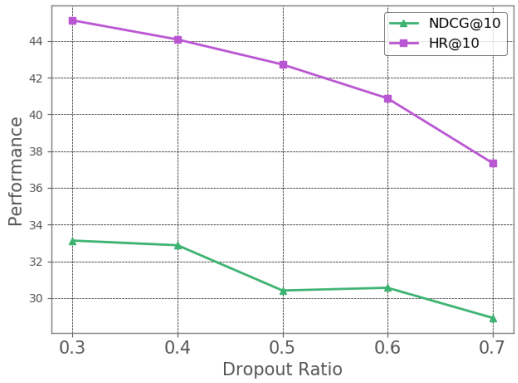
5.4.4 How does DropRec perform with respect to different key hyperparameters?

Due to limited space, we study the effect of the embedding size d, dropout ratio ρ, regularization parameter λ, and number of hidden layers l on the performance of S-DropRec (C-DropRec has a similar pattern) on Ciao dataset as shown in Figure 5.4(a), 5.4(b), 5.4(c), 5.4(d), respectively (**RQ3**). In general, we adjust the corresponding hyperparameter and keep other settings unchanged to observe the oscillated performance of S-DropRec. From the figures, it is evidently that the selection of the key hyperparameters has a big impact on the model performance. To be specific, Figure 5.4(a) explains our chosen latent dimension d based on the performance. Moreover, we notice that including the additional regularization parameters or dropout ratio does not help to enhance the performance on top of attention mechanisms, in which the performance of S-DropRec even decreases in average of -16.62% in terms of NDCG@10, as illustrated in Figure 5.4(b) and 5.4(c). Thus, our proposed adaptive architecture dropout demonstrates the effectiveness in boosting the performance even without including additional regularization parameters.

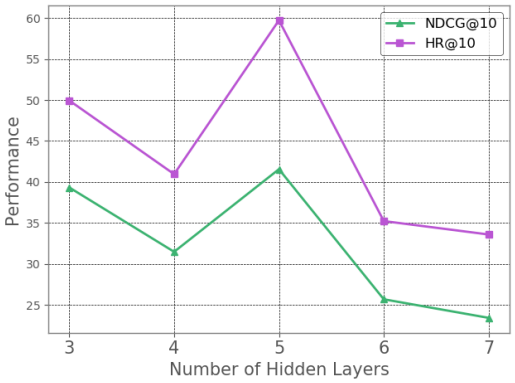


(a) Latent Dimension

(b) Regularization Parameter



(c) Dropout Ratio



(d) Number of Hidden Layers

Figure 5.4: Sensitivity of S-DropRec to hyperparameters on Epinions dataset (*Best viewed in color*).

5.5 Discussions

The focus of this chapter is to introduce a new concept of exploring attention mechanism directly on hidden layers as a regularization effect in the application of recommendation domain. Thus, the performance of DropRec is understandably limited compared to recent advanced complex networks as proposed in [TLLK19, ESF18, ZWC19, CWTY19, SZLM18, LKHJ18]. However, taking attention mechanism as a regularization approach to incorporate into deep complex architectures could be a promising direction to be explored.

Moreover, our proposed architecture in this chapter can be considered as a general deep learning module, which could be utilized to incorporate into different deep recommendation architectures. Specifically, we observe that lots of recent recommendation models exploit representation learning techniques via MLP [ZYS+18, HLZ+17, ZACC17, TLLK19, TSL19, HC17, XHZ+19]. Since our method is developed based on the structure

of MLP, the framework has the potential to be applied into hidden layers of any other MLP-like structures. For example, [TLLK19] can combine our module with the Signed Distance-based Perceptron (SDP) component, or [ZYS+18] can incorporate our module into their user-based and item-based architectures. In addition, it is interesting to point out that with our designed architecture, we could easily replace any one-tower MLP in MLP-like recommenders with our attention based two-tower structures. Therefore, revisiting recent recommendation models with our framework is also an interesting direction that we can further investigate, where we will leave it for future work. On a side note, as we focus on proposing a general module, we choose to revisit the baselines that were heavily built upon MLP-like structures to demonstrate the effectiveness of our proposed framework.

Besides, a part of the architecture proposed in [TZLH18] seems to have similar approach as ours; however, there exists differences between the two proposals. Firstly, while the authors focus on developing a very deep network such as ResNet [HZRS16] in recommendation domain, our main motivation is to avoid redundancy and activate only specific parts of the network by exploring the information passing between hidden layers. Secondly, the fusion of user and item in [TZLH18] actually represents the normal MLP, i.e., the one-tower MLP that usually uses to replace the dot product. As such, stacking more layers to the MLP does not work in recommender system, and incorporating attention mechanism on top could possibly lead to overfitting, in which we also demonstrate our claim in Figure 5.4(d).

Chapter 6

One-Off Comparison between Personalized Recommenders

6.1 Models for Evaluation

In this section, we restate our representation learning techniques introduced in the evaluation from Chapter 3, 4, and 5. These recommendation models represent the strong methods in the context of personalized recommendation, especially for implicit feedback task. They cover: 1) general standard dot product representation techniques, 2) metric learning based representation techniques, and 3) attention based representation techniques. All in all, we group them based on their recommendation techniques and summarize these architectures in Table 6.1.

Methods	Dot product based Recommenders		Metric learning based Recommenders		Attention based Recommenders	
	Matrix Factorization	Neural method	Euclidean	Hyperbolic	Co-attention	Self-attention
MF-BPR [RFGS09]	✓					
PMF [SM07]	✓					
MLP [HLZ+17]		✓				
JRL [ZACC17]		✓				
NeuMF++ [HLZ+17]	✓	✓				
CMN++ [ESF18]		✓			✓	
CML [HYC+17]			✓			
LRML [TATH18]			✓			
W-MLC (Chapter 3)			✓			
S-MLC (Chapter 3)			✓			
HyperML (Chapter 4)				✓		
C-DropRec (Chapter 5)		✓			✓	
S-DropRec (Chapter 5)		✓				✓

Table 6.1: Summary of the recommendation architectures in our evaluation.

In this chapter, we take into account totally 13 methods: 8 of them are the baselines across 3 chapters, 2 of them are from Chapter 3, 1 of them from Chapter 4 and 2 of them

from Chapter 5. For the methods that have "++" after their name, it means that we use the pre-trained version of that methods to boost the performance. Next, we give further discussions on these methods. For the details of each method, please refer to Section 1.3.

Firstly, in terms of MF-based methods, we notice that MF-BPR [RFGS09] has been remaining as one of the most competitive baselines in personalized recommendation task. The idea of MF-BPR is simply to learn a dot-product matching function that would rank positive user-item pair higher than negative user-item pair. In fact, [RZK19, RKZA20] have pointed out that even in some cases, MF-based methods could perform much better than dot-product based neural networks. This phenomenon is also observed in some previous works [KWM18, RZK19, DBCJ19, DCJ19]. Thus, it is worth noting that fine-tuning MF-BPR is an important task in the personalized recommendation problem.

Secondly, as introduced in previous chapters, metric learning based recommenders have caught the attention of the recommendation research community recently [WDWK11, CHL05, XNJR02, KTW+12, CHL05, HYC+17, TATH18, WBS05, LL16]. Next, we revisit and extend the toy example in Chapter 1 for metric learning as illustrated in Figure 6.1.

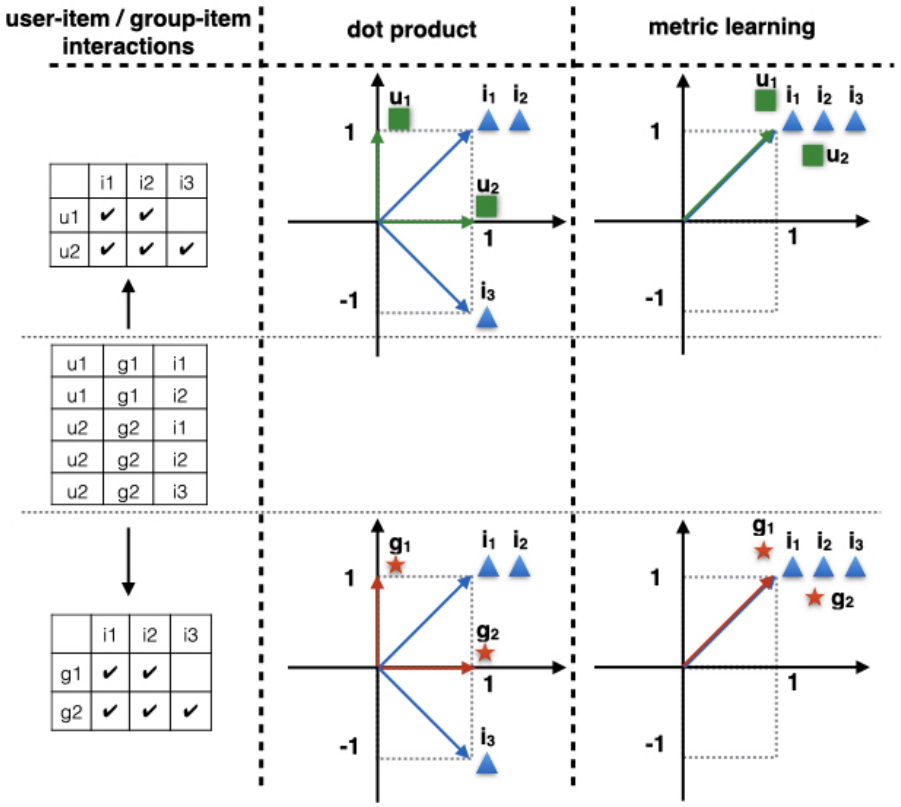


Figure 6.1: Learning with dot product vs. metric learning for user-item and group-item interactions.

As shown in Figure 6.1, assuming that we have an interaction table of users, groups and items, we extract two separate interaction tables: one is for user-item interactions and another one is for group-item interactions. Similar to our motivations in Chapter 1, we realize that metric learning yields more reasonable results, where item i_3 is correctly portrayed in both cases due to triangle inequality property. Therefore, we perceive that the evolution of metric learning is essential in recommendation domain, which also explains our motivations of exploring metric learning with different aspects in this dissertation. On a high-level note, the idea of metric learning methods is similar to dot product based methods, in which they try to pull the positive user-item pair closer in the space and simultaneously push the negative user-item pair further apart. Particularly, CML [HYC+17] was originally developed to replicate this behavior. Then, LRML [TATH18] modified the method to add an adaptive vector in a similar fashion. There exists other proposed metric learning based methods at the same timeline with CML and LRML such as [LZZ+20, HKM18, PKXY18], but the heart of all these methods could be considered as equivalent.

Thirdly, attention mechanism is another important piece that we would like to consider [VSP+17] in this dissertation. Particularly, we notice that since the introducing of attention mechanism back in 2014, the usage of attention mechanism spans across various tasks in different domains due to its strong performance. Indeed, recent superb models[1] such as Transformer [VSP+17] or BERT [DCLT18] are also advanced based on the foundation of this mechanism. In the context of recommender systems, many representation learning techniques employed attention mechanisms have been proposed [ESF18, TLLK19, XHZ+19, TSL19, SLW+19]. That being said, revisiting attention mechanism in recommender systems becomes a need, especially when there are so many complicated architectures that have been proposed in the user-item interaction based problem. As a result, we revisit co-attention and self-attention mechanism to provide a general module as introduced in the previous chapter.

In this one-off comparison, we compare our proposed representation learning techniques to the baselines with 13 methods in total. The evaluation settings and results of our extensive experiment will be introduced in the later sections.

6.2 Evaluation Settings

Generally speaking, we specifically focus on implicit feedback task of personalized recommendation problem in this dissertation. Thus, we do not consider all the datasets that introduced in Section 1.3 as some of them are only used for group recommendation (e.g., Plancast) or sequential recommendation (e.g., Tafeng). Since interaction data such as clicks or purchases is low-cost in practice compared to rating data, we believe that it is worth paying attention to this aspect of recommendation domain, i.e., implicit feedback.

[1] https://huggingface.co/transformers/model_summary.html

Following [DCJ19, DBCJ19, RKZA20], we consider the *reproducibility* as the key attention to measure the result quality of the experiments, in which we focus on answering the following question: To what extent are experimental results in the comparison reproducible with reasonable effort?

Moreover, we also understand that some representation learning techniques are actually not fully-deterministic. For example, neural network based methods usually use random initialization to initialize the embeddings, or PMF [SM07] explored probabilistic processes for matrix factorization. Therefore, we give the details of the reproducible steps in this chapter to give the readers the full view of the whole process. Notably, the variability of obtaining the similar results should be low given the detailed information. In general, the whole experiments of this extensive one-off comparison could be reproduced by following two steps: 1) data preprocessing and 2) hyper-parameters tuning, which will be introduced later in this section.

Since the proposed representation learning methods address the same task with the same protocol, we decide to compare them across 12 datasets to provide the same line of comparison. The details of these datasets could be recalled in Section 1.3. Here, we provide the statistics of these datasets in Table 6.2 for convenient.

Dataset	# of users	# of items	# of actions	density %	k-core
MovieLens20M	53K	27K	16M	1.2%	100
MovieLens1M	6K	4K	1M	4.5%	20
MovieLens100k	0.9K	1.7K	100K	6.3%	20
MovieLens HetRec	2K	10K	860K	4.0%	5
Epinions	23K	137K	631K	0.02%	5
Yelp	22K	18K	1M	0.3%	5
Goodbooks	53K	10K	6M	1.1%	5
Meetup	47K	17K	248K	0.03%	5
Clothing, Shoes, and Jewelry	39K	23K	358K	0.04%	5
Sports and Outdoors	36K	18K	368K	0.06%	5
Cell Phones	28K	10K	250K	0.09%	5
Automotive	3K	2K	26K	0.5%	5

Table 6.2: Summary statistics of datasets in our evaluation.

For data preprocesing in this one-off experiment, we firstly remove duplicated items in all the dataset. Then, we adopt the popular *k-core* preprocessing step [HLZ+17, HM16a] to filter inactive users with less than k interactions. Notably, we only filter out inactive users with less than k actions but not items with less than k consumptions [TLLK19]. Finally, we consider all observed ratings as positive interactions and the remaining as negative interactions. As mentioned in Section 1.3, we adopt the widely-used leave-one-out as the common protocol; NDCG@10 (Normalized Discounted Cumulative Gain) and

HR@10 (Hit Ratio) as evaluation metrics; and randomly select 100 negative items for ranking. For non-timestamped datasets, the order of interactions are kept as similar as the original datasets. Finally, we report the testing performance based on the best developing performance.

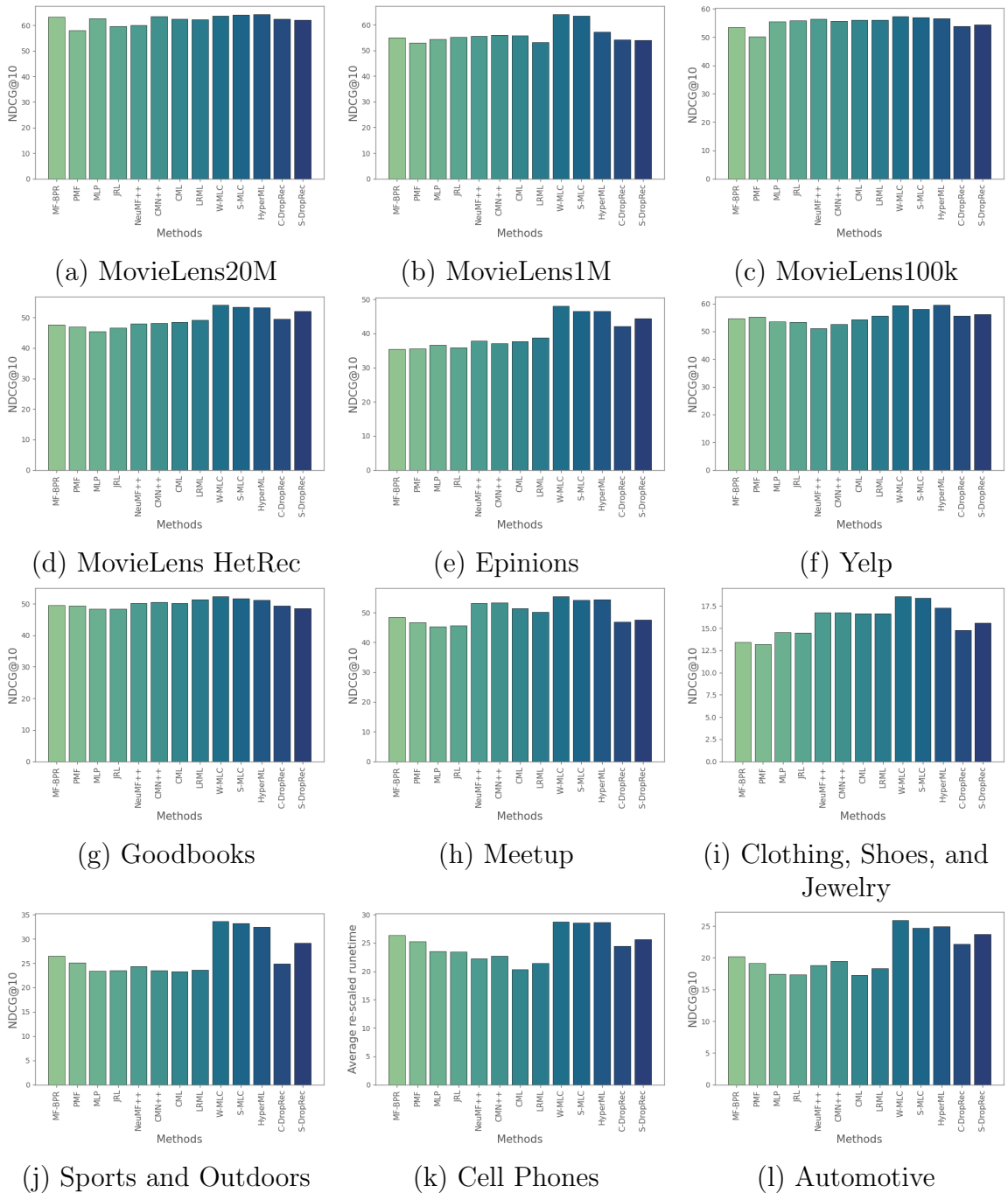
For hyper-parameters tuning, we report the values as follows: We choose Adam [KB15] or AdaGrad [DHS11] optimizer with learning rates selected from $\{0.01, 0.001, 0.0001, 0.00001\}$. For MF-based models (MF-BPR, PMF) and standard neural methods (MLP, JRL), we re-use the best parameters chosen from the previous chapters. For NeuMF++, the number of MLP layers are chosen from $\{1, 2, 3\}$. For CMN++, the number of hops are chosen from $\{1, 2, 3, 4\}$. The number of negative samples for each positive instance is chosen from $\{1, 2, 3, 4\}$. The embedding size d of all models is chosen from $\{8, 16, 32, 64, 128\}$ and the batch size is tuned amongst $\{256, 512, 1024, 2048\}$. For the models that use multi-objective learning weights (i.e., γ, η, etc.), we empirically chose the values from $\{0, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$. For models that optimize the hinge loss with a fixed margin, the margin m is selected from $\{0.1, 0.2, 0.5\}$. All the embeddings and parameters are randomly initialized using the random uniform initializer $\mathcal{U}(-\alpha, \alpha)$ with $\alpha = 0.01$. For metric learning models, we empirically set $\alpha = (\frac{3\beta^2}{2d})^{\frac{1}{3}}$ with $\beta = 0.01$ so that the embedding parameters arbitrarily close to the origin of the balls. We also empirically set the dropout rate $\rho = 0.5$. For other parameters that specific to the proposed models, we also re-use the best parameters chosen from the previous chapters. Due to the resource constraint, we decided to repeat each experiment twice and average the results. Most importantly, all the hyper-parameters are tuned using the development set [DCJ19, DBCJ19].

6.3 Summary of Results and New Insights

In this section, we report all the comparison results between the methods in Table 6.1 and the datasets in Table 6.2. Figures 6.2 depicts the overall comparison of 13 methods with respect to top-K recommendations on 12 datasets.

At first glance, we observe that the performance of metric learning based methods tend to have better results. Particularly, across all datasets, W/S-MLC and HyperML improve upon other representation learning techniques by 0.41-3.92%. Thus, this observation also explains our motivations of preferring metric learning over dot product. In fact, we also perceive the consistent performance of previous proposed metric learning methods even with their simple representation techniques [HYC+17, TATH18]. Moreover, it seems that for the datasets with low density (e.g., Sports and Outdoors with 0.06%), our proposed metric learning based techniques perform largely good. One possible explaination comes from the foundation of metric learning: the pull and push mechanism. Specifically, the distances between users and items might not be separate enough if we have a very dense

dataset. Thus, the very close positions of the representations in the space could then harm the performance.



(a) MovieLens20M (b) MovieLens1M (c) MovieLens100k

(d) MovieLens HetRec (e) Epinions (f) Yelp

(g) Goodbooks (h) Meetup (i) Clothing, Shoes, and Jewelry

(j) Sports and Outdoors (k) Cell Phones (l) Automotive

Figure 6.2: Performance comparison between 13 methods on 12 datasets in terms of NDCG@10.

-	MF-BPR	PMF	MLP	JRL	NeuMF++	CMN++
MovieLens20M	+8.98/-1.57	+0.00/-9.68	+7.92/-2.52	+2.56/-7.36	+3.51/-6.51	+9.10/-1.41
MovieLens1M	+3.79/-14.20	+0.00/-17.33	+2.55/-15.23	+4.20/-13.86	+4.77/-13.39	+5.86/-12.49
MovieLens100k	+6.42/-6.81	+0.00/-12.43	+10.67/-3.09	+11.18/-2.64	+12.22/-1.73	+10.80/-2.98
MovieLens HetRec	+4.89/-12.11	+3.39/-13.37	+0.00/-16.21	+2.74/-13.91	+5.80/-11.35	+6.29/-10.94
Epinions	+0.00/-26.53	+0.53/-26.13	+3.40/-24.03	+1.23/-25.62	+6.85/-21.49	+4.73/-23.05
Yelp	+7.13/-8.16	+8.06/-7.37	+5.00/-9.99	+4.59/-10.34	+0.00/-14.28	+3.16/-11.57
Goodbooks	+2.55/-5.30	+2.28/-5.55	+0.04/-7.62	+0.00/-7.66	+3.96/-4.00	+4.46/-3.53
Meetup	+6.84/-12.60	+2.91/-15.82	+0.00/-18.20	+0.77/-17.57	+17.12/-4.20	+17.43/-3.95
Clothing	+1.87/-27.60	+0.00/-28.93	+10.27/-21.63	+9.86/-21.93	+26.97/-9.76	+26.85/-9.85
Sports	+13.82/-21.07	+7.92/-25.16	+0.49/-30.31	+1.00/-29.96	+4.74/-27.37	+1.11/-29.88
Cell Phones	+29.65/-8.39	+24.49/-12.03	+15.93/-18.08	+15.15/-18.64	+9.43/-22.68	+11.90/-20.93
Automotive	+17.42/-21.97	+10.91/-26.29	+1.29/-32.69	+0.86/-32.97	+9.09/-27.5	+12.88/-24.99

-	CML	LRML	W-MLC	S-MLC	HyperML	C-DropRec	S-DropRec
MovieLens20M	+7.54/-2.87	+7.29/-3.10	+9.57/-1.04	+10.16/-0.50	+10.72/-0.00	+7.45/-2.95	+6.70/-3.63
MovieLens1M	+5.24/-13.00	+0.22/ -17.15	+20.96/-0.00	+19.69/-1.05	+7.94/-10.76	+2.11/-15.59	+1.98/-15.69
MovieLens100k	+11.64/-2.24	+11.79/-2.11	+14.20/-0.00	+13.64/-0.49	+12.73/-1.28	+7.42/-5.93	+8.46/-5.02
MovieLens HetRec	+7.04/-10.31	+8.38/-9.18	+19.34/-0.00	+17.86/-1.24	+17.28/-1.73	+9.18/-8.51	+14.89/-3.73
Epinions	+6.40/-21.83	+9.76/-19.35	+36.10/-0.00	+31.37/-3.47	+31.69/-3.24	+19.12/-12.48	+25.76/-7.60
Yelp	+6.42/-8.78	+8.99/-6.57	+16.18/-0.41	+13.83/-2.42	+16.66/-0.00	+8.80/-6.73	+10.06/-5.65
Goodbooks	+3.91/-4.04	+6.16/-1.97	+8.29/-0.00	+6.91/-1.27	+5.89/-2.21	+2.26/-5.57	+0.26/-7.41
Meetup	+13.23/-7.38	+10.57/-9.56	+22.25/-0.00	+19.35/-2.38	+20.02/-1.83	+3.42/-15.41	+4.79/-14.28
Clothing	+25.83/-10.57	+25.94/-10.5	+40.71/-0.00	+39.25/-1.04	+30.77/-7.06	+11.95/-20.43	+18.1/-16.07
Sports	+0.00/-30.65	+1.55/-29.58	+44.21/-0.00	+42.36/-1.28	+39.28/-3.41	+6.78/-25.95	+25.06/-13.28
Cell Phones	+0.00/-29.34	+5.58/-25.4	+41.52/-0.00	+40.33/-0.84	+40.90/-0.44	+20.24/-15.04	+25.93/-11.02
Automotive	+0.00/-33.55	+6.30/-29.36	+50.48/-0.00	+43.13/-4.88	+44.8/-3.77	+28.69/-14.48	+37.58/-8.57

Table 6.3: Summary relative improvements/deteriorations of all methods across datasets.

In addition, we observe a comparable to better performance of W-MLC compared to HyperML across the datasets. Indeed, we could actually expect this results. To recall the proposed metric learning general framework, W-MLC appears to have stronger connections between embeddings in the space, while also provide the flexibility through adaptive margin loss. In contrast, HyperML inherits the same foundation framework and simply leverages the exponential expansion property of hyperbolic space. On a side note, it is reasonable to ask the question of whether it is possible to combine the designs of the two methods to come out with a more powerful unified model. In short, it is feasible but not straightforward to perform the metric learning chain on hyperbolic space. The reason is because the metric learning chain in hyperbolic space would require several new definitions of formulas. Firstly, we need to define the projection in hyperbolic space to create a chain. Secondly, the multi-objective loss functions need to be reformulated to suit the hyperbolic properties. Lastly, we possibly expect a gradient descent based method to update parameters in hyperbolic space for optimization. In fact, [GBH18b] also re-defined RNN/GRU in hyperbolic space with the similar process.

On the other hand, we also discover the enhance performance of C/S-DropRec compared to MLP-like architectures, which also matches with our observations previously. Moreover, we notice that DropRec especially performs good on small-to-medium size data. One possible explaination is because the core framework of DropRec uses only

simple MLP. Thus, this might limit the performance of this module on large datasets due to the limited performance of MLP.

Finally, in order to give a big picture of the relative improvements/deteriorations of all methods across datasets, we further provide the information in the Table 6.3. Here, the format of each entry in the table is x/y. Specifically, x (%) and y (%) denote the relative improvement and deterioration of a method over the least and best favourable method for a particular dataset, respectively.

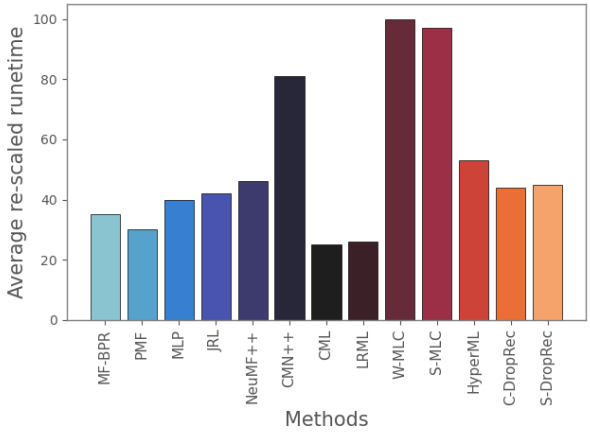


Figure 6.3: Average re-scaled runtime of 13 methods across datasets.

6.4 Runtime Comparison

To compare model runtimes, we used a Nvidia GeForce GTX 1080Ti with a batch size of 512 and embedding size of 64. To conduct this experiment, we first run 5 epochs of each method on each dataset. Then, we access the average runtime (seconds per epoch) of these methods and re-scale the values to $[0, 100]$ for one-off comparison, since we observe similar patterns across datasets. Figure 6.3 shows the runtimes (seconds per epoch) of all the representation learning methods. In terms of practicality, we observe that HyperML, CML and LRML are the three metric learning based methods that are worth to consider. Besides, MF-BPR still remains as a strong model with reasonable runtime compared to other methods. Additionally, methods with hops/chains (i.e., CMN++, W/S-MLC) indeed require much longer time for each epoch, which yields a concern about the efficient in practical usage.

Chapter 7

Learning Representation for Group Recommendation

7.1 Introduction

People often participate in activities in groups, e.g., having dinners with colleagues, watching movies with partners, and shopping with friends. This calls for effective techniques for group recommendation. Unfortunately, existing recommendation algorithms designed for individuals are not effective for group recommendation. With the availability of group event data, it comes with the need to further research on how to make effective recommendations for a group of users instead of individuals [ARC+09, CM13, GLRW13, LTYL12, YLL12, YCL14, KBV09, BMR10, OCKR01, MSC+06] to facilitate groups making decisions, and helps social network services improve user engagement. This chapter is concerned with designing highly effective recommender systems that are targeted at modeling group preferences as opposed to individual preferences, where groups are ad-hoc (any combination of individuals) rather than pre-defined.

Group preferences are not straightforward to model, given the inherent complexity of group dynamics. To this end, this chapter aims to exploit the interactions between group members in order to drive the model towards highly effective group-level recommendations. Moreover, it is natural that collective decisions have a tendency to be dynamic, i.e., a user's preference may be highly influenced by the other members in the group. Group-level agreement tends to require a *consensus* amongst group members, in which this consensus largely depends on each member's roles and expertise. While this is not explicitly captured with any semantic information or meta-data, we hypothesize that

This chapter is published as ***Lucas Vinh Tran***, *Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong and Xiaoli Li. Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (**SIGIR 2019**). Pages 255-264, Paris, France.* [TPT+19].

this can be implicitly captured with simply interaction data. As such, it is crucial to model the *interactions* among group members. However, existing proposals for group recommendation fail to model the interactions of group members well. Most of existing solutions belong to memory based methods that are based on either preference aggregation [KBV09] or score aggregation strategy [BMR10, OCKR01] and do not consider the interactions of group members. These strategies neglect the interactions between members in a group but instead using simple methods to aggregate its members' preferences. Some existing solutions are model-based approaches and try to exploit user interactions for group recommendation. However, they cannot fully utilize the user interactions as discussed in the literature reviews.

In our work, to model the interactions among group members, we propose a new neural architecture for group recommendation. Specifically, our architecture is a new variant of the attention mechanism in which each group member is represented with a single sub-attention network. Subsequently, a group of users is then represented as a *'medley'* of sub-attention networks that is responsible for making the overall recommendation. The role of each sub-attention network is to capture the preference of its representative group member, *with respect to* all other members in the group. As such, our proposed model leverages user-user interactions for making group recommendation decisions, and is not only well-aligned with the fundamental intuition of group-level dynamics but also expressive in the sense that it considers the user-user interactions. In fact, our experiments demonstrate that a simple attentive aggregation of user representations is insufficient and has roughly identical performance to that of an average pooling matrix factorization (MF) baseline (More details will be discussed in Section 7.3). On the other hand, our experiments show that our model is significantly better than seven state-of-the-art baselines. All in all, our core intuition serves as an inductive bias for our model, providing more effective group recommender performance on multiple benchmark datasets.

Our Contributions. Overall, the key contributions of this chapter are summarized as follows:

- We propose MoSAN (*Medley of Sub-Attention Networks*), a novel deep learning architecture for the group recommendation problem. Our model distinguishes itself from all prior work in group recommendation based on the fact that it considers user-user interactions using sub-attention networks. To the best of our knowledge, this is the first neural model that explores the usage of user-user interactions for the group recommendation task.

- We conduct extensive experiments on four publicly available benchmark datasets. Our experimental results demonstrate that MoSAN achieves state-of-the-art performance, outperforming a myriad of strong competitors in the task at hand.

- In addition to comparison against well-studied baselines, we conduct ablation studies against two baselines AVG-MF (Average Matrix Factorization) and ATT-AVG (Attentive Aggregation) and observe that our approach significantly outperforms both approaches. This shows that our proposed model provides a more useful inductive bias for the task at hand.
- We show that the attention weights of MoSAN are interpretable, i.e., it is able to discover the different weights of each user across groups, highlighting the impact of each user in different groups.

7.2 Our Proposed Framework

Generally speaking, our proposed MoSAN model consists of two levels: 1) user interaction learning through sub-attention network modules in which each group member is represented with a single sub-attention network module to simulate the decision making process of other members under the influential of that group member; and 2) the medley of sub-group decisions which concludes the final decision to recommend items for the group. Next, we first present the input encoding of the group recommendation problem in Section 7.2.1. We then introduce the two key components of our proposed model in Section 7.2.2. Lastly, we discuss the optimization method in Section 7.2.3.

7.2.1 Input Encoding

Let $U = \{u_1, u_2, \ldots, u_M\}$ and $I = \{i_1, i_2, \ldots, i_N\}$ be the sets of M users and N items, respectively. We denote a history log (i.e., training instances) as $H = \{\langle g_1, s_1\rangle, \langle g_2, s_2\rangle, \ldots, \langle g_L, s_L\rangle\}$, where $g_l \subset U$ denotes an ad-hoc group and $s_l = i_k$ denotes the selected item by the group.

Given a target group g_t, we aim to generate a recommendation list of items that group members $u_{g_t,i}$ in the group g_t may be interested in. Note that the target group can be an ad-hoc group. The group recommendation problem can be defined as follows:

Input: Users U, items I, historical log H, and a target group g_t.
Output: A function that maps an item to a real value to represent the item score for the target group $f_{g_t} : I \rightarrow \mathbb{R}$.

For each training instance, our model accepts a list of group members (users) and an item. Each user and item are represented as one-hot vectors, which map onto a dense low-dimensional vector by looking up an user and item embedding matrix, respectively. These user and item embeddings are referred to as *user-latent* and *item-latent* vectors (denoted as $\mathbf{u}_m$ and $\mathbf{v}_j$, respectively) in the rest of the chapter. Moreover, we introduce

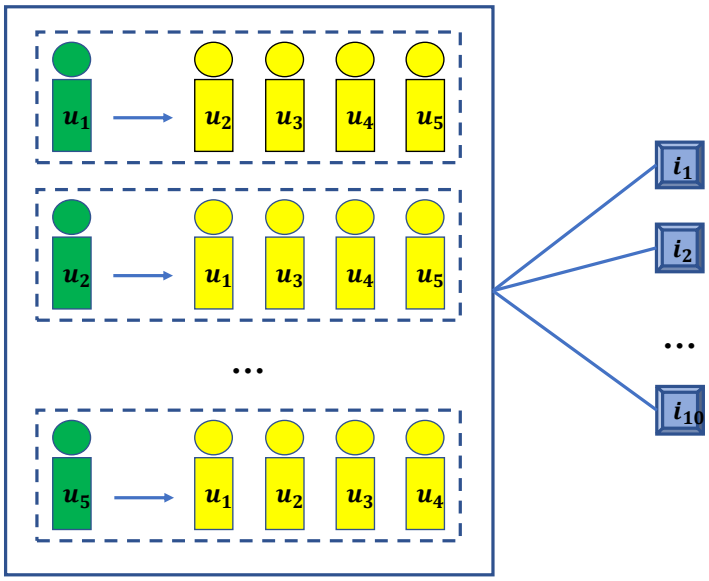


Figure 7.1: Illustration of group decision making process of MoSAN, in which green users represent *user-context* and yellow users represent *user-latent*.

an additional user embedding matrix, referred to as *user-context* embedding (denoted as $\mathbf{c}_l$) which is specifically designed to denote the owner of each sub-attention network. Embedding matrices are trainable parameters of the model's architecture network and are trained end-to-end with the rest of the parameters. Finally, our model trains by pairwise ranking, which essentially requires negative item samples. We define the negative items as the items that are not selected by all members in the group.

Figure 7.1 illustrates the group decision making process we simulate in this chapter. The solid rectangle represents the final decision of the group with each item. The inner dashed rectangles represent user interactions within the group, in which each dashed rectangle illustrates how a user (*user-context*) influences other users (*user-latent*) in the decision making process.

7.2.2 Medley of Sub-Attention Networks

This subsection introduces our proposed neural framework for group recommendation. To recapitulate, the key motivation behind our neural architecture is to enable group-level recommendations by modeling interactions between group members. Figure 7.2 illustrates the architecture of MoSAN.

Motivation. It is not straightforward to model the group preferences, given the inherent complexity of group dynamics. Therefore, typical aggregation algorithms may

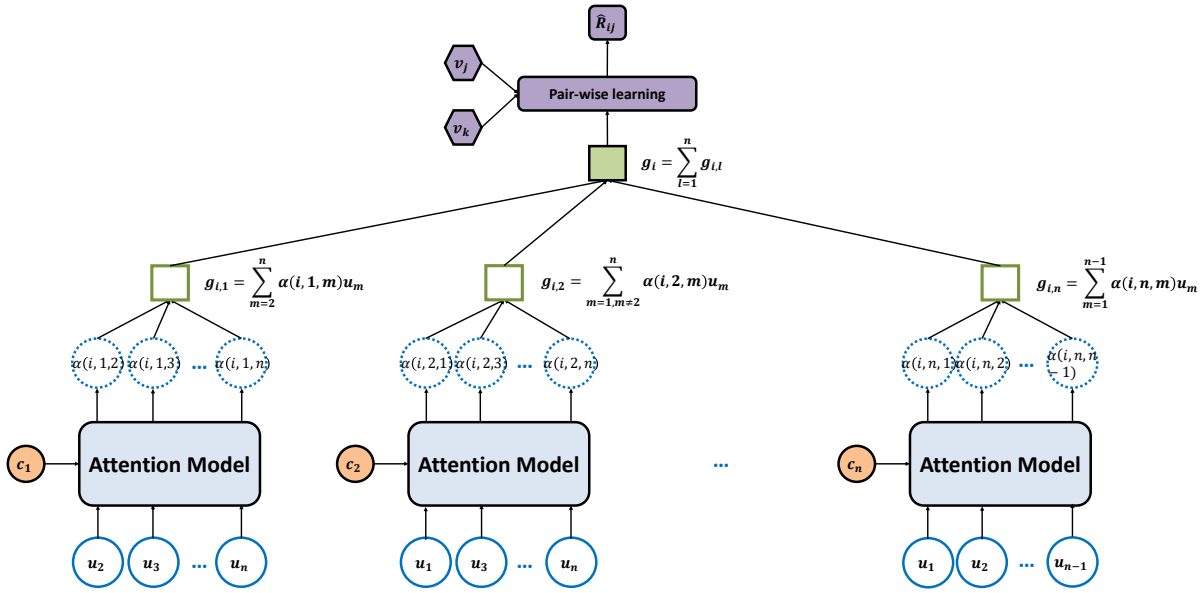


Figure 7.2: High level overview of our proposed Medley of Sub-Attention Networks (MoSAN) model. Each sub-attention network is representative of a single group member, interacting with all other group members in order to learn its preference score.

be insufficient for the task. Different aggregation strategies have been proposed such as average [BMR10, BF10], least misery [ARC+09], or maximum satisfaction [BC11]. In general, these aggregation strategies are also known as pre-defined strategies, where they first predict the scores across individuals for candidate items, and then aggregate those predicted scores of each member in a group via the strategies to obtain the group's preferences.

We argue that these existing aggregation strategies are not sufficient to model the complexity and dynamics of the group due to its inflexibility in adjusting the weights of members in the group. For example, user A may have higher impact weight than user B in a given group when the group makes decision on which movie to watch, but have lower weight than user B when the group makes decision on which restaurant to dine at. Recently, neural attention mechanism has been proposed as one of the most exciting advancements in deep learning [VTBE15, BCB15, CBS+15]. The concept of attention is that when people visually access an object, we tend to focus on (pay attention to) certain important parts of the object instead of the whole object in order to come up with a response. Our model adopts the attention mechanism to learn attentive and dynamic weight of each user, in which higher weights indicate the corresponding users are more important; thus, their contributions are more important for the group's final decision.

Our Method. We focus on designing novel and effective neural architectures for group recommendation under representation learning framework. Specifically, we propose

novel representation learning technique for learning-to-rank group-of-users and item pairs. Under the representation learning paradigm, we are able to model group representation via an end-to-end representation learning.

Let $\mathbf{u}_m$ and $\mathbf{v}_j$ be the embedding vector for user m and item j, respectively. We aim to obtain an embedding vector $\mathbf{g}_i$ for each group to estimate the group's preference on the item j. Formally, it can be defined as:

$$\mathbf{g}_i = f_a(\{\mathbf{u}_m\}_{m \in I_i}, \mathbf{v}_j) \tag{7.1}$$

in which $\mathbf{g}_i$ denotes the representation learning of group i which represents its preference on item j; I_i contains the user indexes of group i; and f_a is the aggregation function to be specified. In MoSAN, our model architecture presenting group embedding consists of two levels: 1) Sub-Attention Network Module, and 2) the Medley of Sub-Attention Networks.

We next elaborate the two-level structure of MoSAN.

Sub-Attention Network Module. For a given group $\mathbf{g}_i$, we create n attention sub-networks. Each attention sub-network l takes the user-context[1] vector $\mathbf{c}_l$ (filled orange circle) and the set of member user-latent vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{l-1}, \mathbf{u}_{l+1}, \ldots, \mathbf{u}_n\}$ (solid blue circles) as input, and then returns the attention weight $\alpha(i, l, m)$ of each user m $(m \neq l)$ (dashed blue circles) of sub-network l in group $\mathbf{g}_i$.

Intuitively, each attention sub-network models the interactions between each member l and the rest of the group to learn the preference votes of user l for other members in the group. Recall that this satisfies our key intuition and desiderata set out in the exposition of this chapter. Given a user-context vector $\mathbf{c}_l$ and a set of user-latent vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\} \setminus \{\mathbf{u}_l\}$, we use a two-layer network to compute the attention score $a(i, l, m)$ as:

$$a(i, l, m) = \mathbf{w}^T \phi(\mathbf{W}_c \mathbf{c}_l + \mathbf{W}_u \mathbf{u}_{m, m \neq l} + \mathbf{b}) + d, \quad l, m = \overline{1, n} \tag{7.2}$$

in which the matrices $\mathbf{W}_c$ and $\mathbf{W}_u$ are weight matrices of the attention network that convert user-context embedding and user-latent embedding to hidden layer respectively, and $\mathbf{b}$ is the bias vector of the hidden layer; the weight vector $\mathbf{w}$ and bias d are the parameters of the second layer that we use to project the hidden layer to the score $a(i, l, m)$. We simply use a linear $\phi(x) = x$ as an activation function, but one can also use other functions like a ReLU function $\phi(x) = \max(0, x)$.

We normalize $a(i, l, m)$ using the Softmax function to obtain the final attention weights:

[1]Note that the user-context vector is mainly used to differentiate the ownership of the current attention sub-network.

$$\alpha(i,l,m) = \frac{\exp(a(i,l,m))}{\sum_{m=1,m\neq l}^{n}\exp(a(i,l,m))} \tag{7.3}$$

Finally, the output of each attention sub-network l is calculated as the weighted sum $\mathbf{g}_{i,l} = \sum \alpha(i,l,m)\mathbf{u}_m$ (solid green square), which represents the group given the user-context $\mathbf{c}_l$. As such, $\mathbf{g}_{i,l}$ can be treated as the l-th representation of the group i, which captures the decisions of each member given that these group members' decisions are influenced by the decision of member l.

Medley of Sub-Attention Networks. The final representation of the group $\mathbf{g}_i$ is then computed as the summation $\mathbf{g}_i = \sum_l \mathbf{g}_{i,l}$ (filled green square). This final representation can be interpreted as the feature representation of the group of users, which can be easily matched with the item embedding to determine the recommendation score. More concretely, after we obtain the representation of the group i, the predicted score $\hat{R}_{ij}$ for group i and item j is computed as follows:

$$\begin{aligned}\hat{R}_{ij} &= \mathbf{g}_i^T\mathbf{v}_j \\ &= \left(\sum_l \mathbf{g}_{i,l}\right)^T\mathbf{v}_j \\ &= \left(\sum_l\sum_{m\neq l}\alpha(i,l,m)\mathbf{u}_m\right)^T\mathbf{v}_j,\end{aligned} \tag{7.4}$$

in which $\mathbf{v}_j$ is the item latent vector for item j.

One may argue that such a simple summation could fail to consider the different number of group members, e.g., the summation of $\mathbf{g}_i$ over a 3-member group tends to be much smaller than a 10-member group. As a result, according to Eq. (7.4), the group preference score on item v of the 3-member group will be much less than that of the 10-member group due to the huge number of members, even if the 10-member group may not like this item v as much as the 3-member group. However, since a large group (i.e., 10-member group) has large representation, it produces larger loss value in terms of absolute value. Hence, its gradient tend to be larger, and their group members representations are updated more during the training. If we choose appropriate learning rate, the performance will not much be affected. Moreover, in the testing phrase, we only consider and compare items to a group, so we do not have to concern about the large/small group problem. Usually the number of members in a group is small (i.e., less than 20 members), so this concern can be negligible. The normalization can also be taken into account at this layer; however, it will also be considered as an average

pooling of the sub-attention networks, which makes the distinguish observation between each sub-attention network becomes less important.

Alternatively, one could consider an additional attentive aggregation at the second layer. However, our preliminary experiments showed this yielded no improvements in performance. This is intuitive, as our model already *'reasons'* over group members using the sub-attention network modules at earlier layers. Specifically, regarding the summation, the difference of each member is already captured by the sub-attention network. This is because each sub-attention network already captures the relationship of one member against the group, which aligns well with this intuition. As such, an additional attention layer did not provide any benefit to the overall network structure.

7.2.3 Optimization and Learning

Objective function. Our proposed MoSAN leverages BPR [RFGS09] pair-wise learning objective to optimize the pair-wise ranking between the positive and negative items. The objective function can be rewritten as follows:

$$\arg\min_{\Theta} \sum_{(i,j,k)\in\mathcal{D}_s} -\ln\sigma\Bigg\{\Bigg(\sum_{l}\sum_{m\neq l}\alpha(i,l,m)\mathbf{u}_m\Bigg)^T\mathbf{v}_j - \Bigg(\sum_{l}\sum_{m\neq l}\alpha(i,l,m)\mathbf{u}_m\Bigg)^T\mathbf{v}_k\Bigg\} + \lambda(\|\Theta\|^2), \quad (7.5)$$

in which Θ represents the model parameters; and $\alpha(i,l,m)$ is the weight of user l votes for user m in group i. Subsequently, the model can be trained end-to-end with an optimizer such as Adaptive Moment Estimation (Adam).

Learning details. We next describe some details for learning our proposed model which are useful to replicate.

Mini-batch training. We perform mini-batch training. Each mini-batch contains interactions of group members and the item adopted by the group. Specifically, we shuffle all the observed interactions, and then sample a mini-batch of those observed ones. Lastly, we form the training instances by sampling a fixed number of negative instances for each observed interaction.

Dropout. We also employ dropout [SHK+14] to improve our proposed model's performance. Specifically, we drop with the dropout rate of ρ on the first layer of our

neural sub-attention network. We empirically found that applying dropout on the hidden layer of the neural attention network did boost our generalization performance. On a side note, we only apply dropout during the training phrase and disable it during the testing phase.

7.3 Experiments

In this section, we report experimental results of comparing MoSAN and seven state-of-the-art baseline techniques on four datasets. We also report the learned attention weights to evaluate the dominant users in group decision making. Such results will offer explanation for group recommendation result, which is an additional advantage of MoSAN. In general, our experiments aim to answer the following research questions (RQ):

- **RQ1:** How does MoSAN perform as compared to existing state-of-the-art methods?
- **RQ2:** Are the dynamic weights learned by MoSAN more preferable than the fixed weights learned by existing methods? How effective is our attention model? What is the advantage of our attention model over the vanilla attentive aggregation baseline?
- **RQ3:** How does MoSAN perform with different group sizes?

7.3.1 Experimental Settings

Datasets. We conduct extensive experiments on four real-world datasets. The first dataset is from an event-based social network (EBSN), Plancast,[2] which is used in [LHT+12]. Plancast allows users to directly follow the event calendars of other users. An event in Plancast consists of a user group and a venue. We therefore consider an event a group, and each user in the event a group member. Members in the group will select a venue (the candidate item) to host the event. Our goal is to recommend a venue for the group event.

Our second dataset is the dataset crawled from the EBSN Meetup,[3] which is from the work [PLCZ16]. We select the NYC data, which contains events held in New York City, as the dataset for our experiments. Similar to Plancast, we aim to recommend a venue for a given group to host an event. The statistics of this dataset is different from which reported in [LTYL12] due to the difference in the period of crawling.

[2]https://www.plancast.com
[3]https://www.meetup.com/

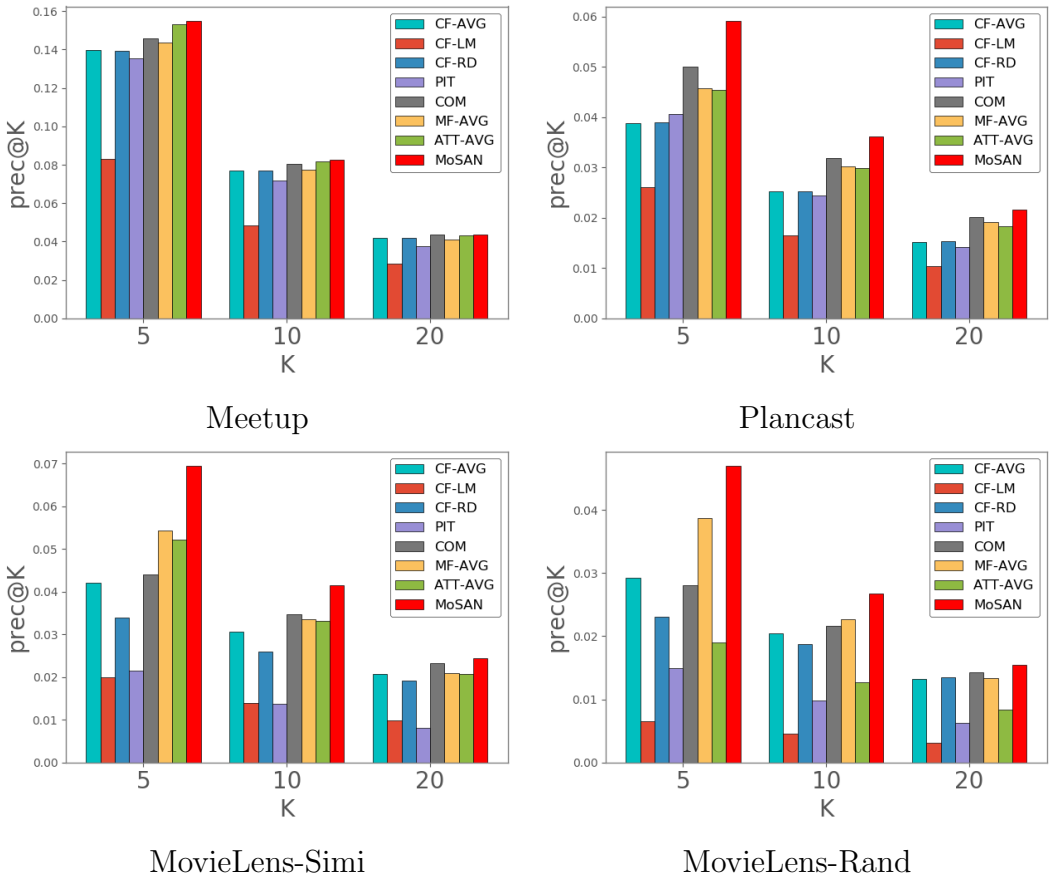


Figure 7.3: Performance of Group Recommendation Methods in terms of prec@K ($p < 0.0001$) (*Best viewed in color*).

The final two datasets are obtained from the MovieLens 1M Data.[4] The MovieLens 1M Data contains one million movie ratings from over 6,000 users on approximately 4,000 movies. Following the approach in [BMR10], we extract from the MovieLens 1M Data two datasets: MovieLens-Simi and MovieLens-Rand. MovieLens-Simi contains groups with high user-user similarity. We select top 33% in terms of similarity of all possible pairs to form groups from there. MovieLens-Rand contains groups that are formed without the restrictions above. For a given group in both cases, if every member gives 4 stars or above to a movie, we assume that the movie is adopted by the group. Users in the MovieLens-Simi data are assigned into the same group when they have high inner group similarity, while users in the MovieLens-Rand data are grouped randomly. MovieLens-Simi and MovieLens-Rand groups thereby resemble two typical real life situations: groups can either include people with similar preferences, or form between unrelated people. For

[4]https://grouplens.org/datasets/movielens/

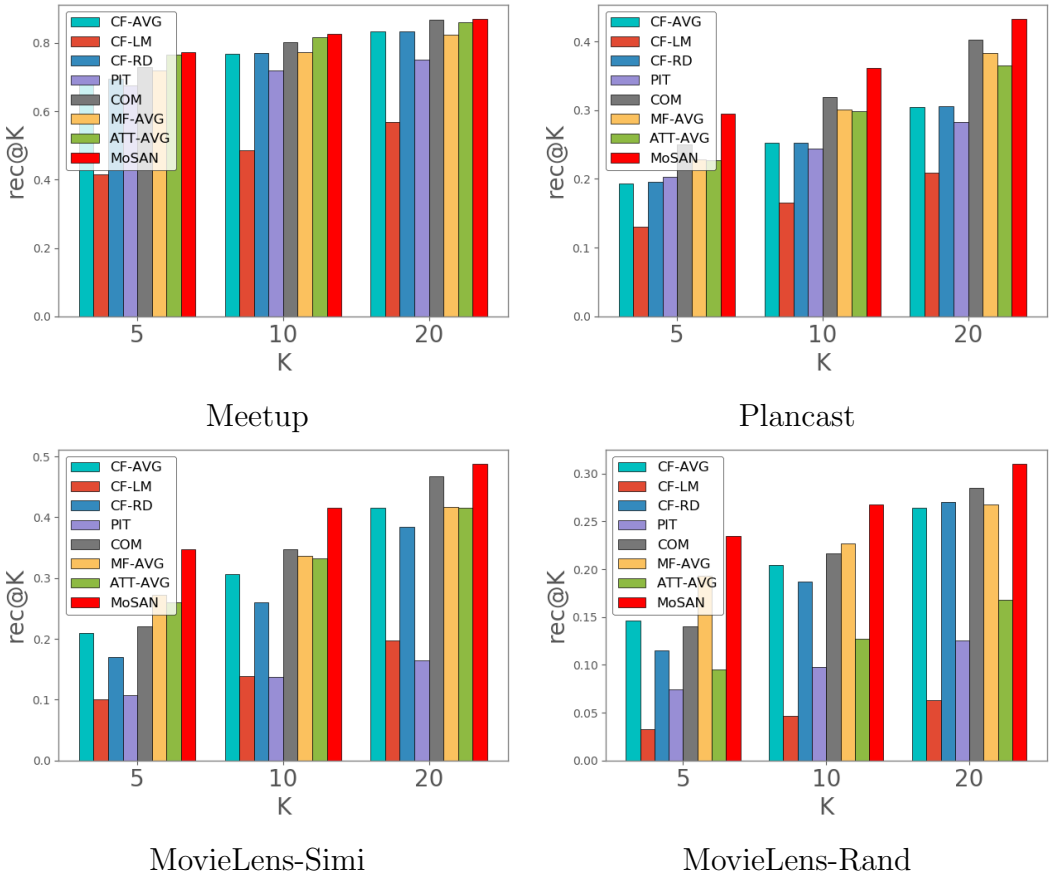


Figure 7.4: Performance of Group Recommendation Methods in terms of rec@K ($p < 0.0001$) (*Best viewed in color*).

example, a group of close friends has high inner group similarity, whereas people on the same bus can be considered a random group.

Table 7.1 reports descriptive statistics of the four datasets. We randomly split each dataset into training, tuning and testing data with the ratio of 70%, 10% and 20% respectively. Note that the groups in our setting are ad-hoc, i.e., it is possible that a group appears only in test data, but not in training data.

Evaluation Metrics. Following previous work [BMR10, LTYL12, PLCZ16, YCL14], we evaluate model performance using three widely used evaluation metrics: *precision* (*prec@K*), *recall* (*rec@K*), and normalized discounted cumulative gain (NDCG) (*ndcg@K*). Here K is the number of recommendations. We evaluate recommendation accuracy with $K = \{5, 10, 20\}$. *precision@K* is the fraction of top-K recommendations selected by the group, while *recall@K* is the fraction of relevant items (true items) that have been retrieved in the top K relevant items. We average the *precision@K* and *recall@K* values

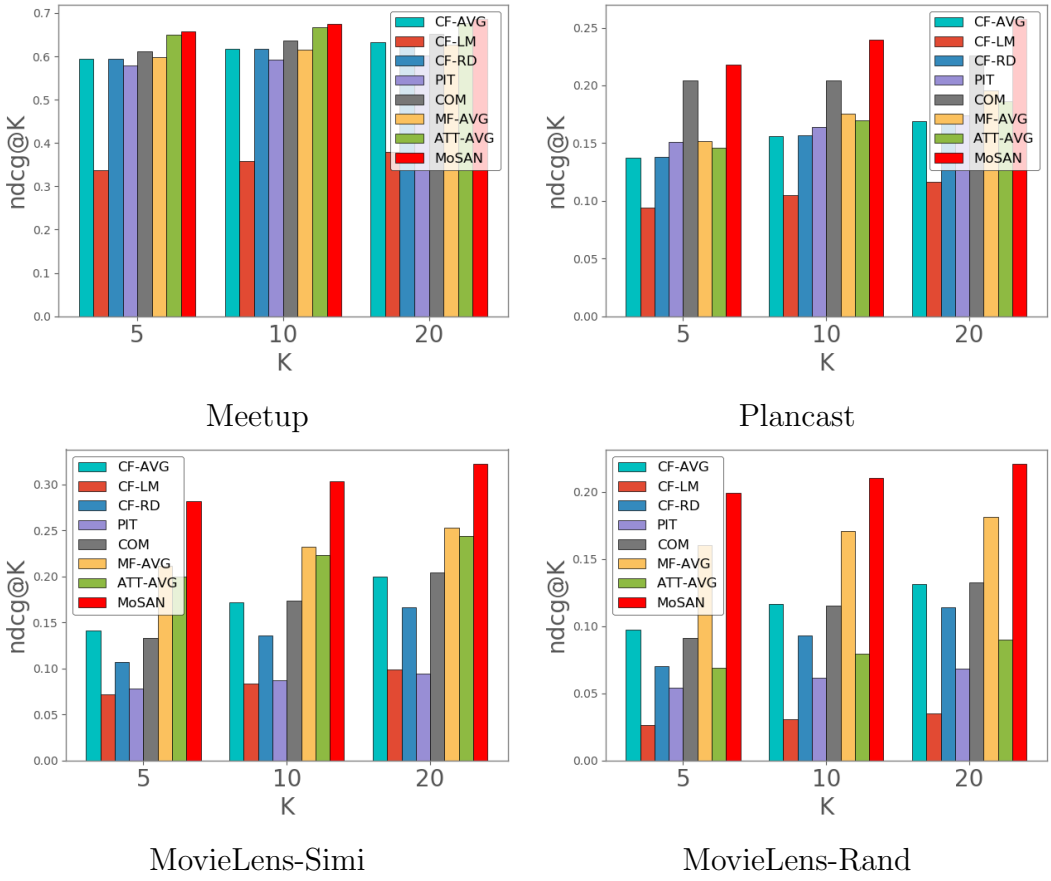


Figure 7.5: Performance of Group Recommendation Methods in terms of ndcg@K ($p < 0.0001$) (*Best viewed in color*).

across all testing groups to calculate *prec@K* and *rec@K*. We also use the *NDCG* metric to evaluate the rankings of true items in the recommendation list. We average the *NDCG* values across all testing groups to obtain the *ndcg@K* metric. For all of the three metrics, a larger metric value indicates better recommendations.

Compared Baselines. We compare with seven state-of-the-art baselines in our experiments: CF-AVG, CF-LM, CF-RD [ARC+09], PIT [LTYL12], COM [YCL14], MF-AVG and ATT-AVG.

- **User-based CF (CF-AVG, CF-LM, CF-RD)** [ARC+09]: The baselines are standard user-based collaborative filtering methods by integrating pre-defined aggregation strategies which are averaging strategy (CF-AVG), least-misery strategy (CF-LM) and relevance and disagreement strategy (CF-RD).

Dataset	Plancast	Meetup	MovieLens -Simi	MovieLens -Rand
Total Users	41,065	42,747	5,759	5,802
Total Groups	25,447	13,390	29,975	54,969
Total Items	13,514	2,705	2,667	3,413
Avg. Group Size	12.01	16.66	5.00	5.00
Avg. Record/User	7.44	5.22	26.03	47.37
Avg. Record/Item	1.88	4.95	11.24	16.11

Table 7.1: Dataset Statistics

- **Personal impact topic model (PIT)** [LTYL12]: PIT is an author-topic model. Assuming that each user has an impact weight that represents the influence of the user to the final decision of the group, PIT chooses a user with a relatively large impact score as the group's representative. The selected user then chooses a topic based on her preference, and then the topic generates a recommended item for the group.

- **Consensus model (COM)** [YCL14]: COM relies on two assumptions: (i) the personal impacts are topic-dependent, and (ii) both the group's topic preferences and individuals' preferences influence the final group decision.

- **Average Matrix Factorization (MF-AVG)**: This baseline is a simplified version of MoSAN and considers the average embedding of all users in the group. All users are weighted equally. We represent a group as $g = \sum_i w_i u_i$ where $w_i = \frac{1}{n}$, and we optimize the BPR objective to predict group recommendation scores.

- **Attentive Aggregation (ATT-AVG)**: This baseline is also a simplified version of MoSAN, and represents a group embedding using a vanilla attentive aggregation over all the user embeddings in the group. The user embeddings are optimized with the BPR objective function.

On a side note, [HCX+14] recently proposed a deep-architecture model called DLGR that learns high-level comprehensive features of group preferences to avoid the vulnerability of the data. Similar to the work [SYMM11], DLGR only focuses on pre-defined groups instead of ad-hoc groups, and thus cannot be applied to our setting. Additionally, a recent work exploits neural attention for a group recommendation setup called AGREE [CHM+18]. However, AGREE is different from our model as AGREE also focuses on pre-defined groups where it requires additional group preference representation information as one component to learn the final representation of a group. In addition, if we separate the group preference representation from AGREE, the architecture becomes completely different from the original design and it similarly becomes one of our baselines

Model	Meetup		Plancast		MovieLens -Simi		MovieLens -Rand	
	rec@5	*ndcg@5*	*rec@5*	*ndcg@5*	*rec@5*	*ndcg@5*	*rec@5*	*ndcg@5*
MF-AVG	0.851962	0.632753	0.467176	0.211513	0.321079	0.191142	0.530135	0.274007
ATT-AVG	0.883432	0.681998	0.439959	0.200346	0.221269	0.099581	0.520669	0.263710
MoSAN	**0.888513**	**0.689149**	**0.502620**	**0.270515**	**0.360627**	**0.230099**	**0.578485**	**0.338959**
p-value	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

Table 7.2: Performance comparison between MF-AVG and ATT-AVG (Ablation study) on four datasets. Results show that our proposed attention mechanism is significantly better than a standard attention-based aggregation.

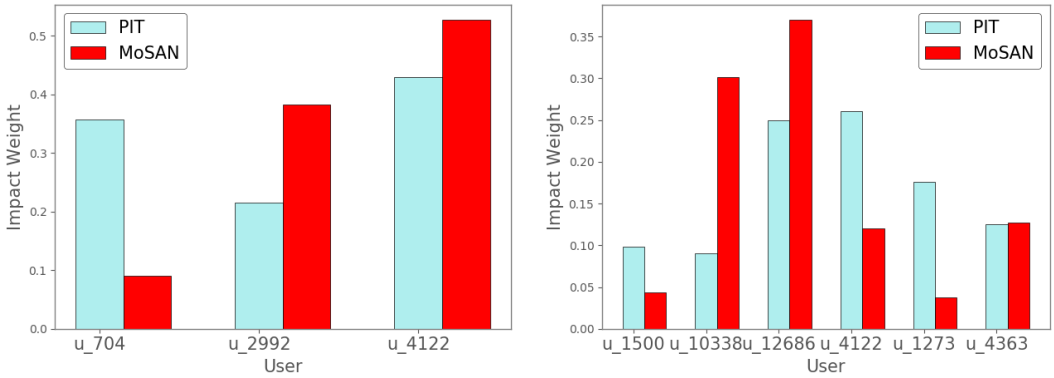
above called Attentive Aggregation (ATT-AVG) model. Hence, their framework cannot be directly applied to our setting. Moreover, our overall intuition and framework (usage of sub-attention and user-user interactions) significantly distinguish our work from theirs. Therefore, we acknowledge DLGR [HCX+14] and AGREE [CHM+18], but we do not compare to them in this chapter.

Hyperparameter Settings. For PIT and COM, we tuned the number of topics and kept other hyperparameters as default. With regard to the MF-AVG/ATT-AVG and MoSAN models, we first randomly initialize the parameters using the Gaussian distribution with mean of 0 and standard deviation of 0.05, and then use Adaptive Moment Estimation (Adam) to optimize our objective functions. We also tested the batch size of [128, 256, 512], the learning rate of [0.001, 0.005, 0.01, 0.05, 0.1], and different regularizers of [0.001, 0.01, 0.1, 0]. We empirically set the embedding size of MF-AVG/ATT-AVG and MoSAN with the dimension of 50. We obtain the optimal setting with the batch size of 256, learning rate of 0.001, and regularizers of 0.01. We randomly set dropout rate $\rho = 0.5$ and sample 3 negative items for each training instance.

7.3.2 Overall Performance Comparison

This subsection compares the recommendation results from MoSAN to those from the baseline models (**RQ1**). Figure 7.3, Figure 7.4 and Figure 7.5 report the *prec@K*, *rec@K* and *ndcg@K* values for the four datasets with $K = \{5, 10, 20\}$. We observe from the three figures that:

- MoSAN consistently achieves the best performance across all methods, including score-aggregation approaches (CF-AVG, CF-LM, CF-RD) and probabilistic model approaches (PIT, COM).
- Although the group information of the two MovieLens datasets is generated manually instead of already observed as Meetup and Plancast, our model can still be able to show the recommendation flexibility in adopting to randomness datasets.



(a) Group A (b) Group B

Figure 7.6: Attention Weights Learned by PIT and MoSAN.

- MoSAN and MF-AVG models produce good results in comparison to the previous state-of-the-art probabilistic models. MF-AVG performs better than PIT and COM on the MovieLens-Simi and MovieLens-Rand datasets in terms of *ndcg@K*, but not on the Meetup and Plancast datasets. One explanation is that the simplistic setup of MF-AVG cannot model the complexity of real life group interactions.

We observe that there is no obvious winner among the baseline solutions. For each dataset, we define the baseline that has the greatest performance among the proposed ones as *the best baseline* method. The *prec@5* metric values show that MoSAN outperforms the best baseline method significantly by 0.94%, 17.91%, 27.71%, 21.36% on Meetup, Plancast, MovieLens-Simi and MovieLens-Rand, respectively. We observe the same improvements for *rec@5*. In general, MoSAN's recommendations are consistently better than the baseline methods', with the p-value less than 0.0001 for all the results and thus statistically significant.

It is noticeable that PIT's performance is not comparable with those of other baseline models. One possible reason is that as a Meetup or Plancast group usually has a large number of participants, many of whom only join a few groups and thus have very limited historical data. Hence, the user impacts learned by PIT for such participants are not reliable. Another possible reason for PIT's poor performance is that the assumptions underlying PIT do not hold in the context of MovieLens data: since MovieLens users select movies independently from one another, there is no representative user in a MovieLens group.

We also observe that MoSAN does not outperform the baselines models on the Meetup dataset as significantly as it does on the other three datasets. One explanation is that since a Meetup group often has few venue options, and group members tend to choose the

place they are most familiar with, making it relatively easy to recommend the venue to the group [PLCZ16]. While Meetup users form a big group before hosting an event and choosing the venue, Plancast allow users to follow other users' event calendars and choose to participate in existing events. Plancast groups therefore tend to be more diverse than Meetup groups, and Plancast event venues are not as easily predicted as Meetup event venues.

In addition, one may concern the ability of MoSAN to deal with large groups. However, it should not be an issue as we observe that typically most of groups have fewer than 10 users. For larger groups, we can always reduce the number of users to a smaller number (e.g., 20 users). Then, we can use our model to firstly compute the attention weights of users and remove the users with very small weights. We can always perform this step because in reality, the number of group leaders/experts who participates in making the final decision for the group is always very small.

In general, MoSAN achieves remarkable recommendation results on the variety of datasets consistently. Our experiments show the flexibility of MoSAN in making group recommendation given different data types.

7.3.3 The Role of Attention Mechanism

In this section, we study the role of attention mechanism (**RQ2**) through various ablation studies and experiments.

Ablation Study. We further conduct paired t-tests on the performances of MoSAN against MF-AVG and ATT-AVG models to verify that MoSAN's improvements over MF-AVG and ATT-AVG are statistically significant at the five percent significance level. While MoSAN employs a user-user attention mechanism to calculate the weights, MF-AVG assigns a normalized constant weight to each group member. Similarly, ATT-AVG is a simple attentive pooling over all users. Even though it also adopts an attention mechanism, it does not consider user-user interactions. We conduct paired t-tests on the performance metrics of top-K recommended lists ($K \in [10, 100]$) for MoSAN, MF-AVG and ATT-AVG to see whether MoSAN statistically significantly outperforms MF-AVG and ATT-AVG. We also observe that ATT-AVG does not always outperform MF-AVG, signifying that the standard attention mechanism is insufficient.

Table 7.2 compares the performances of MoSAN and MF-AVG. We observe that the mean pooling strategy of MF-AVG always performs worse than the attention mechanism of MoSAN. The good performance of MF-AVG on MovieLens-Rand data is expected because MovieLens-Rand groups satisfy MF-AVG assumptions: randomly-grouped users in MovieLens-Rand data tend to equally contribute to the groups' final decisions. The

Number of users K to be removed	Meetup		Plancast	
	MoSAN	PIT	MoSAN	PIT
$K = 1$	0.643932	0.56832	0.197189	0.14193
$K = 3$	0.593263	0.56690	0.118760	0.14085
$K = 5$	0.500849	0.56424	0.081018	0.14013
$K = 7$	0.377586	0.56238	0.058164	0.13954
$K = 9$	0.284848	0.56163	0.046473	0.13950

Table 7.3: Performance comparison between MoSAN and PIT on Meetup and Plancast datasets in terms of ndcg@5 by removing top-K high weight users.

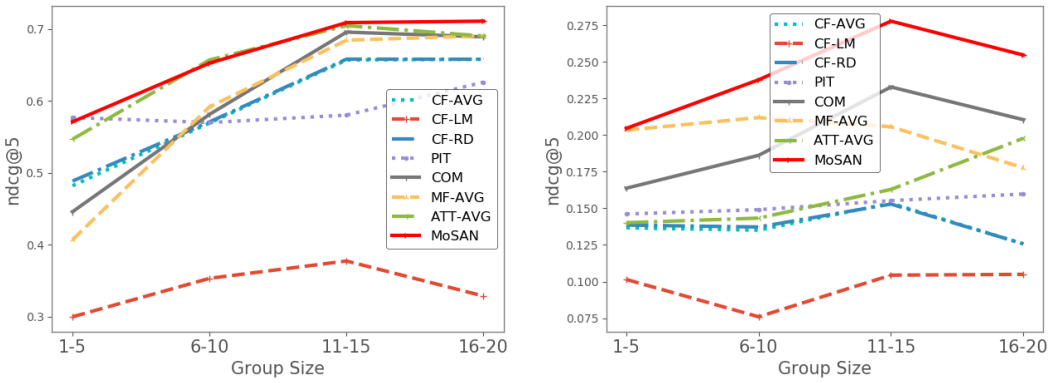
reported p-values are nominal for all tests, indicating that MoSAN's performances are statistically much stronger than MF-AVG's performances.

Attention Weight Visualization. As noted previously, an advantage of MoSAN is that the method allows us to calculate the attention weight values for explanation of group recommendation results. Figure 7.6 visualizes the attention weights learned by MoSAN and PIT for two randomly-chosen groups from our experiments. We compare with the weights learned by PIT because similar to MoSAN, PIT is able to learn the personal impact weight for each user [LTYL12]. Since COM does not learn the personal impact weight for each user [YCL14], we do not compare MoSAN with COM here.

Figure 7.6(a) and Figure 7.6(b) show user attention weights learned by PIT and MoSAN of two randomly-chosen groups that share a user no. 4122 ("u_4122"). Figure 7.6(a) reports the personal impact weights of three users in the first group (group A). According to both models, the user no. 4122 has the highest weights in group A and therefore dominates group A's decision making. Figure 7.6(b), however, shows that PIT continues to assume that user no. 4122 is the most influential user in group B, whereas MoSAN is able to detect other users who play more important roles in group B's decision making than user no. 4122 does. While PIT's personal impact parameter cannot differentiate the roles of one user in different groups and thus may fail to recognize influential members in a group, the attention mechanism of MoSAN can capture the dynamic user impacts across groups in group decision making.

The importance of high weight users. In addition, we also conduct another ablation experiment to analyze the importance of users by comparing the recommendation results of MoSAN and PIT with and without high weight users on Meetup and Plancast datasets. We rank users by their impact weight in decreasing order and remove top-K users from group recommendation, and report the change in MoSAN's and PIT's performance, in which $K \in \{1, 3, 5, 7, 9\}$.

Table 7.3 shows the performance comparison between MoSAN and PIT in terms of *ndcg@5* by removing top-K high weight users. It is noticeable that the performance



(a) Meetup (b) Plancast

Figure 7.7: Performance on Different Group Sizes (*Best viewed in color*).

of PIT is stable as the number of removed users increases, whereas the performance of MoSAN drops dramatically. Specifically, the average performance drop of MoSAN for removing every additional two users is 18.15% for Meetup dataset and 29.97% for Plancast dataset, while the performance of PIT is nearly unchanged. The decreasing performance of MoSAN shows that the removed users are important users in the group while the top-ranked users in PIT are not really important users; and thus, the attention mechanism in MoSAN can really capture the dynamic user impacts better than PIT. It reflects the effectiveness of MoSAN in analyzing and weighting high/low impact users in different groups, which is critical for group recommendation systems. All in all, the ablation experiment of the importance of users shows the preciseness of MoSAN over PIT in learning the attentive weight for each user.

7.3.4 Model Performances for Different Group Sizes

To study the performance of each recommendation method on different group sizes (**RQ3**), we run the experiments for four levels of group size (1-5 members, 6-10 members, 11-15 members, and 16-20 members) using Meetup and Plancast groups. We keep the same setting as illustrated for all the models, and classify the groups into bins based on group size. Since the number of groups with more than 20 members is very small, we exclude these groups in this experiment. Figure 7.7 plots the resulting *rec@5* and *ndcg@5* curves. Note that since the group size of the MovieLens-Simi and MovieLens-Rand dataset is fixed, we do not study the different levels of group sizes on these two datasets.

Figure 7.7 shows that MoSAN achieves better performance than other baseline methods across different group sizes. We have the following observations: 1) MoSAN shows

clear improvements in recommendations for groups of larger sizes. MoSAN improves 0.59% and 2.93% on the Meetup dataset over the best baseline method for groups of 11-15 members and 16-20 members respectively, while these numbers are 19.30% and 20.97% for the Plancast dataset. This indicates the significance of MoSAN in addressing groups of larger sizes. 2) CF approaches often deliver good performance when the group size is small, as low diversity within a group facilitates smooth aggregations. As the number of members in the group increases, we need relatively complex methods such as probabilistic models or neural networks models to make adequate recommendations.

Chapter 8

Conclusions, Discussions, and Future Work

8.1 Conclusions

In this dissertation, we study the task of developing effective neural architectures by exploring various representation learning techniques for recommender systems. Specifically, Chapter 3, 4, 5 introduce new representation learning approaches for personalized recommendation, and Chapter 7 presents a novel representation learning approach for group recommendation.

In Chapter 3, we propose a novel W-MLC model, which exploits a series of metric learning that outperform standard metric learning recommenders. To the best of our knowledge, we are the first to explore metric learning chain approach in recommender systems. Moreover, we design multiple loss functions, i.e., pull and push loss, adaptive margin loss, and transformation loss, to incorporate different aspects of metric learning. We conduct extensive experiments on eight real world datasets and demonstrate the superiority of our proposed methods against various state-of-the-art recommendation methods. In addition, we evaluate the performance of W-MLC on three different tasks to indicate the generality of our proposed approach, which can be adopted for different applications.

In Chapter 4, we introduce a new effective and competent recommendation model called HyperML. To the best of our knowledge, HyperML is the first model to explore metric learning in hyperbolic space in recommender system. Additionally, we also introduce a distortion term, which is essential to control good representations in hyperbolic space. Through extensive experiments on ten datasets, we are able to demonstrate the effectiveness of HyperML over other baselines in Euclidean space, even state-of-the-art metric learning models such as CML or LRML. The promising results of HyperML may

inspire other future works to explore hyperbolic space in solving recommendation problems.

In Chapter 5, we introduce a new concept of exploring attention mechanism on hidden layers as a regularization effect. Through extensive experiments on six widely adopted benchmark datasets, we are able to demonstrate the effectiveness of our proposed DropRec, i.e. C-DropRec and S-DropRec, of going beyond shallow MLP compared to many recent strong multi-layered models. Indeed, DropRec achieves significantly and consistently competitive results, outperforming all the baselines by a wide margin from +6.07% to +33.41%.

In Chapter 6, we introduce an extended extensive experiment to give new insights of all the proposed representation learning techniques and the baselines in the context of personalized recommnedation. We indeed show interesting findings, insights and discussions of this one-off experiment. Moreover, we also conduct analyses on runtime of these methods to raise a concern on the practicality of the architectures.

In Chapter 7, we propose a new effective neural recommender for group recommendation. The major contributions of MoSAN are that the model not only dynamically learns different impact weights of each given user for different groups, but it also considers the interactions between users in the group. Thus, MoSAN is able to better model the group decision making process. In addition, MoSAN can tell the relative importance of users in a group, and thus make explainable recommendation possible. We conduct extensive experiments on four real-world datasets, demonstrating that MoSAN is capable of outperforming a myraid of strong state-of-the-art baselines.

After all, we summarize the various aspects of representation learning techniques for recommendations in this dissertation. Firstly, we discover that the MF based representation learning still remains as the strong baselines, in which several works also recognize this similar observations [RZK19, RKZA20]. Thus, it is fair to say that MF based methods, especially MF-BPR [RFGS09], should be *required* as a default group of baselines to be compared in the personalized recommendation task. These methods improve representation learning from the aspect of dot product. Moreover, BPR [RFGS09] is also acknowledged to be a strong pairwise learning objective. Secondly, we observe the better peformance of metric learning representation techniques in comparison to dot product techniques. The successful of metric learning in recommendation task could possibly be because of the triangle inequality property as discussed. Hence, metric learning could be considered as a potential direction for future research of implicit feedback recommendation. This class of methods improves representation learning from the aspect of embedding space. Lastly, attention mechanisms [VSP+17] are also certified as a revolution in the deep learning community, in which we also observe their powerful representation in both personalized and group recommendation task throughout this dissertation. Therefore, these methods improve representation learning from the aspect of extracting and aggregating representations selectively with the strength of 'attention'.

8.2 Discussions

As we focus on building effective neural architectures, our methods could be potentially generalized and applied to various recommendation tasks. Indeed, the representation learning techniques introduced can be decomposed into different submodules for other tasks. For example, the transformation process in Chapter 3 could be possibly used for embedding behavior chains of users [WM18] or the personalized adaptive margin module could also be integrated into any hinge loss based functions [HYC+17]; the hyperbolic metric embedding technique in Chapter 4 could also be applied to other recommendation tasks that explore tree-like structure data or data that follows power-law distribution [FVTC+20]; the framework proposed in Chapter 5 could be incorporated into any other MLP-like architecture; and the attention submodules in Chapter 7 could be potentially applied to aggregate information of any groups of users. Therefore, our representation learning techniques in this dissertation are considered as general modules that could be generally incorporated into other tasks.

Moreover, there are recent papers focusing on analyzing results of deep recommendation models which caught some attentions of the research community recently [DBCJ19, DCJ19]. In particular, the authors revisit conventional methods (e.g., based on the nearest-neighbor heuristics), compare them to deep neural approaches, and observe that only a few neural methods can consistently better than existing learning-based techniques (e.g., MF-based or linear models). However, to my personal perspective, the limitation of recent methods is acceptable for two reasons: Firstly, many different recommendation algorithms of different types have been extensively proposed over the past 25 years. To this end, lots of them have already shown remarkable results. Thus, it is not easy to significantly improve the accuracy performance beyond these benchmarks, given that the existing recommenders nearly reach to the saturated point. Secondly, even though it is hard to create milestones in recommendation domain at the moment, we still need to make progress because we will never know about the future. It is possible that one of the directions that we are focusing right now could become a foundation of future advanced recommendation models. It is also the reason that motivates and inspires me to continue doing research in the field, although sometimes the results are not impressed as people expected.

8.3 Future Directions and Challenges

This section identifies several potential directions that are exciting for future work. While there might be no principled solution exists, many of these future directions remain as promising and challenging for recommendation domain.

8.3.1 Efficient Models for Hyperbolic Representation

This thesis demonstrates promising results of hyperbolic representation in recommender systems. However, the preliminary results only focus on exploring the notion of user-item interaction representations [VTTZ+20, CHW+19]. To this end, incorporating side information into hyperbolic space might be an interesting future direction. However, this is not trivial to integrate those extra information into the neural architecture if the data does not follow any tree-like structures. Therefore, this direction remains a highly technical and theoreotical challenge for innovation.[1]

In addition, there are also several ways to endow different subsets of Euclidean space with a hyperbolic metric, which leads to different models of hyperbolic space beside Poincaré ball model such as Lorentz model (Hyperboloid model) or Klein model. Although all these models of hyperbolic space are necessarily the same, they however have different coordinate systems that offer different advantages in computation. For instance, using the distance function in Lorentz model can help to avoid numerical instabilities that arise from the fraction in the Poincaré distance [NK18]. Therefore, exploring different hyperbolic models in recommendation would be interesting. Moreover, [GBH18b] has shown that adapting recurrent models to hyperbolic space is reasonable. Hence, it would also be interesting to investigate techniques to exploit recurrent or convolutional architectures in Hyperbolic space for different tasks.

On a side note, as discussed in Chapter 6, combining the Wasserstein distance with hyperbolic space is also worth exploring. As mentioned, one of the main challenges of this unified framework is to define the new projection in hyperbolic space to create a chain. Moreover, since the hyperbolic works [NK18, GBH18b, VTTZ+20] usually focus on representations in hyperbolic unit ball, it is also a challenge to consider the case when the chain jumps out of the unit ball. Therefore, even after sucessfully creating the chain, we need to explore the full Riemannian stochastic gradient descent (RSGD) for parameters update, as the approximate gradient algorithms will likely cause the new embeddings come out of the ball [NK18, VTTZ+20]. Besides, we also require new multi-objective loss functions to suit the hyperbolic properties. In general, these challenges are very interesting and could be explored as one of the future directions.

8.3.2 Evaluation Metrics and Scalability

In this dissertation, our researches mainly focus on improving the accuracy of the top-K recommendations through different evaluation metrics such as HR (Hit Ratio), NDCG (Normalized Discounted Cumulative Gain), Precision and Recall. However, in reality, user experience is far from only concerned with the top-K accuracy. To this

[1] http://hyperbolicdeeplearning.com/

end, we believe that it is critical to explore other evaluation metrics for recommendation to achieve high-quality items. Indeed, a good recommender should also consider other aspects such as diversity, usability, privacy, explainability and interpretability [ZC18]. On a side note, some recent works about group/bundle recommendation have incorporated other metrics to balance the quality and diversity [QMPT16, CC19, BZS+19, LRFS19].

Additionally, scalability is another important factor for building real-world recommender systems, especially in the era of big data [YHC+18, HKM18, CHW+19]. Thus, we need to further design effective representation models that are capable of not only dealing with large amount of data, but also extracting fast top-K recommendations for users in real-time. Therefore, there exists a few challenges such as dealing with streaming data (e.g., large volume of new incoming users and items), or controlling the computational efficiency of the model with the exponential growth of parameters.

8.3.3 Reinforcement Learning for Decision Making

Although the works that we presented in this thesis have achieved remarkable results on many recommendation tasks, we believe that it is essential to further investigate the ability of modelling sequential behaviour data in recommender systems. In typical real-world recommender systems, ranking items in a search session is usually considered as a multi-step decision-making problem [PYC+19, HDZ+18, ZXZ+18, TZH+19, ZZZ+18]. However, most of the proposed methods usually do not consider the *sequence*. As such, it is natural to explore reinforcement learning for recommendation, where the end goal is to learn effective representations throughout the process of making decisions of users. Nevertheless, traditional reinforcement learning methods would perform badly due to some special features of recommendation such as large discrete space of items or sparse rewards with different types (e.g., click, favourite, add to cart, purchase, etc.). As such, this becomes a challenge to apply reinforcement learning to solve recommendation problems. Several works have proposed reinforcement models pertaining this direction [SED+15, CHH+18, CBC+19, CYD+18, DESC15, ZZD+18, DQH+19, ZXD+19].

Appendix A

List of Publications

- **Lucas Vinh Tran**, Yi Tay, Shuai Zhang, Gao Cong, Xiaoli Li. HyperML: A Boosting Metric Learning Approach for Recommender Systems. *Proceedings of the 13th International Conference on Web Search and Data Mining* (**WSDM 2020**), Pages 609-617, Houston, TX, USA. (**Best Paper Award Runner-Up**)

- Shanshan Feng, **Lucas Vinh Tran**, Gao Cong, Lisi Chen, Jing Li and Fan Li. HME: A Hyperbolic Metric Embedding Approach for Next-POI Recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (**SIGIR 2020**). Xi'an, China.

- **Lucas Vinh Tran**, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong and Xiaoli Li. Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (**SIGIR 2019**). Pages 255-264, Paris, France.

- Shuai Zhang, Lina Yao, **Lucas Vinh Tran**, Aston Zhang, Yi Tay. Quaternion Collaborative Filtering for Recommendation. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (**IJCAI 2019**). Pages 4313-4319, Macao, China.

- Yi Tay, Shuai Zhang, Anh Tuan Luu, Siu Cheung Hui, Lina Yao, **Lucas Vinh Tran**. Holographic Factorization Machines for Recommendation. *Proceedings of the 33th AAAI Conference on Artificial Intelligence* (**AAAI 2019**). Pages 5143-5150, Honolulu, Hawaii, USA.

- **Lucas Vinh Tran**, Thanh Tran, Shanshan Feng, Gao Cong, Xiaoli Li. Wasserstein Distance based Metric Learning Chain for Recommender Systems. *Under Review.*

- **Lucas Vinh Tran**, Gao Cong, Xiaoli Li. DropRec: On Adaptive Architecture Dropout for Personalized Recommendation. *Under Review.*

References

[AC10] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 91–100, 2010.

[AGS05] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savare. Gradient flows in metric spaces and in the space of probability measures. 01 2005.

[ARC+09] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765, 2009.

[BC11] Ludovico Boratto and Salvatore Carta. State-of-the-art in group recommendation and new approaches for automatic identification of groups. In *Information Retrieval and Mining in Distributed Environments*, pages 1–20. 2011.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[BF10] Shlomo Berkovsky and Jill Freyne. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 111–118, 2010.

[BG19] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[BK16] Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. In *Proceedings of the Poster Track of the 10th ACM Conference on Recommender Systems (RecSys 2016), Boston, USA, September 17, 2016*, volume 1688 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[BMR10] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 119–126, 2010.

[Bon13] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Trans. Automat. Contr.*, 58(9):2217–2229, 2013.

[BS13] Pierre Baldi and Peter J. Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2814–2822, 2013.

[BZS+19] Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. Personalized bundle list recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 60–71. ACM, 2019.

[CAS16] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 191–198, 2016.

[CBC+19] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 456–464. ACM, 2019.

[CBS+15] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 577–585, 2015.

[CC17] Rose Catherine and William W. Cohen. Transnets: Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 288–296. ACM, 2017.

[CC19] Edgar Eduardo Ceh-Varela and Huiping Cao. Recommending packages of multi-criteria items to groups. In *2019 IEEE International Conference on Web Services, ICWS 2019, Milan, Italy, July 8-13, 2019*, pages 273–282. IEEE, 2019.

[CCD17] Benjamin Paul Chamberlain, James R. Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *CoRR*, abs/1705.10359, 2017.

[CD14] Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 685–693. JMLR.org, 2014.

[CDPB19] Hyunghoon Cho, Benjamin Demeo, Jian Peng, and Bonnie Berger. Large-margin classification in hyperbolic space. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1832–1840. PMLR, 2019.

[CHH+18] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. Reinforcement learning based recommender system using biclustering technique. *CoRR*, abs/1801.05532, 2018.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA, pages 539–546. IEEE Computer Society, 2005.

[CHM+18] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. Attentive group recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 645–654, New York, NY, USA, 2018. ACM.

[CHW+19] Benjamin Paul Chamberlain, Stephen R. Hardwick, David R. Wardrope, Fabon Dzogang, Fabio Daolio, and Saúl Vargas. Scalable hyperbolic recommender systems. *CoRR*, abs/1902.08648, 2019.

[CKH+16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 7–10. ACM, 2016.

[CM13] Lucas Augusto Montalvão Costa Carvalho and Hendrik Teixeira Macedo. Users' satisfaction in recommendation systems for groups: an approach based on noncooperative games. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 951–958, 2013.

[CMTJ12] Shuo Chen, Joshua L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 714–722. ACM, 2012.

[CT17] Wei-Ta Chu and Ya-Lun Tsai. A hybrid recommendation system considering visual information for predicting favorite restaurants. *World Wide Web*, 20(6):1313–1331, 2017.

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2292–2300, 2013.

[CvMG+14] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.

[CWTY19] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. Collaborative similarity embedding for recommender systems. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2637–2643. ACM, 2019.

[CXWS12] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[CYD+18] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1187–1196. ACM, 2018.

[CZH+17] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 335–344, 2017.

[CZJC18] Jin Yao Chin, Kaiqi Zhao, Shafiq R. Joty, and Gao Cong. ANR: aspect-based neural recommender. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 147–156. ACM, 2018.

[DBCJ19] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *CoRR*, abs/1911.07698, 2019.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.

[DCJ19] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 101–109. ACM, 2019.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[DESC15] Gabriel Dulac-Arnold, Richard Evans, Peter Sunehag, and Ben Coppin. Reinforcement learning in large discrete action spaces. *CoRR*, abs/1512.07679, 2015.

[DFC+18] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. Hyperspherical variational auto-encoders. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 856–865. AUAI Press, 2018.

[DHS11] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

[DLZ17] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 152–160. ACM, 2017.

[Doe16] Carl Doersch. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908, 2016.

[DQH+19] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. Reinforced negative sampling for recommendation with exposure data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2230–2236. ijcai.org, 2019.

[DR15] Gintare Karolina Dziugaite and Daniel M. Roy. Neural network matrix factorization. *CoRR*, abs/1511.06443, 2015.

[DSN+18] Bhuwan Dhingra, Christopher J. Shallue, Mohammad Norouzi, Andrew M. Dai, and George E. Dahl. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing, TextGraphs@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 6, 2018*, pages 59–69. Association for Computational Linguistics, 2018.

[dSTXZ16] Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016.

[ESF18] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 515–524. ACM, 2018.

[ESH15] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 278–288. ACM, 2015.

[FCAC17] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. Poi2vec: Geographical latent representation for predicting future visitors. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 102–108. AAAI Press, 2017.

[FLZ+15] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2069–2075. AAAI Press, 2015.

[FML+19] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 417–426. ACM, 2019.

[FMS19] Charlie Frogner, Farzaneh Mirzazadeh, and Justin Solomon. Learning embeddings into entropic wasserstein spaces. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[FVTC+20] Shanshan Feng, Lucas Vinh Tran, Gao Cong, Lisi Chen, Jing Li, and Fan Li. Hme: A hyperbolic metric embedding approach for next-poi recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020, Xi'an, China, July 26-31, 2020*, 2020.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[GBH18a] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1632–1641. PMLR, 2018.

[GBH18b] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*

31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada, pages 5350–5360, 2018.

[GBR+06] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 513–520. MIT Press, 2006.

[GCB+19] Aude Genevay, Lénaïc Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1574–1583. PMLR, 2019.

[GDM+19] Çaglar Gülçehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter W. Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[GH12] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13:307–361, 2012.

[GLRW13] Jagadeesh Gorla, Neal Lathia, Stephen Robertson, and Jun Wang. Probabilistic group recommendation via information matching. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 495–504, 2013.

[GLT19] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention, please! A critical review of neural attention models in natural language processing. *CoRR*, abs/1902.02181, 2019.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6645–6649. IEEE, 2013.

[GPC18] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617. PMLR, 2018.

[GS84] Clark R. Givens and R. M. Shortt. A class of wasserstein metrics for probability distributions. 1984.

[GSGR19] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[GZTY14] Guibing Guo, Jie Zhang, Daniel Thalmann, and Neil Yorke-Smith. ETAF: an extended trust antecedents framework for trust prediction. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2014, Beijing, China, August 17-20, 2014*, pages 540–547. IEEE Computer Society, 2014.

[HC17] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 355–364. ACM, 2017.

[HCX+14] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Wei Cao. Deep modeling of group preferences for group-based recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1861–1867, 2014.

[HDW+18] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2227–2233. ijcai.org, 2018.

[HDW+20] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 639–648. ACM, 2020.

[HDZ+18] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 368–377. ACM, 2018.

[HHS+18] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. NAIS: neural attentive item similarity model for recommendation. *IEEE Trans. Knowl. Data Eng.*, 30(12):2354–2366, 2018.

[HKBT16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[HKM18] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation: A scalable method for modeling sequential behavior. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5264–5268. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 263–272. IEEE Computer Society, 2008.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[HM16a] Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM, 2016.

[HM16b] Ruining He and Julian J. McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 144–150. AAAI Press, 2016.

[HM18] Drew A. Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[HQKT16] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 241–248. ACM, 2016.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSK+12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[HYC+17] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 193–201. ACM, 2017.

[HZKC16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 549–558. ACM, 2016.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[JL17] Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 306–310. ACM, 2017.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[KBV09] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[KL51] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[KNK13] Santosh Kabbur, Xia Ning, and George Karypis. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 659–667, New York, NY, USA, 2013. Association for Computing Machinery.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 426–434. ACM, 2008.

[KPK+10] Dmitri V. Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *CoRR*, abs/1006.5169, 2010.

[KPO+16] Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 233–240. ACM, 2016.

[KS10] Mohammad Khoshneshin and W. Nick Street. Collaborative filtering via euclidean embedding. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 87–94. ACM, 2010.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[KTW+12] Dor Kedem, Stephen Tyree, Kilian Q. Weinberger, Fei Sha, and Gert R. G. Lanckriet. Non-linear metric learning. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 2582–2590, 2012.

[KW17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[KWM18] Wang-Cheng Kang, Mengting Wan, and Julian J. McAuley. Recommendation through mixtures of heterogeneous item relationships. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1143–1152. ACM, 2018.

[LCH+06] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

[LHT+12] Xingjie Liu, Qi He, Yuanyuan Tian, Wang-Chien Lee, John McPherson, and Jiawei Han. Event-based social networks: linking the online and offline social worlds. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1032–1040, 2012.

[LKF15] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 811–820, New York, NY, USA, 2015. Association for Computing Machinery.

[LKHJ18] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 689–698. ACM, 2018.

[LL16] Dung D. Le and Hady Wirawan Lauw. Euclidean co-embedding of ordinal data for multi-type visualization. In *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*, pages 396–404. SIAM, 2016.

[LLSZ19] Marc Teva Law, Renjie Liao, Jake Snell, and Richard S. Zemel. Lorentzian distance learning for hyperbolic representations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3672–3681. PMLR, 2019.

[LRC+17] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1419–1428. ACM, 2017.

[LRFS19] Qinghua Liu, Andrew Henry Reiner, Arnoldo Frigessi, and Ida Scheel. Diverse personalized recommendations with uncertainty from implicit preference data with the bayesian mallows model. *Knowl. Based Syst.*, 186, 2019.

[LS17] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 305–314. ACM, 2017.

[LTYL12] Xingjie Liu, Yuan Tian, Mao Ye, and Wang-Chien Lee. Exploring personal impact for group recommendation. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 674–683, 2012.

[LWW+16] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. Context-aware sequential recommendation. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1053–1058. IEEE Computer Society, 2016.

[LYBP16] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 289–297, 2016.

[LZZ+20] Mingming Li, Shuai Zhang, Fuqing Zhu, Liangjun Zang, Wanhui Qian, Jizhong Han, and Songlin Hu. Symmetric metric learning with adaptive margin for recommendation. In *AAAI*, 2020.

[MA07] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys 2007, Minneapolis, MN, USA, October 19-20, 2007*, pages 17–24. ACM, 2007.

[MB05] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*. Society for Artificial Intelligence and Statistics, 2005.

[MHN13] A.L. Maas, A.Y. Hannun, and A.Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning*, Atlanta, Georgia, 2013.

[MLZW17] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4068–4074. ijcai.org, 2017.

[MSC+06] Kevin McCarthy, Maria Salamó, Lorcan Coyle, Lorraine McGinty, Barry Smyth, and Paddy Nixon. CATS: A synchronous approach to collaborative group recommendation. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA, May 11-13, 2006*, pages 86–91, 2006.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[NCL18] Wei Niu, James Caverlee, and Haokai Lu. Neural personalized ranking for image recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 423–431. ACM, 2018.

[NK17] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6338–6347, 2017.

[NK18] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3776–3785. PMLR, 2018.

[OCKR01] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. Polylens: A recommender system for groups of user. In *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work, 16-20 September 2001, Bonn, Germany*, pages 199–218. Kluwer, 2001.

[OTOT17] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1933–1942, 2017.

[PC19a] François-Pierre Paty and Marco Cuturi. Subspace robust wasserstein distances. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5072–5081. PMLR, 2019.

[PC19b] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[PKXY18] Chanyoung Park, Donghyun Kim, Xing Xie, and Hwanjo Yu. Collaborative translational metric learning. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 367–376. IEEE Computer Society, 2018.

[PLCZ16] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. A general recommendation model for heterogeneous networks. *IEEE Trans. Knowl. Data Eng.*, 28(12):3140–3153, 2016.

[PYC+19] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. Value-aware recommendation based on reinforced profit maximization in e-commerce systems. *CoRR*, abs/1902.00851, 2019.

[PZC+08] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 502–511. IEEE Computer Society, 2008.

[QCJ18] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, 51(4):66:1–66:36, 2018.

[QKHC17] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 130–137. ACM, 2017.

[QMPT16] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Recommending packages to groups. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 449–458. IEEE Computer Society, 2016.

[QRT16] Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. Recommending new items to ephemeral groups using contextual user influence. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 285–292. ACM, 2016.

[Ren10] Steffen Rendle. Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000. IEEE Computer Society, 2010.

[RFGS09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press, 2009.

[RKZA20] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 240–248. ACM, 2020.

[RSL17] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. Inter-session modeling for session-based recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2017, Como, Italy, August 27, 2017*, pages 24–31. ACM, 2017.

[RVM+11] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 833–840, Madison, WI, USA, 2011. Omnipress.

[RZK19] Steffen Rendle, Li Zhang, and Yehuda Koren. On the difficulty of evaluating baselines: A study on recommender systems. *CoRR*, abs/1905.01395, 2019.

[Sar11] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *Graph Drawing - 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers*, volume 7034 of *Lecture Notes in Computer Science*, pages 355–366. Springer, 2011.

[SED+15] Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *CoRR*, abs/1512.01124, 2015.

[SGM16] Florian Strub, Romaric Gaudel, and Jérémie Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 11–16. ACM, 2016.

[SHK+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SK09] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, 2009:421425:1–421425:19, 2009.

[SLW+19] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 1441–1450. ACM, 2019.

[SM07] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264. Curran Associates, Inc., 2007.

[SMSX15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 111–112. ACM, 2015.

[SSGR18] Frederic Sala, Christopher De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4457–4466. PMLR, 2018.

[SYMM11] Shunichi Seko, Takashi Yagi, Manabu Motegi, and Shin-yo Muto. Group recommendation using feature space representing behavioral tendency and power balance among members. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 101–108, 2011.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[SZLM18] Shaoyun Shi, Min Zhang, Yiqun Liu, and Shaoping Ma. Attention-based adaptive model to unify warm and cold starts recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 127–136. ACM, 2018.

[TATH18] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of*

the 2018 World Wide Web Conference, WWW '18, page 729–739, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[TBG19] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincare glove: Hyperbolic word embeddings. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[TD19] Jie Tang and Yuxiao Dong. Representation learning on networks: Theories, algorithms, and applications. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1321–1322. ACM, 2019.

[TLH18] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2309–2318, 2018.

[TLLK19] Thanh Tran, Xinyue Liu, Kyumin Lee, and Xiangnan Kong. Signed distance-based deep memory recommender. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1841–1852. ACM, 2019.

[TPT+19] Lucas Vinh Tran, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong, and Xiaoli Li. Interact and decide: Medley of sub-attention networks for effective group recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 255–264. ACM, 2019.

[TQC+15] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 30music listening and playlists dataset. In *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16, 2015*, volume 1441 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[TSL19] Thanh Tran, Renee Sweeney, and Kyumin Lee. Adversarial mahalanobis distance-based attentive song recommender for automatic playlist continuation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 245–254. ACM, 2019.

[TTH18] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 583–591. ACM, 2018.

[TW18] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 565–573, New York, NY, USA, 2018. Association for Computing Machinery.

[Twa16] Bartlomiej Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 273–276. ACM, 2016.

[TXL16] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 17–22. ACM, 2016.

[TZH+19] Ryuichi Takanobu, Tao Zhuang, Minlie Huang, Jun Feng, Haihong Tang, and Bo Zheng. Aggregating e-commerce search results from heterogeneous sources via hierarchical reinforcement learning. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1771–1781. ACM, 2019.

[TZLH18] Yi Tay, Shuai Zhang, Anh Tuan Luu, and Siu Cheung Hui. Self-attentive neural collaborative filtering. *CoRR*, abs/1806.06446, 2018.

[Ung09] Abraham Albert Ungar. *A Gyrovector Space Approach to Hyperbolic Geometry*. Synthesis Lectures on Mathematics & Statistics. Morgan & Claypool Publishers, 2009.

[vdBKW17] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017.

[vdODS13] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2643–2651, 2013.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010, 2017.

[VTBE15] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164, 2015.

[VTTZ+20] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 609–617, New York, NY, USA, 2020. Association for Computing Machinery.

[WB11] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 448–456. ACM, 2011.

[WBS05] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 1473–1480, 2005.

[WCW19] Shoujin Wang, Longbing Cao, and Yan Wang. A survey on session-based recommender systems. *CoRR*, abs/1902.04864, 2019.

[WDWK11] Jun Wang, Huyen Do, Adam Woznica, and Alexandros Kalousis. Metric learning with multiple kernels. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 1170–1178, 2011.

[WDZE16] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, pages 153–162. ACM, 2016.

[WHW+19] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 165–174. ACM, 2019.

[WL18] Benjamin Wilson and Matthia Leimeister. Gradient descent in hyperbolic space. *arXiv preprint arXiv:1805.08207*, 2018.

[WLW+15] Shengxian Wan, Yanyan Lan, Pengfei Wang, Jiafeng Guo, Jun Xu, and Xueqi Cheng. Next basket recommendation with neural networks. In *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16, 2015*, volume 1441 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[WM18] Mengting Wan and Julian J. McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 86–94. ACM, 2018.

[WPC+19] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

[WSF+19] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 235–244. ACM, 2019.

[WWT+17] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 391–400. ACM, 2017.

[WWY15] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1235–1244. ACM, 2015.

[WZG+19] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2091–2102. ACM, 2019.

[XHZ+19] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon M. Jose. Relational collaborative filtering: Modeling multiple item relations for recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 125–134. ACM, 2019.

[XNJR02] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 505–512. MIT Press, 2002.

[XYH+17] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature

interactions via attention networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3119–3125. ijcai.org, 2017.

[YCL14] Quan Yuan, Gao Cong, and Chin-Yew Lin. COM: a generative model for group recommendation. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 163–172, 2014.

[YHC+18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018.

[YLL12] Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation: a generative model approach. In *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*, pages 671–680, 2012.

[YLW+16] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 729–732. ACM, 2016.

[YZHG06] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. TV program recommendation for multiple viewers based on user profile merging. *User Model. User-Adapt. Interact.*, 16(1):63–82, 2006.

[ZACC17] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1449–1458. ACM, 2017.

[ZC18] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192, 2018.

[ZCWZ18] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. Deep variational network embedding in wasserstein space. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2827–2836, New York, NY, USA, 2018. Association for Computing Machinery.

[ZCZ18a] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *CoRR*, abs/1812.04202, 2018.

[ZCZ+18b] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

[ZLJ+18] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 311–319. ACM, 2018.

[ZNY17] Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 425–434. ACM, 2017.

[ZTY+19] Shuai Zhang, Yi Tay, Lina Yao, Aixin Sun, and Jake An. Next item recommendation with self-attentive metric learning. 9, 2019.

[ZWC19] Ziwei Zhu, Jianling Wang, and James Caverlee. Improving top-k recommendation via jointcollaborative autoencoders. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 3483–3482. ACM, 2019.

[ZXD+19] Lixin Zou, Long Xia, Zhuoye Ding, Dawei Yin, Jiaxing Song, and Weidong Liu. Reinforcement learning to diversify top-n recommendation. In *Database*

Systems for Advanced Applications - 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22-25, 2019, Proceedings, Part II, volume 11447 of *Lecture Notes in Computer Science*, pages 104–120. Springer, 2019.

[ZXZ+18] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 95–103. ACM, 2018.

[ZYS+18] Shuai Zhang, Lina Yao, Aixin Sun, Sen Wang, Guodong Long, and Manqing Dong. Neurec: On nonlinear transformation for personalized ranking. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3669–3675. ijcai.org, 2018.

[ZYST19] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1):5:1–5:38, 2019.

[ZYX+17] Shuai Zhang, Lina Yao, Xiwei Xu, Sen Wang, and Liming Zhu. Hybrid collaborative recommendation via semi-autoencoder. In *Neural Information Processing - 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part I*, volume 10634 of *Lecture Notes in Computer Science*, pages 185–193. Springer, 2017.

[ZZD+18] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1040–1048. ACM, 2018.

[ZZZ+18] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 167–176. ACM, 2018.