

Singapore Management University

Institutional Knowledge at Singapore Management University

Dissertations and Theses Collection (Open Access)

Dissertations and Theses

9-2019

Preference learning and similarity learning perspectives on personalized recommendation

Duy Dung LE

Singapore Management University, ddle.2015@phdis.smu.edu.sg

Follow this and additional works at: https://ink.library.smu.edu.sg/etd_coll



Part of the [Computer Engineering Commons](#), and the [Programming Languages and Compilers Commons](#)

Citation

LE, Duy Dung. Preference learning and similarity learning perspectives on personalized recommendation. (2019).

Available at: https://ink.library.smu.edu.sg/etd_coll/241

This PhD Dissertation is brought to you for free and open access by the Dissertations and Theses at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Dissertations and Theses Collection (Open Access) by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

PREFERENCE LEARNING AND SIMILARITY LEARNING
PERSPECTIVES
ON PERSONALIZED RECOMMENDATION

Le Duy Dung

SINGAPORE MANAGEMENT UNIVERSITY

2019

Preference Learning and Similarity Learning
Perspectives on Personalized Recommendation

Le Duy Dung

Submitted to School of Information Systems in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Hady W. Lauw (Supervisor/Chair)
Associate Professor of Information Systems
Singapore Management University

Zheng Baihua
Associate Professor of Information Systems
Singapore Management University

Yuan Fang
Assistant Professor of Information Systems
Singapore Management University

Leong Tze Yun
Professor of Computer Science (Practice)
National University of Singapore

Singapore Management University

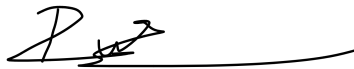
2019

Copyright (2019) Le Duy Dung

I hereby declare that this dissertation is my original work
and it has been written by me in its entirety.

I have duly acknowledged all the sources of information
which have been used in this dissertation.

This dissertation has also not been submitted for any degree
in any university previously

A handwritten signature in black ink, appearing to read 'Le Duy Dung', is positioned above a horizontal line. The signature is stylized and includes a long horizontal stroke extending to the right.

Le Duy Dung

15 September 2019

Preference Learning and Similarity Learning Perspectives on Personalized Recommendation

Le Duy Dung

Abstract

Personalized recommendation, whose objective is to generate a limited list of items (e.g., products on Amazon, movies on Netflix, or pins on Pinterest, etc.) for each user, has gained extensive attention from both researchers and practitioners in the last decade. The necessity of personalized recommendation is driven by the explosion of available options online, which makes it difficult, if not downright impossible, for each user to investigate every option. Product and service providers rely on recommendation algorithms to identify manageable number of the most likely or preferred options to be presented to each user. Also, due to the limited screen estate of computing devices, this manageable number maybe relatively small, yet the selection of items to be recommended is personalized to each individual users.

The basic entities of a personalized recommendation system are items and users. Personalization can be achieved through custom alternatives for delivering the right experience to the right user at the right time on the right device. Therefore, personalized recommendation can appear in many forms, depending on the characteristics of the items and the desired experience that the system wants users to have. In this thesis, we encompass two perspectives on personalized recommendation: *preference learning* and *similarity learning*. The former refers to the personalization in which the recommendation is tailored towards users' preference. The latter, on the other hand, refers to personalization approach in which recommendation is generated based on the users' personal perceptions of similarity between the items.

In the *preference learning* perspective, we focus on the task of retrieving recommendations efficiently and propose two techniques for this objective. For the first technique, we rely on Euclidean embedding to learn user and item latent vectors from users' ordinal preferences. Since they operate

in the Euclidean space, these latent vectors natively support efficient nearest neighbor search using geometric structures such as spatial trees. For the second technique, our key idea is to desensitize the effect of vector magnitudes when modelling users' preferences over items. That effectively reduces the recommendation retrieval problem to the nearest neighbor search problem with cosine similarity, which can be solved efficiently with various indexing methods such as locality sensitive hashing, spatial trees, or inverted index. Extensive experiments on publicly available datasets show significant improvement of proposed techniques over the baselines.

In the *similarity learning* perspective, we are interested in the setting where there are multiple similarity perceptions in the data. Towards modelling these perceptions effectively, we propose two approaches that are natively multiperspective. One is a graph-theoretic framework that yields a similarity measure for any pair of objects for a perspective. Another is a geometric framework that learns multiple low-dimensional representation of objects, each for one perspective. Experiments in both studies show that the adoption of *multiperspective* approach allows us to better model the similarity between objects, as compared to classical uniperspective methods, which ignore the multiperspectivity in the data.

Contents

List of Figures	vi
List of Tables	vii
List of Notations	viii
1 Introduction	1
1.1 Personalized Recommendation	2
1.1.1 Preference Learning	3
1.1.2 Similarity Learning	8
1.2 Thesis Outline and Contributions	11
2 Related Work	15
2.1 Efficient Retrieval of Personalized Recommendations	15
2.1.1 Efficient Candidate Screening	17
2.1.2 Efficient Inner Product Computation	21
2.2 Modelling Multiple Similarity Perspectives	24
2.2.1 Supervised Similarity Learning	24
2.2.2 Unsupervised Similarity Learning	26

I Preference Learning:	
Efficient Recommendation Retrieval	30
3 Collaborative Ordinal Embedding	31
3.1 Introduction	31
3.2 Framework	35
3.2.1 Problem Formulation	35
3.2.2 Generative Model	36
3.2.3 Triple Probability Function	38
3.2.4 Learning Algorithms	40
3.3 Experiments	42
3.3.1 Comparison to Embedding Baselines	44
3.3.2 Efficient Retrieval of Recommendation with KD-Tree	49
3.4 Discussion	52
4 Indexable Bayesian Personalized Ranking	53
4.1 Introduction	54
4.2 Framework	56
4.3 Experiments	61
4.3.1 Top-k Recommendation with LSH Index	63
4.3.2 Top-k Recommendation with KD-Tree Index	66
4.3.3 Top-k Recommendation with Inverted Index	67
4.4 Analysis on LSH-friendliness of Indexable BPR	70
4.5 Discussion	75

II Similarity Learning:

Modelling Multiple Perspectives	76
5 Multiperspective Graph-Theoretic Similarity Measure	77
5.1 Introduction	78
5.2 Framework	79
5.2.1 Pipelined-SimRank	83
5.2.2 Joint Solution: MP-SimRank	86
5.3 Experiments	89
5.3.1 Experimental Settings	89
5.3.2 Comparison to Baselines	92
5.3.3 Inter-Perspective Similarities	94
5.3.4 Illustrative Case Study	94
5.4 Efficiency Analysis	96
5.4.1 Complexity Analysis	96
5.4.2 Heuristic for More Efficient MP-SimRank	98
5.5 Discussion	99
6 Spherical Conditional Ordinal Embedding	101
6.1 Introduction	101
6.2 Framework	103
6.2.1 Problem Formulation	103
6.2.2 Proposed Methodology	104
6.2.3 Model	105
6.2.4 Parameter Learning	109
6.3 Experiments	110
6.3.1 Experimental Setup	111
6.3.2 Comparison to Baselines	113

6.3.3	Perspective Relatedness	116
6.3.4	Multiple Maps vs. Single Map	117
6.3.5	SCORE vs. MP-SIMRANK	119
6.4	Coordinate Transformation	120
6.5	Discussion	122
7	Conclusion	123
7.1	Summary	123
7.2	Future Work	124
	Bibliography	126

List of Figures

1.1	An illustration for the incompatibility of inner product kernel for spatial tree index (Euclidean distance) and inverted index (cosine similarity).	6
1.2	An example of three pin collections (boards) of a Pinterest user.	9
1.3	An overview of this thesis.	12
2.1	Approaches for Efficient MF-based Recommendation Retrieval	16
2.2	An Illustration of Candidate Screening with Indexing	18
2.3	A taxonomy for multiperspective similarity modelling	25
3.1	Euclidean Embedding of Users & Items	33
3.2	Collaborative Ordinal Embedding (COE)	37
3.3	Triple Probability Function	40
3.4	Example Visualization of Users (triangles) and Items (crosses) in MovieLens-100K	47
3.5	nRecall@k with KD-Tree Indexing.	50
3.6	nRecall@10 vs. speedup with KD-tree Indexing.	52
4.1	Number of triples (per user) vs. number of items.	63
4.2	nRecall@k with Hash Table Lookup Strategy ($T = 10$ hash tables).	64
4.3	nRecall@10 vs. Speedup with Hashtable Lookup Strategy ($T = 10$ hash tables).	65
4.4	nRecall@k with KD-Tree Indexing.	67
4.5	nRecall@10 vs. Speedup with KD-tree Indexing.	68

4.6	nRecall@k with Inverted Indexing.	69
4.7	nRecall@10 vs. Speedup with Inverted Indexing.	70
4.8	LSH Friendly Measurement at $d = 20$	72
4.9	nDCG@10 at $d \in \{5, 10, 20, 30, 50, 75, 100\}$	74
5.1	Illustration of the Hypergraph Representation	81
5.2	PIPELINED-SIMRANK: Bipartite graph for computing similarity between perspective nodes	84
5.3	Recall values of all models	92
5.4	PRES values of all models	92
5.5	Illustrative example of multiperspective similarity from <i>Paris Attractions</i> dataset.	97
5.6	PRES, Recall, and running time of CLUSTEREDMP-SIMRANK with different number of clusters k	99
6.1	Approaches for Conditional Ordinal Embedding.	104
6.2	Representations of three objects i, j, k , two perspectives p, p'	107
6.3	The probability of observing $\langle p i, j, k \rangle$ is a combination of the perspective-specific probability σ_{ijk}^p and the global probability σ_{ijk}	108
6.4	Overall and hidden preservation accuracy	113
6.5	10-NN classification accuracy at $r = 0.5$	114
6.6	Overall preservation accuracies at various split ratios.	115
6.7	Pearson Correlation of Angular similarities vs. NMIs.	116
6.8	Visualization maps for <i>type, #legs, predator (Zoo)</i>	117
6.9	Visualizations for three attributes: <i>immigration, education-spending, crime (HouseVote)</i>	118
6.10	Transformation of objects' coordinates from 3- d to 2- d	121

List of Tables

1	List of Notations	viii
3.1	Datasets Summary	44
3.2	Rating-Based Dataset (MovieLens-100K): COE vs. Other Baselines	46
3.3	Rating-Based Dataset (Netflix): COE vs. Other Baselines	46
3.4	Cooccurrence-Based Dataset (Last.fm): COE vs. Cooccurrence Embedding	48
3.5	Cooccurrence-Based Dataset (20News): COE vs. Cooccurrence Embedding	48
4.1	Datasets Summary	63
4.2	<i>Absolute</i> and <i>Relative</i> $nDCG@10$ of all models as the length of LSH codes (b) varies.	73
5.1	Correlation between NMI scores and inter-perspective similarities for <i>Zoo</i>	95
5.2	Correlation between NMI scores and inter-perspective similarities for <i>HouseVote</i>	96
5.3	Cluster data of four users from <i>Paris Attractions</i>	97
5.4	Complexity analysis (per iteration) of all SimRank-based methods	98
6.1	Performance of SCORE: multi-maps vs. single-map.	117
6.2	Prediction Accuracies on Unseen Triplets of SCORE and MP-SIMRANK.	120

List of Notations

Symbol	Description
\mathcal{U}	collection of all users
\mathcal{I}	collection of all items
u	a specific user
i	a specific item
r_{ui}	rating of user u for item i
x_u	latent vector representation for user u
y_i	latent vector representation for item i
d	latent space dimension
k	the number of recommendations
\mathcal{P}	collection of perspectives
\mathcal{O}	collection of objects
p	a specific perspective
o	a specific object
\mathcal{X}	set of all vertices, i.e., $\mathcal{P} \cup \mathcal{O}$
\mathcal{E}	set of all hyperedges, i.e., $\{(p, o_i, o_j) : o_i \text{ and } o_j \text{ are related, according to } p\}$
\mathcal{H}	3-uniform hypergraph, $\mathcal{H} = \{\mathcal{X}, \mathcal{E}\}$
$N_p(o_i)$	$\{o_j \in \mathcal{O} (p, o_i, o_j) \in \mathcal{E}\}$
$S_p(o_i, o_j)$	similarity score between vertices o_i and o_k , according to perspective p
\mathcal{S}_p	perspective-specific inter-object similarity score matrix
$\text{sim}(p, p')$	similarity score between two perspective vertices p, p'
$\langle t i, j, k \rangle$	a quadruple of one perspective and three objects, where the first-mentioned object is more similar to the centered object, compared to the third-mentioned object
\mathcal{N}_t	the set of observed quadruples for a perspective t , i.e., $\mathcal{N}_t = \{\langle t i, j, k \rangle i \neq j \neq k \in \mathcal{O}\}$.
x_t	embedding of a perspective t
y_o	embedding of an object o

Table 1: List of Notations

Acknowledgments

First and foremost, I would like express my sincere gratitude to my advisor, Prof. Hady Wirawan Lauw for his continuous support of my PhD, for his inspiration, patience, and immense knowledge. Under his guidance, I have learnt to appreciate the importance of being perseverant and curious, to strive for success but also to embrace failure. I could not have imagined having a better advisor and mentor for my PhD.

Besides my advisor, I would like to send special thanks to the rest of my thesis committee: Prof. Zheng Baihua, Prof. Fang Yuan, and Prof. Leong Tze Yun for their insightful comments and constructive suggestions to make this dissertation more complete and comprehensive.

I want to thank my teammates: Dr. Aghiles Salah, Dr. Maksim Tkachenko, Dr. Nguyen Thanh Son, Dr. Le Duc Trong, Quoc-Tuan, Trung-Hoang, Thanh-Binh, Phu-Minh, Ween Jiann, Chong Cher, Zhang Ce, Guo Jingyao, Darryl, and other former colleagues for the amazing and stimulating discussions, for the fun that we experienced together, and for the close-knit friendships we build over the years.

I would like to thank the administrative staff of the PhD program in School of Information Systems: Pei Huan and Caroline for dedicating their time and effort to incentivize me to progress towards the end of the PhD journey. I also want to thank Yong Fong from SMU PGR Office, who has helped to improve my communication skills significantly.

Last but not the least, I would like to thank my family: my parents, my wife, and my brother for always accompanying me in all sweet and sour moments of my PhD. Without the unconditionally love and the endless support from them, this thesis would not have been possible.

Chapter 1

Introduction

Today, we frequently face a multitude of options in various spheres of life, e.g., deciding which product to buy on Amazon, selecting which movie to watch on Netflix, choosing which article to read or image to view on social media. However, as their systems are growing rapidly, the number of options is becoming immense. Recent estimates¹ put the number of unique products sold at Amazon.com at close to 400 million. In the realm of digital artefacts, the scale is even larger. The number of photos on Facebook and Flickr is estimated² to have reached billions, and still growing. On one hand, such a large quantity of information in the Web makes it difficult, if not downright impossible, for users to investigate every option. On the other hand, it is also challenging for merchants to present appropriate products to users in a timely manner. That problem leads to the need for a system for intelligent information access and personalized support in sifting through large amounts of available information, according to user interests and preferences. One of such systems is based on the idea of *personalized recommendation*, whose objective is to learn the user preferences from their historical feedback, and generate a curated set of items that might be of interest to the users.

¹<http://bit.ly/2k6aIrr>, <http://bit.ly/2g7wVUI>, <http://bit.ly/2iMehgY>

²<http://read.bi/1Niwlvs>, <http://bit.ly/1v1m6zR>

1.1 Personalized Recommendation

Over the past decades, recommendation algorithms have been one of the key technologies for Web services. Statistics from McKinsey³ have shown that recommendation systems brought Amazon 35 percent of its revenue and increased up to 75 percent of video consumption on Netflix. Recommendation feature is also responsible for 70 percent of views on YouTube⁴. Pinterest reported that their recommendation engines have powered over 40% of user engagement on the platform [64]. A recommendation system is a type of information filter, which can learn users' interests and hobbies according to their profile or historical behaviors, and then predict their ratings or preferences for a given item. It changes the way businesses communicate with users and strengthens the interactivity between them.

The fundamental entities of a recommender system are items, which are products and services and users, who are consumers. The basic principle of recommender systems is to deliver the right experience to the right user at the right time on the right device. Therefore, personalized recommendation can appear in many forms, depending on the characteristics of the Web services and the desired experience that the system wants users to have. In this thesis, we encompass two of such perspectives on personalization, namely: *preference learning* and *similarity learning*. The former refers to the personalization in which the recommendation list is tailored towards user preferences. Examples of preference observations are explicit ratings given by users or implicit feedbacks such as users' consumption behaviors. The latter, on the other hand, refers to personalization approach in which recommendation list is generated based on the personal perspectives of users on the similarity among objects (e.g., products, images). This focuses on learning from "clustering" feedback (i.e., grouping items into clusters), which enables users to express their own views on the similarity or relatedness between different items. In the subsequent sections, we elaborate further on these two perspectives on personalized recommendation.

³<https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers>

⁴<https://www.cnet.com/news/youtube-ces-2018-neal-mohan/>

1.1.1 Preference Learning

The *first* perspective is the form of personalization that derives recommendations based on users' historical feedbacks. In an increasingly digitized world, users are leaving ever greater traces of their personal preferences. Users express their preferences in diverse forms, both explicitly through ratings and rankings, as well as implicitly through their consumption behaviors. Due to its great and ascending importance, preference data has been studied intensively, particularly in the context of recommender systems to build and update user profiles in order to provide personalization. Example techniques include collaborative filtering [88], user-item graph models [4], regression based models [107], deep learning based models [121], and matrix factorization [49]. In this part of the thesis, we focus on matrix factorization (MF), which is an established and prevalent methodology in the literature, thanks to its efficiency in dealing with large and sparse user-item rating matrices. MF-based methods have improved upon the efficiency of other collaborative filtering approaches such as neighborhood-based models ([23]). By representing users and items as low-dimensional latent vectors, MF-based methods can avoid the expensive similarity weight computation (user-to-user or item-to-item) in high-dimensional space of neighborhood-based collaborative filtering models.

In this thesis, we seek to further improve the efficiency the recommendation process by jointly investigating the two phases of a MF-based recommender system: *learning* and *retrieval*.

- *Learning phase*: this phase analyzes user's historical feedback (e.g., ratings, click behaviors) to learn user's preference. Specifically, MF-based methods such as [87], [108] derive from those inputs a latent vector $x_u \in \mathbb{R}^d$ for each user $u \in \mathcal{U}$, and a latent vector $y_i \in \mathbb{R}^d$ for each item $i \in \mathcal{I}$, where d is the dimensionality of the vector space. The degree of preference of user u for item i is modeled as the inner product score $\hat{r}_{ui} \propto x_u^T y_i$ of vectors x_u, y_i . A higher inner product score implies a higher chance of the user to prefer the item.
- *Retrieval phase*: Given the output latent vectors from the *learning* phase, to arrive at the

recommendation list for a target user u , we need to identify the top- k preferred items according to preference scores $x_u^T y_i, \forall 1 \leq i \leq n$. In the scenarios where there are bias terms in modelling the preference score, i.e., $\hat{r}_{ui} \propto x_u^T y_i + b_u + b_i$, we can convert each user vector x_u to $\tilde{x}_u = [x_u, b_u, 1]$ and each item vector y_i to $\tilde{y}_i = [y_i, 1, b_i]$. The problem now become identifying the top- k items with the highest inner product scores $\tilde{x}_u^T \tilde{y}_i, \forall 1 \leq i \leq n$. For ease of readability, from this point onward we refer to \tilde{x}_u and \tilde{y}_i as x_u and y_i .

For the *learning* phase, the objective is to optimize for the accuracy in identifying the items a user is likely to prefer. For the *retrieval* phase, the objective is to achieve high efficiency in constructing the recommendation list in real time for each user. The real-time nature of the retrieval task is necessitated by the response time expected by end users. Ideally, all item vectors in $\{y_1, y_2, \dots, y_n\}$ are examined and sorted according to their inner product scores. However, the catalog of items is often too large to allow an exhaustive computation of all the inner products within a budgeted retrieval time. Therefore, having a faster alternative for this process of recommendation retrieval is desirable.

MF-based Recommendation Retrieval Problem

MF-based recommendation retrieval is essentially a similarity search problem. Indeed, the retrieval of top- k recommendations is equivalent to the task of ranking every item y_i ($\forall 1 \leq i \leq n$) for each user x_u with respect to the inner product score $x_u^T y_i$ and returning the k items at the top of the rank list. This can be reduced to the fundamental problem of finding the index of the item that returns the highest inner product score, as in Equation 1.1.

Problem 1.1. For each user vector $x_u, u \in \mathcal{U}$, determine the index i of an item in \mathcal{I} such that:

$$i = \arg \max_{1 \leq i \leq n} x_u^T y_i \tag{1.1}$$

Problem in Equation 1.1, known as *Maximum Inner Product Search* or MIPS, arises naturally

in many large-scale tasks [91], when inner product-based comparisons are done between the embedding vector of a query and many potential candidate objects’ vectors.

Complexity of Exhaustive Search Solution. The MIPS problem is not trivial as a naive brute-force solution scales linearly with the number of objects to score. In the context of recommendation systems, the challenge is obvious as the number of items is typically very large (e.g., at the scale of millions). Naively scanning these millions of items to identify the few most relevant ones may inhibit truly real-time retrieval performance. Per user (or per query), the cost of such naive exhaustive approach is $\mathcal{O}(n \times d)$, which scales linearly with the number of items n and the number of factors d . However, real-world systems are constrained to perform top- k retrieval in a few milliseconds and scoring and ranking all items are impossible.

Pre-computing the top- k recommendations for all m users requires the storage cost of $\mathcal{O}(m \times k)$, which is not practical for systems where the number of users and items is in the scale of millions. Also, pre-computing the recommendations restricts the flexibility of the systems in capturing the rapidly changing preference of users (e.g., target user’s vector is updated online according to his/her behaviors) and adoptions of new items. Therefore, to achieve real-time retrieval for large-scale system that handles tens of thousands of users (number of queries), reducing the retrieval cost of online top- k recommendations is important.

Approaches. An effective approach to improve retrieval efficiency is to use indexing structures such as locality-sensitive hashing (LSH) [91], spatial trees (e.g., KD-tree [14]), and inverted index [15]. By indexing items’ latent vectors, we can quickly retrieve a small candidate set for k “most relevant” items to the user query vector, probably in sub-linear time with respect to the number of items n . This avoids an exhaustive search, and saves on the computation for those large number of items that the index considers irrelevant.

We focus on indexing as a faster alternative to exhaustively searching over all items to identify top- k items. Indexing is preferred over pre-computation of recommendation lists for all users, which is impractical [15, 47] as user interests change over time and new items appear. By indexing,

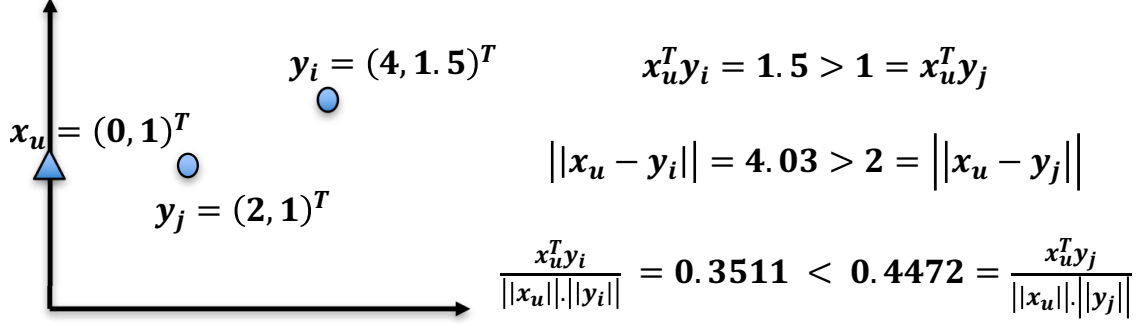


Figure 1.1: An illustration for the incompatibility of inner product kernel for spatial tree index (Euclidean distance) and inverted index (cosine similarity).

we avoid the storage requirement of dealing with all possible user-item pairs. Index storage scales only with the number of items n , while the number of queries could be larger.

Issues. However, getting the top nearest neighbors by the inner products using aforementioned index structures is trickier than conventional distance metrics like Euclidean or Cosine distance. The challenge is that the inner product does not form a proper metric space. As the inner product scores are unbounded, a point might not be its own nearest neighbour. This violates the triangle inequality and invalidates some common approaches for approximate nearest neighbours. Specifically, it has been established that there cannot exist any LSH family for maximum inner product search [91]. Also, retrieval on a spatial tree index finds the nearest neighbors based on the Euclidean distance, which are not equivalent to those with maximum inner product [9].

Figure 1.1 shows an example to illustrate the above analysis. In Figure 1.1, the inner product $x_u^T y_i$ is *greater* than $x_u^T y_j$, implying that u prefers i to j . However, the Euclidean distance computation shows that y_j is *closer* to x_u than y_i is to x_u . Also, the cosine similarity between x_u and y_j is *smaller* than that between x_u and y_i . This means that the nearest neighbor according to the inner product kernel is not the same as that according to Euclidean distance and Cosine similarity.

In this thesis, we propose to resolve this issue with the two following approaches:

- **Euclidean Embedding.** Euclidean embedding takes as input distances between data points (or their ordinal relationships), and outputs low-dimensional latent coordinates for each

point so that the inter-point Euclidean distances have the same ordering as of the input distances/ordinal relationships [51]. Because they operate in the Euclidean space, the coordinates support efficient nearest neighbor search using geometric index structures such as spatial trees.

There exist recent works on using Euclidean distance kernel to model user preferences over items. One of them is Collaborative Filtering via Euclidean Embedding or CFEE [46], which fits a rating \hat{r}_{ui} by user u on item i in terms of the squared Euclidean distance between x_u and y_i . However, fitting ratings directly does not preserve the pairwise comparisons. Therefore, we propose Collaborative Ordinal Embedding or COE [54] that is based on ordinal triples. It expresses an ordinal triple t_{uij} , indicating that a user u prefers an item i to another item j , through the Euclidean distance difference $\|x_u - y_j\| - \|x_u - y_i\|$. COE’s objective is to maximize this difference for each observation t_{uij} . Chapter 3 presents this work in detail.

- **Indexable Bayesian Personalized Ranking.** The key reason behind the incompatibility between inner product search that matrix factorization relies on, and the aforesaid index structures is how a user u ’s degree of preference for an item i , expressed as the inner product $x_u^T y_i$, is sensitive to the respective magnitude of the latent vectors $\|x_u\|, \|y_i\|$. Therefore, one insight towards achieving geometric compatibility is to desensitize the effect of vector magnitudes. The challenge is how to do so while still preserving the accuracy of the top- k retrieval.

There are a couple of recent approaches in this direction. One approach [9] is a post-processing transformation that expands the latent vectors learnt from matrix factorization with an extra dimensionality to equalize the magnitude of all item vectors. Because the transformation is a separate process from learning the vectors, such a workaround would not be as effective as working with natively indexable vectors in the first place. Another approach [29] extends the Bayesian Probabilistic Matrix Factorization [87], by making the

item latent vectors natively of fixed length. Fitting inner product to absolute rating value may not be suitable when only implicit feedback (not rating) is available. Moreover, we note that top- k recommendation is inherently an expression of “relative” rather than “absolute” preferences, i.e., the ranking among items is more important than the exact scores. Therefore, we propose *Indexable Bayesian Personalized Ranking* or INDEXABLE BPR that learns from ordinal triplets t_{uij} and produces native geometrically indexable latent vectors for accurate and efficient top-k personalized recommendation. We describe INDEXABLE BPR in detail in Chapter 4.

1.1.2 Similarity Learning

The *second* perspective on personalized recommendation is to consider how users perceive the similarity or relatedness between different items/objects. In this setting, a user can cluster or group related items together, and obtain personalized recommendations based on these clusters. For example, on Pinterest⁵, a visual discovery platform, as users pin images onto boards. Each board consists of photos that belong to an overall concept defined by the user. Pinterest users may wish to discover other related images to expand their boards (Figure 1.2). On Amazon, two products may be “related” in different ways: browsed together, purchased together, same manufacturer, etc. Amazon users may want to get recommended products that are similar or complementary to those they have purchased or browsed for before. In such scenarios, in order to generate high quality recommendations, machine learning models should capture the varying similarity perspectives of different users. We refer to this learning objective as *multiperspective similarity modelling*. Here, a similarity perspective could be a human subject or an aspect or a concept being used as reference for comparing the similarity among objects.

This form of personalization is distinct from multicriteria recommender systems (MCRS) in the literature [2, 61, 69], which uses multi-criteria preference ratings to further improve the rec-

⁵www.pinterest.com

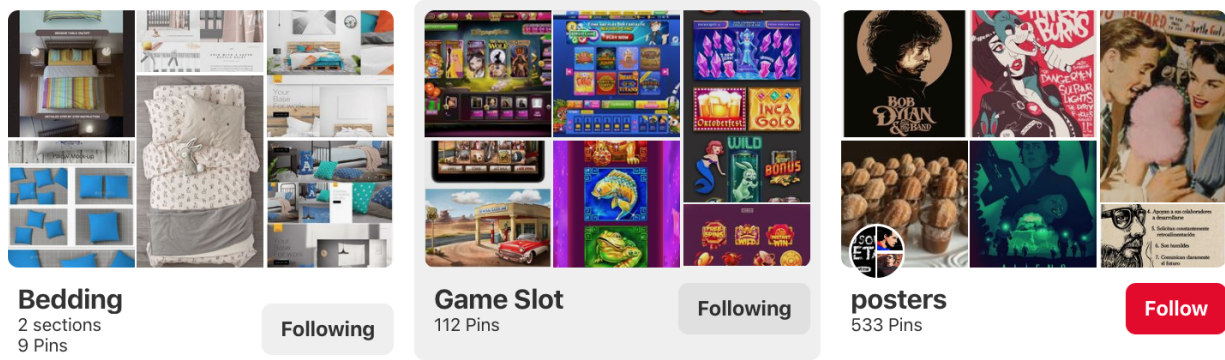


Figure 1.2: An example of three pin collections (boards) of a Pinterest user.

ommendation quality. MCRS models aim at capturing not only the user’s overall preference for a given item but also his/her preferences along specific aspects of the items. This setting is different from our as we are interested in the scenarios where the observations are similarity comparisons among objects generated by multiple users’ perspectives. MCRS’s objective is to arrive at a final ranking of items according to the preferences of the targeted users, whereas we seek to model how different perspectives perceive the similarity among the items.

Challenges. There are two main challenges for learning from such “clustering” behaviors, according to [116]. For one, since each user only clusters a subset of items, we need to simultaneously learn from multiple users’ feedbacks. However, different users may have different similarity perspectives e.g., two images or products may be similar/related according to one user, but dissimilar/unrelated to another. For example, on Pinterest, two images can be saved to the same board by one user, but to different boards by other users, depending on the overall concept depicted by the boards. Conventional approaches that do not account for personal perceptions cannot effectively learn similarity measures that are personalized to different users. For another, it is generally difficult to design a content-based features that can effectively capture similarity preferences of different users. From earlier examples of Pinterest images, it is unclear that visual features from images are sufficient to reflect multiple similarity perspectives of different users.

Approaches. Our assumption is that as the perspectives concern the same set of items, they

are related or similar to some extent. This is similar to the underlying principle of collaborative filtering techniques for recommender systems. We therefore adopt the *collaborative* approaches for the multiperspective similarity modelling problem. The term *collaborative* refers to the idea of learning from all perspectives simultaneously, in which a perspective can learn from observations of other similar perspectives, instead of solely learning from its' own similarity observations. With this principle in mind, we propose two different frameworks that both adopt the multiperspective approach: one is based on *graph-theoretic similarity measure* and the other is based on *ordinal embedding*:

- **Graph-theoretic Similarity Measure.** When the basis for similarity is a set of object-to-object relationships, it is a natural option to rely on graph-theoretic measure. Specifically, each object can be represented as a node on the graph. Two nodes are connected by an edge if the corresponding objects are similar or related. One seminal technique for measuring the structural-context similarity between a pair of graph nodes is SimRank [40], whose underlying intuition is that two nodes are similar if they are connected by other similar nodes. However, by design, SimRank as well as its variants capture only a single perspective of similarity. We propose a natively *multiperspective* approach to measuring graph-theoretic similarity. As input, we are given multiple graphs of the same object nodes and each graph reflects relationships among objects from a specific perspective. As output, we seek to measure the similarity between any pair of objects according to a particular perspective. The key intuition underlying this formulation is to model not only the perspective-specific *inter-object* similarity between any pair of objects, but also the *inter-perspective* similarity between any two perspectives. Learning these two similarity measures simultaneously renders an advantage in sharing information across similar perspectives, which helps to address the problem where observations for each perspective are under-sampled. Details of this approach can be found in Chapter 5.
- **Ordinal Embedding.** Embedding deals with finding a low-dimensional representation of

data points or objects based on observations of their similarities. The proximity between the low-dimensional vectors expresses a specific viewpoint or perspective about the similarity between objects. Particularly, ordinal embedding is the class of embedding methods that rely on relative comparisons of similarity – given an object, which of two other objects is more similar to it, rather than on exact similarity values (that may not always be observed). However, most of classical embedding approaches assume there is only one valid perspective of similarity and therefore seek to produce only a single embedding map. In the scenarios where ordinal comparisons are derived from multiple perspectives, we hypothesize that these perspectives would be better represented by multiple embedding maps. We formulate this problem as *conditional ordinal embedding*, which learns a distinct low-dimensional embedding map for each perspective, yet allows information sharing across perspectives via a shared representation. Our geometric approach is novel in its use of a shared spherical representation and multiple perspective-specific embedding maps on their respective tangent hyperplanes. We describe this framework in detail in Chapter 6.

1.2 Thesis Outline and Contributions

Organization. Figure 1.3 shows a graphical overview of this thesis. In Chapter 2, we review related works in the literature for both problems. *Next*, in Chapters 3 and 4, we respectively describe two approaches for efficient retrieval of personalized recommendations. In Chapters 5 and 6, we respectively present two frameworks for modeling diverse similarity perspectives in the data: one is based on graph-theoretic similarity measure and the other is based on ordinal embedding. *Finally*, we conclude and outline future research directions in Chapter 7.

Contributions. In this thesis, we explore two perspectives on personalized recommendation: *preference learning* and *similarity learning*. For *preference learning*, we seek to further improve upon the efficiency of MF-based methods by focusing on the *retrieval step* of recommendations

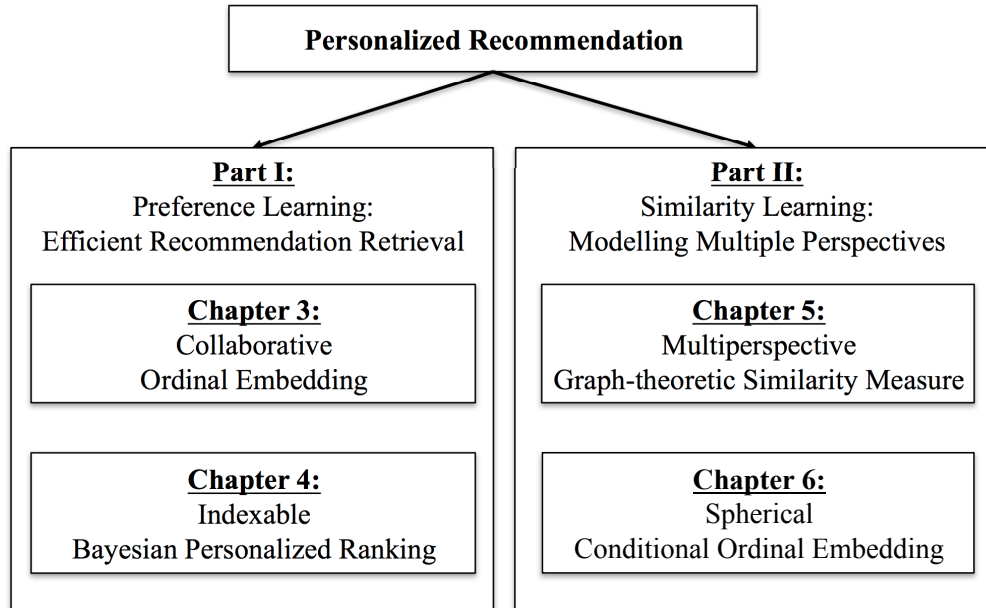


Figure 1.3: An overview of this thesis.

(after learning the latent vectors). As exhaustive search over all items scales linearly with the number of items, it may not be scalable to very large-scale systems. Particularly, we focus on using indexing structures (e.g., locality sensitive hashing, spatial tree indexing, and inverted indexing) as an efficient alternative to exhaustive search over all items. We propose two recommendation algorithms that produce vector representation for users and items that guarantees high recommendation accuracy returned by searching with the indexing structures. This could potentially lead to the development of a new category of recommender systems that not only optimize for the accuracy of the recommendations but also the efficiency in retrieving these recommendations in real-time. For *similarity learning*, our motivation is based on the observation that there are increasingly more scenarios where there exist multiple similarity perspectives in the data. By modelling these perspectives effectively, we can better understand the diverse similarity preferences of different users, via which we can personalize the recommendations to each user based on their "clustering" feedbacks. This is a distinct form of personalization as compared to the conventional setting as described in the *preference learning* perspective.

Overall, this thesis will cover the following papers, which are my publications as the first author:

- For preference learning perspective, we propose two different learning models that optimize for both recommendation accuracy and retrieval efficiency:
 - The first approach is based on *Euclidean embedding*, which learns from users' ordinal preferences and models the user-item relationship as the Euclidean distance between their respective low-dimensional latent vectors. The search for the most preferred items is converted to the *nearest neighbor search* (NNS) problem, which can be solved effectively using spatial indexing methods. This work was published in SIAM International Conference on Data Mining in 2016 [54].
 - The second approach is based on the idea of desensitizing the effect of vector magnitudes in modeling user-item relationship of MF-based methods. Our INDEXABLE BPR is formulated with a kernel based on angular distance shows a balance of accuracy and run-time efficiency, achieving higher recommendation accuracy than the baselines at the same retrieval speedup level, and higher retrieval speedup at the same accuracy level. This work was published in ACM Conference on Information and Knowledge Management in 2017 [55].
- For similarity learning perspective, we propose two approaches that support modelling multiple similarity perspectives in the data effectively:
 - In the first approach, we rely on the notion of graph-theoretic similarity measure and propose a framework that supports multiperspectivity. The proposed framework yields a similarity score for any pair of items for a specific perspective. Experiments on publicly available datasets showcase the utility of modeling multiple similarity perceptions compared to *uniperspective* methods. This work was accepted in the 27th ACM Conference on Information and Knowledge Management in 2018 [56].

- In the second approach, we propose a geometric framework for the conditional ordinal embedding problem, i.e., learning multiple maps from a pool of conditional ordinal triplets, each for one similarity perspective. Experiments on public datasets showcase the utility of collaborative learning over baselines that learn multiple maps independently. This work has been accepted for publication in the proceeding of the 28th International Joint Conference on Artificial Intelligence in 2019.

Chapter 2

Related Work

In this chapter, we review techniques and approaches that are closely related to our topics and methods in the subsequent chapters. For *efficient recommendation retrieval*, we provide a systematic review of relevant approaches in literature in Section 2.1. For *multiperspective similarity modelling*, we relate our work to the closet branches in literature in Section 2.2.

2.1 Efficient Retrieval of Personalized Recommendations

As illustrated in Figure 2.1, retrieving the recommendations involves the following steps:

1. **Candidate Screening:** Given a user vector x_u as query, a filtering procedure determines a candidate set C_u , with an objective of reducing the number of candidates.
2. **Candidate Ranking:** For the candidates in C_u , we compute their inner products against the query vector x_u , and sort them accordingly to identify the top- k recommendations. The computational complexity of this process is $O(|C_u| \times d + |C_u| \times \log(|C_u|))$.

For the exhaustive search approach (linear scanning), Step 1 essentially means doing nothing, passing all n items as candidates, i.e., $|C_u| = n$. Step 2 involves $O(nd + n \log n)$ operations for full inner product computations and sorting.

Any effort to optimize the retrieval efficiency would have to improve the running times of either or both of the steps outlined above. Therefore, we use these steps as the primary axis for taxonomizing the works, as shown in Figure 2.1.

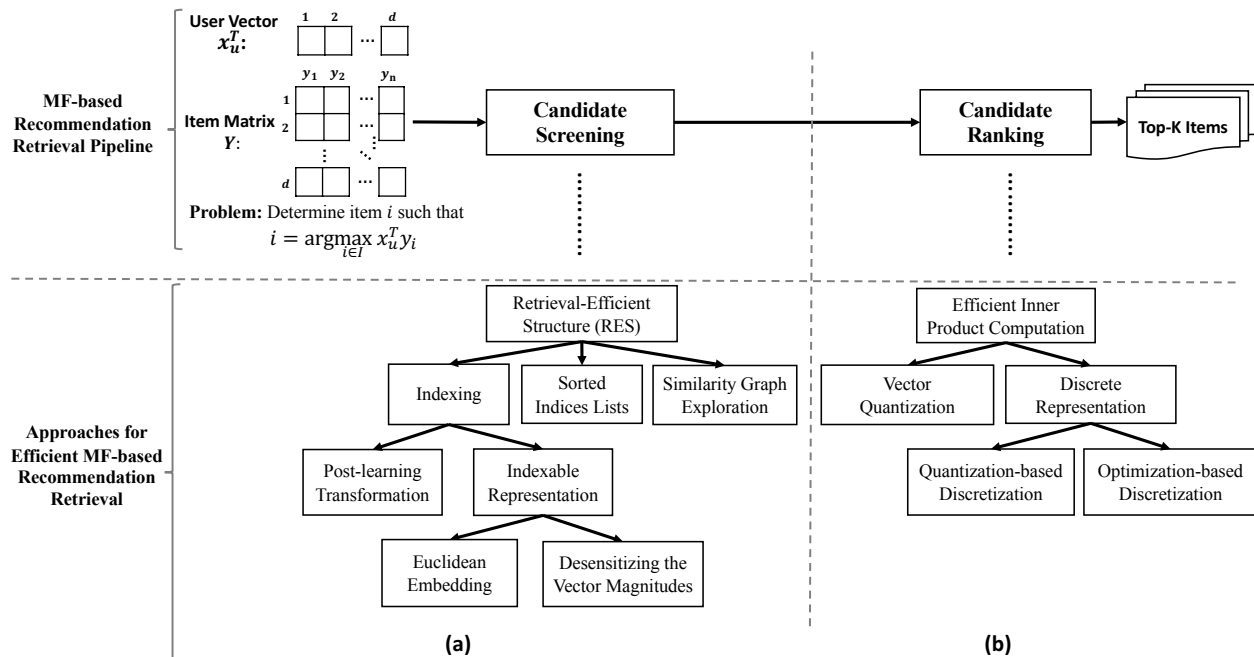


Figure 2.1: Approaches for Efficient MF-based Recommendation Retrieval

Efficient Candidate Screening. Figure 2.1(a) (left) organizes works that seek to quickly discard potentially irrelevant items, resulting in a candidate set C_u in which $|C_u|$ is substantially smaller than n (however, usually larger than k). The underlying idea is to trade off the cost of slightly lower recall from potentially missing out on the false negatives, for the benefit of faster retrieval through smaller candidate sets. There are two main lines of such strategies. *Exact* methods attempt to return an identical top- k to linear scanning, based on sequential scanning [60, 98, 99]. However, in the worst case, these exact methods still need to scan through all items, i.e., the complexity is $\mathcal{O}(n)$. Therefore, *approximate* reduction strategies, which may generate a slightly different top- k , are more prevalent in the literature. In Section 2.1.1, we describe those strategies that use *retrieval-efficient structures*, which are closely related to our proposed methods in Chapters 3 and 4.

Efficient Inner Product Computation. Figure 2.1(b) (right) organizes works that reduce the cost of *each* inner product score computation, originally operating in the d -dimensional real-valued latent feature space. One idea *vector quantization* decomposes the d -dimensional space into a Cartesian product of lower dimensional subspaces and quantizing each subspace separately. The inner product in the original d -dimensional space is approximated as the sum of inner products in these subspaces. Another idea *discrete representation* reduces the computational cost of operating on real-valued latent factors by representing users and items as binary codes in the Hamming space. The inner product score can be converted to computing the Hamming distance between user and item binary vectors requiring simple XOR operations [118]. That speeds up the generation of recommendations, even if we have to exhaustively scan over all items. We will elaborate further on these two ideas in Section 2.1.2.

2.1.1 Efficient Candidate Screening

In the following, we discourse on the class of approximate methods using retrieval-efficient structures for top- k recommendation retrieval.

Approximate Methods via Retrieval-Efficient Structure (RES)

This approach performs query-independent preprocessing of item vectors and stores them in a data structure that supports efficient candidate filtering upon query, probably in sub-linear time with respect to the number of items n . There are several possible structures, namely: indexing structures, sorted indices list, and similarity graph exploration.

RES#1: Indexing By indexing item latent vectors, we can quickly retrieve a small candidate set for the “most relevant” items to the user query vector, as illustrated in Figure 2.2. As opposed to exhaustive search, indexing offers a speed advantage, probably in sub-linear time with respect

to the number of items n , at the cost of some storage. As opposed to full precomputation whose storage requirement scales with the number of items and users, indexing is significantly more storage-efficient (only items need to be indexed), while offering greater flexibility for k , the size of recommendation list to be retrieved. Popular indexing structures include locality-sensitive hashing (LSH) [92] hash tables, spatial indexing (e.g., KD-tree [14]), and inverted index [15]. Depending on the structure, it might require a query-dependent processing step to convert the query vector x_u to a suitable form for querying.

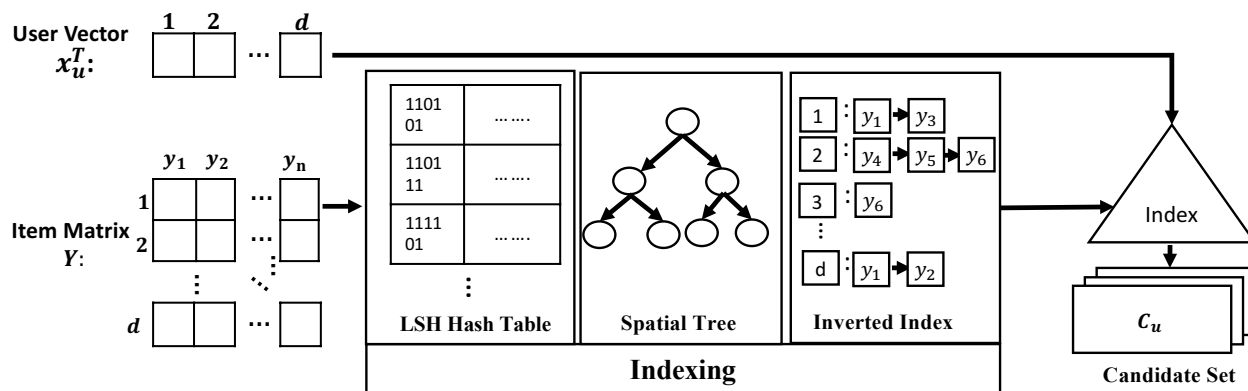


Figure 2.2: An Illustration of Candidate Screening with Indexing

Issues of using Indexing for MIPS. Early attempts towards having efficient MF-based retrieval include [48], which constructs a ball tree with item vectors and employs a branch and bound algorithm and [82], which proposes a dual-tree based search using cone trees to handle many queries simultaneously. However, these solutions partition the data space based on Euclidean distance, which could produce different top- k recommendations from the inner product.

The reason is because inner product does not form a proper metric space: for $\forall x \in \mathcal{R}^d$, you can always find $y_1, y_2 \in \mathcal{R}^d$ so that $x^T y_1 < x^T x < x^T y_2$. A point might not be its own nearest neighbor. This violates the triangle inequality and invalidates some common approaches for approximate nearest neighbors search. This means that the inner product kernel does not satisfy the underlying assumption for the effectiveness of a spatial tree, or an inverted index.

To resolve this issue, there are two possible solutions:

1. Post-Learning Transformation The *first* solution is to reduce MIPS problem to either *Nearest Neighbor Search* (NNS) problem defined as:

$$i = \arg \min_{1 \leq i \leq n} \|x_u - y_i\| \quad (2.1)$$

or *Maximum Cosine Similarity Search* (MCSS) problem:

$$i = \arg \max_{1 \leq i \leq n} \frac{x_u^T y_i}{\|x_u\| \cdot \|y_i\|} \quad (2.2)$$

The three problems are equivalent if all item vectors are of the same length. This can be solved via post-learning transformations applied to user and item vectors. This typically adds extra dimensions to user vectors $\{x_u\}_{u \in \mathcal{U}}$ and item vectors $\{y_i\}_{i \in \mathcal{I}}$ so that solving MIPS in the original space is equivalent to solving NNS/MCSS in the transformed space. After the reduction, there are several indexing solutions for the transformed NNS/MCSS such as locality sensitive hashing (LSH) [76, 92, 93], PCA-tree ([10]).

There are many choices for the transformation itself. For example, [92, 93] augment the vectors to higher dimensional space. Later, [10, 76] extend the output latent vectors by one dimension to equalize the magnitude of item vectors:

$$x_u \rightarrow \left[\frac{x_u}{\|x_u\|}, 0 \right] \forall u \in \mathcal{U}; y_i \rightarrow \left[\tilde{y}_i, \sqrt{1 - \|\tilde{y}_i\|^2} \right] \forall i \in \mathcal{I}, \quad (2.3)$$

in which $\tilde{y}_i = \frac{y_i}{\max_{i \in \mathcal{I}} \|y_i\|} \forall i \in \mathcal{I}$.

Extensions include [38] that minimizes the distortion error in reducing MIPS to NNS via Asymmetric LSH scheme and Query Normalized First transformation and [43] that uses randomized partition trees instead of LSH for a better theoretical guarantee on choosing the best MIPS-to-NNS/MCSS reduction strategies.

Also under this category, [15] transforms user and item dense latent vectors to sparse representation, where the sparsity patterns reflect the closeness of original vectors. It then uses inverted indexing with the resulting sparse vectors for efficient retrieval.

2. Indexable Representation The *second* solution is to avoid the need for transformation by designing recommendation algorithms whose latent output vectors can be immediately sub-linearly searchable using indexing. For instance, CFEE [46] uses Euclidean distance to model the user preference over items. The retrieval of top recommendations becomes nearest neighbor search (NNS) with Euclidean distance. Another work on indexable representation learning is *Indexable Probabilistic Matrix Factorization* or IPMF [29], which keeps the classic formulation of matrix factorization preference learning models, but incorporates additional constraint that all item vectors have the same magnitude. IPMF produces output representation in which MIPS is equivalent to both NNS and MCSS. Therefore, one can immediately solve MIPS efficiently with LSH or spatial trees. Our two approaches (in Chapters 3 and 4) fall under this category.

RES#2: Sorted Indices Lists To quickly construct the candidate set, [114] proposes an algorithm *Greedy-MIPS* based on the following working assumption:

$$x_u^T y_i > x_u^T y_j \Leftrightarrow \max_{1 \leq l \leq d} x_u^{(l)} y_i^{(l)} > \max_{1 \leq l \leq d} x_u^{(l)} y_j^{(l)}, \quad (2.4)$$

In other words, the ranking of inner products between user vector x_u and item vectors $\{y_i\}_{i=1}^n$ can be approximated by the ranking of products of their maximum elements. Before observing any query, *Greedy-MIPS* constructs d different lists, in which the l -th list consists of sorted indices of elements in that dimension of all item vectors, i.e., $\{y_i^{(l)}\}$. At the querying phase, a max-heap is employed to iteratively traverse (j, l) entries of matrix $Z = [x_u^{(l)} y_j^{(l)}], \forall j \in [1, 2, \dots, n]$ and $l \in [1, 2, \dots, d]$ in a greedy sequence, and the first newly visited item indices j will be added to the candidate set C_u .

RES#3: Similarity Graph Exploration Based on the assumption that similar item vectors may constitute relevant results to the same query, *ip-NSW* [74] proposes to solve MIPS with the use of similarity graph based on the inner product between item vectors. In constructing a s -Delaunay similarity graph, at each step *ip-NSW* adds the next item i to the graph, connecting it by directed edges to H vertices, corresponding to most similar item vectors that are already in the graph. At the query phase, greedy walks on this graph are employed to efficiently determine the candidates for recommendation. At this stage, *ip-NSW* maintains a priority queue of size Q of neighbors that should be visited by the search process. Both H and Q determine the balance/trade-off between the runtime and search accuracy.

Our two proposed approaches COE and INDEXABLE BPR fall under the *indexable representation* category. However, unlike CFEE [46], IPMF [29] that fit ratings, we learn from ordinal triples. In the experiments, we include CFEE and IPMF as the baselines to validate the effectiveness of learning from ordinal triples. Another comparable baseline is BPR [83], which also relies on triples, followed by the post-learning transformation to reduce MIPS to MCSS ([10, 77]).

2.1.2 Efficient Inner Product Computation

The number of inner product computations is determined solely by the size of the candidate set C_u . Therefore, a valid strategy in the *candidate ranking* step (Figure 2.1(b)) is to reduce the cost of *each* inner product computation. There are two main approaches in this branch. *Vector quantization* attacks the dimensionality d of the latent features by operating at lower subspaces. *Discrete representation* attacks the real-valued operations by operating at binary representations.

Vector Quantization

[34] describes a vector quantization-based technique to approximate inner product search. In particular, each vector vector is mapped to M subspaces using simple chunking.

$$x_u = [x_u^{(1)}; x_1^{(2)}; \dots; x_1^{(M)}]; y_i = [y_i^{(1)}; y_i^{(2)}; \dots; y_i^{(M)}]$$

where $x_u^{(t)}, y_i^{(t)} \in \mathbb{R}^l, l = \lceil \frac{d}{M} \rceil$. The t^{th} subspace containing the t^{th} blocks of all the item vectors $\{y_i^{(t)}\}$, is quantized by a code book $U^{(t)} \in \mathbb{R}^{l \times n_C}$, where n_C is the number of quantizers in subspace t . Each item vector y_i is quantized in the t^{th} subspace as $y_i^{(t)} \sim U^{(t)} \alpha_{y_i}^{(t)}$, where $\alpha_{y_i}^{(t)}$ is one-hot indication vector. The inner product of a user query vector and an item vector is approximated as the sum of inner products with the subspace quantizers, with the cost of $\mathcal{O}(Mn_C \lceil \frac{d}{M} \rceil)$:

$$x_u^T y_i = \sum_{k=1}^K x_u^{(k)T} y_i^{(k)} \approx \sum_{k=1}^K x_u^{(k)T} U^{(k)} \alpha_{y_i}^{(k)}$$

The codebook $U^{(k)}$ and indication vector $\alpha_{y_i}^{(k)}$ are learned by minimizing the inner product quantization error for held-out queries Z with known top- k , at the cost of $\mathcal{O}(nn_C|Z|)$.

Discrete Representation

In this approach, each user/item is represented as a sequence of binary values. The inner product preference score in the Hamming space can be converted to computing the Hamming distance between user and item binary vectors, which requires simple XOR operations [118]. However, due to the binary constraints, learning the binary codes in hashing-based learning frameworks to fit the data generally requires solving a NP-hard discrete optimization problem. To arrive at the binary vector representation for users and items, there are two main approaches: *quantization-based discretization* or *optimization-based discretization*.

1. Quantization-Based Discretization consists of two phases. The first phase is relaxed opti-

mization with some specific constraints to obtain continuous latent representations for users and items. The second phase is binary quantization to convert the continuous latent representations into binary codes. [123] constructs binary codes such that the Hamming distances of a user and her preferred items are small. [117] proposes a two-stage process: constraining the learning process, so that users' preferences can be well approximated by user-item similarities and quantization algorithm that generates the binary hashing code from the learned real-valued user/item features. [63] imposes the uncorrelated constraint of binary codes for learning compact latent vectors.

2. Optimization-Based Discretization adopts classic matrix factorization formulations, while imposing further constraints on balance and decorrelation for the binary codes. For instance, DCF [118] learns from explicit feedback, while DPR [122] learns from implicit feedback with ranking-based AUC objective function. DCMF [62] also takes into account context information (such as user's age and gender, item's category and textual content), while DDL [120] combines Deep Belief Network (DBN) and Collaborative Filtering (CF). DFM [65] also learns binary codes for any side feature based on the factorization machine framework. Meanwhile DRMF [119] is based on each user's pairwise preferences, with self-paced learning.

Approaches under this category are different from our proposed methods. For one, we aim at reducing the number of candidates for recommendation at the retrieval step. For another, our models produce real-valued vectors, whereas discrete representation models produce vectors in the Hamming space, whose performances are usually bounded by conventional matrix factorization methods ([118]).

2.2 Modelling Multiple Similarity Perspectives

Similarity learning and measurement is a broad topic. Since our key thrust is incorporating in multiperspectivity, in the following we relate our work to the closest branches in the literature. Figure 2.3 presents a taxonomy for related techniques, which are summarized into two main categories: *supervised* and *unsupervised similarity learning*.

The former category refers to the setting where the feature vectors and the similarity labels are known, and the objective is to learn a mapping function from the feature domain to the label domain. We further divide this category into two approaches: *uniperspective* – methods that assume only one similarity perspective in the data and *multiperspective* – methods that assume the existence of multiple similarity perceptions in the data.

The latter category, i.e., *unsupervised similarity learning* refers to the setting where the feature vectors are unknown and only similarity observations are given. The objective is to discover the underlying similarity structure between objects or the representation of objects that explains the observations. There is further consideration on the form of similarity observations and the corresponding learning strategies. As such, we summarize the work into two subcategories. One is *structure-based similarity measure*, which takes in object-to-object relation graph where two objects are connected by an edge if they are considered similar and yields a similarity measure between any pair of objects. Another is *ordinal embedding*, which learns low-dimensional representations of objects from ordinal similarity comparisons between objects. We split each of these subcategories into either *uniperspective* or *multiperspective* approaches.

2.2.1 Supervised Similarity Learning

Uniperspective. Supervised similarity learning [113] is mostly uniperspective, dealing with the problem where the feature vectors of objects are known. It relies on training labels in the form of specific similarity values, binary labels (similar vs. dissimilar), or ranked comparisons. The

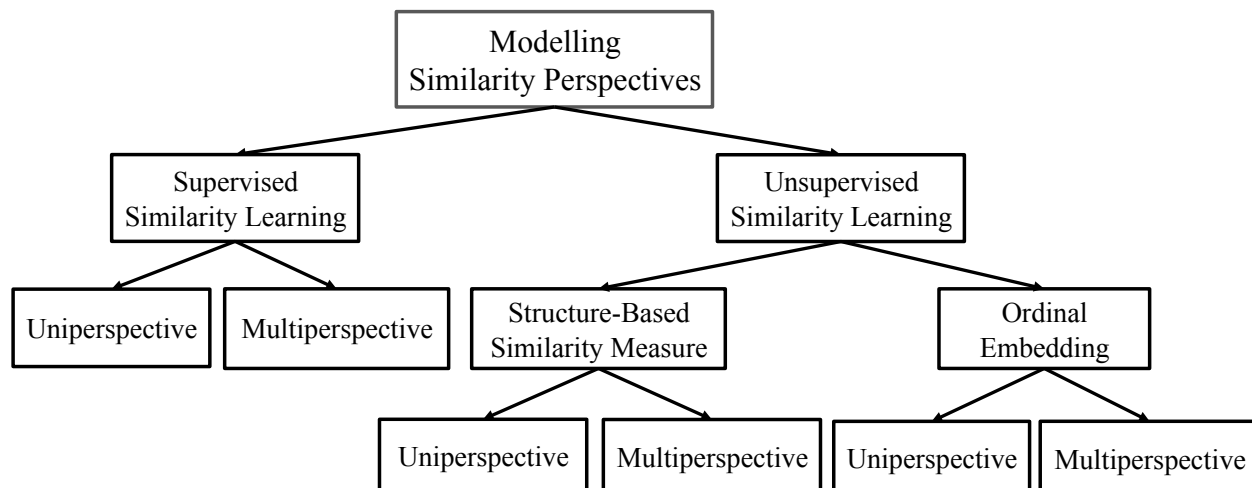


Figure 2.3: A taxonomy for multiperspective similarity modelling

focus is on learning the similarity values from features, and the use case is primarily for clustering [33, 116] or classification [109]. For multi-modal similarity, [72] uses ordinal triplets but only as side information, and still primarily relies on features.

Multiperspective. There exist other methods that could be interpreted as learning content-based multiperspective similarities. The closest is *Personalized Collaborative Clustering* or PCC [116], which uses matrix factorization to learn personalized clustering of objects; each person could be seen as a perspective and two objects are similar if they belong to the same cluster. However, instead of graph theory, it is framed in terms of similarity learning [20], where the objective is to derive a function mapping features to similarity labels. In the absence of features, it turns into learning latent representations from similar labels. In Chapter 5, we include PCC as one of our baselines. [106] proposes Conditional Similarity Networks (CSN) that jointly learns a disentangled embedding and masks that select and reweight relevant dimensions to induce embedding for different notions of similarities. However, CSN’s disentangled embedding is learnt from images’ features, which are assumed not known in the general setting. [80] proposes MVE, a network embedding technique that learns node representations by leveraging information from multiple-view and labeled data. Although it deals with multiple types of proximities between objects, it seeks to

learn a robust node representations rather than to reflect a variety of similarity perspectives over the graph nodes.

2.2.2 Unsupervised Similarity Learning

Structure-based Similarity Measure

In this section, we review related work on structured-based similarity measure.

Uniperspective: Most of the previous works in structural-based similarity measurement are based on SimRank [40]. We first briefly review SimRank. Given a graph $G(V, E)$, SimRank measures the similarity between two graph nodes based on the graph structure.

Formally, the SimRank score between two vertices a, b is defined as follows:

$$S(a, b) = \begin{cases} \frac{C}{|N(a)||N(b)|} \sum_{i=1}^{|N(a)|} \sum_{j=1}^{|N(b)|} S(N_i(a), N_j(b)), & \text{if } a \neq b, \\ 1, & \text{if } a = b \end{cases} \quad (2.5)$$

in which C is the damping factor between 0 and 1; $N(a)$ and $N(b)$ comprise the neighbors of a and b respectively. In other words, the SimRank score between a, b is defined in terms of the SimRank scores of their neighbors. The base case is the similarity between a vertex and itself, which is always 1. If a vertex a has no neighbor, then we have $S(a, b) = 0$ for any vertex $b \neq a$.

SimRank has been extended in diverse directions, of which we cite a few here. [58] proposed non-iterative computation for dynamically changing graphs. [36] parallelized the similarity computation using GPUs. [59] optimized the computation when the target was computing the similarity of a single pair of objects. [53, 67, 102, 115] sought to speed up the computation for extremely large graphs. In the context of translation lexicons, [24] presented a modification of SimRank to measure similarity across two graphs; this is distinct from the notion of multiperspective as there is only one perspective.

Besides SimRank, there are other notions of graph-based similarity. Most are based on random walk variants [103] (e.g., Personalized PageRank [39, 84] or hubs and authorities [30]). [95] was concerned with metapaths in heterogeneous information networks.

Multiperspective: We are not aware of any SimRank extension incorporating multiperspectivity similarity. In Chapter 5, we further explore this direction and propose a graph-theoretic similarity measure MP-SIMRANK that deals with multiple input graphs over the same object nodes, each representing one similarity perspective and producing a score of any two objects for a specific perspective. The design of MP-SIMRANK natively supports multiperspectivity.

Ordinal Embedding

This section reviews techniques and methods related to our second multiperspective approach SCORE (Chapter 6).

Uniperspective: Since SCORE learns from a collection of ordinal comparisons between objects, it falls under the umbrella of ordinal embedding. Previous work in ordinal embedding focuses on the single map scenario, and are not designed for multi-perspective scenario. Some are based on quadruplets of objects (distance between the first pair in comparison to that between the second pair). Generalized Non-metric MultiDimensional Scaling or GNMDS [3] is based on semi-definite programming. The state-of-the-art is Soft Ordinal Embedding or SOE [101]. Other methods are based on ordinal triplets of objects, such as t-Stochastic Triplet Embedding or tSTE [105], which has been compared favorably to Crowd Kernel Learning or CKL [97]. As there is no existing solution for multi-perspective ordinal embedding, we compare to the latest models SOE and tSTE in Chapter 6.

Multiperspective: We propose a *collaborative* approach to learning multiple maps from conditional ordinal comparisons by considering the perspectives jointly via a shared representation, while still respecting each perspective via perspective-specific representations. While the concept of multiple embedding maps for the same objects has been introduced in different contexts

[6, 104], our framework SCORE with shared representation is novel. [104] considers a different type of input (non-metric distances). Meanwhile, Multi-View Triplets Embedding or MVTE in [6] also considers ordinal comparisons, but in a different scenario where the associations between perspectives and object triplets are not known and to be derived. Their objective is to split the ordinal observations into a specified number of latent groups, each is associated with an embedding map. In Chapter 6, we includes MVTE as one of our baselines for comparison.

Multi-view network embedding (MVNE) that deals with multi-view graphs, which consists of several distinct sets of edges over the same entities/nodes, has received extensive attention recently ([66, 90, 96]). However, existing works in multi-view network embedding aim to combine the information from all the views and obtain a single low-dimensional feature representation of the nodes that preserves the structural and semantic relations among them. Though having similar input as our graph-theoretic framework MP-SIMRANK to some extent, this is distinct from us as our formulation aims at learning multiple perspective-specific similarity measure for any pair of objects. The final output of MVNE reflects a common view on the relationships between graph nodes. There, the focus is not so much to reflect a variety of perspectives as to arrive at the common consensus.

Multi-task learning [18] is a class of learning problems that leverage on the commonality between several sufficiently related tasks. It is most advantageous when there may not be sufficient training data for each task. Aside from having a common general framework, our work is a distinct formulation from previous works. The particular realization of multi-task learning depends on the specific problem at hand. In our distinct case, that is ordinal embedding. Related formulations include metric learning for nearest neighbor classification [79, 112], feature selection [7], and collaborative clustering [116]. Aside from different formulations, most of the works rely on features rather than ordinal similarity comparisons.

The term “embedding” is also used in contexts of representation learning such as distributed representation [13] or distributional representation [16]. We do not rely on features; rather we

learn from similarities or distances (or ordinal comparisons thereof). Ours are low-dimensional Euclidean coordinates that directly support visual analysis, whereas the above works would require a separate method for visualization.

Part I

Preference Learning: Efficient Recommendation Retrieval

Chapter 3

Collaborative Ordinal Embedding

In this chapter, we present a collaborative filtering approach via Euclidean ordinal embedding for efficient recommendation retrieval. Euclidean embedding takes as input distances (or their ordinal relationships), and outputs low dimensional latent coordinates for each point that would preserve the input as much as possible. Because they operate in the Euclidean space, the coordinates support nearest neighbor search using geometric index structures such as spatial trees. In addition to supporting efficient recommendation retrieval, Euclidean embedding could also enable other applications such as visualization (i.e., when $d = 2$ or 3). The proposed model Collaborative Ordinal Embedding or COE is based on generative modelling of ordinal triples. Experiments on publicly available datasets show that COE outperforms the baselines both in terms of retrieval efficiency and information preservation for ordinal data.

3.1 Introduction

We are interested in ordinal embedding approach for collaborative filtering. Each user and item is represented by a coordinate in a low-dimensional Euclidean space, and the relationships among data points are modelled through Euclidean distances in that space. Most of the previous works

on embedding focus on metric embedding, whose objective is to preserve the pairwise similarities among data points [22, 86, 89, 100]. This is applicable when the main relationships among objects is similarity, e.g., images of handwritten digits or human faces [22].

Ordinal data refers to data where the ranking established by numerical values is more significant than the exact values. Such a representation is very common in the domain of preferences, where users express how much they like various items. For instance, after purchasing a product on Amazon, a user may leave an explicit rating. While listening to music at Spotify, a user leaves implicit traces of her liking for a track or an artist by the frequencies at which she consumes them. In both explicit and implicit cases, it is important to model the relative sense of whether an item is preferred to another.

Problem. Embedding for ordinal preference data seeks to preserve the ordinal preference of users over items through their Euclidean coordinates. Our goal is to achieve ordinal *co-embedding*, where multiple entities are involved (e.g., users and items), and cross-type ordinal relationships are key (e.g., users express preferences over items). Suppose for each user, we are given his or her pairwise rankings over items. A triple t_{uij} indicates that a user u prefers an item i to a different item j . As output, every user and every item would be respectively assigned a latent coordinate (to be learned) in a d -dimensional Euclidean space. User u 's preference for item i to item j is expressed through a shorter distance between u and i than between u and j . Such a representation supports *efficient retrieval* for recommendation queries, such as which items are the closest (most preferred) to a user. Euclidean geometry fits the mould of spatial data management, allowing it to benefit from such developments as spatial indexing [12] and efficient nearest-neighbor query processing [85]

In addition to recommendation retrieval, embedding could also enable other applications arising from its Euclidean metric properties. One potential application is *visualization* for preference analytics. Figure 3.1 illustrates an example 2d embedding for three users (blue triangles) and three items (purple crosses), specifying their respective coordinates. Through our spatial perception of

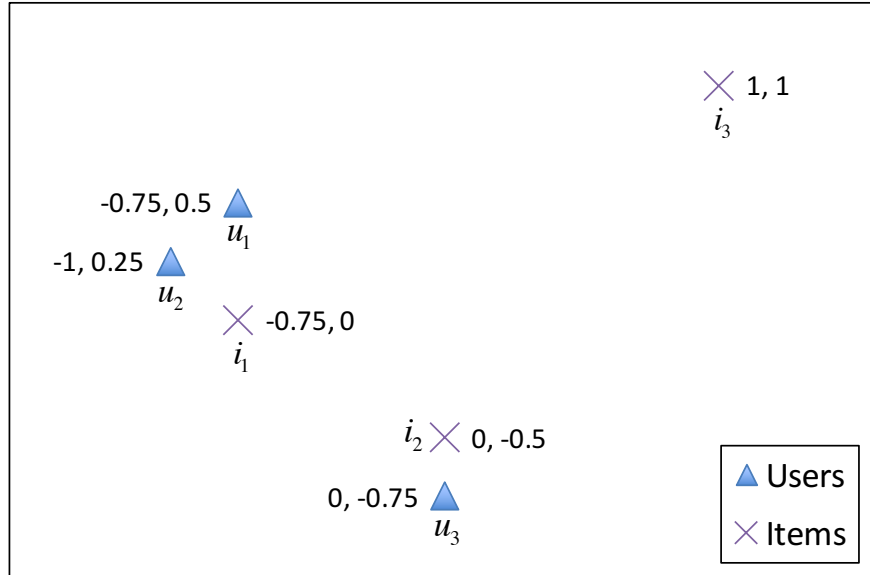


Figure 3.1: Euclidean Embedding of Users & Items

the relative distances, we can immediately tell that the user u_1 prefers item i_1 the most (closest), followed by item i_2 , and item i_3 the least (furthest). Such information leaps out at us without our having to consciously compute the distances. For another potential application, as embedding relies on building a compact model for user preferences, it may eventually enable an interactive interface for training recommender systems. In text domain [70], we may seek an embedding that preserves the relative importance of words to a document (for summarization).

Approach. While there has been prior work on ordinal embedding [3, 52, 101], our work is novel in a couple of fundamental respects. First, the “classical” ordinal embedding is formulated mainly for one object type, e.g., cities [101], images [3]. It enforces that for same-type quadruple of objects $\langle i, j, k, l \rangle$, if i is closer to j in the original data than k is to l , the same ordinal relationship should hold in the embedding space. This presumes that the primary information is similarity among objects. In contrast, our primary objective is based on *ordinal preference ranking*, i.e., the ranking of items according to user preferences. For instance, it is possible for two users to be “similar”, say in terms of their demographics or their habits of watching horror movies, and yet to have different rankings over specific items.

Moreover, because classical ordinal embedding deals with within-type ordinal relationships, it implicitly assumes that there is one underlying reality to approximate, e.g., distances of cities in the map [101]. However, for many ordinal datasets, there may not be a singular ground-truth reality. For preference data, each user imposes his or her own ranking on the items, and these rankings may be different and at times conflicting. This fundamental difference motivates two distinguishing aspects of our approach. Because a common embedding space needs to accommodate the diverse preferences of users, we harness the collaborative effect among users and among items. In order to capture the variance in the rankings induced by preferences of different users or items in a principled way, we also formulate our model in terms of probabilistic generative modelling.

Contributions and Organization. We provide the formal problem statement in Section 3.2.1. We make the following contributions towards the problem. *First*, in Section 3.2, we propose a new embedding model, called *Collaborative Ordinal Embedding* or COE. This model is notable in its generative modeling of ordinal embedding allowing various types of triples, as well as in its objective function with both a penalty component for violated observations and a reward component for preserved observations on a smooth continuous spectrum modeled by probabilistic Sigmoid distribution. *Second*, in Section 3.2.4, we describe COE’s learning algorithm to derive the embedding coordinates that maximize the posterior probability of the generative model based on stochastic gradient ascent. *Third*, in Section 3.3, comprehensive experiments on publicly available datasets show that COE outperforms the baselines in: preserving the observed pairwise comparisons and predicting unseen pairwise comparisons expressed as relative distances in the Euclidean space, and achieving efficient recommendation retrieval via spatial indexing. We then briefly summarize the work in Section 3.4.

3.2 Framework

3.2.1 Problem Formulation

We formally define the problem addressed in this chapter, which is *co-embedding* of users and items based on ordinal preference triplets.

Input. The set of users is \mathcal{U} , and u or v refers to a user. The set of items is \mathcal{I} , and i or j refers to an item. The input is a multiset of triples $\mathcal{T} = \mathcal{T}_A \cup \mathcal{T}_B$, consisting of “type-A” triples $\mathcal{T}_A \subset \mathcal{U} \times \mathcal{I} \times \mathcal{I}$ and “type-B” triples $\mathcal{T}_B \subset \mathcal{U} \times \mathcal{U} \times \mathcal{I}$. A type-A triple $t_{uij} \in \mathcal{T}_A$ relates a user $u \in \mathcal{U}$ and two different items $i, j \in \mathcal{I}$, indicating u ’s preferring i to j . A type-B $t_{uvi} \in \mathcal{T}_B$ indicates a user u has greater preference over i than user v does.

Such triples form a general representation of preferences over one object type as expressed by the other object type. There are examples abound in both explicit and implicit feedback scenarios. Triples can be derived from ratings, e.g., when u assigns a higher rating to i than to j . Other than ratings, it could also model implicit feedback [83]. For cable TV, u may watch the channel i but not j , or spend a longer time watching i than j [37]. For Web search, u may click on the result i after skipping j [81]. Outside of preference domain, in text, a word i may be more frequent than another word j in document u . Alternatively, document u may be more relevant to word i than document v does. In addition to their greater availability, triples are also more reliable than ratings. Studies in psychology [44] showed that human users were more reliable at determining relative ordering than at assigning absolute numbers.

While we focus on cross-type triples, it is feasible to accommodate triples involving three objects of the same type, e.g., user u is more “similar” to user v than to user v' . Here, we will not concentrate on such similarity-based triples. More generally, we can use triple form $(o_{\tau_1}^1, o_{\tau_2}^2, o_{\tau_3}^3)$, where $o_{\tau_i}^i$ are objects of types τ_i , ($i = 1, 2, 3$) respectively, to represent ordinal relations among multiple objects. The framework can be extended naturally by adding latent variables for objects of each type. For simplicity, we only present our model with two types.

Output. Given \mathcal{T} , the goal is to assign a coordinate $x_u \in \mathbb{R}^d$ to each user $u \in \mathcal{U}$, as well as a coordinate $y_i \in \mathbb{R}^d$ to each item $i \in \mathcal{I}$, such that their distances in \mathbb{R}^d preserve the relative ordering indicated by the triples. We denote the collection of all user coordinates as X and the collection of all item coordinates as Y . The coordinates of users and items lie in the same d -dimensional Euclidean space.

Problem 3.1 (Ordinal Co-Embedding). *Given a set of triples \mathcal{T} , find the set of user coordinates X and item coordinates Y , so as to meet the following respective condition for as many triples in \mathcal{T} as possible, i.e.,*

$$\begin{aligned} t_{uij} \in \mathcal{T}_A &\Rightarrow \|x_u - y_i\| < \|x_u - y_j\|, \\ t_{uvi} \in \mathcal{T}_B &\Rightarrow \|x_u - y_i\| < \|x_v - y_i\| \end{aligned}$$

We now describe our proposed model, called *Collaborative Ordinal Embedding* or COE. The challenge is integrating the diverse triples into the same low-dimensional Euclidean space. The input triples \mathcal{T} may also suffer from sparsity, variance, and uncertainties, in the form of incompleteness (not all possible triples are specified), inconsistency (some triples are conflicting), and repetitions (some triples may occur more than once). Yet the final objective is a unified view for all items and users.

3.2.2 Generative Model

To achieve this, we harness the “collaborative” effect. Since item coordinates are shared across users, users with similar coordinates would have similar ordinal relationships with items. To develop this probabilistically, we design a graphical model as is illustrated in Figure 3.2.

We model each user coordinate and each item coordinate as real-valued latent random variables x_u and y_i respectively. For each triple $\langle u, i, j \rangle$ where $i < j$, we associate it with a binary random variable c_{uij} . When c_{uij} takes on the value of 1, it corresponds to an instance of $t_{uij} \in \mathcal{T}$. When

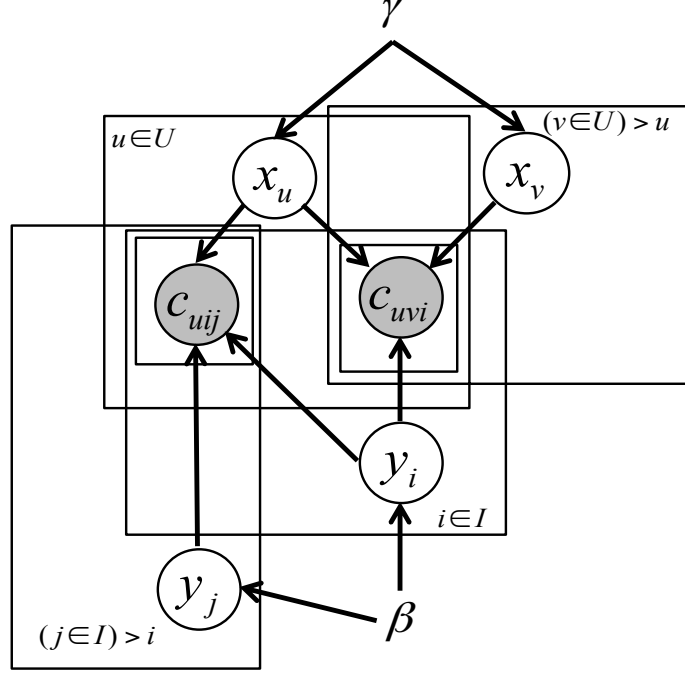


Figure 3.2: Collaborative Ordinal Embedding (COE)

$c_{uij} = 0$, it corresponds to an instance of $t_{uji} \in \mathcal{T}$. In Figure 3.2, c_{uij} is shaded and lies within its own plate, i.e., it is observed and there could be multiple instances. Correspondingly, for each triple $\langle u, v, i \rangle$ where $u < v$, we associate it with a variable c_{uvi} . The state of c_{uij} (or c_{uvi}) and the generation of t_{uij} (or t_{uvi}) are related to user and item coordinates through the following generative process.

The generative process of COE is as follows:

1. For each user $u \in \mathcal{U}$:
 - Draw u 's coordinate: $x_u \sim \text{Normal}(0, \gamma^2 \mathbf{I})$,
2. For each item $i \in \mathcal{I}$:
 - Draw i 's coordinate: $y_i \sim \text{Normal}(0, \beta^2 \mathbf{I})$,
3. For each triple $\langle u, i, j \rangle \in \mathcal{T}_A$:
 - Draw $c_{uij} \sim \text{Bernoulli}(P(c_{uij} = 1 \mid x_u, y_i, y_j))$,
 - If $c_{uij} = 1$, generate a triple instance t_{uij} ,

- Else ($c_{uij} = 0$), generate a triple instance t_{uji} .
4. For each triple $\langle u, v, i \rangle \in \mathcal{T}_B$:
- Draw $c_{uvi} \sim \text{Bernoulli}(\text{P}(c_{uvi} = 1 \mid x_u, x_v, y_i))$.
 - If $c_{uvi} = 1$, generate a triple instance t_{uvi} ,
 - Else ($c_{uvi} = 0$), generate a triple instance t_{vui} .

In Step 1 and Step 2, we generate the users' and items' coordinates, placing zero-mean multivariate spherical Gaussian priors on these coordinates, with γ^2 and β^2 controlling the respective variances of the Normal distributions. \mathbf{I} denotes the identity matrix.

In Step 3, we generate type-A triples involving one user and two items, by drawing the outcome for c_{uij} from a Bernoulli process, where the parameter is specified by the probability $\text{P}(c_{uij} = 1 \mid x_u, y_i, y_j)$ of generating a triple instance t_{uij} . In Step 4, we generate type-B triples involving two users and one item.

3.2.3 Triple Probability Function

A crucial component is how the latent coordinates of users and items would generate the pairwise comparisons in \mathcal{T} . This bridge between the hidden variables and the observations is the triple probability function. To keep the discussion streamlined, in the following we discourse on type-A triple of the form t_{uij} , but a similar principle applies in a symmetric manner to type-B triples.

The principle in relating latent coordinates to a triple t_{uij} is: if u prefers i to j , the distance from x_u to y_i is shorter than that from x_u to y_j . The more evidence there is that u prefers i to j , the closer x_u should be to y_i than to y_j . To realize this intuition, we express the probability $\text{P}(c_{uij} = 1 \mid x_u, y_i, y_j)$ in terms of the distances $\|x_u - y_i\|$ and $\|x_u - y_j\|$. Let Δ_{uij} be a quantity expressed in terms of these distances, such that Δ_{uij} is higher the more u prefers i to j . One

realization of Δ_{uij} is Equation 3.1.

$$\Delta_{uij} = \|x_u - y_j\| - \|x_u - y_i\| \quad (3.1)$$

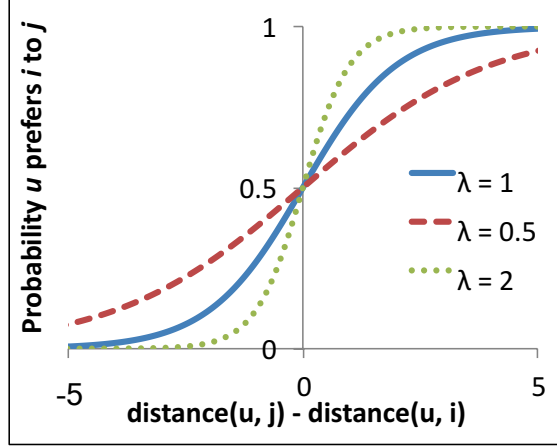
Because t_{uij} and t_{uji} are two opposite sides of the same coin, we have $P(t_{uij} | x_u, y_i, y_j) = 1 - P(t_{uji} | x_u, y_i, y_j)$. Δ_{uij} has a bearing on these probabilities as follows.

- For $\Delta_{uij} > 0$, t_{uij} is more likely, $P(t_{uij}|x_u, y_i, y_j) > 0.5$.
- For $\Delta_{uij} < 0$, t_{uji} is more likely, $P(t_{uji}|x_u, y_i, y_j) < 0.5$.
- For $\Delta_{uij} = 0$, they are indifferent, $P(t_{uij}|x_u, y_i, y_j) = 0.5$.

To relate Δ_{uij} to these probabilities, here we propose the sigmoid $\sigma(\cdot)$ function, as shown in Equation 3.2. This is without prejudice on other potential functions that could realize the above properties as well. As shown in Figure 3.3, the probability that u prefers i to j tends towards 1 as $\Delta_{uij} \rightarrow \infty$, and tends towards 0 as $\Delta_{uij} \rightarrow -\infty$. To model the probabilities of triples as a function of Δ_{uij} (or Δ_{uvi}), we identify two possible functions.

$$P(c_{uij} = 1 | x_u, y_i, y_j) = \frac{1}{1 + e^{-\lambda \cdot \Delta_{uij}}} \quad (3.2)$$

This function allows us to model both a penalty for violating observed triples (probability mass < 0.5), and a reward for preserving observed triples (probability mass > 0.5). This is different from classical ordinal embedding. For instance, the state-of-the-art SOE [101] only has a penalty component, but no reward. This holds two advantages for COE. First, there is a smoother spectrum of penalty and reward over a continuous function vs. the cliff effect for SOE. Second, there is discrimination among triples with more vs. less evidence earning different probability masses. The scaling parameter λ controls the slope of the function. The greater is λ , the steeper is the penalty/reward. The λ setting may empirically tuned.



Sigmoid Function

Figure 3.3: Triple Probability Function

3.2.4 Learning Algorithms

Given \mathcal{T} as input observations, our goal is to learn the latent coordinates X and Y with the highest posterior probability $P(X, Y|\mathcal{T})$. Through Bayes' Theorem, we have $P(X, Y|\mathcal{T}) = P(\mathcal{T}, X, Y)/P(\mathcal{T})$. Since $P(\mathcal{T})$ does not affect the model parameters, the goal is to maximize the joint probability, as shown in Equation 3.3.

$$\arg \max_{X, Y} P(\mathcal{T}, X, Y|\gamma, \beta) \quad (3.3)$$

The joint probability is decomposed into four terms:

$$P(\mathcal{T}, X, Y|\gamma, \beta) = P(X|\gamma) \times P(Y|\beta) \times P(\mathcal{T}|X, Y),$$

in which,

$$\begin{aligned}
P(X|\gamma) &= \prod_{u \in \mathcal{U}} (2\pi\gamma^2)^{-\frac{D}{2}} e^{-\frac{1}{2\gamma^2}\|x_u\|^2}, \\
P(Y|\beta) &= \prod_{i \in \mathcal{I}} (2\pi\beta^2)^{-\frac{D}{2}} e^{-\frac{1}{2\beta^2}\|y_i\|^2}, \\
P(\mathcal{T}_A|X, Y) &= \prod_{t_{uij} \in \mathcal{T}_A} P(c_{uij} = 1 \mid x_u, y_i, y_j), \\
P(\mathcal{T}_B|X, Y) &= \prod_{t_{uvi} \in \mathcal{T}_B} P(c_{uvi} = 1 \mid x_u, x_v, y_i).
\end{aligned}$$

Maximizing the joint probability is equivalent to maximizing its logarithm, shown below. To simplify the parameters, we set $\gamma = \beta$, and equate both $\frac{1}{\gamma^2}$ and $\frac{1}{\beta^2}$ to a common regularization parameter η .

$$\mathcal{L} = \ln P(X|\gamma) + \ln P(Y|\beta) + \ln P(\mathcal{T}|X, Y) = \ln P(\mathcal{T}|X, Y) - \eta \sum_{u \in \mathcal{U}} \|x_u\|^2 - \eta \sum_{i \in \mathcal{I}} \|y_i\|^2$$

To find the coordinates that maximize the joint probability, we employ stochastic gradient ascent for computational efficiency, an important factor given the potentially huge size of pairwise comparisons.

The gradient of \mathcal{L} w.r.t. each user coordinate x_u is:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial x_u} &= \sum_{\{i,j: t_{uij} \in \mathcal{T}_A\}} \frac{\lambda e^{-\lambda \Delta_{uij}}}{1 + e^{-\lambda \Delta_{uij}}} \left(\frac{x_u - y_j}{\|x_u - y_j\|} - \frac{x_u - y_i}{\|x_u - y_i\|} \right) \\
&+ \sum_{\{i,v: t_{uvi} \in \mathcal{T}_B\}} \frac{\lambda e^{-\lambda \Delta_{uvi}}}{1 + e^{-\lambda \Delta_{uvi}}} \left(\frac{y_i - x_u}{\|y_i - x_u\|} \right) \\
&+ \sum_{\{i,v: t_{vui} \in \mathcal{T}_B\}} \frac{\lambda e^{-\lambda \Delta_{vui}}}{1 + e^{-\lambda \Delta_{vui}}} \left(\frac{-y_i + x_u}{\|y_i - x_u\|} \right) - \eta \cdot x_u
\end{aligned}$$

The gradient w.r.t. each item coordinate y_i is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial y_i} = & \sum_{\{u,v: t_{uvi} \in \mathcal{T}_B\}} \frac{\lambda e^{-\lambda \Delta_{uvi}}}{1 + e^{-\lambda \Delta_{uvi}}} \left(\frac{y_i - x_v}{\|y_i - x_v\|} - \frac{y_i - x_u}{\|y_i - x_u\|} \right) \\ & + \sum_{\{u,j: t_{uj} \in \mathcal{T}_A\}} \frac{\lambda e^{-\lambda \Delta_{uj}}}{1 + e^{-\lambda \Delta_{uj}}} \left(\frac{x_u - y_i}{\|x_u - y_i\|} \right) \\ & + \sum_{\{u,j: t_{uji} \in \mathcal{T}_A\}} \frac{\lambda e^{-\lambda \Delta_{uji}}}{1 + e^{-\lambda \Delta_{uji}}} \left(\frac{-x_u + y_i}{\|x_u - y_i\|} \right) - \eta \cdot y_i \end{aligned}$$

Algorithm 2 describes the stochastic gradient ascent algorithm for COE. It first initializes the coordinates of users and items. In each iteration, a triple is randomly selected from \mathcal{T} , and the model parameters are updated based on the gradients above, with a decaying learning rate ϵ over time. The complexity is $\mathcal{O}(|\mathcal{U}| \times |\mathcal{I}|^2 + |\mathcal{U}|^2 \times |\mathcal{I}|)$. In case of having triples of multi-type ordinal relations among multiple objects, the complexity is still a polynomial of variables with highest degree is 3.

3.3 Experiments

Our objective is to investigate the effectiveness of the proposed COE, for visualization in low-dimensional Euclidean space and for efficient retrieval of personalized recommendations.

Datasets. While COE assumes ordinal triples as inputs, we experiment with publicly available datasets with numerical values and derive the triples accordingly. This allows us to compare to baselines that work directly with the numerical values. We work with four datasets of two categories, and their sizes are listed in Table 3.1.

The first category includes rating-based preference datasets: *MovieLens-100K*¹ and *Netflix*². The object types are users and movies (items). The raw observations are ratings. As in [26], we apply Z-score normalization, which compensates for different rating means and rating spreads to

¹<http://grouplens.org/datasets/movielens/100k>

²<http://www.cs.uic.edu/~liub/Netflix-KDD-Cup-2007.html>

Algorithm 1 Stochastic Gradient Ascent for COE

```
1: Initialize  $x_u$  for  $u \in \mathcal{U}$ 
2: Initialize  $y_i$  for  $i \in \mathcal{I}$ 
3:
4: while not converged do
5:   Draw a triple at random from  $\mathcal{T}$ .
6:   if it is a type-A triple  $t_{uij} \in \mathcal{T}_A$  then
7:      $x_u \leftarrow x_u + \epsilon \cdot \left[ \frac{\lambda e^{-\lambda \Delta_{uij}}}{1 + e^{-\lambda \Delta_{uij}}} \left( \frac{x_u - y_j}{\|x_u - y_j\|} - \frac{x_u - y_i}{\|x_u - y_i\|} \right) - \eta \cdot x_u \right]$ 
8:      $y_i \leftarrow y_i + \epsilon \cdot \left[ \frac{\lambda e^{-\lambda \Delta_{uij}}}{1 + e^{-\lambda \Delta_{uij}}} \left( \frac{x_u - y_i}{\|x_u - y_i\|} \right) - \eta \cdot y_i \right]$ 
9:      $y_j \leftarrow y_j + \epsilon \cdot \left[ \frac{\lambda e^{-\lambda \Delta_{uij}}}{1 + e^{-\lambda \Delta_{uij}}} \left( \frac{-x_u + y_j}{\|x_u - y_j\|} \right) - \eta \cdot y_j \right]$ 
10:   end if
11:
12:   if it is a type-B triple  $t_{uvi} \in \mathcal{T}_B$  then
13:      $x_u \leftarrow x_u + \epsilon \cdot \left[ \frac{\lambda e^{-\lambda \Delta_{uvi}}}{1 + e^{-\lambda \Delta_{uvi}}} \left( \frac{y_i - x_u}{\|y_i - x_u\|} \right) - \eta \cdot x_u \right]$ 
14:      $x_v \leftarrow x_v + \epsilon \cdot \left[ \frac{\lambda e^{-\lambda \Delta_{uvi}}}{1 + e^{-\lambda \Delta_{uvi}}} \left( \frac{-y_i + x_v}{\|y_i - x_v\|} \right) - \eta \cdot x_v \right]$ 
15:      $y_i \leftarrow y_i + \epsilon \cdot \left[ \frac{\lambda e^{-\lambda \Delta_{uvi}}}{1 + e^{-\lambda \Delta_{uvi}}} \left( \frac{y_i - x_v}{\|y_i - x_v\|} - \frac{y_i - x_u}{\|y_i - x_u\|} \right) - \eta \cdot y_i \right]$ 
16:   end if
17: end while
18:
19: Return  $\{x_u\}_{u \in \mathcal{U}}$  and  $\{y_i\}_{i \in \mathcal{I}}$ 
```

make ratings more comparable across users. We then generate a type-A triple t_{uij} for each instance where a user u has higher normalized rating on an item i than on item j , and a type-B triple t_{uvi} for each instance where a user u has higher normalized rating on i than v does. We do not generate any triple involving non-rated items. For *MovieLens-100K*, *Netflix*, each user has been pre-conditioned by the original dataset to have at least 20 ratings. We further ensure that each item has at least 4 ratings. We find similar practice in other works [83].

The second category are based on cooccurrences: *Last.fm*³ and *20News*⁴. *Last.fm* contains users' listening frequencies to music artists (items). As in above, we retain users with at least 20 items, and items with at least 4 users. To show applicability beyond preferences, we include

³<http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip>

⁴<http://web.ist.utl.pt/acardoso/datasets/>

	users/ docs	items/ words	ratings/ observations	type-A triples $\langle u, i, j \rangle$	type-B triples $\langle u, v, i \rangle$
MovieLens-100K	943	1,413	99,543	7.80×10^6	8.22×10^6
Netflix	429,102	17,769	99,841,834	2.29×10^{10}	2.51×10^{11}
Last.fm	1,772	3,521	72,955	1.50×10^6	3.87×10^6
20News	15,744	14,414	1,076,900	5.61×10^7	2.19×10^8

Table 3.1: Datasets Summary

the text-based *20News*, which has documents (“users”) and words (“items”). We downloaded the dataset with stop words removed and the remaining words stemmed. Following the standard practice by the baseline [32], we filter out extremely infrequent words (less than 5 documents), and extremely frequent words (top 100 most frequent). For both datasets, the raw observation is the term frequency of a word (or an item) in a document (or a user). To normalize the effect of document length, we divide each word’s frequency by the document length, and generate triples from these normalized term frequencies.

3.3.1 Comparison to Embedding Baselines

For the applications we have in mind (e.g., visualization), Euclidean dimensionalities of 2 or 3 make the most sense. In this section, we investigate all models in terms of how well the learnt Euclidean embeddings preserve the *seen* information and generalize to *unseen* information. Because of the different natures of the two categories of datasets, which involve some different comparative baselines, in the following we organize the experiments into two sections, one for each dataset category.

Rating-Based Datasets

The first baseline is the embedding designed to fit the numerical rating values, i.e., CFEE [45]. As its authors have not made their implementation available, we implement it in Java. The second baseline is matrix factorization based on pairwise comparisons BPR(MF) [83] with one dimen-

sion, followed by [8]’s Euclidean transformation into two dimensions, denoted as BPR(MF)+. For BPR(MF), we use the Java implementation in LibRec⁵. We tune the respective parameters for the best performance on each dataset.

Metrics. We apply several metrics that allow an evaluation of the various methods in terms of information preservation in two-dimensional Euclidean space.

As is common for dimensionality reduction [42], the primary aim is how well the reduced dimensionality preserves the observed data. The first and main metric is *preservation accuracy*, the extent to which the information within the observed triples is preserved by the coordinates. For a user u , let $\mathcal{T}_{observed}^u$ denote the triples involving u . For u , the preservation accuracy is defined as the fraction of her triples for which the coordinates reflect the preference direction in the triples. Overall, the preservation accuracy is the average of users’ preservation accuracies, as shown in Equation 3.4. By doing so, it is not biased towards few users with many ratings at the expense of many users with few ratings.

$$\frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\{t_{uij} \in \mathcal{T}_{observed}^u : \|x_u - y_i\| < \|x_u - y_j\|\}|}{|\mathcal{T}_{observed}^u|} \quad (3.4)$$

As mentioned in Section 3.2.1, we do not presume that the input set of triples is complete. It is therefore interesting to study how well the learnt coordinates could generalize to unseen triples. We introduce a secondary metric, *prediction accuracy*, the extent to which the coordinates can infer the preference directions of hidden triples \mathcal{T}_{hidden} . For an embedding solution as a whole, the prediction accuracy is derived from user-level accuracies, as shown in Equation 3.5.

$$\frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|\{t_{uij} \in \mathcal{T}_{hidden}^u : \|x_u - y_i\| < \|x_u - y_j\|\}|}{|\mathcal{T}_{hidden}^u|} \quad (3.5)$$

The above definitions are for type-A triples. A corresponding version is defined for type-B triples.

⁵<http://www.librec.net/>

We split the ratings randomly into 80% $\mathcal{R}_{observed}$ and 20% \mathcal{R}_{hidden} , in a stratified manner to maintain the same ratio for every user. The observed set of triples $\mathcal{T}_{observed}$ is formed within $\mathcal{R}_{observed}$. The hidden set of triples \mathcal{T}_{hidden} includes triples formed within \mathcal{R}_{hidden} , as well as triples involving one rating each from $\mathcal{R}_{observed}$ and \mathcal{R}_{hidden} . Ordinal-based methods learn from $\mathcal{T}_{observed}$, while the rest learn from with $\mathcal{R}_{observed}$. Both preservation and prediction accuracies range from 0% (worst) to 100% (best). For statistical significance, we average the results across 10 random (80:20) splits.

These metrics are general for ordinal triples. Since the ordinal triples are derived from ratings, we include a rating-based third measure: *average rating among k -nearest neighbors (k -NN)*. Intuitively, a good embedding with high preservation should place higher-rated items closer to the user. Given a user, we identify the k -nearest rated items based on their Euclidean distances in the embedding space, and average the user’s ratings on those items. Symmetrically, this can be measured from each item’s point of view. We average this across users and items for $k = 5$.

	Preservation Accuracy			Prediction Accuracy			5-NN Avg Rating		
	Type-A	Type-B	H-Mean	Type-A	Type-B	H-Mean	Users	Items	H-Mean
COE	75.0%	65.0%	69.6%	64.0%	59.0%	61.4%	4.33	3.58	3.92
CFEE	67.2%	62.4%	64.7%	59.7%	60.3%	60.0%	4.03	3.50	3.75
BPR(MF)+	68.4%	60.9%	64.5%	62.1%	59.1%	60.5%	4.13	3.40	3.73

Table 3.2: Rating-Based Dataset (MovieLens-100K): COE vs. Other Baselines

	Preservation Accuracy			Prediction Accuracy			5-NN Avg Rating		
	Type-A	Type-B	H-Mean	Type-A	Type-B	H-Mean	Users	Items	H-Mean
COE	75.2%	66.3%	70.4%	63.3%	61.2%	62.2%	4.51	3.74	4.09
CFEE	66.0%	62.4%	64.2%	58.9%	61.4%	60.2%	4.10	3.74	3.91
BPR(MF)+	68.2%	60.2%	64.0%	60.3%	58.8%	59.6%	4.00	3.15	3.52

Table 3.3: Rating-Based Dataset (Netflix): COE vs. Other Baselines

In Table 3.2, we compare COE to the baselines with *MovieLens-100K* dataset. CFEE, which fits rating values directly, generally achieves lower accuracies. Since rating and visualization spaces are distinct, forcing their unification may not obtain the best embedding to preserve the

triples. BPR(MF)+, which learns matrix factorization by pairwise ranking, followed by Euclidean transformation, also achieves lower results. As mentioned in Section 2.1, the Euclidean transformation applied to BPR(MF)'s output could only preserve the pairwise comparisons of either type-A triples or type-B triples (not both at once). However, we present the best results for both transformations, which evidently are still lower than COE's. This signifies that for visualization, directly modelling Euclidean distance, such as in COE, leads to better visualization.

Table 3.3 shows the results for the much-larger *Netflix* dataset, which also support the major observations made above. The differences between COE and the baselines are statistically significant.

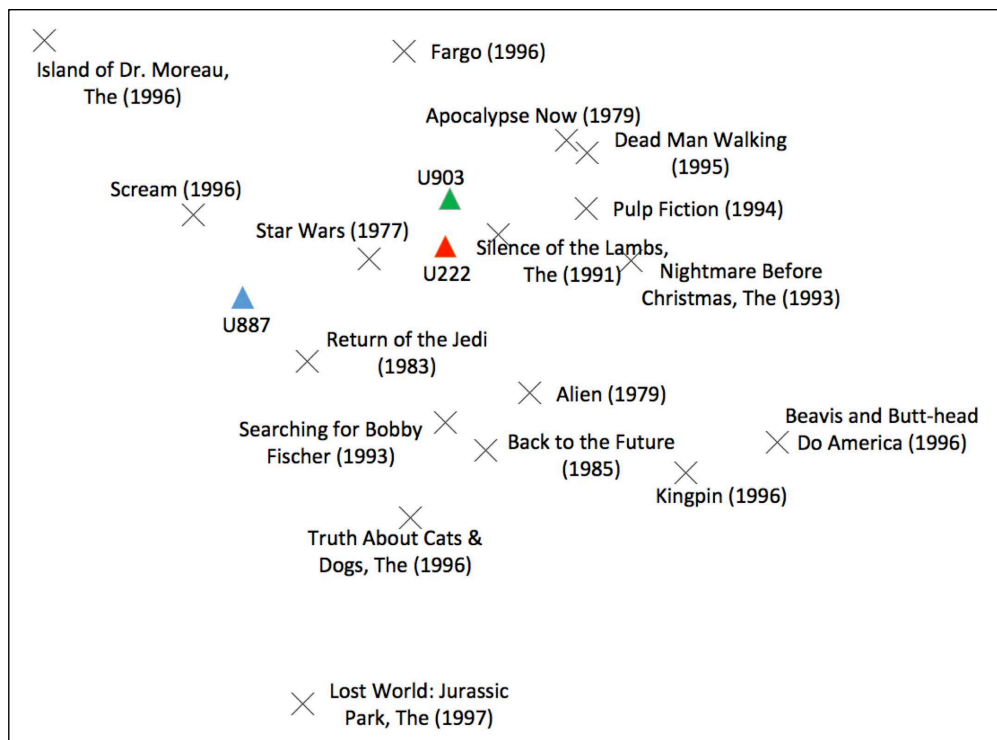


Figure 3.4: Example Visualization of Users (triangles) and Items (crosses) in MovieLens-100K

Visualization. Figure 3.4 shows an example of three users U887 (blue), U222 (red), U903 (green) in *MovieLens-100K*, and the 17 items (crosses) that all three have rated. For instance, U222 and U903 are closer to Fargo (which they rated 5) than U887 is (who rated it 2). Interestingly, U222

is closer to U903 than U222 is to U887, supported by the Pearson correlation of their ratings on items: 0.31 between (U222, U903), and -0.21 between (U222, U887). The layout of movies are also intuitive. Horror films *Scream* and *Island of Dr. Moreau* are on the top left. Science fictions *Star Wars*, *Return of the Jedi*, and *Back to the Future* are at the centre. Darker dramas *Fargo*, *Apocalypse Now* are on the top right. Comedies such as *Kingpin* and *Beavis and Butt-head* are on the far right. Family-oriented *Searching for Bobby Fischer* and *Lost World* are towards the bottom.

Learning efficiency is not our major focus here. The learning algorithms can be run offline. On MovieLens-100K and Netflix, COE takes approximately a minute on a PC with Intel Core i5 3.2GHz CPU and 12GB RAM. For 20News, the running time of COE is around 15 minutes. Our efficiency is comparable to other models running on pairwise comparisons, e.g., BPR(MF), and is much faster than ordinal embedding, i.e., SOE.

	Preservation Accuracy			Prediction Accuracy			5-NN Avg Frequency		
	Type-A	Type-B	H-Mean	Type-A	Type-B	H-Mean	Users	Items	H-Mean
COE	64.5%	85.6%	73.5%	51.7%	63.2%	56.9%	0.041	0.032	0.036
CODE	53.3%	52.8%	53.1%	49.8%	54.7%	52.2%	0.032	0.032	0.032

Table 3.4: Cooccurrence-Based Dataset (Last.fm): COE vs. Cooccurrence Embedding

	Preservation Accuracy			Prediction Accuracy			5-NN Avg Frequency		
	Type-A	Type-B	H-Mean	Type-A	Type-B	H-Mean	Docs	Words	H-Mean
COE	78.9%	90.3%	84.3%	51.0%	69.2%	58.7%	0.039	0.029	0.037
CODE	59.7%	56.2%	57.9%	48.7%	52.8%	50.7%	0.033	0.020	0.025

Table 3.5: Cooccurrence-Based Dataset (20News): COE vs. Cooccurrence Embedding

Cooccurrence-Based Datasets

We now discuss the comparisons for the other two datasets based on cooccurrences: *Last.fm* and *20News*. Here, we focus on the comparison to CODE [32], which fits co-occurrence frequencies.

For the metrics, we again rely on preservation and prediction accuracies. In addition, we adapt the “average rating” concept to the cooccurrence scenario. Since the raw observation is normalized

term frequency, we evaluate the average term frequencies among the k -nearest neighbors of a document or a word respectively. The higher it is, the more successful is the embedding in placing the closest words to a document (vice versa).

Table 3.4 for *Last.fm* and Table 3.5 for *20News* show that COE has significantly higher preservation and prediction accuracies than the baseline CODE. This experiment showcases that the information within ordinal triples is not easily approximated by fitting probabilities of co-occurrences (which is semantically closer to similarity/distance-based embedding). This is also evident from the comparison of average normalized term frequencies among the 5-NN. The values seem deceptively low, these frequencies are actually high, considering that each document consists of many words.

3.3.2 Efficient Retrieval of Recommendation with KD-Tree

The key advantage of modelling user-item interaction through the Euclidean between their respective latent vectors over matrix factorization is that the recommendation candidates retrieval task is equivalent to the nearest neighbors search (NNS) problem. Hence, there are numerous methods in literature on spatial indexing for nearest neighbor search that can be used to find recommendation candidates efficiently. We consider a well-known tree structure, KD-tree. Approximate kNN retrieval can be achieved by restricting the searching time on the tree ([29]). The implementation of KD-tree in [75] controls this by c , the number of nodes to explore on the tree. For this experiment, we use the two large rating-based preference datasets *MovieLens 20M*⁶ and *Netflix* for comparison, and learn only from the Type-A triples for simplicity. We first derive high-dimensional vectors representation for users and items (e.g., $d = 20$)⁷ and employ KD-tree as indexing structure. We then compare the post-indexing performances of all models.

⁶<http://grouplens.org/datasets/movielens/20m/>

⁷Similar trends are observed across other dimensionalities

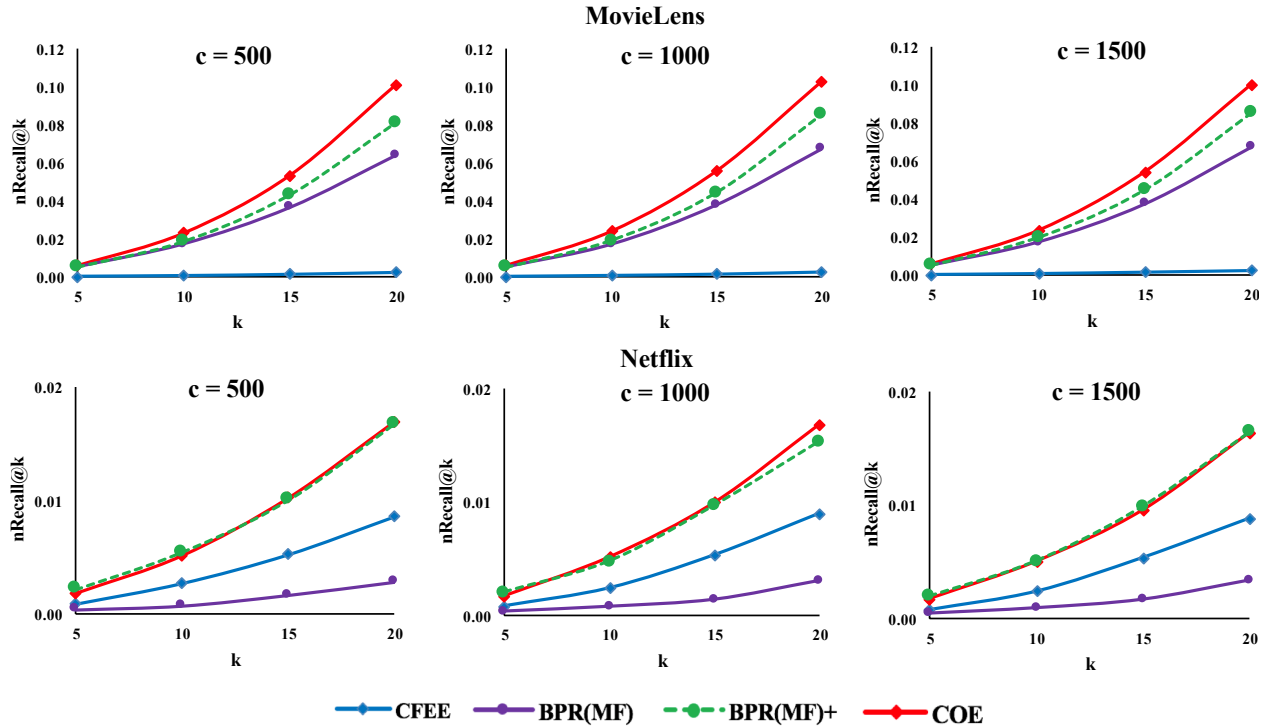


Figure 3.5: nRecall@k with KD-Tree Indexing.

Metrics. We assume that the goal of top- k recommendation is to recommend *new* items to a user, among the items not seen in the training set. When retrieval is based on an index, the evaluation of top- k necessarily takes into account the operation of the index. Because we maintain one index for all items to be used with all users, conceivably items returned by a top- k query may belong to one of three categories: those in the training set (to be excluded for new item recommendation), those in the test set (of interest as these are the known ground-truth of which items users prefer), and those not seen/rated in either set (for which no ground-truth of user preference is available). It is important to note the latter may not necessarily be bad recommendations, they are simply unknown. Precision of the top- k may penalize such items. We reason that among the rated items in the test set, those that have been assigned the maximum rating possible by a user would be expected to appear in the top- k recommendation list for that user. A suitable metric is the recall of items in the test set with maximum rating. For each user u with at least one highest rating item in the test set (for the two datasets, the highest possible rating value is 5), we compute the percentage

of these items that are returned in the top- k by the index. The higher the percentage, the better is the performance of the model at identifying the items a user prefers the most. Equation 3.6 presents the formula for $Recall@k$:

$$Recall@k = \frac{1}{|\mathcal{U}_{\max}|} \sum_{u \in \mathcal{U}_{\max}} \frac{|\{i \in \psi_k^u : r_{ui} = \text{max rating}\}|}{|\{i \in \mathcal{I} : r_{ui} = \text{max rating}\}|}, \quad (3.6)$$

in which \mathcal{U}_{\max} is the set of users who have given at least one item with rating of 5 and ψ_k^u is the top- k returned by the index. We exclude training items for u from both numerator and denominator. We normalize $Recall@k$ with the ideal $Recall@k$ that a perfect algorithm can achieve, and denote the metric as $nRecall@k$.

To investigate the efficacy of using the indexing schemes for top- k recommendation, we introduce the second metric *speedup*, which is the ratio between the time taken by exhaustive search to return the top- k , to the time taken by an index.

$$Speedup = \frac{\text{Retrieval time taken by exhaustive search}}{\text{Retrieval time taken by the index}}. \quad (3.7)$$

We will discuss the results in terms of trade-off between recall and speedup. There are index parameters that control the degree of approximation, i.e., higher speedup at the expense of lower recall. Among the comparative recommendation algorithms, a better trade-off means higher speedup at the same recall, or higher recall at the same speedup. For each comparison below, we control for the indexing scheme, as different schemes vary in ways of achieving approximation, implementations, and deployment scenarios.

Figure 3.5 shows the $nRecall@k$ with various $c \in \{500, 1000, 1500\}$. We also experimented with $c \in \{50, 150, 300, 750, 2000\}$ and get similar trends. COE consistently outperforms the CFEE and BPR(MF) at all values of c . Notably, COE outperforms BPR(MF)+ on MovieLens dataset and is comparable on Netflix dataset. The likely reason is that we are experimenting in

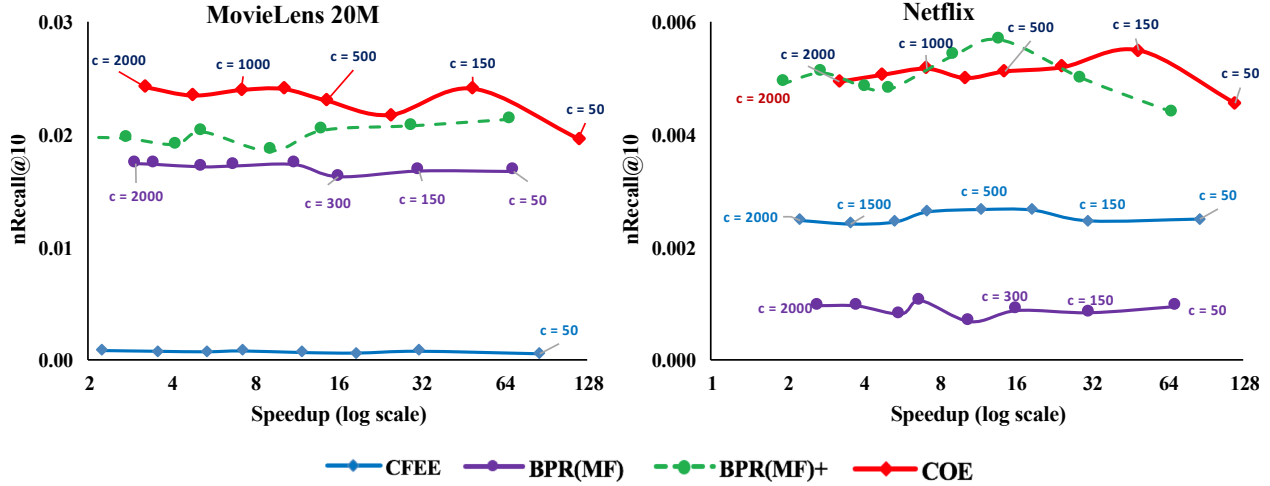


Figure 3.6: nRecall@10 vs. speedup with KD-tree Indexing.

a high-dimensional space ($d = 20$), therefore BPR(MF) is capable of learning the user preference accurately, and the Euclidean transformation is useful in preserving the performance after indexing. Figure 3.6 plots the accuracy in terms of $nRecall@10$ vs. the retrieval efficiency in terms of speedup. As we increase c , a longer searching time on KD-tree is allowed, resulting in higher quality of the returned top- k . Here too, COE achieves higher accuracy at the same speedup, higher speedup at the same accuracy, as compared to the baselines.

3.4 Discussion

We address the problem of ordinal co-embedding based on cross-type ordinal relationships, whereby every user and every item is respectively associated with a latent coordinate in a low-dimensional Euclidean space. The objective is to place a user closer to a more preferred item. This accommodates datasets including ratings and co-occurrences. Experiments on public datasets show that Collaborative Ordinal Embedding or COE outperforms comparable baselines in both information preservation in the visualization space and efficient retrieval of recommendation candidates.

Chapter 4

Indexable Bayesian Personalized Ranking

As mentioned in Section 2.1, one direction towards retrieval efficiency is to formulate retrieval as approximate k most similar neighbors (kNN) search aided by indexing schemes, such as locality-sensitive hashing, spatial trees, and inverted index. These schemes, applied on the output representations of recommendation algorithms, speed up the retrieval process by automatically discarding a large number of potentially irrelevant items when given a user query vector. However, many previous matrix factorization recommendation algorithms commonly use inner product as the predictor that may not necessarily align well with the structural properties of these indexing schemes, eventually resulting in a significant loss of accuracy post-indexing. In this chapter, we introduce *Indexable Bayesian Personalized Ranking* (INDEXABLE BPR) that learns from ordinal preference to produce latent representation that is inherently compatible with the aforesaid indices. Experiments on publicly available datasets show superior performance of the proposed model compared to state-of-the-art methods on top- k recommendation retrieval task, achieving significant speedup while maintaining high accuracy.

4.1 Introduction

As discussed in Chapter 1, two overriding goals for personalized top- k recommendations include *accuracy* in placing items that user u prefers most into u 's recommendation list and *retrieval efficiency* in delivering the recommendation list upon request. Faster retrieval helps the system to cope with a large number of consumers, and minimize their waiting time to receive recommendations. In contrast, learning efficiency or minimizing model learning time, while useful, is arguably less mission-critical, as it can be done offline and involves mainly machine time, rather than human time. Therefore, we seek to keep the learning time manageable, while improving retrieval efficiency. Many previous recommendation algorithms focus mainly on accuracy and the efficiency of the learning process. One challenge in practice is the need for an exhaustive search over all candidate items to identify the top- k , which is time-consuming when the number of items n is extremely large [47].

In this work, we seek to further improve upon the efficiency of MF-based methods by focusing on the retrieval step of recommendations (after learning the latent vectors). Particularly, we focus on using indexing structures (e.g., locality sensitive hashing, spatial tree indexing, and inverted indexing) as an efficient alternative to exhaustive search over all items.

Approach. The key reason behind the incompatibility between inner product search that matrix factorization relies on, and the aforesaid index structures is how a user u 's degree of preference for an item i , expressed as the inner product $x_u^T y_i$, is sensitive to the respective magnitude of the latent vectors $\|x_u\|, \|y_i\|$. Therefore, one insight towards achieving geometric compatibility is to desensitize the effect of vector magnitudes. The challenge is how to do so while still preserving the accuracy of the top- k retrieval.

There are a couple of recent approaches in this direction. One approach [9] is a post-processing transformation that expands the latent vectors learnt from matrix factorization with an extra dimensionality to equalize the magnitude of all item vectors. Because the transformation is a separate

process from learning the vectors, such a workaround would not be as effective as working with natively indexable vectors in the first place. Another approach is Indexable Probabilistic Matrix Factorization, proposed in [29], which extends the Bayesian Probabilistic Matrix Factorization [87], by making the item latent vectors natively of fixed length. Fitting inner product to absolute rating value may not be suitable when only implicit feedback (not rating) is available. Moreover, we note that top- k recommendation is inherently an expression of “relative” rather than “absolute” preferences, i.e., the ranking among items is more important than the exact scores. Therefore, we propose to work with ordinal expressions of preferences. Ordinal preferences can be expressed as a triple (u, i, j) , indicating that a user u prefers an item i to a different item j . Ordinal representation is prevalent in modeling preferences [83], and also accommodates both explicit (e.g., ratings) and implicit feedback.

Contributions. We make the following contributions in this work. *First*, we propose *Indexable Bayesian Personalized Ranking* model or INDEXABLE BPR in short, which produces native geometrically indexable latent vectors for accurate and efficient top- k recommendation. BPR [83] is a generic framework modeling ordinal triples. Each instantiation is based on a specific kernel [35, 50, 57, 78]. [83] has inner product kernel, which is not well-fitted to indexing structures. In contrast, our INDEXABLE BPR is formulated with a kernel based on angular distances (see Section 5.2). *Second*, we describe how the resulting vectors are used with LSH, spatial tree, and inverted index for top- k recommendation in Section 4.3. We conduct experiments with available datasets to compare INDEXABLE BPR with baselines. Empirically, we observe that INDEXABLE BPR achieves a balance of accuracy and run-time efficiency, achieving higher accuracy than the baselines at the same speedup level, and higher speedup at the same accuracy level. *Third*, to support the observation on the robustness of INDEXABLE BPR, we provide a theoretical analysis in the context of LSH, further bolstered with empirical evidence, on why our reliance on angular distances results in more index-friendly vectors, smaller loss of accuracy post-indexing, and balanced all-round performance.

4.2 Framework

Problem. We consider a set users \mathcal{U} and a set of items \mathcal{I} . We consider as input a set of triples $\mathcal{T} \subset \mathcal{U} \times \mathcal{I} \times \mathcal{I}$, which has been defined in Section 3.2.1. The goal is to derive a d -dimensional latent vector $x_u \in \mathbb{R}^d$ for each user $u \in \mathcal{U}$, and a latent vector $y_i \in \mathbb{R}^d$ for each item $i \in \mathcal{I}$, such that the relative preference of a user u over two items i and j can be expressed as a function (to be defined) of their corresponding latent vectors x_u , y_i , and y_j . We denote the collection of all user latent vectors and item latent vectors as X and Y respectively.

Framework. Given the input triples \mathcal{T} , we seek to learn the user and item vectors X, Y with the highest posterior probability.

$$\arg \max_{X, Y} P(X, Y | \mathcal{T}) \quad (4.1)$$

The Bayesian formulation for modeling this posterior probability is to decompose it into the likelihood of the triples $P(\mathcal{T} | X, Y)$ and the prior $P(X, Y)$, as shown in Equation 4.2.

$$P(X, Y | \mathcal{T}) \propto P(\mathcal{T} | X, Y) P(X, Y) \quad (4.2)$$

We will define the prior later when discussing the generative process. We now focus on defining the likelihood, which can be decomposed into the probability for individual triple t_{uij} :

$$P(\mathcal{T} | X, Y) = \prod_{t_{uij} \in \mathcal{T}} P(t_{uij} | x_u, y_i, y_j) \quad (4.3)$$

Weakness of Inner Product Kernel for Top- k Retrieval. To determine the probability for an individual triple, we need to define a kernel function. The kernel proposed by the matrix factorization-based (not natively indexable) BPR [83] is shown in Equation 4.4 (σ is the sigmoid function). This assumes that if $x_u^T y_i > x_u^T y_j$, then user u is likely to prefer item i to j .

$$P(t_{uij} | x_u, y_i, y_j) = \sigma(x_u^T y_i - x_u^T y_j) \quad (4.4)$$

Since our intended application is top- k recommendation, once we learn the user and item latent vectors, the top- k recommendation task is reduced to searching for the k nearest neighbors to the query (user vector) among the potential answers (item vectors).

As discussed in Section 1.1.1, an indexing-based approach which prioritize or narrow the search to a smaller search space, is a faster alternative to an exhaustive search over all the items. For the nearest neighbors identified by an index to be as accurate as possible, the notion of similarity (or distance) used by the index should be compatible with the notion of the similarity of the underlying model that yields that user and item vectors. Therein lies the issue with the inner product kernel in Equation 4.4. It is not necessarily compatible with indexing structures that rely on similarity functions other than inner products.

First, we examine its incompatibility with spatial tree index. Suppose that all item latent vectors y_i 's are inserted into the index. To derive the recommendation for u , we use x_u as the query. Nearest neighbor search on spatial tree index is expected to return items that are closest in terms of Euclidean distance. The relationship between Euclidean distance and inner product is expressed in Equation 4.5. It implies that items with the closest Euclidean distances may not have the highest inner products, due to the magnitudes $\|x_u\|$ and $\|y_i\|$. Spatial tree index retrieval may be inconsistent with Equation 4.4.

$$\|x_u - y_i\|^2 = \|x_u\|^2 + \|y_i\|^2 - 2x_u^T y_i \quad (4.5)$$

Second, we examine its incompatibility with inverted index that relies on cosine similarity (Equation 4.6). Similarly, the pertinence of the magnitudes $\|x_u\|$ and $\|y_i\|$ implies that inverted index retrieval may be inconsistent with maximum inner product search.

$$\cos(x_u, y_i) = \frac{x_u^T y_i}{\|x_u\| \cdot \|y_i\|} \quad (4.6)$$

Third, in terms of its incompatibility with LSH, we note that it has been established that there cannot exist any LSH family for maximum inner product search [91], while there exist LSH families for Euclidean distances and cosine similarity respectively.

Proposed Angular Distance Kernel. To circumvent the limitation of the inner product kernel, we propose a new kernel to express the probability for a triple t_{uij} in a way that is insensitive to vector magnitudes. A different kernel is a non-trivial, even significant, change as it requires a different learning algorithm. Our proposed kernel is based on angular distance.

Let θ_{xy} denote the angular distance between vectors x and y , evaluated as following:

$$\theta_{xy} = \cos^{-1}\left(\frac{x^T y}{\|x\| \cdot \|y\|}\right) \quad (4.7)$$

Proposing the angular distance, i.e., the arccos of the cosine similarity, to formulate the user-item association is a novel and appropriate design choice for the following reasons.

- *Firstly*, since arccos is a monotone function, the closest point according to the angular distance is the same as the point with the highest cosine similarity, resulting in its compatibility with the inverted index structure.
- *Secondly*, since angular distances are not affected by magnitudes, it preserves all the information learnt by the model. Before indexing, the learnt vectors could be normalized to unit length for compatibility with indexing that relies on either Euclidean distance or cosine similarity.
- *Lastly*, the angular distance is also compatible to LSH indexing. A theoretical analysis and empirical evidence on this compatibility is provided in Section 4.4.

While the vectors x_u and item y_i could be of varying lengths, the magnitudes are uninformative as far as the user preferences encoded by the triples are concerned. This advantageously allows greater flexibility in parameter learning, while still controlling the vectors via the regularization terms, as opposed to constraining vectors to fixed length during learning (as in [29]).

We formulate the probability of a triple t_{uij} for INDEXABLE BPR as in Equation 4.8:

$$P(t_{uij}|x_u, y_i, y_j) = \sigma(\theta_{x_u y_j} - \theta_{x_u y_i}) \quad (4.8)$$

The probability is higher when the difference $\theta_{x_u y_j} - \theta_{x_u y_i}$ is larger. If u prefers i to j , the angular distance between x_u and y_i is expected to be smaller than that between x_u and y_j .

Generative Process. The proposed model INDEXABLE BPR as a whole could be expressed by the following generative process:

1. For each user $u \in \mathcal{U}$: Draw $x_u \sim \text{Normal}(0, \eta^2 \mathbf{I})$,
2. For each item $i \in \mathcal{I}$: Draw $y_i \sim \text{Normal}(0, \eta^2 \mathbf{I})$,
3. For each triple of one user $u \in \mathcal{U}$ and two items $i, j \in \mathcal{I}$:
 - Draw a trial from Bernoulli($P(t_{uij}|x_u, y_i, y_j)$),
 - If “success”, generate a triple instance t_{uij} ,
 - Otherwise, generate a triple instance t_{uji} .

The first two steps place zero-mean multi-variate spherical Gaussian priors on the user and item latent vectors. η^2 denotes the variance of the Normal distributions; for simplicity we use the same variance for users and items. \mathbf{I} denote the identity matrix.

The prior $P(X, Y)$ is defined as following:

$$P(X, Y) = (2\pi\eta^2)^{-\frac{D}{2}} \prod_{u \in \mathcal{U}} e^{-\frac{1}{2\eta^2} \|x_u\|^2} \prod_{i \in \mathcal{I}} e^{-\frac{1}{2\eta^2} \|y_i\|^2} \quad (4.9)$$

Triples in \mathcal{T} are generated from users and items’ latent vectors according to the probability $P(t_{uij}|x_u, y_i, y_j)$ as defined in Equation 4.8.

Parameter Learning. The objective is to maximize the log-posterior in Equation 4.2:

$$\mathcal{L} = \ln P(\mathcal{T}|X, Y) + \ln P(X, Y) = \alpha \ln P(\mathcal{T}|X, Y) - \frac{1}{\eta^2} \sum_{u \in \mathcal{U}} \|x_u\|^2 - \frac{1}{\eta^2} \sum_{i \in \mathcal{I}} \|y_i\|^2 \quad (4.10)$$

Let us denote $\Delta_{uij} = \theta_{x_u y_j} - \theta_{x_u y_i}$, $\tilde{x}_u = \frac{x_u}{\|x_u\|} \forall u \in \mathcal{U}$ and $\tilde{y}_i = \frac{y_i}{\|y_i\|} \forall i \in \mathcal{I}$. The gradient of \mathcal{L} w.r.t each user vector x_u is:

$$\frac{\partial \mathcal{L}}{\partial x_u} = \sum_{\{i,j: t_{uij} \in \mathcal{T}\}} \frac{1}{\|x_u\|^2} \frac{e^{-\Delta_{uij}}}{1 + e^{-\Delta_{uij}}} \times \left(\frac{-\tilde{y}_j \cdot \|x_u\| + \cos(x_u, y_j) \cdot x_u}{\sqrt{1 - \cos(x_u, y_j)^2}} - \frac{-\tilde{y}_i \cdot \|x_u\| + \cos(x_u, y_i) \cdot x_u}{\sqrt{1 - \cos(x_u, y_i)^2}} \right),$$

in which, $\cos(x_u, y_i) = \frac{x_u^T y_i}{\|x_u\| \|y_i\|} \forall u \in \mathcal{U}$ and $\forall i \in \mathcal{I}$.

The gradient of \mathcal{L} w.r.t each item vector y_k is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial y_k} &= \sum_{\{u,j: t_{ukj} \in \mathcal{T}\}} \frac{1}{\|y_k\|^2} \frac{e^{-\Delta_{ukj}}}{1 + e^{-\Delta_{ukj}}} \cdot \frac{\tilde{x}_u \cdot \|y_k\| - \cos(x_u, y_k) \cdot y_k}{\sqrt{1 - \cos(x_u, y_k)^2}} \\ &+ \sum_{\{u,i: t_{uik} \in \mathcal{T}\}} \frac{1}{\|y_k\|^2} \frac{e^{-\Delta_{uik}}}{1 + e^{-\Delta_{uik}}} \cdot \frac{-\tilde{x}_u \cdot \|y_k\| + \cos(x_u, y_k) \cdot y_k}{\sqrt{1 - \cos(x_u, y_k)^2}}. \end{aligned}$$

Algorithm 2 describes the learning algorithm with full gradient ascent. It first initializes the users and items' latent vectors. In each iteration, the model parameters are updated based on the gradients, with a decaying learning rate ϵ over time. The output is the set of normalized user vectors \tilde{x}_u and item vectors \tilde{y}_i . On one hand, this normalization does not affect the accuracy of the top- k recommendation produced by INDEXABLE BPR, since the magnitude of the latent vectors does not affect the ranking. On the other hand, normalized vectors can be used for approximate kNN search using various indexing data structures later. The time complexity of the algorithm is linear to the number of triples in \mathcal{T} , i.e., $\mathcal{O}(|\mathcal{U}| \times |\mathcal{I}|^2)$.

Algorithm 2 Gradient Ascent for INDEXABLE BPR

Require: Ordinal triples set $\mathcal{T} = \{t_{uij}, \forall u \in \mathcal{U}, i \neq j \in \mathcal{I}\}$.

```
1: Initialize  $x_u$  for  $u \in \mathcal{U}$ ,  $y_i$  for  $i \in \mathcal{I}$ 
2:
3: while not converged do
4:   for each  $u \in \mathcal{U}$  do
5:      $x_u \leftarrow x_u + \epsilon \cdot \frac{\partial \mathcal{L}}{\partial x_u}$ 
6:   end for
7:
8:   for each  $i \in \mathcal{I}$  do
9:      $y_i \leftarrow y_i + \epsilon \cdot \frac{\partial \mathcal{L}}{\partial y_i}$ 
10:  end for
11: end while
12:
13: Return  $\{\tilde{x}_u = \frac{x_u}{\|x_u\|}\}_{u \in \mathcal{U}}$  and  $\{\tilde{y}_i = \frac{y_i}{\|y_i\|}\}_{i \in \mathcal{I}}$ 
```

4.3 Experiments

The key idea is achieving speedup in the retrieval time of top- k recommendation via indexing, while still maintaining high accuracies via better representations that minimize any loss of information post-indexing. Hence, in the following evaluation, we are interested in both the accuracy of the top- k recommendation returned by the index, and the speedup in retrieval time due to indexing as compared to exhaustive search.

To showcase the generality of INDEXABLE BPR in accommodating various index structures, we experiment with three indexing schemes: locality-sensitive hashing, spatial tree index, and inverted index. Note that our focus is on the relative merits of recommendation algorithms, rather than on the relative merits of index structures. It is our objective to investigate the effectiveness of INDEXABLE BPR, as compared to other algorithms, for top- k recommendation when using these index structures. Yet, it is *not* our objective to compare the index structures among themselves.

Comparative Methods. In this study, we consider the following baselines:

- BPR(MF): the non-index friendly BPR with inner product (MF) kernel [83]. This would validate whether our angular distance kernel is more index-friendly.

- BPR(MF)+: a composite of BPR(MF) and the Euclidean transformation described in [9] to make the item vectors indexable as post-processing. This allows validation of our learning inherently indexable vectors in the first place.
- IPMF: matrix factorization that learns fixed-length item vectors but fits rating scores [29]. This allows validation of our modeling of ordinal triples.
- CFEE: Euclidean embedding that fits rating scores [46]. This allows validation of our modeling of ordinal triples.
- COE: Euclidean embedding that fits ordinal triples [54]. Comparison to CFEE and COE allows validation of our compatibility with non-spatial indices such as some LSH families as well as inverted index.

We tune the hyper-parameters of all models for the best performance. For IPMF, we adopt the parameters provided by its authors for *Netflix* dataset. For the ordinal-based algorithms (BPR, COE, and INDEXABLE BPR), the learning rate and the regularization are 0.05 and 0.001. For CFEE, they are 0.1 and 0.001. All models use $d = 20$ dimensionalities in their latent representations. Similar trends are observed across other dimensionalities (see Sec. 4.4).

Datasets. We experiment on two publicly available rating-based datasets and derive ordinal triples accordingly. One is *MovieLens 20M*¹, the largest among the MovieLens collection. The other is *Netflix*². Table 4.1 shows a summary of these datasets. By default, *MovieLens 20M* includes only users with at least 20 ratings. For consistency, we apply the same to *Netflix*. For each dataset, we randomly keep 60% of the ratings for training and hide 40% for testing. We conduct stratified sampling to maintain the same ratio for each user. We report the average results over five training/testing splits. For training, we generate a triple t_{uij} if user u has higher rating for item i than for j , and triples are formed within the training set.

As earlier mentioned, our focus in this work is on online retrieval speedup. We find that the

¹<http://grouplens.org/datasets/movielens/20m/>

²<http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a>

	#users	#items	#ratings	#training ordinal triples
MovieLens 20M	138,493	27,278	20,000,263	5.46×10^8
Netflix	480,189	17,770	100,480,507	2.29×10^{10}

Table 4.1: Datasets Summary

model learning time, which is offline, is manageable. Our learning times for *MovieLens 20M* and *Netflix* are 5.2 and 9.3 hours respectively on a computer with Intel Xeon E2650v4 2.20GHz CPU and 256GB RAM. Algorithm 2 scales with the number of triples, which in practice grows slower than its theoretical complexity of $\mathcal{O}(|\mathcal{U}| \times |\mathcal{I}|^2)$. Figure 4.1 shows how the average number of triples per user grows with the number of items, showing that the actual growth is closer to linear and lower than the quadratic curve provided as reference.

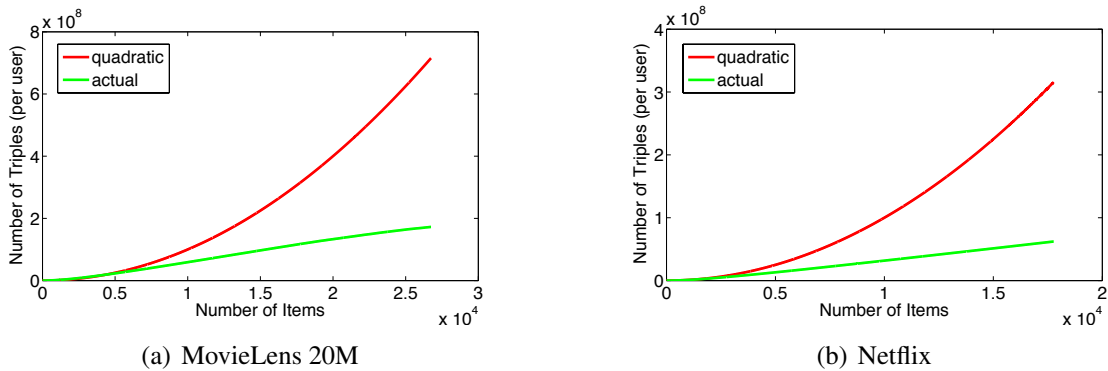


Figure 4.1: Number of triples (per user) vs. number of items.

Metrics. For evaluation, we employ two metrics defined in Section 3.3.2: *Recall* and *Speedup*.

4.3.1 Top-k Recommendation with LSH Index

We first briefly review LSH and how it is used for top- k recommendation. Let $\mathbf{h} = (h_1, h_2, \dots, h_b)$ be a set of LSH hash functions. Each function assigns a bit for each vector. \mathbf{h} will assign each user u a binary code $\mathbf{h}(x_u)$, and each item i a binary hashcode $\mathbf{h}(y_i)$, all of length b . Assuming that x_u prefers y_i to y_j , \mathbf{h} is expected to produce binary hashcodes with a smaller Hamming distance $\|\mathbf{h}(x_u) - \mathbf{h}(y_i)\|_H$ than the Hamming distance $\|\mathbf{h}(x_u) - \mathbf{h}(y_j)\|_H$.

The most frequent indexing strategy for LSH is *hash table lookup*. We store item codes in hash tables, with items having the same code in the same bucket. Given a query (user) code, we can determine the corresponding bucket in constant time. We search for the top-k only among items in that bucket, reducing the number of items on which we need to perform exact similarity computations. We use the LSH package developed by [5]. The LSH family for INDEXABLE BPR for generating hashcodes is SRP-LSH, which is also used for IPMF following [29]. We apply it to BPR(MF) and BPR(MF)+, as [94], [77] claim it to be the more suitable family for transformed vectors. In turn, the LSH scheme for COE and CFEE is L2-LSH, since both use l_2 distance.

When using hash tables, one specifies the number of tables T and the code length b . We experiment with various T , and $T = 10$ returns the best performance (consistent with [29]). We also vary b and larger b is expected to lead to fewer items in each bucket.

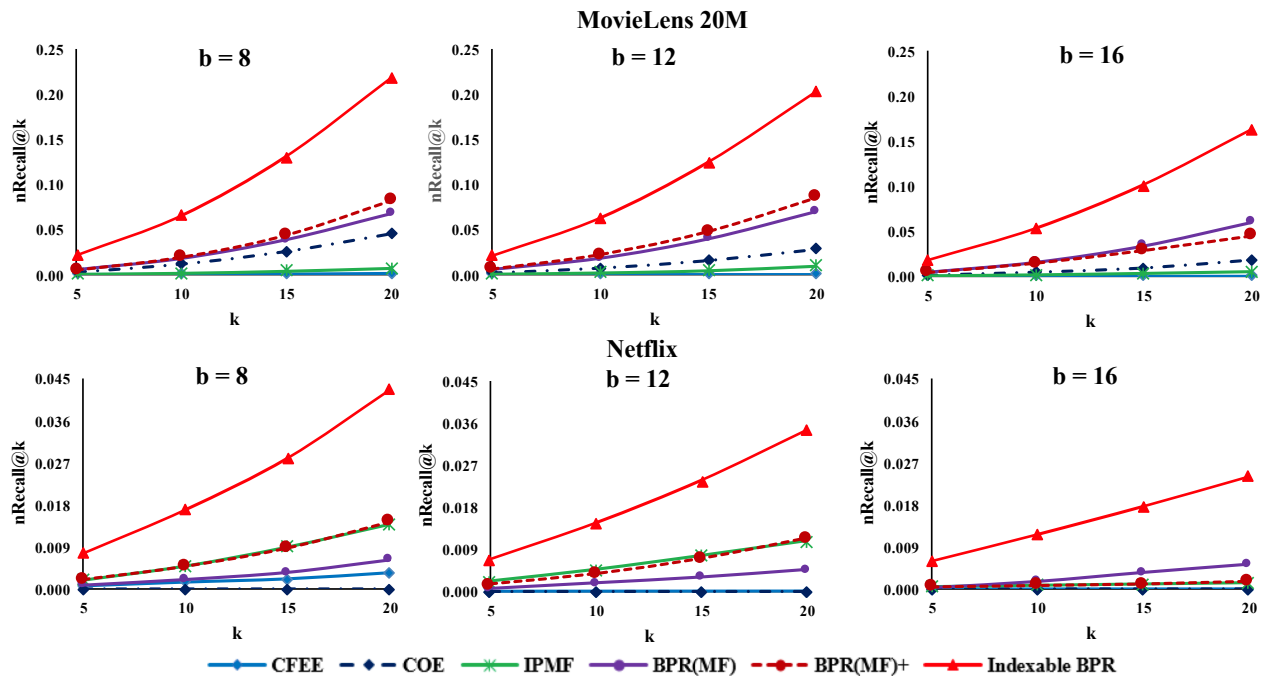


Figure 4.2: $nRecall@k$ with Hash Table Lookup Strategy ($T = 10$ hash tables).

Figure 4.2(a) shows the $nRecall@k$ using *hash table lookup* with $T = 10$ tables and different values of code length $b = 8, 12, 16$ for *MovieLens20M*. Across the b 's, the trends are similar. INDEXABLE BPR has the highest $nRecall@k$ values across all k . It outperforms BPR(MF)+ that

conducts vector transformation as post-processing, which indicates that learning inherently indexable vectors is helpful. In turn, BPR(MF)+ outperforms BPR(MF), which indicates that the inner product kernel is not conducive for indexing. Interestingly, INDEXABLE BPR also performs better than models that fit ratings (IPMF, CFEE), suggesting that learning from relative comparisons may be more suitable for top- k recommendation.

Figure 4.2(b) shows the results for *Netflix*. Again, INDEXABLE BPR has the highest $nRecall@k$ values across all k . The relative comparisons among the baselines are as before, except that IPMF now is more competitive, though still lower than INDEXABLE BPR.

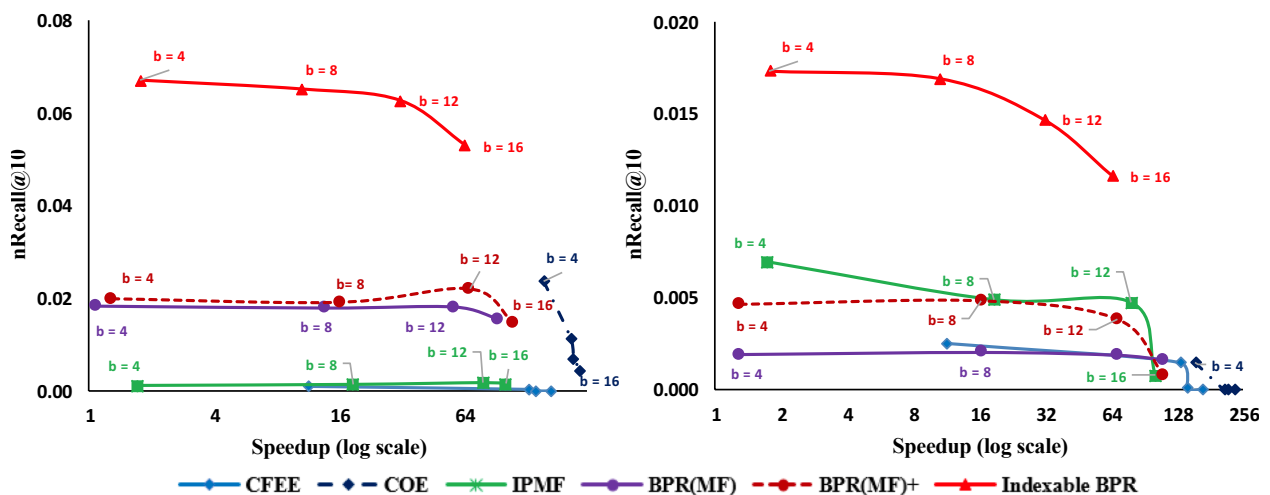


Figure 4.3: $nRecall@10$ vs. Speedup with Hashtable Lookup Strategy ($T = 10$ hash tables).

We also investigate the tradeoff between the speedup achieved and the accuracy of the top- k returned by the index. Figure 4.3 shows the $nRecall@10$ s and the speedup when varying the value of b . Given the same speedup, INDEXABLE BPR can achieve significantly higher performance compared to the baselines. As b increases, the speedup increases and $nRecall@10$ decreases. This is expected, as the longer the hashcodes, the smaller the set of items on which the system needs to perform similarity computation. This reflects the trade-off of speedup and approximation quality.

4.3.2 Top-k Recommendation with KD-Tree Index

Spatial trees refer to a family of methods that recursively partition the data space towards a balanced binary search tree, in which each node encompasses a subset of the data points [73]. For algorithms that model the user-item association by l_2 distance, spatial trees can be used to index the item vectors. Top- k recommendation is thus equivalent to finding kNN to the query. The tree will locate the nodes that the query belongs to, and exact similarity computation is performed only on the points indexed by those nodes.

For INDEXABLE BPR, Algorithm 2 returns two sets of normalized vectors $\tilde{x}_u \forall u \in \mathcal{U}$ and $\tilde{y}_i \forall i \in \mathcal{I}$. We observe that:

$$\|\tilde{x}_u - \tilde{y}_i\| < \|\tilde{x}_u - \tilde{y}_j\| \Leftrightarrow \tilde{x}_u^T \tilde{y}_i > \tilde{x}_u^T \tilde{y}_j \Leftrightarrow \theta_{\tilde{x}_u \tilde{y}_i} < \theta_{\tilde{x}_u \tilde{y}_j}, \quad (4.11)$$

i.e., the ranking of items according to l_2 distance on normalized vectors is compatible to that according to angular distance, implying INDEXABLE BPR’s output can support kNN using spatial tree.

In this experiment, we consider a well-known tree structure, KD-tree. Approximate kNN retrieval can be achieved by restricting the searching time on the tree ([29]). The implementation of KD-tree in [75] controls this by c , the number of nodes to explore on the tree. Figure 4.4 shows the $nRecall@k$ with various $c \in \{500, 1000, 1500\}$. We also experimented with $c \in \{50, 150, 300, 750, 2000\}$ and get similar trends. INDEXABLE BPR consistently outperforms the baselines at all values of c . Notably, INDEXABLE BPR outperforms BPR(MF)+, which in turn outperforms BPR(MF), validating the point made earlier about native indexability. Figure 4.5 plots the accuracy in terms of $nRecall@10$ vs. the retrieval efficiency in terms of speedup. As we increase c , a longer searching time on KD-tree is allowed, resulting in higher quality of the returned top- k . Here too, INDEXABLE BPR achieves higher accuracy at the same speedup, higher speedup at the same accuracy, as compared to the baselines.

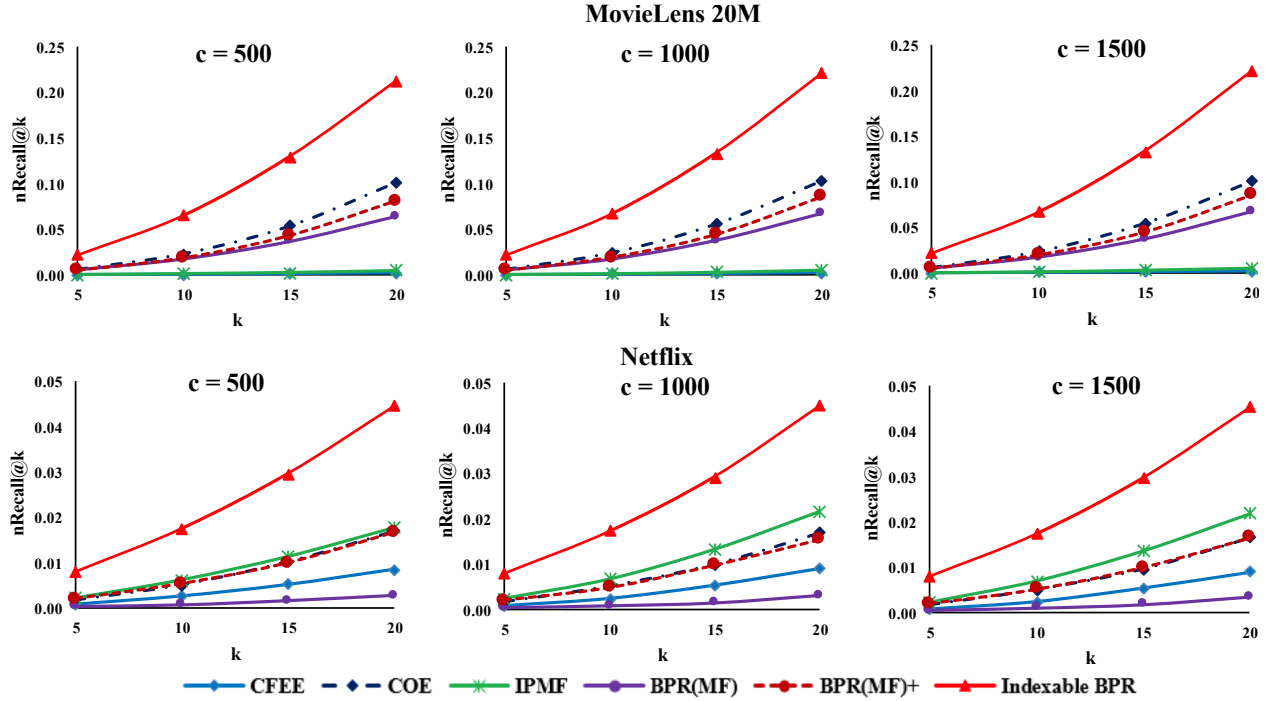


Figure 4.4: nRecall@k with KD-Tree Indexing.

4.3.3 Top-k Recommendation with Inverted Index

For recommendation retrieval, [15] presents an inverted index scheme, where every user or item is represented with a sparse vector derived from their respective dense real-valued latent vectors via a transformation. Given the user sparse vector as query, the inverted index will return items with at least one common non-zero element with the query as candidates. Exact similarity computation will be performed only on those candidates to find out the top- k .

Here, we describe very briefly the indexing scheme. For an extended treatment, please refer to [15]. The sparse representations for users and items are obtained from their dense latent vectors (learnt by the recommendation algorithm, e.g., INDEXABLE BPR) through a set of geometry-aware permutation maps Φ defined on a tessellated unit sphere. The tessellating vectors are generated from a base set $\mathcal{B}_d = \{-1, -\frac{d-1}{d}, \dots, -\frac{1}{d}, 0, \frac{1}{d}, \dots, \frac{d-1}{d}, 1\}$, characterized by a parameter d . The obtained sparse vectors have the sparsity patterns that are related to the angular closeness between the original latent vectors. The angular closeness between user vector x_u and item vector y_i is

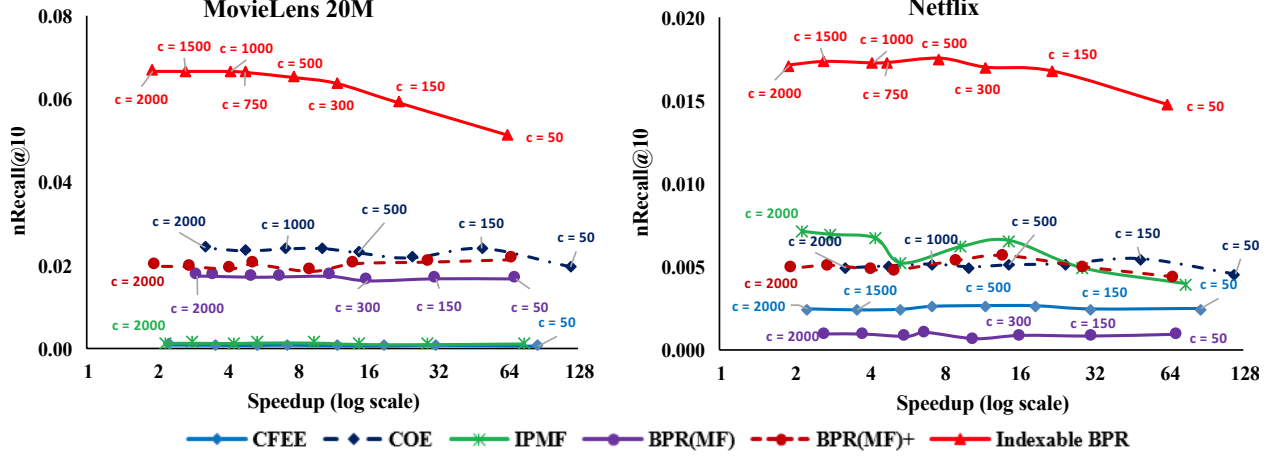


Figure 4.5: nRecall@10 vs. Speedup with KD-tree Indexing.

defined as $d_{ac}(x_u, y_j) = 1 - \frac{x_u^T y_j}{\|x_u\| \cdot \|y_j\|}$.

In the case of $\|x_u\| = \|y_i\| = 1 \forall u \in \mathcal{U}, i \in \mathcal{I}$, we have ($\forall i \neq j \in \mathcal{I}$):

$$d_{ac}(x_u, y_i) < d_{ac}(x_u, y_j) \Leftrightarrow \underbrace{\frac{x_u^T y_i}{\|x_u\| \cdot \|y_i\|}}_{\theta_{x_u y_i}} > \underbrace{\frac{x_u^T y_j}{\|x_u\| \cdot \|y_j\|}}_{\theta_{x_u y_j}} \quad (4.12)$$

The item ranking according to d_{ac} is equivalent to that according to θ -angular distance, indicating that INDEXABLE BPR based on angular distance would be compatible with this structure.

The parameter d can be managed to control the trade-off between the efficiency and the quality of approximation of kNN retrieval. Increasing the value of d leads to a higher number of discarded items using the inverted index, which leads to higher speedup of the top- k recommendation retrieval.

We run the experiments with different values of parameter d to explore the trade-off between speed and accuracy. Figure 4.6 presents the $nRecall@k$ of the two datasets at $d \in \{150, 300, 500\}$. In all cases, INDEXABLE BPR outperforms the baselines in terms of $nRecall@k$. This suggests that INDEXABLE BPR produces a representation that has greater degree of compatibility in terms of angular closeness d_{ac} between users and their preferred items. As a result, the corresponding

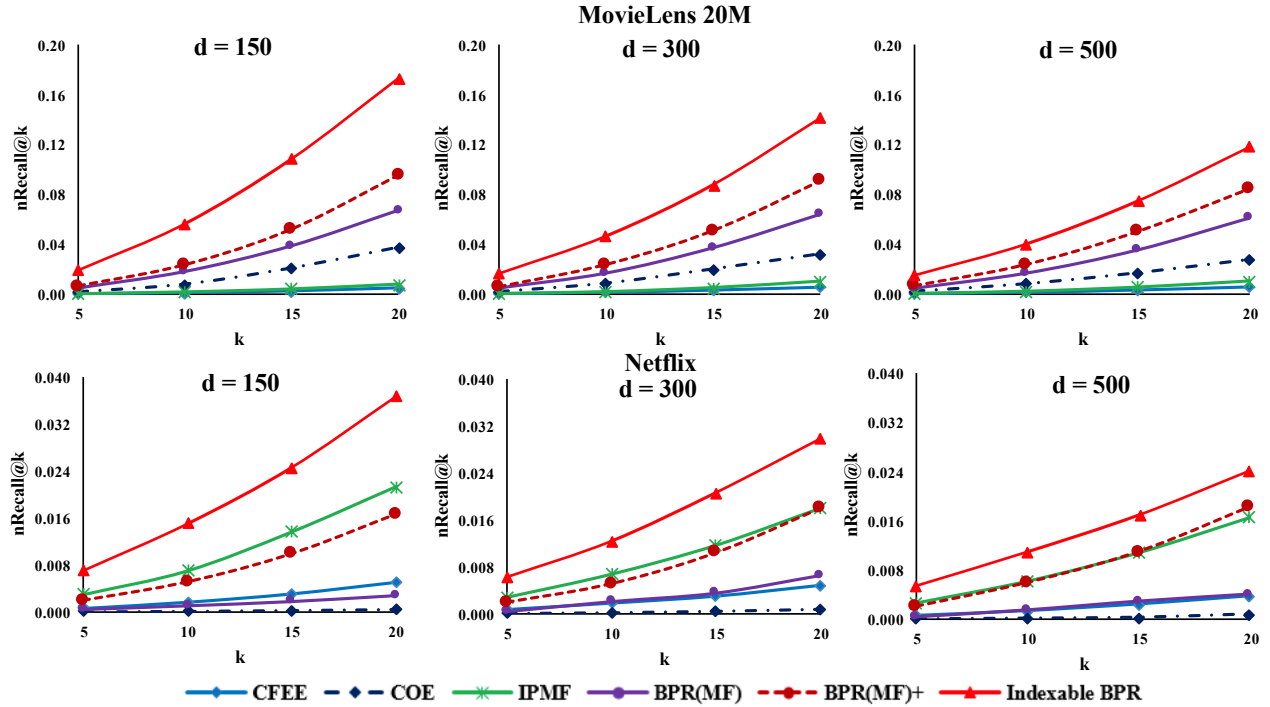


Figure 4.6: $nRecall@k$ with Inverted Indexing.

sparse vectors will have highly similar sparsity patterns, which enhances the quality of kNN using inverted indexing. Figure 4.7 shows the speedup using the inverted index as we vary the value of parameter d . We observe that the speedup increases as d increases. INDEXABLE BPR shows superior performance as compared to other models, given the same speedup.

Overall, INDEXABLE BPR works well on the indexing schemes. Effectively, we develop a model that work with multiple indices, and leave the choice of index structure to the respective application based on need. Our focus is on indexable recommendation algorithms. Here, several consistent observations emerge. INDEXABLE BPR produces representations that are more amenable to indexing, as compared to baselines BPR(MF)+ and BPR(MF). This validates the aim of INDEXABLE BPR in learning natively indexable vectors for users and items. It also outperforms models that fit ratings, as opposed to ordinal triples, for top- k recommendations.

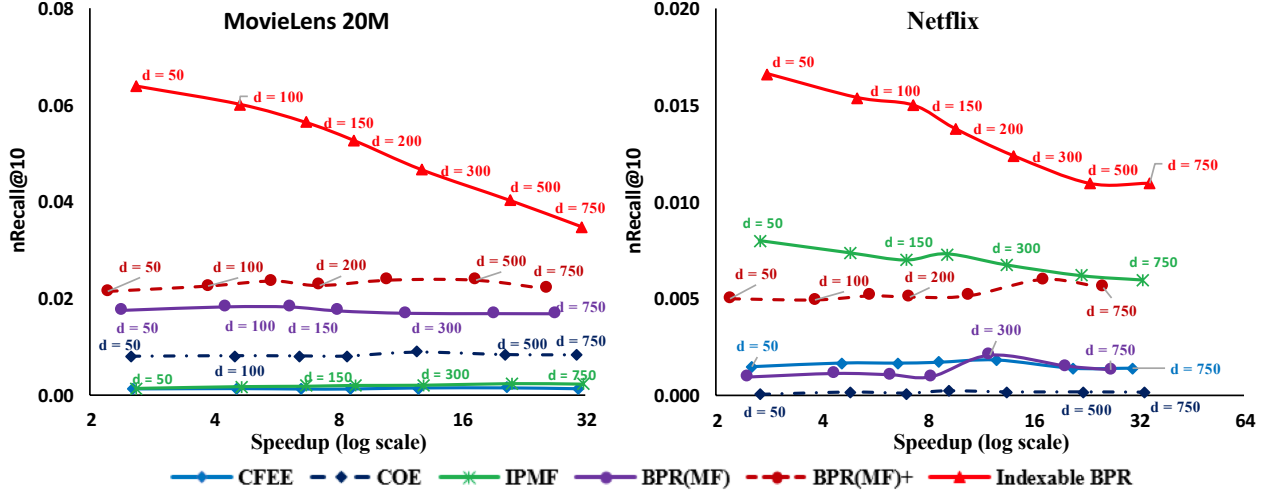


Figure 4.7: nRecall@10 vs. Speedup with Inverted Indexing.

4.4 Analysis on LSH-friendliness of Indexable BPR

Since LSH is inherently an approximate method, the loss of information caused by random hash functions is inevitable. Informally, a representation is LSH-friendly if the loss after hashing is as minimal as possible. To achieve such small loss, a user’s ranking of items based on the latent vectors should be preserved by the hashcodes.

Analysis. For x_u, y_i, y_j in \mathbb{R}^D , one can estimate the probability of the corresponding hashcodes to preserve the correct ordering between them. Since the hash functions h_1, h_2, \dots, h_b are independent of one another, $\|\mathbf{h}(x_u) - \mathbf{h}(y_i)\|_H$ follows the binomial distribution with mean $bp_{x_u y_i}$ and variance $bp_{x_u y_i}(1 - p_{x_u y_i})$, where $p_{x_u y_i}$ is the probability of x_u and y_i having different hash values. Since binomial distribution can be approximated by a normal distribution with same mean and variance, and the difference between two normal distributions is also a normal distribution, we have the following estimation on the probability that $\|\mathbf{h}(x_u) - \mathbf{h}(y_j)\|_H > \|\mathbf{h}(x_u) - \mathbf{h}(y_i)\|_H$:

$$\begin{aligned} \Pr(\|\mathbf{h}(x_u) - \mathbf{h}(y_j)\|_H - \|\mathbf{h}(x_u) - \mathbf{h}(y_i)\|_H > 0) \\ \sim \text{Normal}(bp_{x_u y_j} - bp_{x_u y_i}, bp_{x_u y_j}(1 - p_{x_u y_j}) + bp_{x_u y_i}(1 - p_{x_u y_i})) \end{aligned} \quad (4.13)$$

Due to the shape of the normal distribution, Equation 4.13 implies that a higher mean and smaller variance would lead to a higher probability of the hashcode of x_u is more similar to the hashcode of y_i than to the that of y_j . Therefore, for a fixed length b , if indeed u prefers i to j , we say that x_u, y_i, y_j is a more LSH-friendly representation for u, i , and j if the mean value $(p_{x_u y_j} - p_{x_u y_i})$ is higher and the variance $(p_{x_u y_j}(1 - p_{x_u y_j}) + p_{x_u y_i}(1 - p_{x_u y_i}))$ is smaller.

Hence, the mean and the variance in Equation 4.13 could potentially reveal which representation is more LSH-friendly, i.e., preserves information better after hashing. For each user $u \in \mathcal{U}$, let τ_k^u be the set of items in the top- k by a method before hashing, and $\bar{\tau}_k^u$ be all the other items *not* returned by the models. We are interested in whether after hashing, the items in τ_k^u would be closer to the user than the items in $\bar{\tau}_k^u$.

To account for this potential, we introduce two measures *MeanNorm@k* and *VarNorm@k*, defined as following:

$$\text{MeanNorm@k} = \frac{1}{|\mathcal{U}|} \sum_{i \in \tau_k^u} \sum_{j \in \bar{\tau}_k^u} \frac{(p_{x_u y_j} - p_{x_u y_i})}{|\tau_k^u| \cdot |\bar{\tau}_k^u|}$$

$$\text{VarNorm@k} = \frac{1}{|\mathcal{U}|} \sum_{i \in \tau_k^u} \sum_{j \in \bar{\tau}_k^u} \frac{p_{x_u y_j}(1 - p_{x_u y_j}) + p_{x_u y_i}(1 - p_{x_u y_i})}{|\tau_k^u| \cdot |\bar{\tau}_k^u|}$$

To achieve LSH-friendly representation, *MeanNorm@k* should be high and *VarNorm@k* should be low. Figure 4.8 shows the bar charts displaying values of those metrics. From Figure 4.8, INDEXABLE BPR shows higher mean values *MeanNorm@10* (i.e., $k = 10$) at $d = 20$ (we observe the same results with other values of D and k). Though BPR(MF) and BPR(MF)+ have smaller variance, their mean values are among the lowest. This result gives us a hint that INDEXABLE BPR can preserve information after hashing more effectively.

Compatible Hash Function. There is an explanation for the superior numbers of INDEXABLE BPR in Figure 4.8. Specifically, the probability $p_{x_u y_i}$ depends on the LSH family. In particular, signed random projections [19, 41] or SRP-LSH is meant for angular similarity. The angular

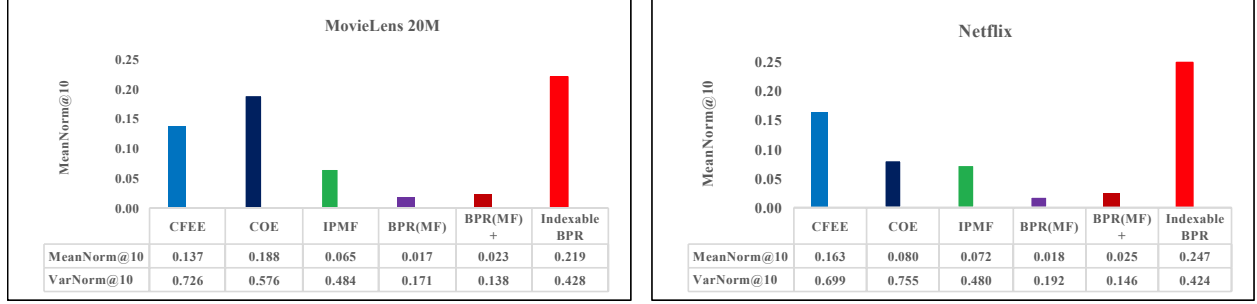


Figure 4.8: LSH Friendly Measurement at $d = 20$.

similarity between x, y is defined as $sim_{\angle}(x, y) = 1 - \cos^{-1}(\frac{x^T y}{\|x\| \cdot \|y\|})/\pi$.

The hash function is defined as $h_a^{srp}(x) = \text{sign}(a^T x)$, in which the parameter a is a random vector chosen with each component from i.i.d normal.

The probability of x, y having different hash values is:

$$p_{xy} = \Pr(h_a^{srp}(x) \neq h_a^{srp}(y)) = \cos^{-1}(\frac{x^T y}{\|x\| \cdot \|y\|})/\pi = \frac{\theta_{xy}}{\pi}, \quad (4.14)$$

For INDEXABLE BPR, as shown in Equation 4.8, for each observation “ u prefers i to j ”, we would like to maximize the difference $\theta_{x_u y_j} - \theta_{x_u y_i}$. From Equation 4.14, we observe that the probability $p_{x_u y_i}$ is a *linear* function of the angular distance $\theta_{x_u y_i}$. Thus, we can infer that INDEXABLE BPR’s objective corresponds to maximizing $p_{x_u y_j} - p_{x_u y_i}$. According to Equation 4.13, this increases the probability that the Hamming distance between u and i is smaller than that between u and j . In other words, the hashcodes are likely to preserve the ranking order. This alignment between the objective of INDEXABLE BPR and the structural property of SRP-LSH helps the model minimize information loss, and show better post-indexing performance.

Also, the appropriate LSH family for methods based on l_2 distance, which includes COE, is L2-LSH [21]. However, there is a question as to how compatible the objective of COE is with the hash functions. The hash function of L2-LSH is defined as follows:

$$h_{a,b}^{L_2}(x) = \lfloor \frac{a^T x + b}{r} \rfloor; \quad (4.15)$$

where r - the window size, a - random vector with each component from i.i.d normal and a scalar $b \sim \text{Uni}(0, r)$. The probability of two points x, y having different hash values under L2-LSH function is:

$$F_r^{L2}(d_{xy}) = \Pr(h_{a,b}^{L2}(x) \neq h_{a,b}^{L2}(y)) = 2\phi\left(-\frac{r}{d_{xy}}\right) + \frac{1}{\sqrt{(2\pi)}(r/d_{xy})} \left(1 - \exp\left(-\left(\frac{r}{d_{xy}}\right)^2/2\right)\right); \quad (4.16)$$

where $\phi(x)$ is cumulative probability function of normal distribution and $d_{xy} = \|x - y\|$ is the l_2 distance between x, y . From Equation 4.16, we see that $F_r^{L2}(d_{xy})$ is a *nonlinear* monotonically increasing function of d_{xy} . COE's objective to maximize $d_{x_u y_j} - d_{x_u y_i}$ does not directly maximize the corresponding mean value of the normal distribution, i.e., $F_r^{L2}(d_{x_u y_j}) - F_r^{L2}(d_{x_u y_i})$, since $F_r^{L2}(d_{x_u y_j})$ is *not a linear* function of l_2 distance $d_{x_u y_j}$. Our hypothesis is that though both rely on ordinal triples, COE may not be as compatible with LSH as INDEXABLE BPR.

Empirical Evidence. For each user u , we rank the items that u has rated in the test set, and measure how closely the ranked list is to the ordering by ground-truth ratings. As metric, we turn to the well-established metric for ranking $nDCG@k$, where k is the cut-off point for the ranked list. Figure 4.9 shows the $nDCG@10$ values for *MovieLens 20M* and *Netflix* respectively at various

	MovieLens 20M - nDCG@10						Netflix - nDCG@10					
	Absolute			Relative			Absolute			Relative		
b	8	12	16	8	12	16	8	12	16	8	12	16
CFEE	0.582	0.582	0.585	0.805	0.806	0.809	0.559	0.561	0.562	0.834	0.836	0.838
COE	0.605	0.609	0.608	0.886	0.891	0.890	0.570	0.565	0.575	0.906	0.898	0.914
IPMF	0.702	0.728	0.704	0.920	0.955	0.923	0.705	0.737	0.747	0.896	0.936	0.949
BPR(MF)	0.599	0.603	0.605	0.831	0.837	0.840	0.560	0.551	0.553	0.863	0.849	0.853
BPR(MF)+	0.603	0.604	0.606	0.837	0.840	0.841	0.569	0.569	0.566	0.877	0.877	0.873
Indexable BPR	0.743	0.745	0.754	0.977	0.980	0.991	0.732	0.761	0.756	0.924	0.960	0.954

Table 4.2: *Absolute and Relative nDCG@10* of all models as the length of LSH codes (b) varies.

dimensionality of the latent vectors D . We observe that, INDEXABLE BPR is among the best, with the most competitive baseline being IPMF (which fits ratings). More important is whether

the models will still perform well when used with index structures. As similar trends are observed with other values of d , subsequently we show results based on $d = 20$.

Here, the objective is to investigate the effectiveness of the LSH hashcodes in preserving the ranking among the *rated* items in the test set. We use *Hamming ranking*, repeating the same experiment in Figure 4.9, but using Hamming distances over hashcodes. This is to investigate how well INDEXABLE BPR preserves the ranking compared to the baselines. As hashing relies on random hash functions, we average results over 10 different sets of functions. Table 4.2 shows

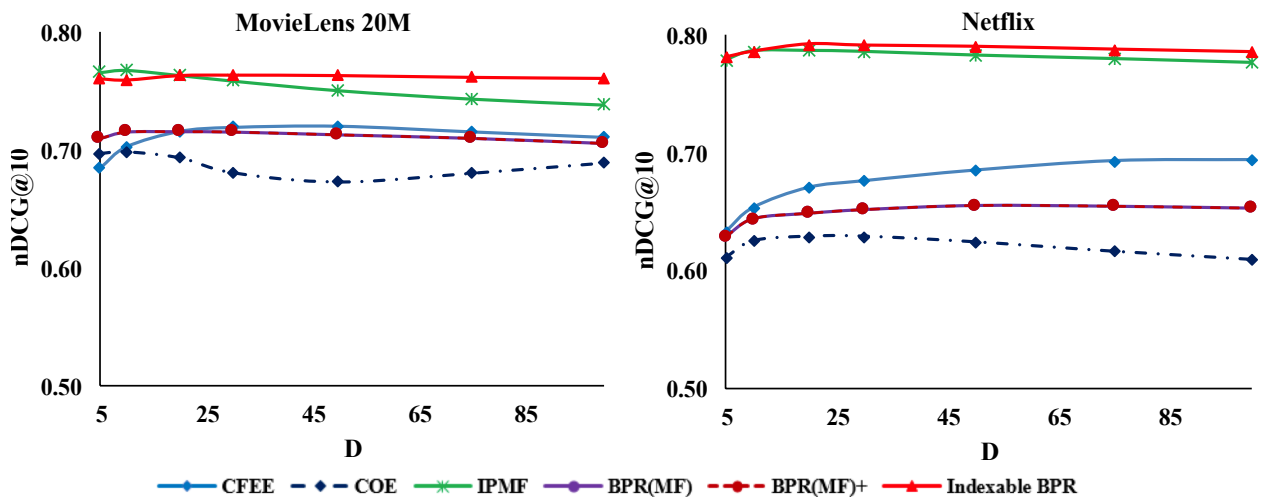


Figure 4.9: $nDCG@10$ at $d \in \{5, 10, 20, 30, 50, 75, 100\}$.

the performances of all models. The two metrics are: *Absolute nDCG@10* is the $nDCG@10$ of LSH hashcodes, and *Relative nDCG@10* is the relative ratio between the Absolute $nDCG@10$ and that of original real-valued latent vectors. INDEXABLE BPR consistently shows better *Absolute nDCG@10* values than the baselines when using LSH indexing. This implies that INDEXABLE BPR coupled with SRP-LSH produces more compact and informative hashcodes. Also, the *Relative nDCG@10* of INDEXABLE BPR are close to 1 and higher than those of the baselines. These observations validate our hypotheses that not only is INDEXABLE BPR competitively effective pre-indexing, but it is also more LSH-friendly, resulting in less loss in the ranking accuracy post-indexing.

4.5 Discussion

In this chapter, we propose a probabilistic method for modeling user preferences based on ordinal triples, which is geared towards top- k recommendation via approximate kNN search using indexing. The proposed model INDEXABLE BPR produces an indexing-friendly representation, which results in significant speedups in top- k retrieval, while still maintaining high accuracy due to its compatibility with indexing structures such as LSH, spatial tree, and inverted index. As future work, a potential direction is to go beyond achieving representations more compatible with existing indexing schemes, to designing novel data structures or indexing schemes that would better support efficient and accurate recommendation retrieval.

Part II

Similarity Learning: Modelling Multiple Perspectives

Chapter 5

Multiperspective Graph-Theoretic Similarity Measure

In many real-world scenarios, there emerge multiple perspectives of similarity, i.e., two objects may be similar from one perspective, but dissimilar from another. For instance, human subjects may generate varied, yet valid, clusterings of objects. In this chapter, we propose a graph-theoretic similarity measure for modelling these multiple perspectives effectively. In our approach, the observed object-to-object relationships due to various perspectives are integrated into a unified graph-based representation, stylised as a hypergraph to retain the distinct perspectives. We then introduce a novel model for learning and reflecting diverse similarity perceptions given the hypergraph, yielding the similarity score between any pair of objects from any perspective. In addition to proposing an algorithm for computing the similarity scores, we also provide theoretical guarantees on the convergence of the algorithm.

5.1 Introduction

There are various ways to measure similarity. Some, such as cosine similarity, are based on content or features, e.g., whether two documents contain the same words, or two products have the same attributes. Others, such as KL-divergence, are based on probability distributions. Yet other measures may be domain-specific, such as sequence alignment [25]. These diverse types of similarity are orthogonal, reflecting various aspects. They are not so much alternatives as complements, and indeed they have been used in conjunction in some applications such as entity resolution.

Problem. Here, we focus on the notion of *graph-theoretic similarity*, based on object-to-object “relationships” (the specific definition of which may be domain-dependent). For instance, a Web page may link to another; two images may belong to the same Pinterest’s board. In each case, object-to-object relationships become the basis for inferring the similarity between any two objects of interest (pages, images). Naturally, such notion of relationship-based similarity lends itself well to a graph-based formulation, with vertices for objects, and edges for relationships between objects.

SimRank [40] lays a foundation for graph-based similarity measurement, premised on the intuition that the similarity between a pair of objects is dependent on the similarity of other object pairs. We consider two objects (i, j) *similar*, if the two objects are respectively related to other objects k (related to i) and l (related to j) that are themselves *similar*. Under this definition, two Web pages are similar if they respectively link to two other pages that are similar. Two images on Pinterest are similar if they respectively belong to the same boards as two other images that are themselves similar. Two users are similar if they respectively adopt similar products.

However, SimRank is a *uniperspective* measure. It assumes only one perception of similarity. In some scenarios, there are actually multiple perspectives of similarity. What may be similar according to one perspective may be different according to another. This may arise due to different facets of relationships, e.g., two products may be “related” in different ways: browsed together, purchased together, same manufacturer, etc. This may also arise due to different agents that ex-

press the relationships, e.g., someone may group tourist attractions based on activities (strolling, amusement park), while another based on artistic value (architecture, museums) or neighborhoods [116]. A uniperspective approach (e.g., SimRank) is not designed for capturing “different strokes for different folks”.

How then do we cope with the presence of multiple perspectives? There are a couple of *naive* approaches. One is to ignore the multiplicity, creating a uniperspective measure by merging the disparate relationships into a single graph and applying the SimRank on this one graph. Another is to isolate each perspective, creating multiperspective measures by maintaining a distinct graph for each perspective and applying SimRank on each graph separately. The former may underfit, due to a lack of capacity to model idiosyncratic nuances of similarity. The latter may overfit, due to the sparsity of relationships within each perspective and the potential to capture incidental relationship instances that may not generalize.

Proposed Approach. Therefore, we propose a natively *multiperspective* approach to measuring graph-theoretic similarity. As input, we are given not one graph, but multiple graphs corresponding to multiple perspectives, with each graph reflecting relationships among objects from a specific perspective. As output, we seek to measure the similarity between a pair of objects according to a particular perspective. The key intuition underlying this formulation is to model not only the perspective-specific *inter-object* similarity between any pair of objects, but also the *inter-perspective* similarity between any two perspectives. Learning these two similarity measures simultaneously renders an advantage in sharing information across similar perspectives, which helps to address the problem where observations for each perspective are under-sampled.

5.2 Framework

In this section, we provide an overview of the problem formulation and solutions. After formally introducing our notations and defining the data representation in terms of our hypergraph

formulation, we outline the solution framework for deriving the perspective-specific inter-object similarities and inter-perspective similarities.

Problem Formulation. Let $\mathcal{O} = \{o_1, o_2, \dots, o_n\}$ be the universal set of objects for which we seek to infer similarities. Suppose that we are interested in modeling m different perspectives $\mathcal{P} = \{p_1, p_2, \dots, p_m\}$ over the similarities of objects in \mathcal{O} . For each perspective $p \in \mathcal{P}$, we are given a graph $G_p(\mathcal{O}, E_p)$, where $E_p \subseteq \mathcal{O} \times \mathcal{O}$ comprises edges between pairs of objects that p considers related. The collection of such graphs $\mathcal{G} = \{G_1, G_2, \dots, G_m\}$ make up the input to the problem.

Because the respective G_p 's are defined over the same set of vertices \mathcal{O} , we seek a unified representation that allows the integration of the m separate graphs. There are several equivalent representations for what is essentially the same data. One is a multi-labeled graph, with perspectives serving as edge labels. Another is a bipartite graph, with perspectives as one type of vertices, and object pairs as the other type. Since we are as concerned with the inter-perspective similarity as we are with the inter-object similarity, for most of the subsequent discussions, we resort to a representation where perspectives and objects are both vertices. A natural candidate for such a representation is a 3-uniform hypergraph, whereby each edge relates exactly three vertices: one *perspective* vertex and two *object* vertices considered similar by the former.

From the input \mathcal{G} , we construct a 3-uniform hypergraph $\mathcal{H} = (\mathcal{X}, \mathcal{E})$ consisting of a set of vertices $\mathcal{X} = \mathcal{P} \cup \mathcal{O}$ and a set of hyperedges $\mathcal{E} = \{(p_k, o_i, o_j) : 1 \leq k \leq m; 1 \leq i, j \leq n\}$, in which $(p_k, o_i, o_j) \in \mathcal{E}$ means that o_i and o_j are related according to perspective p_k , i.e., $(o_i, o_j) \in E_{p_k}$ in G_{p_k} . Figure 5.1 illustrates an example hypergraph with two perspectives $\mathcal{P} = \{p_1, p_2\}$ (red) and four objects $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$ (green). Hyperedges (p_1, o_1, o_4) and (p_1, o_2, o_3) indicate that according to perspective p_1 , object o_1 is related to object o_4 , while object o_2 is related to object o_3 . In contrast, according to p_2 , o_1 is related to o_2 , and o_3 is related to o_4 .

Given a multiperspective hypergraph $\mathcal{H}(\mathcal{X}, \mathcal{E})$, the similarity score of two objects $o_i, o_j \in \mathcal{O}$ according to perspective $p \in \mathcal{P}$ is denoted as $S_p(o_i, o_j)$, whose value is bounded by $[0, 1]$. A special

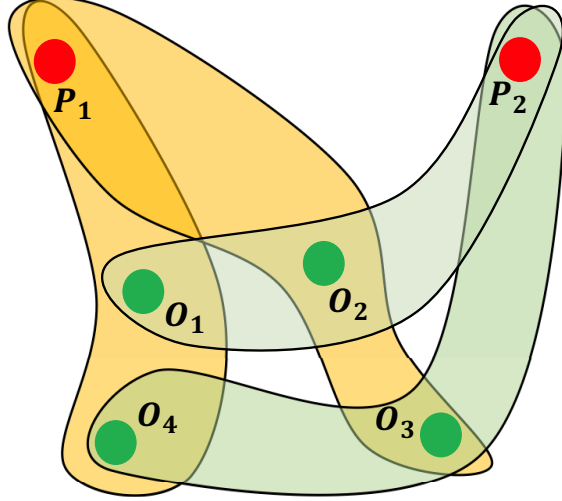


Figure 5.1: Illustration of the Hypergraph Representation

case is $S_p(o_i, o_j) = 1$ when $i = j$. We are now ready to state the problem formally as follows.

Problem 5.1 (Multiperspective Similarity). *Given a multiperspective hypergraph \mathcal{H} , determine the similarity score $S_p(o_i, o_j)$ for each perspective $p \in \mathcal{P}$ and pair of objects $o_i \neq o_j \in \mathcal{O}$.*

Proposed Methodology. To recall how SimRank measures the similarity between two vertices based on the graph structure, we reproduce its similarity measure here. Formally, the SimRank similarity score $S(a, b)$ between two vertices a, b is defined as follows:

$$S(a, b) = \begin{cases} \frac{C}{|N(a)||N(b)|} \sum_{i=1}^{|N(a)|} \sum_{j=1}^{|N(b)|} S(N_i(a), N_j(b)), & \text{if } a \neq b, \\ 1, & \text{if } a = b \end{cases} \quad (5.1)$$

in which C is the damping factor between 0 and 1; $N(a)$ and $N(b)$ comprise the neighbors of a and b respectively. If a vertex a has no neighbor, then we have $S(a, b) = 0$ for any vertex $b \neq a$.

A naive solution to Problem 5.1 is to run SimRank (Equation 5.1) on each perspective's component graph G_p separately. We refer to this solution as *Disjoint-SimRank*. While this produces perspective-specific inter-object similarities, the main issue is that there may not be sufficient information within each G_p to learn the similarities among objects effectively. If every perspective

is distinct and unique, then perhaps we could do no better than this. However, realistically, the various perspectives may share some degree of agreement in how they perceive the similarities among objects. If so, then there would be an opportunity to let a perspective collaborate with other similar perspectives, filling the gaps in each other’s knowledge of object similarities.

Therefore, for a truly multiperspective solution, we advocate enabling information sharing across perspectives, to a degree correlated with the similarity among the corresponding perspectives. Let’s denote $\text{sim}(p, p') \in [0, 1]$ to be the similarity between two perspectives $p, p' \in \mathcal{P}$. How these values may be derived for $p, p' \in \mathcal{P}$ will be discussed shortly.

To infer the similarity $S_p(o_i, o_j)$ between o_i and o_j according to p , we propose to expand the definition in Equation 5.1 to incorporate inter-perspective similarity $\text{sim}(p, p')$, in such a way that $S_p(o_i, o_j)$ is expressed in terms of the corresponding object similarities according to other perspectives p' as well, as shown in Equation 5.2. Here, $N_p(o_i)$ comprises the neighbors of o_i in G_p .

$$S_p(o_i, o_j) = \frac{C}{|\mathcal{P}|} \sum_{p' \in \mathcal{P}} \text{sim}(p, p') \sum_{o_k \in N_{p'}(o_i)} \sum_{o_l \in N_{p'}(o_j)} \frac{S_{p'}(o_k, o_l)}{|N_{p'}(o_i)||N_{p'}(o_j)|}, \quad (5.2)$$

Equation 5.2 captures a couple of fundamental principles. First, the similarity between two objects depends on the similarities between other objects related to those objects of interest. Second, distinctly in our formulation, the similarity between two objects of interest according to a specific perspective also depends on the similarities between related objects as seen by similar perspectives.

Let $\mathcal{S}_p = [S_p(o_i, o_j)]_{n \times n}$ be the matrix representation of the perspective-specific inter-object similarity scores, and W_p be the column-normalized matrix of the adjacency matrix with respect to $p \in \mathcal{P}$. We can express Equation 5.2 in matrix form as follows:

$$\mathcal{S}_p = \frac{C}{|\mathcal{P}|} \sum_{p' \in \mathcal{P}} \text{sim}(p, p') \cdot W_{p'}^T \mathcal{S}_{p'} W_{p'} \quad (5.3)$$

In this multiperspective framework, one important component is the inter-perspective simi-

larity $\text{sim}(p, p')$, determining the degree to which information is shared between one perspective and another. The straightforward solution is to treat it as a pipeline: first compute the similarity between perspectives, then solve Equation 5.2 to compute the perspective-specific inter-object similarities. We refer to this as PIPELINED-SIMRANK (Section 5.2.1). In Section 5.2.2, we further propose a refined formulation MP-SIMRANK to compute the inter-perspective similarities and the perspective-specific inter-object similarities simultaneously. We expect that jointly learning both types of similarities would reinforce the performance of the framework at lower complexities than the former solution.

5.2.1 Pipelined-SimRank

We now describe PIPELINED-SIMRANK, which enables information sharing across perspectives through a pipelined solution. The key idea is to induce unidirectional dependency from the inter-perspective $\text{sim}(p, p')$ to the inter-object $S_p(o_i, o_j)$, but not the other way around. This directionality implies that $\text{sim}(p, p')$ has to be inferred from the hypergraph \mathcal{H} itself.

Inter-Perspective Similarity

As mentioned in Section 5.2, each perspective p is associated with a graph of object-to-object relationships G_p . Intuitively, we consider two perspectives p and p' to be similar, if their corresponding graphs G_p and $G_{p'}$ are similar, which implies that when p considers two objects related, it is likely that p' does as well. We express this intuition in graph-theoretic form as follows.

Let us transform the input hypergraph $\mathcal{H} = (\{\mathcal{P}, \mathcal{O}\}, \mathcal{E})$ into a bipartite graph \mathcal{B} with two types of vertices, as illustrated in Figure 5.2 (unrelated to Figure 5.1). The first type are perspective vertices \mathcal{P} (left). The second type are “object-pair” vertices $\mathcal{O} \times \mathcal{O}$, formed from all pairs of non-identical objects (right). An edge from a perspective p to an object-pair vertex o_{ij} exists in this bipartite graph \mathcal{B} iff $(p, o_i, o_j) \in \mathcal{E}$ in the original hypergraph \mathcal{H} .

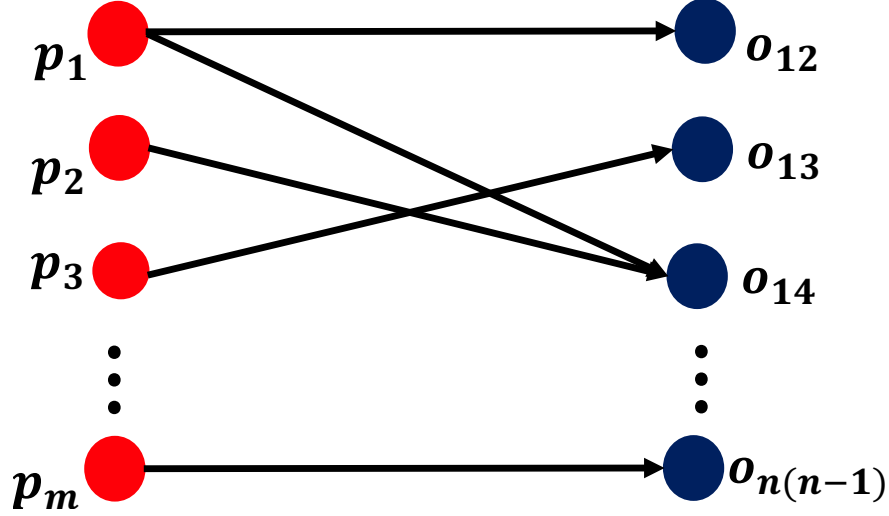


Figure 5.2: PIPELINED-SIMRANK: Bipartite graph for computing similarity between perspective nodes

Once the bipartite graph \mathcal{B} is in place, we can apply a graph-theoretic measure such as the bipartite variant of SimRank [40] to compute the inter-perspective similarity $\text{sim}(p, p')$, which will be used in the next phase for computing $S_p(o_i, o_j)$.

Learning Algorithm

Algorithm 3 encapsulates the pipelined solution PIPELINED-SIMRANK, which involves two phases. In the first phase, we compute inter-perspective similarities $\text{sim}(p, p')$ for all $p, p' \in \mathcal{P}$ as described above. Thereafter, in the second phase, we use these inter-perspective similarities in Equation 5.2 to compute the inter-object similarities for each perspective. Note that $\text{sim}(p, p')$ is now fixed in the second phase. The initial values $S_p^{(0)}(*, *) \forall p \in \mathcal{P}$ at the start of the iterations (line 7) of Algorithm 3 are specified in Equation 5.4 below:

$$S_p^{(0)}(o_i, o_j) = 0 \text{ if } i \neq j \text{ and } 1 \text{ if } i = j \quad (5.4)$$

The solution to Equation 5.2 can be reached by iteration to a fixed-point. Finally, the algorithm returns the converged inter-object similarities, as well as the inter-perspective similarities.

Algorithm 3 PIPELINED-SIMRANK

Require: Hypergraph \mathcal{H} (defined as in Section 5.2)

- 1: /*— create bipartite graph from hypergraph —*/
- 2: $\mathcal{B} \leftarrow \text{bipartiteTransform}(\mathcal{H})$
- 3:
- 4: /*— compute the similarity between perspectives —*/
- 5: $\{\text{sim}^{(*)}(p, p')\}_{\forall p, p' \in \mathcal{P}} \leftarrow \text{bipartiteSimRank}(\mathcal{B})$
- 6:
- 7: **Initialize** $S_p^{(0)} \leftarrow \mathbf{I}_n, \forall p \in \mathcal{P}$
- 8: **while** not converged **do** {

$$S_p^{(t+1)}(o_i, o_j) = \frac{C}{|\mathcal{P}|} \sum_{p' \in \mathcal{P}} \text{sim}^{(*)}(p, p') \quad (5.5)$$

$$\times \sum_{o_k \in N_{p'}(o_i)} \sum_{o_l \in N_{p'}(o_j)} \frac{S_{p'}^{(t)}(o_k, o_l)}{|N_{p'}(o_i)| |N_{p'}(o_j)|},$$

(for $1 \leq i \neq j \leq n$)

$$\text{and } S_p^{(t+1)}(o_i, o_i) = 1 \text{ (for } 1 \leq i \leq n)$$

9: **end while**

10:

11: **Return** $\{S_p^{\text{converged}}(o_i, o_j), \forall p \in \mathcal{P}, o_i, o_j \in \mathcal{O}\}$

12: and $\{\text{sim}^{(*)}(p, p'), \forall p, p' \in \mathcal{P}\}$.

Convergence Property

We now prove that Algorithm 3 will eventually converge, showing the existence of a simultaneous solution of Equation 5.2.

Lemma 5.1. *The sequence of perspective-specific similarity score produced by Algorithm 3 is non-decreasing and bounded by $[0, 1]$, i.e., for $p \in \mathcal{P}, o_i, o_j \in \mathcal{O}, t \geq 0$.*

$$1 \geq S_p^{(t+1)}(o_i, o_j) \geq S_p^{(t)}(o_i, o_j) \geq 0,$$

Proof: From the initialization step and update equations (5.5) (described in Algorithm 3), it is straightforward to see that:

$$S_p^{(1)}(o_i, o_j) \geq 0 = S_p^{(0)}(o_i, o_j), \forall p \in \mathcal{P}, o_i \neq o_j \in \mathcal{O}$$

$$\text{and } S_p^{(1)}(o_i, o_i) = 1 = S_p^{(0)}(o_i, o_i), \forall p \in \mathcal{P}, o_i \in \mathcal{O}.$$

That means Lemma 5.1 is true for $t = 0$. By induction, one can verify the statement in Lemma 5.1 still holds true for $\forall t \geq 1$.

Hence, each sequence $\{S_p^{(t)}(o_i, o_j)\}_{t \geq 0}$ is non-decreasing and bounded. By the Completeness Axiom of calculus, each sequence $\{S_p^{(t)}(o_i, o_j)\}_{t \geq 0}$ therefore converges to a limit $S_p(o_i, o_j) \in [0, 1]$. Moreover, $\{S_p(o_i, o_j)\}$ and $\{\text{sim}^*(p, p')\}$ are the solution for Eq. 5.2.

5.2.2 Joint Solution: MP-SimRank

We now describe our proposed *joint* solution MultiPerspective SimRank or MP-SIMRANK. The key idea is to induce bidirectional dependencies between the inter-perspective $\text{sim}(p, p')$ and the inter-object $S_p(o_i, o_j)$ similarities.

Inter-Perspective Similarity

The dependency from the inter-perspective $\text{sim}(p, p')$ to inter-object S_p is already encoded in Equation 5.2. To induce the dependency in the opposite direction, we need to define $\text{sim}(p, p')$ in terms of S_p . While there could be many possible definitions, we propose the following definition in Equation 5.6, which, as we will show later would still preserve the convergence property.

$$\text{sim}(p, p') = 1 - \frac{\|S_p - S_{p'}\|_F}{n}, \quad (5.6)$$

The similarity between two perspectives p, p' is inversely proportional to the Frobenius norm between \mathcal{S}_p and $\mathcal{S}_{p'}$. If they are similar, i.e., $\frac{\|\mathcal{S}_p - \mathcal{S}_{p'}\|_F}{n}$ is close to 0 then $\text{sim}(p, p')$ is close to 1. Otherwise, if \mathcal{S}_p and $\mathcal{S}_{p'}$ are extremely different, i.e., $\frac{\|\mathcal{S}_p - \mathcal{S}_{p'}\|_F}{n}$ is close to 1, then $\text{sim}(p, p')$ is close to 0.

Algorithm 4 MP-SIMRANK

Require: Hypergraph \mathcal{H} (defined as in Section 5.2)

- 1: Initialize $\mathcal{S}_p^{(0)} \leftarrow I_n, \forall p \in \mathcal{P}$
- 2: Initialize $\text{sim}^{(0)}(p, p') = 1$ if $p = p'$ and 0 if $p \neq p'$
- 3:
- 4: **while** not converged **do** {
- 5:

$$\mathcal{S}_p^{(t+1)}(o_i, o_j) = \frac{C}{|\mathcal{P}|} \sum_{p' \in \mathcal{P}} \text{sim}^{(t)}(p, p') \times \sum_{o_k \in N_{p'}(o_i)} \sum_{o_l \in N_{p'}(o_j)} \frac{\mathcal{S}_{p'}^{(t)}(o_k, o_l)}{|N_{p'}(o_i)| |N_{p'}(o_j)|}, (1 \leq i \neq j \leq n)$$

and $\mathcal{S}_p^{(t+1)}(o_i, o_i) = 1$ (for $1 \leq i \leq n$)

- 6: $\text{sim}^{(t+1)}(p, p') = 1 - \frac{\|\mathcal{S}_p^{(t+1)} - \mathcal{S}_{p'}^{(t+1)}\|_F}{n}, \forall p, p' \in \mathcal{P}$
 - 7: **end while**}
 - 8:
 - 9: **Return** $\{\mathcal{S}_p^{\text{converged}}(o_i, o_j), \forall p \in \mathcal{P}, o_i, o_j \in \mathcal{O}\}$ and $\{\text{sim}^{\text{converged}}(p, p'), \forall p, p' \in \mathcal{P}\}$.
-

Learning Algorithm

Algorithm 4 shows the joint-learning solution for Equation 5.2. We initialize the perspective-specific similarity score $\mathcal{S}_p^{(0)}(*, *) \forall p \in \mathcal{P}$ as in Equation 5.4. For the similarity between perspectives, we initialize $\text{sim}^{(0)}(p, p') \forall p, p' \in \mathcal{P}$ as in Equation 5.7 below.

$$\text{sim}^{(0)}(p, p') = 0 \text{ if } p \neq p' \text{ and } 1 \text{ if } p = p' \tag{5.7}$$

In contrast to the two-phase Algorithm 3, in this Algorithm 4 we iterate the computation of inter-object similarity in line 4 and that of inter-perspective similarity in line 5 until both converge.

Convergence Property

For MP-SIMRANK, we show that the computations for both types of similarities will converge to a fixed point.

Lemma 5.2. *The sequence of similarity between perspectives produced by Algorithm 4 is non-decreasing and bounded by $[0, 1]$, i.e., for $t \geq 1$,*

$$1 \geq \text{sim}^{(t+1)}(p, p') \geq \text{sim}^{(t)}(p, p') \geq 0, \forall p, p' \in \mathcal{P}. \quad (5.8)$$

Proof: Proving that, for $t \geq 0$: $\left\| \mathcal{S}_p^{(t+1)} - \mathcal{S}_{p'}^{(t+1)} \right\|_F \leq \left\| \mathcal{S}_p^{(t)} - \mathcal{S}_{p'}^{(t)} \right\|_F$

From Equation 5.3, $\forall p, p' \in \mathcal{P}$ we have:

$$\begin{aligned} \left\| \mathcal{S}_p^{(t+1)} - \mathcal{S}_{p'}^{(t+1)} \right\|_F &= \left\| \frac{C}{|\mathcal{P}|} \sum_{p'' \in \mathcal{P}} (\text{sim}^{(t)}(p, p'') - \text{sim}^{(t)}(p', p'')) W_{p''}^T \cdot \mathcal{S}_{p''}^{(t)} \cdot W_{p''} \right\|_F \\ &\leq \frac{C}{|\mathcal{P}|} \sum_{p'' \in \mathcal{P}} |\text{sim}^{(t)}(p, p'') - \text{sim}^{(t)}(p', p'')| \cdot \left\| W_{p''}^T \cdot \mathcal{S}_{p''}^{(t)} \cdot W_{p''} \right\|_F \\ &= \frac{C}{|\mathcal{P}|} \sum_{p'' \in \mathcal{P}} \left| \left\| \mathcal{S}_p^{(t)} - \mathcal{S}_{p''}^{(t)} \right\|_F - \left\| \mathcal{S}_{p'}^{(t)} - \mathcal{S}_{p''}^{(t)} \right\|_F \right| \cdot \frac{\left\| W_{p''}^T \cdot \mathcal{S}_{p''}^{(t)} \cdot W_{p''} \right\|_F}{n} \\ &= \frac{C}{|\mathcal{P}|} \sum_{p'' \in \mathcal{P}} \left\| (\mathcal{S}_p^{(t)} - \mathcal{S}_{p''}^{(t)}) - (\mathcal{S}_{p'}^{(t)} - \mathcal{S}_{p''}^{(t)}) \right\|_F \cdot \frac{\left\| W_{p''}^T \cdot \mathcal{S}_{p''}^{(t)} \cdot W_{p''} \right\|_F}{n} \\ &\leq \frac{C}{|\mathcal{P}|} \sum_{p'' \in \mathcal{P}} \left\| \mathcal{S}_p^{(t)} - \mathcal{S}_{p'}^{(t)} \right\|_F < \left\| \mathcal{S}_p^{(t)} - \mathcal{S}_{p'}^{(t)} \right\|_F \\ &\Rightarrow \text{sim}^{(t+1)}(p, p') \geq \text{sim}^{(t)}(p, p'). \end{aligned}$$

Since, $0 \leq \frac{\left\| \mathcal{S}_p^{(t)} - \mathcal{S}_{p'}^{(t)} \right\|_F}{n} \leq 1, \forall t \geq 1$ and $p, p' \in \mathcal{P}$, we also have $\text{sim}^{(t)}(p, p') \in [0, 1], \forall t \geq 0$ and $p, p' \in \mathcal{P}$. By the Completeness Axiom of calculus, $\text{sim}^{(t)}(p, p')$ converges to a limit $\text{sim}(p, p')$.

From Lemma 5.2 and by induction, we can prove that Lemma 5.1 still holds true for the perspective-specific sequences produced by Algorithm 4. That means $\{\mathcal{S}_p^{(t)}(o_i, o_j)\}_{t \geq 0}$ converges to a limit $\mathcal{S}_p(o_i, o_j) \in [0, 1]$ and $\{\text{sim}^{(t+1)}(p, p')\}_{t \geq 0}$ converges to a limit $\text{sim}(p, p')$. Moreover, $\mathcal{S}_p(o_i, o_j)$ and $\text{sim}(p, p')$ solve Equation 5.2.

5.3 Experiments

Our experimental objectives are to study the comparative performance of the proposed graph-theoretic multiperspective approach against comparable baselines, and to investigate the role and effectiveness of inter-perspective similarities.

5.3.1 Experimental Settings

Datasets. For experiments, we seek publicly available datasets that could reflect the notion of multiperspectivity. We identify the following three datasets, whereby the first two model multiperspectivity due to different facets or attributes of objects, and the third models multiperspectivity due to different agents.

*Zoo*¹ contains 101 animals with 17 attributes (excluding name), e.g., *#legs*, *type* (mammals, birds, etc.). We treat attribute as perspective and animal as object, and model the varying similarity of animals according to attributes. We form a hyperedge (p, o_i, o_j) if o_i and o_j have the same value for p . For example, one hyperedge is $(\text{\#legs}, \text{elephant}, \text{giraffe})$, since both animals have four legs.

*Congressional Voting Records (or HouseVote)*² contains 435 instances (congress members) and 16 attributes (votes). After excluding instances with missing values, we get a dataset with 232 instances. Considering each attribute as a perspective, we generate hypergraph in the same way as we do with *Zoo* dataset.

¹<https://archive.ics.uci.edu/ml/datasets/Zoo>

²<https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records>

*Paris Attractions*³ has 237 users organize 250 attractions in Paris into clusters. Each is a group of similar attractions from the perspective of a user. We induce hyperedges involving two attractions i and j that the user (perspective) puts into the same cluster.

The density ratio is measured by dividing the number of present hyperedges by the maximum number of hyperedges possible, i.e., $m * n^2$. Paris Attractions has the lowest density at 0.16%, as compared to 57.2% for Zoo and 52.3% for HouseVote.

Task and Metrics. We evaluate similarity methods as follows. In each dataset, a perspective is associated with a clustering of objects (based on attribute values or groupings). For each cluster, we sample 70% of objects for training, and keep the 30% hidden for testing. From the training set, we induce a hypergraph, and learn the similarity scores. At the prediction stage for each perspective, we measure the affinity between a hidden object and the clusters, and assign the object to the highest-affinity cluster. Here, affinity is the average similarity (as measured by the comparative method) between the hidden object and the known objects in the cluster.

While presences of hyperedges indicate similarity, absences may not necessarily indicate dissimilarity (maybe missing values). Thus, we evaluate predictions via two recall-oriented metrics. We conduct stratified sampling to maintain the same ratio for each perspective and report the average results over ten train/test splits.

Recall: For a $p \in \mathcal{P}$, hiding an object from one of its clusters essentially creates hidden hyperedges in the test set involving the perspective, the hidden object, and other objects in the cluster. Correspondingly, at prediction stage, the assignment of a hidden object to the highest-affinity cluster “predicts” another set of hyperedges. Let $\mathcal{E}_p^{\text{hid}}$ denote the former, and $\mathcal{E}_p^{\text{pred}}$ the latter. *Recall* is the fraction of $\mathcal{E}_p^{\text{hid}}$ recovered by $\mathcal{E}_p^{\text{pred}}$, averaged across perspectives.

$$\text{Recall} = \frac{1}{m} \sum_{p \in \mathcal{P}} \frac{|\mathcal{E}_p^{\text{pred}} \cap \mathcal{E}_p^{\text{hid}}|}{|\mathcal{E}_p^{\text{hid}}|} \quad (5.9)$$

³http://projects.yisongyue.com/collab_cluster/

PRES: As the recall measure above relies on discrete assignments, we use a second metric that relies on rankings. For a cluster, we rank the candidate objects based on the affinity scores. We then evaluate the rank positions of the ground-truth hidden objects using *PRES* (Patent Retrieval Evaluation Score) [68], which had been designed for recall-oriented retrieval tasks. Equation 5.10 shows the formula for a cluster of a given perspective, where n is the number of ground-truth objects hidden from this cluster, r_i is the rank order of each ground-truth object in the output, and N_{max} is the total number of candidates. To report the overall result, we average it across the clusters of a perspective, and then across perspectives.

$$PRES = 1 - \frac{\sum r_i - \frac{n+1}{2}}{N_{max}} \quad (5.10)$$

Methods. We compare the two methods described in this work: PIPELINED-SIMRANK and MP-SIMRANK to several baselines. Since our work is related to SimRank, and the key contribution is to incorporate native multiperspectivity, our main baselines are variants of SimRank. For all the graph-theoretic methods, including ours, the damping factor C is set to 0.8, as recommended in [40].

The first two are *uniperspective* SimRank-based methods. *Merged-SimRank* is obtained by taking the union of graphs due to different perspectives, and applying SimRank on the merged graph. *Average-SimRank* is obtained by running SimRank on each perspective’s graph independently, and then averaging the SimRank scores to be used as a common inter-object score. Comparing to these uniperspective variants allows us to see the effect of multiperspectivity.

Disjoint-SimRank recognizes multiperspectivity, but assumes they can be obtained separately. For each perspective, we create a single graph to represent its own similarity viewpoint. We then run classic SimRank on each graph independently. In this mode, each perspective can only learn from its own graph, without collaborating with others. Comparing to this variant allows us to see the effect of inter-perspective collaboration that underpins both of our models.

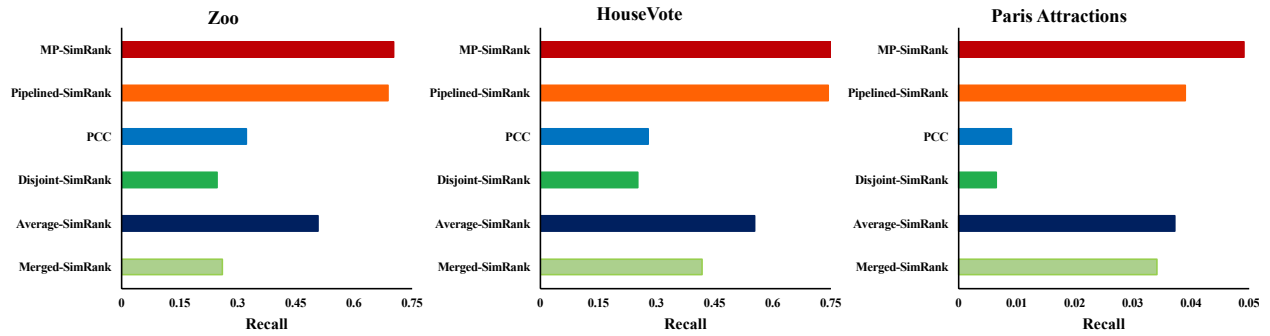


Figure 5.3: Recall values of all models

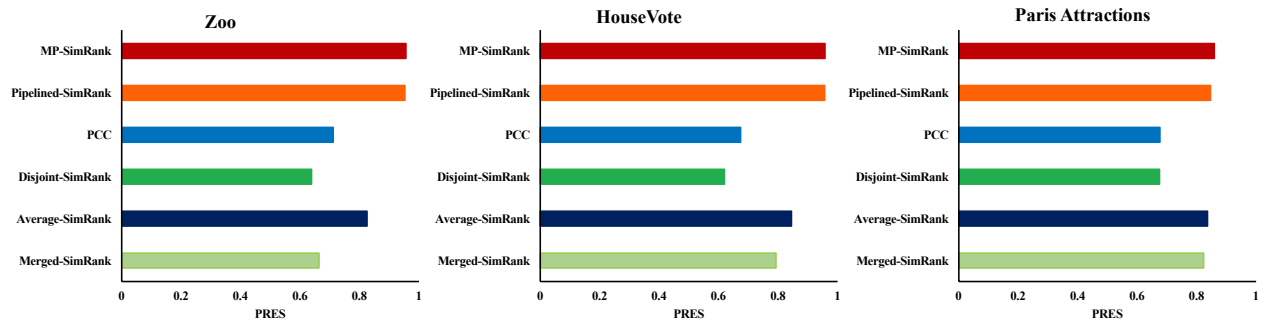


Figure 5.4: PRES values of all models

The final method *Personalized Collaborative Clustering* or *PCC* [116] is not graph-theoretic per se. Given our focus, strictly speaking it is not a baseline. However, it is included for completeness because it supports some notion of multiperspectivity, but relies on matrix factorization. We tune the parameters of PCC (learning rate, dimension of latent space) for its best performance.

5.3.2 Comparison to Baselines

We now discuss the experimental results, focusing on the similarity values among objects. Figure 5.3 shows the *Recall* of all comparative methods across the three datasets.

Disjoint-SimRank is consistently the weakest. Its *Recall* for *Zoo*, *HouseVote*, and *Paris Attractions* are 24.65%, 25.14%, and 0.65%. We attribute this to the lack of information within each perspective, since each only runs SimRank on its own graph.

Merged-SimRank achieves slightly better *Recall* than *Disjoint-SimRank* on three datasets: 26.0%

for *Zoo*, 41.7% for *HouseVote*, and 3.4% for *Paris Attractions*. By pooling together all the perspectives, it learns the consensus view. *Average-SimRank* interestingly achieves the best *Recall* values among all the baselines: 50.7% for *Zoo*, 55.3% for *HouseVote*, and 3.7% for *Paris Attractions*. Perhaps it captures the dominant perspective, as this model would give higher similarity score for those pairs of objects that have been clustered as similar more frequently in the data than the score for other pairs.

The natively multiperspective models show better performances than the uniperspective baselines. PIPELINED-SIMRANK achieves *Recall* of 68.8% for *Zoo*, 74.3% for *HouseVote*, and 3.9% for *Paris Attractions*. MP-SIMRANK is even better, its *Recall* for the three datasets are 70.2%, 75.0%, and 4.9% respectively. This supports our intuition that by modeling multiple perspectives, we can capture nuances specific to some perspectives, and yet still allow collaboration among similar perspectives.

The results for non-graph theoretic *PCC* are middling and mixed, 32.2% for *Zoo*, 27.8% for *HouseVote*, and 0.9% for *Paris Attractions*. It is still better than *Disjoint-SimRank*, ostensibly due to the sharing across perspectives. However, it is not always better than *Merged-SimRank* and is generally worse than *Average-SimRank*.

Figure 5.4 presents the *PRES* values. The trends are consistent with the *Recall* values for all datasets, in terms of the relative performance of the comparative methods. Compared to *Recall*, the *PRES* values tend to be higher. Especially, the *PRES* of the two multiperspective models are close to 1. This indicates that while recalling all ground-truth objects may be challenging, those that we do recall tend to be ranked almost at the top of the candidates.

Comparing the three datasets, *Paris Attractions* is the most sparse, which explains why the *Recall* and *PRES* values of all models for this dataset are relatively lower than that of the two other datasets.

5.3.3 Inter-Perspective Similarities

As a byproduct of determining the similarities among objects, multiperspective models also produce the similarities among perspectives. We are interested in investigating the inter-perspective similarity $\text{sim}(p, p'), \forall p, p' \in \mathcal{P}$ of the two models. Intuitively, for effective information sharing, two “similar” perspectives p, p' should have higher $\text{sim}(p, p')$ value than two “dissimilar” perspectives.

To attempt to understand how meaningful the $\text{sim}(p, p')$ values are, we turn to the concept of Normalized Mutual Information or NMI [27]. In particular, for *Zoo* and *HouseVote*, each perspective corresponds to an attribute, whose values effectively define a clustering of objects. Supposing we see the full dataset, we can quantify how similar two attributes are, using NMI on the two clusterings over the same objects. For each $p \in \mathcal{P}$, we measure the Pearson correlation of its NMI scores and its inter-perspective similarities with other perspectives in \mathcal{P} . We do not include *Paris Attractions*, since each user only clusters a different subset of objects.

Table 5.1 and Table 5.2 show the Pearson correlation values of each perspective for PIPELINED-SIMRANK and MP-SIMRANK respectively. Both achieve high correlation values between the NMI scores and the inter-perspective similarities for each perspective. That means both are able to reflect very well the underlying similarity between two perspectives. The joint learning MP-SIMRANK seems to better learn the similarity between two perspectives than the PIPELINED-SIMRANK. This explains the improvement in the performance of MP-SIMRANK upon PIPELINED-SIMRANK in the earlier experiment.

5.3.4 Illustrative Case Study

To gain an intuition of how multiperspectivity plays a part in the similarity measurement, here we include a small case study. For this example, we use the *Paris Attractions* dataset to showcase the role of multiperspective similarity measure. Table 5.3 shows the clustering data of four users in the

dataset, each represented by user id. Each cluster is separated from another by the symbol $||$. Of particular interest to us are objects with id: 30, 50, 62, 76, and 88 (in bold). We can observe that users may cluster objects similarly or differently from one another. For example, both U53 and U86 place objects 30, 50, 62, and 88 in the same cluster. On the other hand, U94 places object 62 and 88 in the same cluster, but places object 50 in a different cluster. U168, however, places three objects: 30, 76, and 88 in three different clusters.

Perspective	MP-SIMRANK	PIPELINED-SIMRANK
p_1	0.9531	0.6914
p_2	0.8163	0.7702
p_3	0.9576	0.6430
p_3	0.9451	0.6051
p_5	0.8954	0.8514
p_6	0.9669	0.8670
p_7	0.9952	0.9891
p_8	0.9191	0.7410
p_9	0.7920	0.7612
p_{10}	0.8434	0.8208
p_{11}	0.8574	0.9832
p_{12}	0.8445	0.8391
p_{13}	0.9049	0.7840
p_{14}	0.9229	0.8490
p_{15}	0.9139	0.9932
p_{16}	0.9846	0.9400
p_{17}	0.8492	0.6174

Table 5.1: Correlation between NMI scores and inter-perspective similarities for *Zoo*

We apply the MP-SIMRANK on the full *Paris Attractions* dataset and investigate the inter-perspective similarities between the four mentioned users. In Figure 5.5, each big circle represents the clustering data of each user. Clusters are wrapped inside inner circles. The values on the dashed lines represent the Frobenius distance between perspectives (users). We observe that the distance between U53 and U86 is smaller than those between U53 and either U94, U86 or U94. This is expected since U53 and U86 have more similar perspectives. The inter-perspective distances reflect that U53 and U86 are more similar to U94 than to U168. This is reasonable, since U53, U86, and U94 place object 62 and object 88 in the same cluster.

Perspective	MP-SIMRANK	PIPELINED-SIMRANK
p_1	0.9963	0.9763
p_2	0.9999	0.9990
p_3	0.9790	0.8295
p_4	0.9764	0.7586
p_5	0.9733	0.6552
p_6	0.9789	0.8715
p_7	0.9811	0.8732
p_8	0.9749	0.7106
p_9	0.9783	0.7335
p_{10}	1.0000	0.9998
p_{11}	0.9992	0.9975
p_{12}	0.9763	0.7793
p_{13}	0.9865	0.8866
p_{14}	0.9641	0.8197
p_{15}	0.9883	0.9335
p_{16}	0.9713	0.9376

Table 5.2: Correlation between NMI scores and inter-perspective similarities for *HouseVote*

5.4 Efficiency Analysis

This section discusses the theoretical complexity and efficiency of the SimRank-based methods.

5.4.1 Complexity Analysis

First, we look into the theoretical storage and time complexities, which are summarized in Table 5.4. For the uniperspective *Merged-SimRank*, its complexities are the same as the original SimRank’s, which is square to the number of object pairs, i.e., n^2 . Suppose for a given perspective, d_p is the average product of neighbor counts, i.e., $|N_p(o_i)| \cdot |N_p(o_j)|$ across object pairs $o_i, o_j \in \mathcal{O}$. Then d_{\max} is the maximum such average among all perspectives $\forall p \in \mathcal{P}$.

For the methods that require computation for each perspective, *Average-SimRank* and *Disjoint-SimRank*, it is reasonable that the complexity will also scale with m the number of perspectives. However, both of these act independently for each perspective.

For the natively multiperspective methods, there is a need to compute the inter-perspective similarities. For PIPELINED-SIMRANK, this is done by inducing a bipartite graph of perspectives-

ID	Clustering Data
U53	14 21— 30 40 50 62 76 88 —17 156—78 79 106 126 201 232 247
U86	72 78 96 109 164 208—2 30 50 62 88 178 224—79 84 207
U94	7 91 115 140 159 167 248—34 49 62 73 79 88 142 151 238— 50 90 154
U168	40 48 73 84 85 88 89 90 117 154 166 171—45 51 61 76 111 116 126 133 146 200—28 30 52 60 78 86 100 128 132 195—21

Table 5.3: Cluster data of four users from *Paris Attractions*

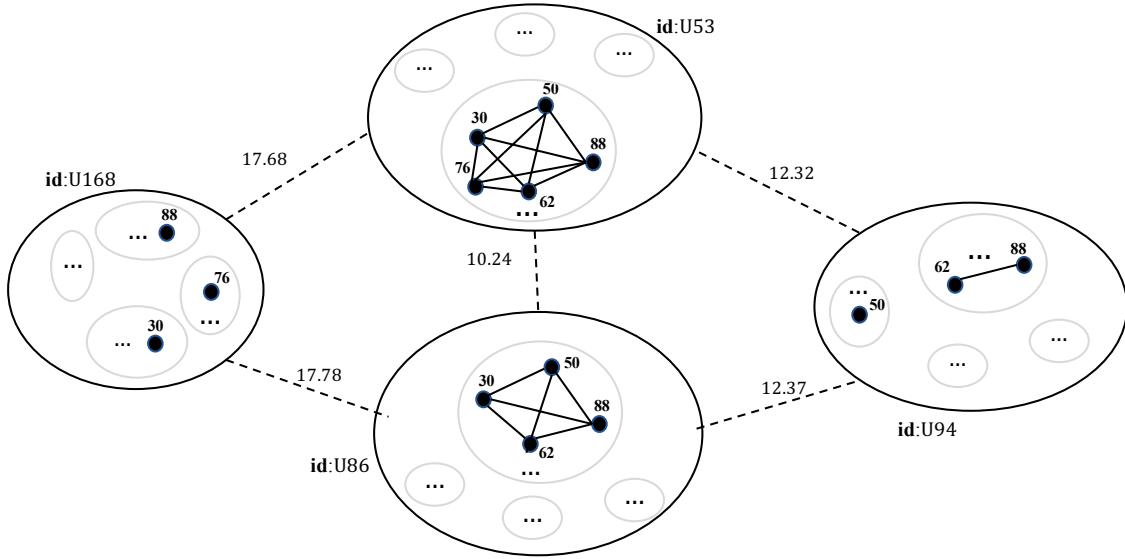


Figure 5.5: Illustrative example of multiperspective similarity from *Paris Attractions* dataset.

by-object pairs. Therefore, in addition to the perspective-specific inter-object similarities (mn^2), we store and compute the inter-perspective similarities (m^2) and the similarity between any two object pairs (n^4). d_{bi} is the average product of neighbor counts in the bipartite \mathcal{B} (Section 5.2.1). In terms of time, we further need to consider the computation of perspective-specific inter-object similarities, iterated over all perspectives, i.e., $m^2n^2d_{max}$. This is computationally intensive, which motivates the development of MP-SIMRANK.

For MP-SIMRANK, the joint computation of both inter-perspective and inter-object similarities avoids the instantiation of the bipartite graph, dropping the n^4 term from the complexities. This

Methods	Storage	Time
Merged-SimRank	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 d_{\max})$
Average-SimRank	$\mathcal{O}(n^2)$	$\mathcal{O}(mn^2 d_{\max})$
Disjoint-SimRank	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn^2 d_{\max})$
PIPELINED-SIMRANK	$\mathcal{O}(m^2 + n^4 + mn^2)$	$\mathcal{O}((m^2 + n^4 + mn^2)d_{\text{bi}} + m^2 n^2 d_{\max})$
MP-SIMRANK	$\mathcal{O}(m^2 + mn^2)$	$\mathcal{O}(m^2 n^2 d_{\max})$

Table 5.4: Complexity analysis (per iteration) of all SimRank-based methods

dramatically improves the running time of MP-SIMRANK.

We are also interested in how many iterations are generally required for convergence in practice. The *convergence rate* of the algorithm is defined as follows:

$$D_t = \frac{1}{m} \sum_{p \in \mathcal{P}} \frac{\|\mathcal{S}_p^{(t+1)} - \mathcal{S}_p^{(t)}\|_F}{n},$$

as the algorithm converges, the value of D_t should approach 0 as t goes to infinity. Overall, both models converge after reasonably few iterations (less than 5 iterations for *Zoo* and *HouseVote*, and less than 8 iterations for *Paris Attractions*).

5.4.2 Heuristic for More Efficient MP-SimRank

Since our main focus is multiperspectivity, one possible avenue to further improve efficiency is to reduce the number of perspectives, by grouping similar perspectives into a cluster with one representative perspective. We test the feasibility of this concept here.

Algorithm 5 describes CLUSTEREDMP-SIMRANK that adopts the idea of clustering perspectives. We first run *Disjoint-SimRank* on each graph and produce $S_p^{\text{disjoint}}, \forall p \in \mathcal{P}$ (Step 1) with computational cost of $\mathcal{O}(mn^2 d_{\max})$. Next, we compute the Frobenious distance between all perspectives, cluster them using the *k-medoids* algorithm ($k \leq m$ is given), and merge graphs of perspectives in the same cluster together (Steps 2 and 3). A medoid here is defined as the perspective with the smallest average distance to all others in the same cluster. These two steps require a

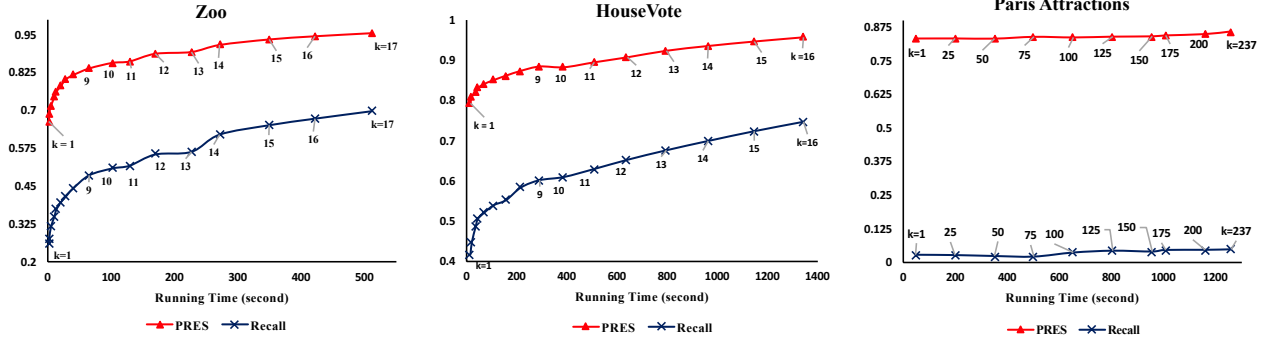


Figure 5.6: PRES, Recall, and running time of CLUSTEREDMP-SIMRANK with different number of clusters k .

computational cost of $\mathcal{O}(m^2 + km)$. We then run MP-SIMRANK on the new hypergraph \mathcal{H}_c , yielding the *cluster-specific inter-object* similarity S_c (Step 4) with the cost of $\mathcal{O}(k^2 n^2 d_{\max})$. Finally, perspectives of the same cluster share the same cluster-specific inter-object scores. The total computational cost is $\mathcal{O}(mn^2 d_{\max} + m^2 + km + k^2 n^2 d_{\max})$, less complex than the cost of MP-SIMRANK, i.e., $\mathcal{O}(m^2 n^2 d_{\max})$.

Figure 5.6 shows the performance of CLUSTEREDMP-SIMRANK and its running time as we vary the number of clusters k . The horizontal axis shows the required running time in second and the vertical axis shows the performances in terms of *Recall* (in blue) and *PRES* (in red). We observe that by choosing a small number of clusters, we can improve significantly the speed of the learning. As k increases, CLUSTEREDMP-SIMRANK approaches the performance of MP-SIMRANK (when $k = m$). With a reasonable choice of k , we can speed up the learning process while still obtaining acceptable level of performances from the learnt similarity scores.

5.5 Discussion

In certain real world applications, there is a need for expressing diverse perspectives of similarity. We propose a multiperspective graph-based framework for learning similarity from data. The proposed framework relies on a unified hypergraph representation of object-to-object relationships.

Algorithm 5 CLUSTEREDMP-SIMRANK

Require: Hypergraph \mathcal{H} and number of clusters k

- 1: /*- Step 1: run disjoint-simrank on each perspective graph - */
 - 2: $S_p^{\text{distjoint}} \leftarrow \text{Disjoint-SimRank}(G_p), \forall p \in \mathcal{P}$.
 - 3:
 - 4: /*- Step 2: compute Frobenius distances between perspectives - */
 - 5: $\mathcal{F} = [F(p, p')]_{p, p' \in \mathcal{P}}$, where
 - 6:
$$F(p, p') = \left\| S_p^{\text{distjoint}} - S_{p'}^{\text{distjoint}} \right\|_F$$
 - 7:
 - 8: /*- Step 3: cluster perspectives and merge graphs - */
 - 9: $\mathcal{C} \leftarrow \text{K-Medoids}(\mathcal{F}, k)$; $\mathcal{H}_c \leftarrow \text{merge-graph}(\mathcal{H}, \mathcal{C})$
 - 10:
 - 11: /*- Step 4: run MP-SIMRANK on the new hypergraph \mathcal{H}_c - */
 - 12: $\{S_c\}_{c \in \mathcal{C}} \leftarrow \text{MP-SIMRANK}(\mathcal{H}_c)$
 - 13:
 - 14: /*- Step 5: assign each perspective the inter-object similarity -
 - 15: - of the cluster it belongs to - */
 - 16: $S_p \leftarrow S_c, \forall p \in \mathcal{P}, c \in \mathcal{C}, \text{ and } p \in c$
 - 17:
 - 18: Return $\{S_p, \forall p \in \mathcal{P}\}$
-

The key is to learn not only the similarity between two objects for each perspective, but also the similarities across perspectives so as to allow information sharing across perspectives. We present two models, PIPELINED-SIMRANK and MP-SIMRANK, and provide their proof of convergence. Experiments on publicly available datasets show that multiperspective similarity models outperform baseline models that either ignores multiplicity of perspectives or treats each perspective separately. As future work, we will investigate strategies for improving the efficiency of the proposed framework, towards creating potential applications involving large-scale networks.

Chapter 6

Spherical Conditional Ordinal Embedding

Ordinal embedding seeks a low-dimensional representation of objects based on relative comparisons of their similarities. This low-dimensional representation expresses a specific view of similarity between objects. Classical embedding approaches assume only one valid perspective of similarity. In this chapter, we are interested in the scenarios involving ordinal comparisons that inherently reflect multiple similarity perspectives, which would be better represented by multiple embedding maps. We formulate this problem as conditional ordinal embedding, which learns a distinct low-dimensional representation for each perspective, yet allows information sharing among similar/related perspectives via a shared representation. Our geometric approach is novel in its use of a shared spherical representation and multiple perspective-specific projection maps on tangent hyperplanes. Experiments on public datasets showcase the utility of collaborative learning approach over baselines that learn multiple maps independently.

6.1 Introduction

Increasingly, there are more scenarios where we know some *relative* comparisons – which object is *more similar* to another, even as their exact similarities are not known. For instance, [3, 111]

investigated human perception of “gloss” by studying how human subjects compared images. It is now commonplace to employ human intelligence tasks to generate categorization labels for images [33, 110]. [116] modeled how different users organized attractions.

Such observations can be represented as object triplets. Observing a triplet $\langle i, j, k \rangle$ indicates the reference (center) object j 's greater similarity to the first-mentioned i than to k . The problem of interest is to arrive at object coordinates in a low-dimensional space – effectively a map as the output, such that their relative distances would preserve the observed quadruples. This problem is known as *ordinal embedding*. The output representation is useful for various applications such as estimation of relative similarities for unseen quadruples or “features” for other machine learning tasks. Another important application that we focus on here is visualization on a scatterplot. Without loss of generality, in the subsequent discussion, we will assume $2d$ for ease of illustration.

Previous works [101, 105] mostly output one visualization map, reflecting a singular similarity perception. However there could be more than one similarity perception. For instance, when the quadruples have been generated by different human subjects, there may be natural “disagreements” on some quadruples. Classically, such disagreements are assumed to be noisy conflicts to be removed in order to uncover the *one* map.

Multiple Maps. We postulate that these quadruples may be expressing multiple perspectives of similarities. The disagreements among quadruples reflect idiosyncratic perspectives of similarity. The varying perspectives are valid, and should be preserved by the embedding. A single visualization map is insufficient to accommodate the different points of view simultaneously. It would be more appropriate to learn *multiple* maps, each of which reflects a particular perspective of similarity.

Hence, we are dealing not with ordinal triplets per se, rather with ordinal quadruples in the form of $\langle p|i, j, k \rangle$ expressing relative comparison between object i, j and k , according to a perspective p . We refer to the problem of learning multiple maps from such conditional ordinal comparisons as *conditional ordinal embedding*. As input, we are given ordinal quadruples where the asso-

ciations among object triplets to perspectives are known. As output, we seek to learn multiple low-dimensional Euclidean maps, one for each perspective.

6.2 Framework

6.2.1 Problem Formulation

Input. The set of objects of interest is denoted \mathcal{O} , e.g., images, documents, items, and the set of perspectives \mathcal{P} . For generality, we assume *no feature* for an object beyond its identity. A perspective could be a human subject, an attribute, etc., whose perception of similarity is to be modeled. Each perspective $p \in \mathcal{P}$ observes conditional ordinal quadruples in the form of $\langle p|i, j, k \rangle$, where $(i \neq j \neq k) \in \mathcal{O}$. Such a quadruple indicates that *according to perspective p , j is more similar to i than to k* . The set of observed quadruples for an perspective $p \in \mathcal{P}$ is: $\mathcal{N}_p = \{\langle p|i, j, k \rangle | i \neq j \neq k \in \mathcal{O}\}$. The input is thus \mathcal{N} , the union of quadruples of all perspectives.

Output. For each perspective $p \in \mathcal{P}$, we derive an embedding map of all objects. For the map associated with perspective p , every object $i \in \mathcal{O}$ is associated with a coordinate $y_i^p \in \mathbb{R}^d$, where d is the desired dimensionality of the target representation. For visualization purpose, we assume $d = 2$ in this study. The objective is to satisfy the following condition for as many quadruples in \mathcal{N}_p specifically, and \mathcal{N} generally, as possible:

$$\langle p|i, j, k \rangle \in \mathcal{N} \iff \|y_j^p - y_i^p\| < \|y_j^p - y_k^p\| \quad (6.1)$$

For $|\mathcal{P}| \geq 2$, we refer to this problem as *conditional ordinal embedding*. For $|\mathcal{P}| = 1$, this problem degenerates to the classical “single-perspective” ordinal embedding problem.

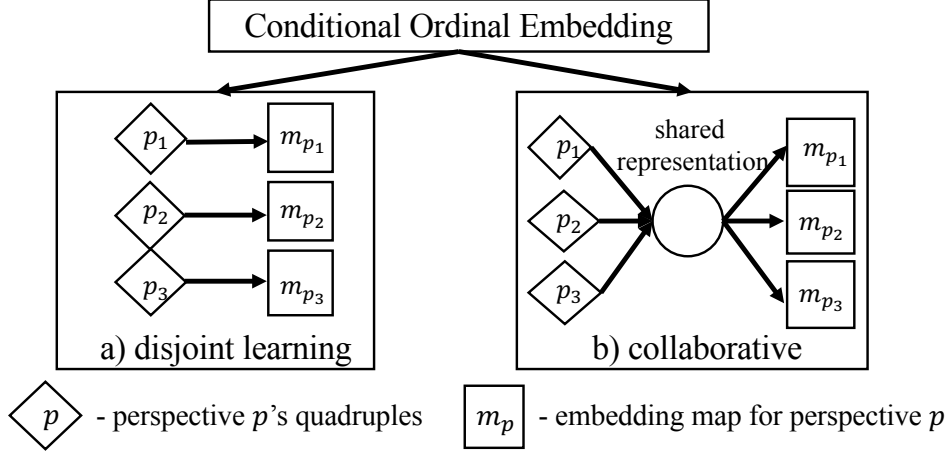


Figure 6.1: Approaches for Conditional Ordinal Embedding.

6.2.2 Proposed Methodology

Figure 6.1 outlines two approaches for conditional ordinal embedding. The straightforward approach is disjoint learning, i.e., deriving a map for each perspective independently. Specifically, the map m_{p_1} is learnt from only perspective p_1 's quadruples, and the various maps m_{p_1} to m_{p_3} are not related (Figure 6.1 left).

We believe that the perspectives are potentially related as they concern the same set of objects. Their latent relationships could render significant advantage when perspectives are sufficiently related, and yet each perspective is under-sampled. In practice, we do not necessarily observe all possible quadruples, but a subset. For sparse data, an perspective may have insufficient information. Furthermore, the quadruples of any one perspective may not cover all objects [3].

We propose a *collaborative* approach as shown on (Figure 6.1 right). The challenge is to design a shared representation that allows “sharing” across perspectives, and yet to still allow each perspective to remain distinct. Here, we adopt a well-known instance of Riemannian manifold [31], namely: *hypersphere*.

Each perspective $p \in \mathcal{P}$ and object $i \in \mathcal{O}$ are respectively associated with a spherical coordinate $x_p, y_i \in \mathbb{S}^d$, where $\mathbb{S}^d = \{v \in \mathbb{R}^{d+1} : \|v\| = 1\}$. The output coordinate $y_i^p \in \mathbb{R}^d$ is the

projection of y_i onto task-specific d -dimensional hyperplane defined by x_p . The projection of the objects' spherical coordinates onto the tangent hyperplane of an perspective constructs a map that reveals that perspective's distinct view of proximities between objects. The intuition for a sphere as the shared representation is that it allows greater flexibility for each perspective to model its own similarity perspective, while each perspective's embedding map is embedded within the same space as the shared sphere representation. Also, one advantage of the sphere is that it is not a higher-dimensional structure, since its effective dimensionality is still d .

6.2.3 Model

To arrive at shared $Y = \{y_i : i \in \mathcal{O}\}$, while accommodating variances among perspectives, we turn to probabilistic modeling.

Generative Process. Let us first consider an individual conditional ordinal quadruple $\langle p|i, j, k \rangle \in \mathcal{N}$. We associate an perspective p and three objects i, j , and k with a binary-valued random variable c_{ijk}^p . When $c_{ijk}^p = 1$, we generate the quadruple $\langle p|i, j, k \rangle \in \mathcal{N}$, i.e., p considers j to be more similar to i than to k . If $c_{ijk}^p = 0$, opposing quadruple $\langle p|k, j, i \rangle \in \mathcal{N}$ is generated.

There are two views of relative proximity, which determines the outcome of c_{ijk}^p . First, there is the *perspective-specific* view of an perspective p , based on the projected coordinates on p 's tangent hyperplane, for which the probability is $P_p(c_{ijk}^p = 1|y_i^p, y_j^p, y_k^p)$. Second, there is the *global* view, based on coordinates on the shared sphere, for which the probability is $P_s(c_{ijk}^p = 1|y_i, y_j, y_k)$. We assume that some quadruples are perspective-specific, generated according to P_p , while other quadruples are generic, generated according to P_s . The balance between the two is modeled by parameter $\delta_p \in [0, 1]$.

Now we describe the generative process for quadruples in \mathcal{N} :

1. For each perspective $p \in \mathcal{P}$:
 - Draw p 's coordinate: $x_p \sim \text{VMF}(\mu_{\mathcal{P}}, \kappa_{\mathcal{P}})$

- Draw p 's parameter δ_p : $\delta_p \sim \mathcal{U}(0, 1)$
2. For each object $i \in \mathcal{O}$:
- Draw i 's coordinate: $y_i \sim \text{VMF}(\mu_{\mathcal{O}}, \kappa_{\mathcal{O}})$
3. For objects $i, j, k \in \mathcal{O}$, $i < k$, $j \neq i, k$:
- If a draw from Bernoulli (δ_p) turns up 1, then: $c_{ijk}^p \sim \text{Bernoulli}(\mathbb{P}_p(c_{ijk}^p = 1 | y_i^p, y_j^p, y_k^p))$
Else: then $c_{ijk}^p \sim \text{Bernoulli}(\mathbb{P}_s(c_{ijk}^p = 1 | y_i, y_j, y_k))$
 - If $c_{ijk}^p = 1$, generate a quadruple instance $\langle p | i, j, k \rangle$,
Else: generate a quadruple instance $\langle p | k, j, i \rangle$.

In the above generative process, x_p and y_i have von Mises-Fisher (vMF) [71] priors, parameterized by mean unit vector μ and concentration κ . Higher κ translates to greater concentration around μ . $\kappa = 0$ models the uniform prior. In this work, we assume that δ_p has a uniform prior.

Perspective-Specific Probability Function. Given the shared spherical representation, and the intention to maintain each perspective's embedding on a Euclidean space, a natural choice is to have the perspective-specific representation lie on the tangent hyperplane of sphere \mathbb{S}^d at x_p , defined as: $T_{x_p}\mathbb{S}^d = \{v \in \mathbb{R}^{d+1} : (x_p)^T v = 0\}$. We define the perspective-specific representation for p to be the projection of objects' coordinates $\{y_i : i \in \mathcal{O}\}$ onto the tangent hyperplane at x_p as following:

$$y_i^p = \text{Proj}_{x_p}(y_i) = \left[I - x_p(x_p)^T \right] y_i. \quad (6.2)$$

where I is $(d+1)$ -dimensional identity matrix. Though $\text{Proj}_{x_p}(y_i)$ is a $(d+1)$ -dimensional vector, it still effectively lies on a d -dimensional tangent hyperplane in the $(d+1)$ -dimensional space. In Section 6.4, we describe in details the d -dimensional coordinate transformation.

Figure 6.2 illustrates an example of the representations y_i, y_j, y_k of three objects i, j, k (red points) and $x_p, x_{p'}$ of two perspectives p, p' (blue points) on the unit sphere. The left tangent hyperplane $T_{x_p}\mathbb{S}^d$ corresponds to the representation map of perspective p . On this map, y_j^p is closer to y_k^p

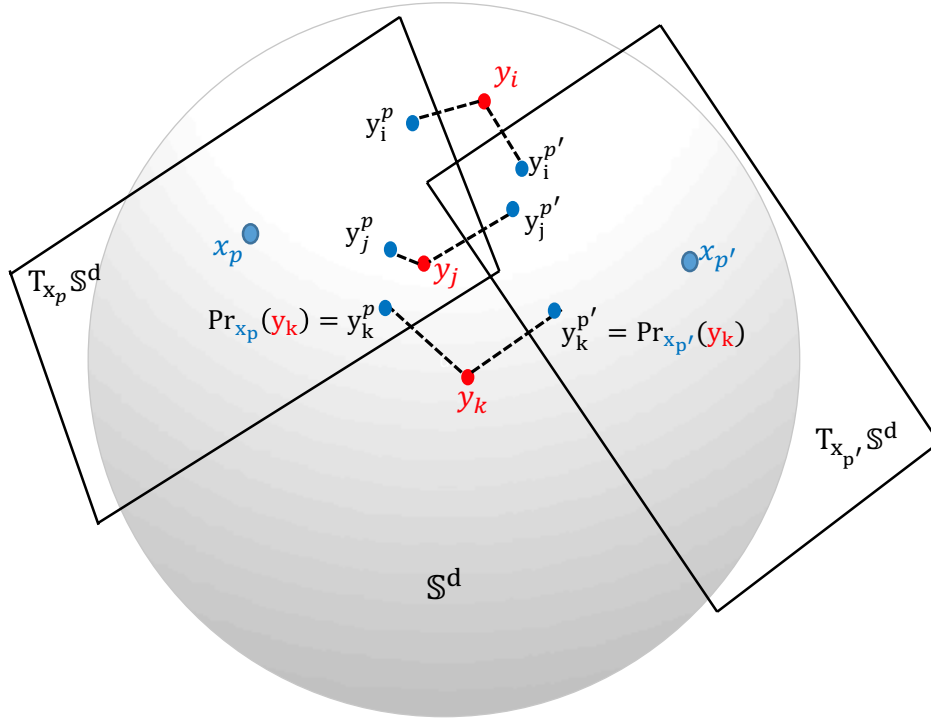


Figure 6.2: Representations of three objects i, j, k , two perspectives p, p' .

than to y_i^p through the projection Proj_{x_p} . The right tangent hyperplane $T_{x_{p'}} S^d$ is the representation map of perspective p' . There, $y_j^{p'}$ is closer to $y_i^{p'}$ than to $y_k^{p'}$. These are “conflicting” ordinal relationships between p and p' , yet they arise from the same spherical coordinates of objects, indicating the role of perspectives’ tangent hyperplanes in accommodating different similarity perceptions. There are also quadruples which both p and p' agree on.

We now express P_p in terms of such projected distances. Let us denote the distance d_{ij}^p between two objects i, j on perspective p ’s map, i.e., $d_{ij}^p = \|\text{Proj}_{x_p}(y_j - y_i)\|$. We express $P_p(c_{ijk}^p = 1 | y_i^p, y_j^p, y_k^p)$ in terms of difference between d_{jk}^p and d_{ij}^p (Equation 6.3). The smaller is d_{ij}^p relative to d_{jk}^p , the higher is this probability. α is the scaling factor.

$$\sigma_{ijk}^p = P_p(c_{ijk}^p = 1 | y_i^p, y_j^p, y_k^p) = \frac{1}{1 + e^{-\alpha \cdot (d_{jk}^p - d_{ij}^p)}} \quad (6.3)$$

Global Probability Function. We now describe the “global” probability P_s - the likelihood of

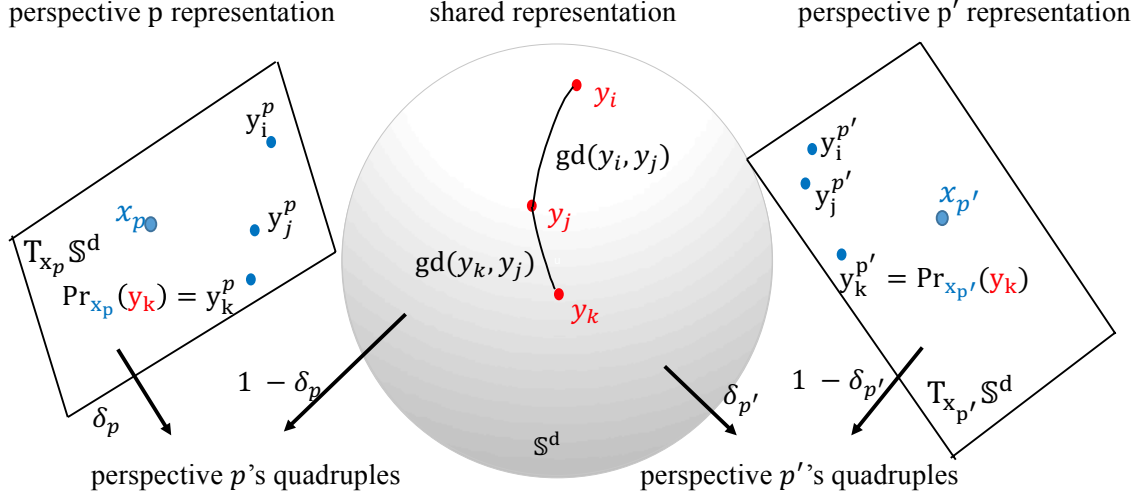


Figure 6.3: The probability of observing $\langle p|i, j, k \rangle$ is a combination of the perspective-specific probability σ_{ijk}^p and the global probability σ_{ijk} .

observing the quadruple $\langle p|i, j, k \rangle$ based on the objects' spherical coordinates. On the unit sphere, the distance between y_i and y_j is the geodesic distance [28]: $\text{gd}(y_i, y_j) = \cos^{-1}(y_i^T y_j)$.

Given $y_i, y_j, y_k \in \mathbb{S}^d$, the following relation holds:

$$\text{gd}(y_i, y_j) < \text{gd}(y_k, y_j) \Leftrightarrow y_i^T y_j > y_k^T y_j \quad (6.4)$$

On one hand, Equation 6.4 implies that the inner product yields the same ordering as the geodesic distance. On the other hand, inner product computation is more computationally efficient compared to the geodesic distance. Therefore, the global probability is defined as follows:

$$\sigma_{ijk} = P_s(c_{ijk}^p = 1 | y_i, y_j, y_k) = \frac{1}{1 + e^{-\alpha \cdot (y_i^T y_j - y_k^T y_j)}}, \quad (6.5)$$

Objective Function. The likelihood of observing the quadruple $\langle p|i, j, k \rangle$ is the normalized weighted sum of P_p and P_s (Figure 6.3). The formula is described in Equation 6.6 below.

$$l_{ijk}^p = \delta_p \cdot \sigma_{ijk}^p + (1 - \delta_p) \cdot \sigma_{ijk} \quad (6.6)$$

The model's parameters are learnt to maximize the joint probability $P(\mathcal{N}, X, Y | \kappa_{\mathcal{P}}, \mu_{\mathcal{P}}, \kappa_{\mathcal{O}}, \mu_{\mathcal{O}})$ of the model across the observed quadruples (Equation 6.7), which can be factorized as product of:

$P(\mathcal{N}_p | X, Y) = \prod_{\langle p|i,j,k \rangle \in \mathcal{N}_p} l_{ijk}^p$ - the likelihood, $P(X | \kappa_{\mathcal{P}}, \mu_{\mathcal{P}})$ and $P(Y | \kappa_{\mathcal{O}}, \mu_{\mathcal{O}})$ - the priors.

$$\begin{aligned} & \arg \max_{X, Y} P(\mathcal{N}, X, Y | \kappa_{\mathcal{P}}, \mu_{\mathcal{P}}, \kappa_{\mathcal{O}}, \mu_{\mathcal{O}}) \\ &= \arg \max_{X, Y} \prod_{p \in \mathcal{P}} P(\mathcal{N}_p | X, Y) \times P(X | \kappa_{\mathcal{P}}, \mu_{\mathcal{P}}) \times P(Y | \kappa_{\mathcal{O}}, \mu_{\mathcal{O}}) \end{aligned} \quad (6.7)$$

Maximizing the joint probability in Equation 6.7 is equivalent to maximizing its logarithm \mathcal{L} (to simplify the parameters, we tie $\kappa_{\mathcal{P}} = \kappa_{\mathcal{O}} = \kappa$ and $\mu_{\mathcal{P}} = \mu_{\mathcal{O}} = \mu$):

$$\begin{aligned} \mathcal{L} &= \sum_{p \in \mathcal{P}} \ln P(\mathcal{N}_p | X, Y) + \ln P(X | \kappa, \mu) + \ln P(Y | \kappa, \mu) \\ &\propto \sum_{p \in \mathcal{P}} \sum_{\langle p|i,j,k \rangle \in \mathcal{N}_p} \ln(l_{ijk}^p) + \sum_{p \in \mathcal{P}} \kappa \cdot \mu^T x_p + \sum_{i \in \mathcal{O}} \kappa \cdot \mu^T y_i. \end{aligned} \quad (6.8)$$

6.2.4 Parameter Learning

Line Search on Manifold. The learning requires solving an optimization problem on the spherical manifold. [1] presents the line-search method on a manifold \mathcal{M} . The update formula is: $x_{k+1} = R_{x_k}(t_k \eta_k)$, - R_{x_k} , p_k , and $\eta_k \in T_{x_k} \mathcal{M}$ are the *retraction map* at x_k , the step size, and the search direction respectively. Retraction map ensures the update process to be performed on the manifold. Here, we consider the following map ([17]):

$$\mathcal{R}_x(\eta) = \arg \min_{y \in \mathbb{S}^d} \|x + \eta - y\| = \frac{x + \eta}{\|x + \eta\|} \quad (6.9)$$

For parameter learning, we adopt the stochastic gradient descent strategy for functions defined on a Riemannian manifold [17], which requires the computation of the Riemannian gradient. According to [28], the gradient on the sphere of a differentiable function $f : \Omega \rightarrow \mathbb{R}$ (let $\Omega \in \mathbb{S}^d$ be

an open set), at $x \in \Omega$ is defined by (where $\nabla f(x)$ is the usual gradient of $f(x)$ at $x \in \Omega$):

$$\text{grad}f(x) = [I - xx^T]\nabla f(x), \quad (6.10)$$

Learning Algorithm. Algorithm 6 shows that in each iteration, a quadruple $\langle p|i, j, k \rangle$ is randomly selected, and the parameters are updated using the line-search optimization technique on the unit sphere. Specifically, we first compute the partial derivatives with respect to x_p, y_i, y_j, y_k . The gradients on the spherical surface are computed through the project map $\text{Proj}(\cdot)$. Then we update the model parameters using the retraction map as described earlier. Learning rate ϵ is decayed over time. The last update guarantees that $\delta_t \in [0, 1]$. The complexity is linear to the size of \mathcal{N} -the set of all quadruples, which is bounded by $\mathcal{O}(|\mathcal{P}| \times |\mathcal{O}|^3)$.

Algorithm 6 SCORE

- 1: Initialize x_p for $p \in \mathcal{P}$ and y_i for $i \in \mathcal{O}$.
 - 2: While not converged
 - 3: Draw a quadruple $\langle p|i, j, k \rangle$ randomly from \mathcal{N} .
 - 4: Compute the likelihood:
 - 5: $l_{ijk}^p = \delta_p \cdot \sigma_{ijk}^p + (1 - \delta_p) \sigma_{ijk}$.
 - 6: Compute the partial derivatives:
 - 7: $\Delta_z \leftarrow \frac{\partial \mathcal{L}}{\partial z}$ for each $z \in \{x_p, y_i, y_j, y_k\}$
 - 8: Update the model parameters:
 - 9: $z \leftarrow \mathcal{R}_z(\epsilon \cdot \text{Proj}_z(\Delta_z))$, for $z \in \{x_p, y_i, y_j, y_k\}$;
 - 10: $\delta_p \leftarrow \delta_p + \epsilon \cdot (\sigma_{ijk}^p - \sigma_{ijk})$; $\delta_p = \arg \min_{\delta \in [0,1]} |\delta_p - \delta|$;
 - 11: Return $\{x_p\}_{p \in \mathcal{P}}$ and $\{y_i\}_{i \in \mathcal{O}}$.
-

6.3 Experiments

Our objective is primarily to investigate the effectiveness of multiple maps for conditional ordinal embedding, which is expressed in terms of how well the resulting output embedding coordinates could preserve the ordering reflected by the quadruples.

6.3.1 Experimental Setup

Datasets. We experiment with three datasets mentioned in Chapter 5, which could model varying similarity perspectives.

- *Zoo*¹ contains 17 attributes of 101 animals (excluding animal name). We model each attribute as a similarity perspective. For perspective p , we form the quadruple $\langle p|i, j, k \rangle$ if i and j have the same attribute value, which is different from k . There are 3.24×10^6 quadruples.
- *Congressional Voting Records (or HouseVote)*² contains 435 instances (congressmen) and 16 attributes (voting issues). After excluding all instances with missing values, we get a fully-observed dataset with 232 instances and 16 attributes. We generate quadruples in the same way as we do with *Zoo* dataset. That induces totally 2.4×10^7 quadruples.
- *Paris Attractions*³ contains 237 users organizing 250 Paris attractions into clusters. Considering each user as an perspective, we induce 3.48×10^5 quadruples, each involves two attractions i and j that the user puts into the same cluster, and another attraction k in a different cluster. As in [116], we exclude attractions uninteresting to users.

Comparative Methods. We compare SCORE to several baselines. The disjoint learning approach learns a distinct map from the quadruples of a perspective. We use two recent ordinal embedding methods: **SOE**⁴ [101] and **tSTE**⁵ [105].

Multiview Triplet Embedding (**MVTE**) [6] divides the collection of quadruples into clusters. The number of views is set to the number of perspectives, i.e., $|\mathcal{P}|$. Since the associations between perspectives and object triplets are unknown, we match each view with a ground-truth perspective using Hungarian maximum bipartite matching algorithm, so as to maximize the accuracy.

Multiview Multidimensional Scaling (**MVMDS**) [11] performs MDS on multi-view data, learns

¹<https://archive.ics.uci.edu/ml/datasets/Zoo>

²<https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records>

³http://projects.yisongyue.com/collab_cluster/

⁴<http://rpackages.ianhowson.com/cran/loe/>

⁵<http://homepage.tudelft.nl/19j49/ste/>

the weights of these views, and produces one consensus map. This is akin to learning a single map by consolidating multiple views. Since MVMDS expects distance matrices, we feed it feature vectors learnt by SOE from the ordinal quadruples.

For visualization purpose, we set the dimensionality of the embedding space $d = 2$. We tune the parameters of all methods for their best performances on the training data. For SCORE, the setting is $\kappa = 10^{-3}$ for *Paris Attractions*, and 0 for *Zoo* and *HouseVote*, vMF mean vector $\mu = (0, 0, 1)$, the learning rate $\epsilon = 0.05$, and the scaling factor $\alpha = 30$. For SOE, the scaling factor is 0.1 for all the datasets. For tSTE, the learning rate and regularization parameter are 2 and 0 respectively for all datasets. For MVTE, the learning rate is 1 for all datasets. For MVMDS, $\gamma = 5$ for all datasets.

Evaluation Measure. The *preservation accuracy* for an perspective p is the fraction of its ordinal quadruples \mathcal{N}_p for which p 's coordinates reflect the correct direction. The fewer the violated quadruples, the higher is the accuracy. The overall *accuracy* is the average of perspectives' preservation accuracies:

$$\frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \frac{|\langle p | i, j, k \rangle \in \mathcal{N}_p : \|y_j^p - y_i^p\| < \|y_j^p - y_k^p\| \}|}{|\mathcal{N}_p|}, \quad (6.11)$$

- y_i^p, y_j^p, y_k^p are p 's embedding coordinates of objects i, j, k .

Since in practice we may not observe all quadruples or even all objects beforehand, we sample a fraction r (*split ratio*) of objects for each perspective, then evaluate the coordinates against the full set of quadruples. For this study, we set $r = 0.5$, which has a relative balance between the information that an perspective sees and the information that it could learn from others. We average the results across 30 random samples.

The running times are reasonable. For the *Paris Attractions*, including all perspectives, SCORE takes 5 minutes on a PC with Intel Core i5 3.2 GHz CPU and 12 GB RAM. The learning times for the *Zoo* and *HouseVote* are less than 10 minutes.

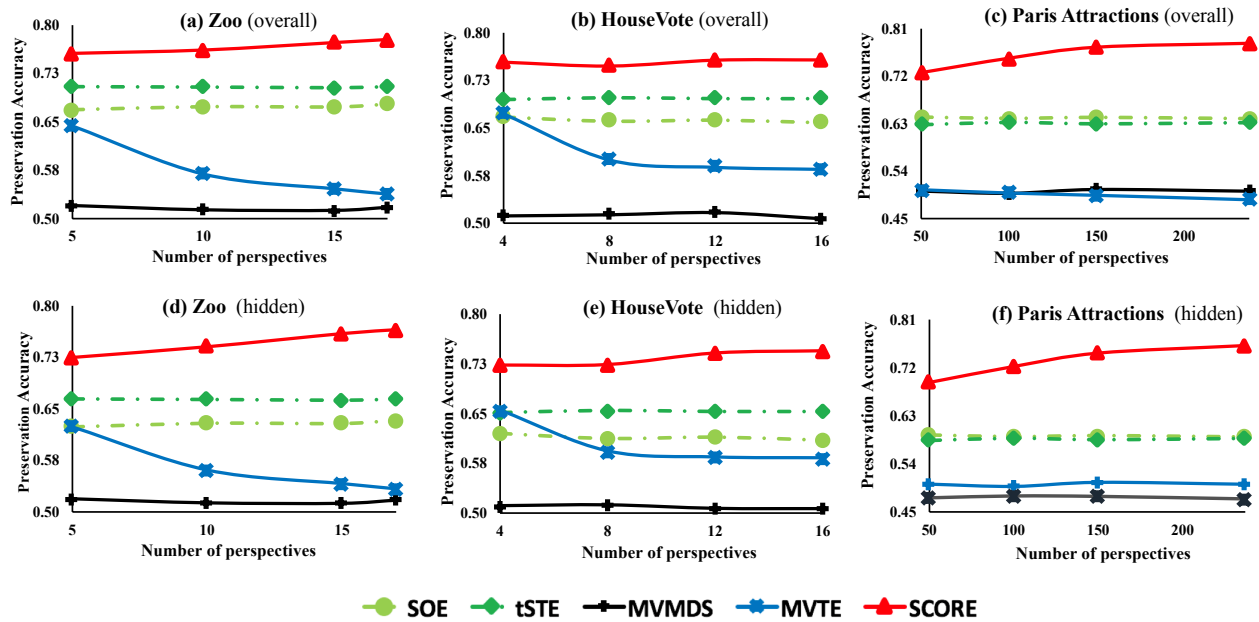


Figure 6.4: Overall and hidden preservation accuracy

6.3.2 Comparison to Baselines

We vary the number of perspectives by randomly sampling perspectives. Figure 6.4 shows the preservation accuracies of all models.

Overall Preservation Accuracy. SCORE shows significantly higher performance than *SOE*, *tSTE* (Figure 6.4(a,b,c)). In the latter, an perspective cannot collaborate with other perspectives, leading to poor performance on unseen quadruples. This highlights the benefit of collaborative learning, as it helps perspectives fill in each other’s missing information. *MVTE* and *MVMSD* show even weaker performances. For *MVTE*, the likely reason is the lack of information about the associations between perspectives and quadruples. If this information is provided, *MVTE* reduces to *tSTE*, which is showing relatively higher performance than *MVTE*. For *MVMSD*, the likely reason is the consolidation into a single map, which, though learnt from multiple distance matrices, cannot fit conflicting quadruples.

With more perspectives, SCORE has even more opportunities to find other related perspectives to collaborate with, resulting in increasing accuracy. *SOE* and *tSTE* do not benefit from more per-

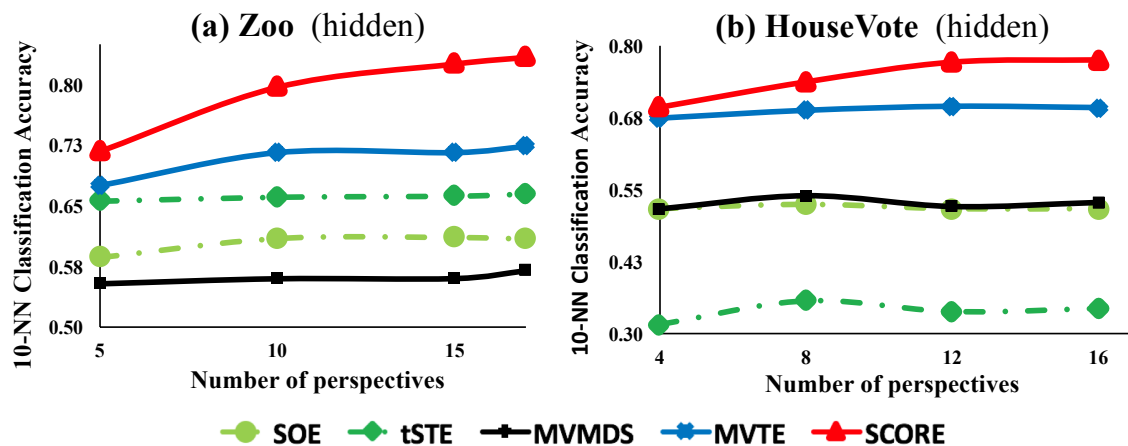


Figure 6.5: 10-NN classification accuracy at $r = 0.5$.

spectives as each perspective has its own map. *MVTE*'s accuracies decrease with more perspectives as it gets more difficult for it to discover clusters that are well-aligned to perspectives. *MVMDS* with one map does not benefit much from more perspectives.

Performance on Predicting Hidden Triplets. The earlier accuracies are evaluated for the full set of quadruples, which is the combination of the *observed* subset (from the r fraction) and the *hidden* subset (the unseen quadruples). To see how well the models generalize to the unseen data, we now investigate the preservation accuracy measured on the *hidden* alone (Figure 6.4(d,e,f)). We observe the same picture as before but with generally lower accuracies than that for full sets (Figure 6.4(a,b,c)), which are expected as these are *unseen* quadruples. The reduction is more dramatic for *SOE* and *tSTE*, which tend to overfit the *observed*, and generalize poorly to the *hidden* quadruples. *SCORE* does commendably well on the *hidden* set, showing greater robustness in generalizing to the unseen quadruples.

For an alternative measure of generalization, we test the learnt coordinates as features to classify the *hidden* objects by attribute values associated with the perspective. An object is assigned the majority label among its 10-nearest neighbors. Figure 6.5 shows the *10-NN classification accuracy*, averaged across perspectives. Only *Zoo* and *HouseVote* have “labels” and are involved in this experiment. *SCORE* has better results than the baselines (statistically significant at 0.05)

in predicting the labels of unseen instances. Interestingly, *MVTE* performs better than disjoint learning baselines in this task. Since quadruples are learned jointly, some quadruples may have been assigned to clusters correlated with the class labels, though the clusters may not reflect the perspective-specific view perfectly.

Exploration on the Split Ratio. To better understand the benefits of multi-perspective model-

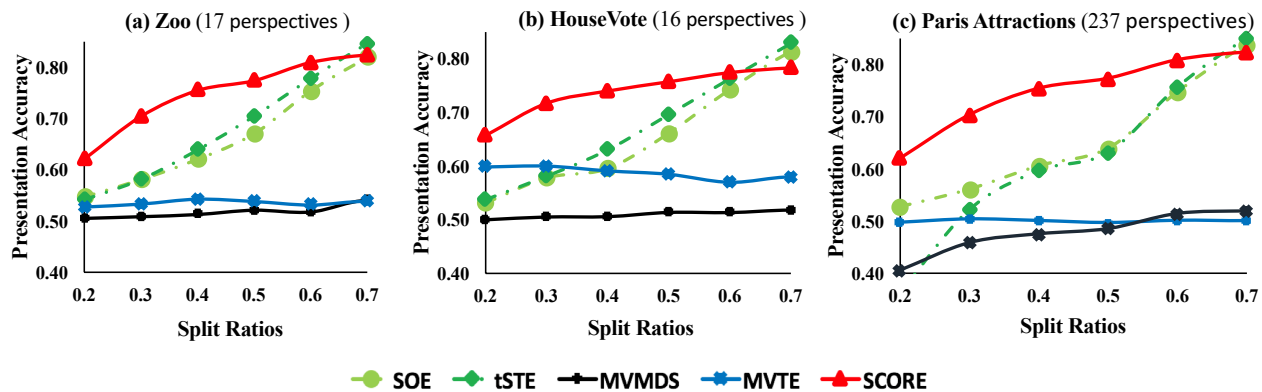


Figure 6.6: Overall preservation accuracies at various split ratios.

ing, we shows the accuracies with varying r for the complete set of perspectives in Figure 6.6.

The *disjoint learning* baselines perform poorly for low value of r . This is expected since the amount of observed data is insufficient for a single task to learn its own map effectively. For extremely high r , e.g., 0.7, the *disjoint learning* baselines tend to do well. For *Zoo*, *HouseVote*, and *Paris Attractions*, $r = 0.7$ respectively corresponds to approximately 1.1M, 16.1M, and 155dquadruples in training, which are 34.23%, 34.17%, and 44.42% of all possible quadruples. With sufficiently large data that cover majority of objects, each perspective has more flexibility to specialize, with little risk in missing out information. Also in Figure 6.6, SCORE shows significantly better performances than MVTE and MVMDs for the same reasons as in previous comparison.

Importantly, SCORE is robust across values of r . It is the best around 0.2-0.6, and never the worst. This result has two implications. *First*, it reiterates the benefit of collaborative approach when the data is under-sampled, yet sufficient to learn the relatedness and specialization of tasks.

Second, in practice it is often unclear whether the data is sufficient. Upon such ambiguity, multi-perspectives ameliorates the risk of performing badly, while providing reasonable performance.

6.3.3 Perspective Relatedness

Two similar perspectives would be expected to be closer on the hypersphere than two dissimilar perspectives. For *Zoo* and *HouseVote*, each perspective corresponds to an attribute, whose values effectively define a clustering of objects. We define the attribute-based similarity between two perspectives as the Normalized Mutual Information or NMI [27] between the two clusterings. We also define the proximity between two perspectives on the shared hypersphere as their *angular similarity*. For each perspective $p \in \mathcal{P}$, we measure the Pearson correlation of the NMI scores and angular similarities between p and other perspectives in \mathcal{P} . We observe positive correlations among the NMI scores and the angular similarities (Figure 6.7). The minimum values for both datasets are non-negative and the median values are quite positive 0.34 and 0.36 for *Zoo* and *HouseVote* respectively, indicating that SCORE captures perspective relatedness during learning, with similar perspectives more likely to be closer on the hypersphere.

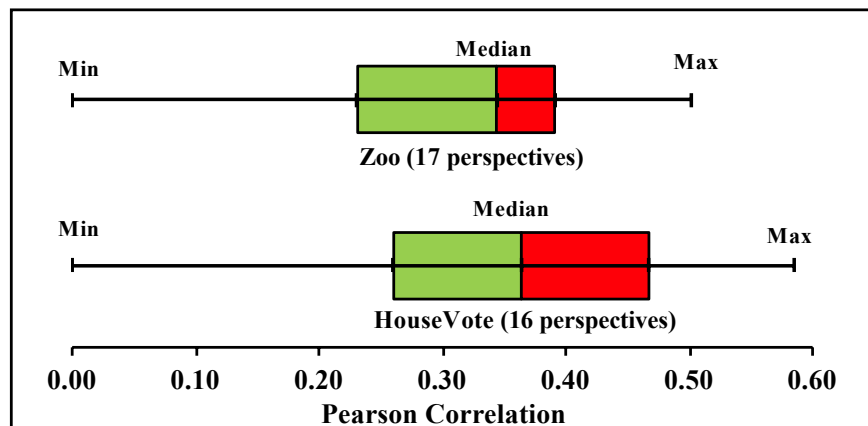


Figure 6.7: Pearson Correlation of Angular similarities vs. NMIs.

6.3.4 Multiple Maps vs. Single Map

To better illustrate the need for multiple maps, we consider a scenario involving three attributes of a dataset (*type*, *#legs*, *predator* from *Zoo*, and *immigration*, *education-spending*, *crime* from *HouseVote*). We compare SCORE in two modes: *multiple maps* when we learn three distinct maps collaboratively and *single map* when we pool quadruples from the three attributes to learn one map. Table 6.1 compares the preservation accuracies of the two modes, showing that multiple maps have significantly higher accuracies, indicating its greater capacity for reflecting multiple perspectives than a single map.

SCORE	Zoo	HouseVote
Single Map	0.82	0.70
Multiple Maps	0.98	0.89

Table 6.1: Performance of SCORE: multi-maps vs. single-map.

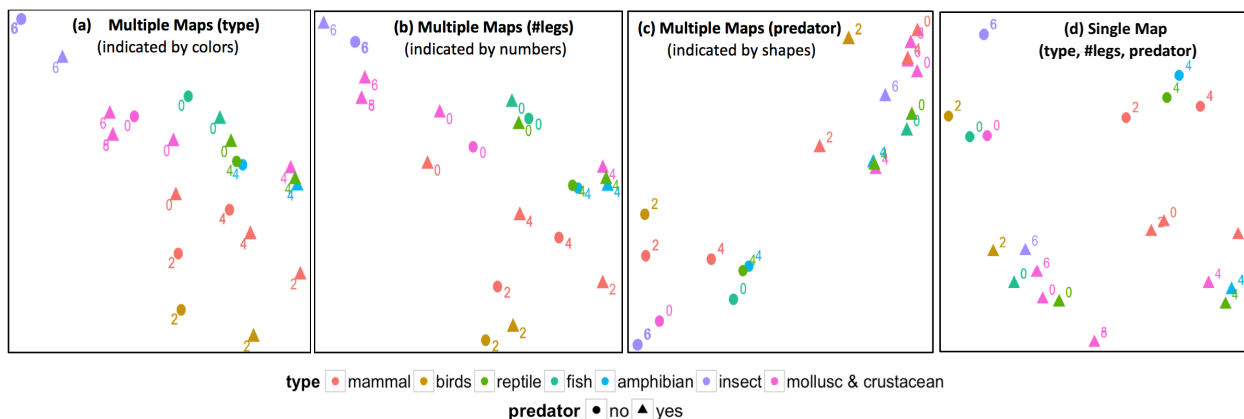


Figure 6.8: Visualization maps for *type*, *#legs*, *predator* (*Zoo*).

As a visual illustration, Figure 6.8(a, b, c) shows the three maps (corresponding to *multiple maps*) concerning animals from *Zoo*. Each animal is shown as a point, whose color, number, and shape indicate *type*, *#legs*, and *predator* attributes respectively. Figure 6.8(a) visualizes animals based on *type* (color). Animals of the same type (color) flock together, e.g., insects (purple) on the top left, birds (yellow) on the bottom right. Figure 6.8(b) visualizes animals in terms of *#legs*.

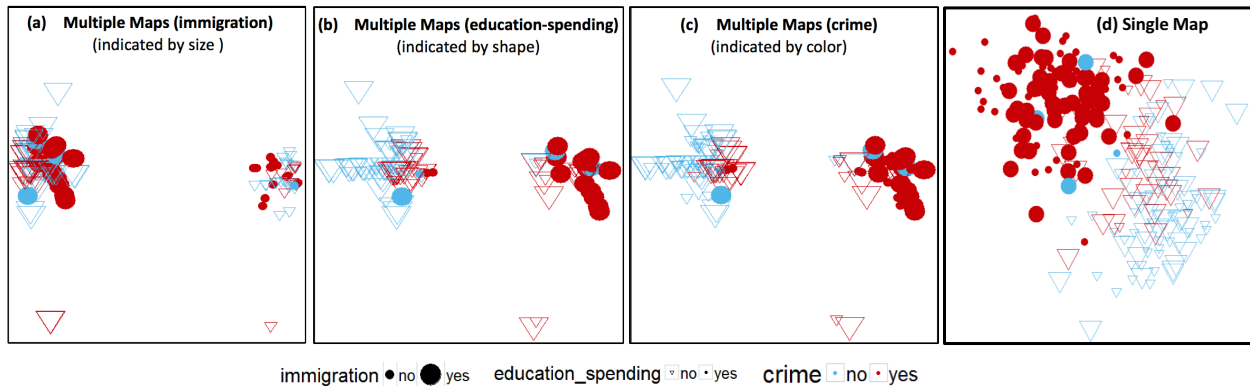


Figure 6.9: Visualizations for three attributes: *immigration*, *education-spending*, *crime* (*HouseVote*).

Animals of the same number of legs tend to be found together, e.g., 6 legs on the top left, 2 legs on the bottom right. Figure 6.8(c) visualizes animals based on whether they are *predator* (shape). The binary separation of predators (triangles) on the top right and non-predators (circles) on the lower left is evident. A single map cannot capture diverse perceptions of similarity. Figure 6.8(d) depicts the *single map* mode for the same three attributes. It could only represent separation by *predator* (shape), but is unable to represent *type* or *#legs* well.

Figure 6.9 provides another example from the *HouseVote* dataset, visualizing three binary attributes: *immigration*, *education-spending*, and *crime*. For *HouseVote*, each object is a congressman. Some objects overlap as they may have similar attribute values. We use size, shape, and color to represent the attribute values of *immigration*, *education-spending*, and *crime* respectively. According to the sizes, the labels, and the colors in Figure 6.9, instances with same attribute values are grouped intuitively. For example, Figure 6.9(a) visualizes *immigration* attribute (represented by size: small indicates a 'no' vote, while large indicates a 'yes' vote). We see the clear separation between points of similar size: small shapes on the right, large shapes on the left. Similarly, Figure 6.9(b) is a visualization for *education_spending* (represented by shape: inverted triangles for 'no' and circles for 'yes'). Triangular objects tend to flock to the left, while circular objects flock to the right. Finally, in Figure 6.9(c) - a map of the *crime* vote (represented by color: blue for

‘no’ and red for ‘yes’). It is evident blues tend to be on the left, while red tend to be on the right. These are multiple maps defined over the same set of objects, yet reflecting different perceptions of similarity.

Upon a closer inspection of the data, we uncover further insights. For instance, the *HouseVote* data contains the political affiliation of congressmen (Republican or Democrat), which was not used for learning the embedding maps. Overall, there is a tendency for Democrats to vote ‘no’ on *education_spending* and *crime*, while Republicans tend to vote ‘yes’ on both counts. On Figure 6.9(b), most of blue triangles on the left are in fact Democrats, while most of the red circles on the right are Republicans. Interestingly, there are red triangles on the left, who are likely to be Republicans voting with Democrats, whereas the red triangles on the right are likely to be Democrats voting with Republicans. Thus the map helps highlight the varying similarities between congressmen depending on the voting issues. Figure 6.9(d) is produced by SCORE, running on single-map mode. Again, a single visualization map is not efficient to capture diverse similarity perceptions. This highlights the need for multiple maps, each for a similarity perspective.

6.3.5 SCORE vs. MP-SIMRANK

Both SCORE and MP-SIMRANK are designed for modelling multiple similarity perspectives and have shown better performances in the scenarios where each perspective’s data is insufficient and under-sampled as compared to uniperspective methods.

SCORE and MP-SIMRANK are not directly comparable, since the two frameworks are designed for different purposes. MP-SIMRANK attempts to estimate similarity values, whereas SCORE estimates similarity rankings. However, for the sake of completeness, we compare the two models on a same task as in Section 6.3.2: *predicting the unseen ordinal quadruples*. Specifically, for MP-SIMRANK, we measure how well the learnt similarity scores can predict correctly the direction of the quadruples. Whereas, for SCORE, we derive such scores based on the Euclidean distances between the object coordinates on each perspective-specific map.

	Zoo	HouseVote	Paris Attractions
MP-SimRank	0.71	0.72	0.67
SCORE	0.77	0.74	0.76

Table 6.2: Prediction Accuracies on Unseen Triplets of SCORE and MP-SIMRANK.

Table 6.2 shows the prediction accuracies of the two models for all three datasets. In this particular task, SCORE shows better performances as compared to MP-SIMRANK in predicting the unseen quadruples. This is expected since this task may be more aligned to SCORE’s objective.

These results, however, do not imply that SCORE is more effective in modelling multiperspectivity than MP-SIMRANK. A complete answer would require more comprehensive experiments to evaluate the output of both models on several other downstream tasks.

6.4 Coordinate Transformation

A necessary step is to transform the $(d + 1)$ –dimensional coordinate of objects on p ’s tangent hyperplane, i.e., $\{\text{Proj}_{x_p}(y_i)\}_{i \in \mathcal{O}}$, to their corresponding d –dimensional coordinates, i.e., $\{y_i^p\}_{i \in \mathcal{O}}$. For the purpose of visualizing the embedding for an perspective p on a scatterplot, we describe how to transform the $3-d$ coordinates of objects on p ’s tangent hyperplane to their corresponding $2-d$ coordinates in the following. However, the analysis below is also applicable to $d > 2$.

Since $x_p, \text{Proj}_{x_p}(y_i), \text{Proj}_{x_p}(y_j), \text{Proj}_{x_p}(y_k)$ lie on the tangent hyperplane $T_{x_p}\mathbb{S}^d$ of the task p , the three vectors are on $T_{x_p}\mathbb{S}^d$ as well:

$$u = \text{Proj}_{x_p}(y_i) - x_p; v = \text{Proj}_{x_p}(y_j) - x_p; w = \text{Proj}_{x_p}(y_k) - x_p$$

As illustrated in Figure 6.10, the cross product $x_p \times u$ is a vector on $T_{x_p}\mathbb{S}^d$ and perpendicular to x_p, u . Let’s denote:

$$e_1 = \frac{u}{\|u\|}, e_2 = \frac{x_p \times u}{\|x_p \times u\|}.$$

We can see that e_1, e_2 form a basis of $T_{x_p} \mathbb{S}^d$ (since $\|e_1\| = \|e_2\| = 1, e_1^T e_2 = 0$). From linear algebra, for each point $y \in T_{x_p} \mathbb{S}^d$, there exists unique $a_y, b_y \in \mathbb{R}$ such as:

$$(y - x_p) = a_y \cdot e_1 + b_y \cdot e_2$$

Consider the following transformation map where $a_y, b_y \in \mathbb{R}$ are defined as above:

$$\begin{aligned} \text{Tr}_p : T_{x_p} \mathbb{S}^d &\rightarrow \mathbb{R}^2 \\ y &\mapsto \text{Tr}_p(y) = (a_y, b_y) \end{aligned} \quad (6.12)$$

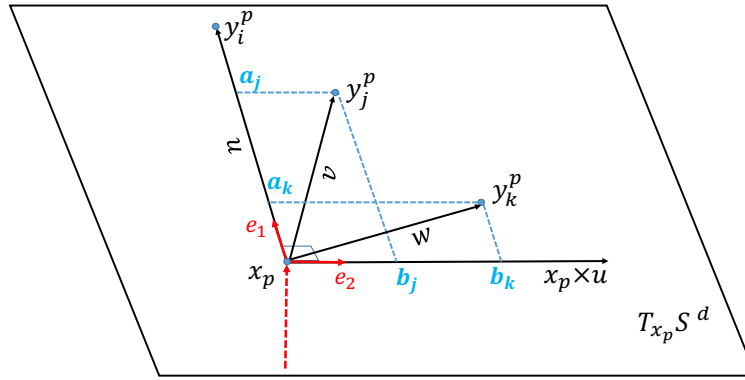


Figure 6.10: Transformation of objects' coordinates from 3-d to 2-d.

Let (a_j, b_j) and (a_k, b_k) be the transformation of $\text{Proj}_{x_p}(y_j)$ $\text{Proj}_{x_p}(y_k)$ respectively:

$$\begin{aligned} \|\text{Proj}_{x_p}(y_j) - \text{Proj}_{x_p}(y_k)\| &= \|(\text{Proj}_{x_p}(y_j) - x_p) - (\text{Proj}_{x_p}(y_k) - x_p)\| \\ &= \|v - w\| = \|(a_j \cdot e_1 + b_j \cdot e_2) - (a_k \cdot e_1 + b_k \cdot e_2)\| \\ &= \|(a_j - a_k) \cdot e_1 + (b_j - b_k) \cdot e_2\| = \sqrt{(a_j - a_k)^2 + (b_j - b_k)^2} \\ &= \|\text{Tr}_p(\text{Proj}_{x_p}(y_j)) - \text{Tr}_p(\text{Proj}_{x_p}(y_k))\|. \end{aligned} \quad (6.13)$$

Equation 6.13 implies that the L_2 -norm between points on $T_{x_p} \mathbb{S}^d$ are preserved through the trans-

formation map Tr_p . Therefore, the ordinal relations between points are also preserved through the transformation. Hence, we express $y_i^p = \text{Tr}_p(\text{Proj}_{x_p}(y_i))$, for all $i \in \mathcal{O}$.

6.5 Discussion

In this work, we formulate the problem of ordinal embedding involving comparisons from multiple perspectives as conditional ordinal embedding. Our proposed geometric approach seeks to represent perspectives and objects on a shared hypersphere, as well as on perspective-specific tangent hyperplanes. Experiments on public datasets show that the proposed framework is robust, and particularly beneficial when there is variance across perspectives yet with insufficient data to learn each perspective’s map separately, thus collaboration across perspectives is helpful.

Chapter 7

Conclusion

7.1 Summary

This thesis considers personalized recommendation problem, whose fundamental entities include items (e.g., products, movies, etc.) and users (who are consumers). The objective of personalized recommender systems is to mine from users' historical feedbacks in order to learn their preferences, and apply the learnt knowledge to filter out items that are tailored towards users' tastes. Personalization can appear in many forms, depending on the characteristics of the items and the desired experience that the system wants users to have. In this thesis, we explore two such perspectives on personalized recommendations: *preference learning* and *similarity learning*.

For the first perspective, i.e., *preference learning*, we focus on the problem of retrieving personalized recommendations via indexing as a faster alternative to an exhaustive search over all items. We drill down to the problem of designing recommendation algorithms whose output representations natively support sub-linear time retrieval of candidates for recommendation requests. In Chapters 3 and 4, we respectively describe two frameworks: Collaborative Ordinal Embedding (COE) and Indexable Bayesian Personalized Ranking (INDEXABLE BPR) that are "retrieval-friendly", i.e., the learning output natively supports efficient top preferred recommendation re-

trieval. Experiments on real-world datasets show that COE and INDEXABLE BPR outperform their baselines both in terms of recommendation accuracy and retrieval efficiency. This shows our approach on redefining the user-item interaction modelling of classic matrix factorization model is useful in improving the retrieval efficiency of the recommendation process.

For the second perspective, i.e., *similarity learning*, we focus our attention on effectively modelling multiple similarity perceptions in the data. Towards this goal, we respectively propose two different approaches in Chapters 5 and 6 that are both multiperspective. One approach adopts the idea of graph-theoretic similarity measure and proposes a framework MP-SIMRANK that yields a similarity measure for any pair of objects for a specific perspective. Another approach is based on multi-perspective ordinal embedding, dealing with the problem of learning multiple low-dimensional maps, each for one perspective, from a collection of multi-perspective ordinal comparisons. Experiments on several public datasets with varying similarity perspectives showcase the utility of multiperspective modelling as compared to the uniperspective baselines.

7.2 Future Work

In this thesis, the two problems of our interest are *efficient recommendation retrieval* and *multiperspective similarity modelling*. Though we proposed several methods towards solving these problems, we acknowledge that there are still many open problems that worth further investigation, namely:

- **Stochastically Robust Representation for LSH Recommendation Retrieval**

From the analysis in Section 4.4, INDEXABLE BPR has shown greater compatibility with LSH indexing structure as compared to the baselines. However, since LSH is inherently a stochastic method, performance degeneration caused by randomly-drawn LSH hash functions is inevitable. One interesting direction for further exploration is to factor in the stochasticity of LSH hash functions when learning real-valued user and item latent vectors. The out-

put vectors will be more robust to the stochasticity of LSH structure, potentially producing superior post-LSH-indexing performances as compared to INDEXABLE BPR.

- **Scalable Multiperspective Similarity Measurement**

Though MP-SIMRANK has shown the utility of multiperspective approaches over conventional uniperspective methods, we are aware of the high computational expenses of this graph-theoretic framework (as analyzed in Section 5.4). This might hinder the adoption of multiperspective similarity approaches for real-world systems. As MP-SIMRANK also suffers the inefficiency from the recursive dependency in the computation as SimRank, one future direction is to investigate earlier works on speeding up SimRank (Section 2.2.2) and look for potential scaling solutions that can be integrated in the multiperspective formulation in Equation 5.2.

Bibliography

- [1] Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2009. *Optimization algorithms on matrix manifolds*. Princeton University Press. 6.2.4
- [2] Adomavicius, G., and Kwon, Y. 2015. Multi-criteria recommender systems. In *Recommender Systems Handbook*. 1.1.2
- [3] Agarwal, S.; Wills, J.; Cayton, L.; Lanckriet, G.; Kriegman, D. J.; and Belongie, S. 2007. Generalized non-metric multidimensional scaling. In *AISTATS*. 2.2.2, 3.1, 6.1, 6.2.2
- [4] Aggarwal, C. C.; Wolf, J. L.; Wu, K.-L.; and Yu, P. S. 1999. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 201–212. ACM. 1.1.1
- [5] Aly, M.; Munich, M.; and Perona, P. 2011. Indexing in large scale image collections: Scaling properties and benchmark. In *IEEE Workshop on Applications of Computer Vision (WACV)*, 418–425. IEEE. 4.3.1
- [6] Amid, E., and Ukkonen, A. 2015. Multiview triplet embedding: Learning attributes in multiple maps. In *ICML*, 1472–1480. 2.2.2, 6.3.1
- [7] Argyriou, A.; Evgeniou, T.; and Pontil, M. 2007. Multi-task feature learning. In *NIPS*, 41–48. 2.2.2
- [8] Bachrach, Y.; Finkelstein, Y.; Gilad-Bachrach, R.; Katzir, L.; Koenigstein, N.; Nice, N.; and Paquet, U. 2014a. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*. 3.3.1
- [9] Bachrach, Y.; Finkelstein, Y.; Gilad-Bachrach, R.; Katzir, L.; Koenigstein, N.; Nice, N.; and Paquet, U. 2014b. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, 257–264. ACM. 1.1.1, 1.1.1, 4.1, 4.3
- [10] Bachrach, Y.; Finkelstein, Y.; Gilad-Bachrach, R.; Katzir, L.; Koenigstein, N.; Nice, N.; and Paquet, U. 2014c. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*, 257–264. 1, 2.1.1
- [11] Bai, S.; Bai, X.; Latecki, L. J.; and Tian, Q. 2017. Multidimensional scaling on multiple input distance matrices. In *AAAI*, 1281–1287. 6.3.1
- [12] Beckmann, N.; Kriegel, H.-P.; Schneider, R.; and Seeger, B. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*. 3.1

- [13] Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *TPAMI* 35(8):1798–1828. 2.2.2
- [14] Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9):509–517. 1.1.1, 2.1.1
- [15] Bhowmik, A.; Liu, N.; Zhong, E.; Bhaskar, B. N.; and Rajan, S. 2016. Geometry aware mappings for high dimensional sparse factors. In *AISTATS*. 1.1.1, 2.1.1, 1, 4.3.3
- [16] Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *JMLR* 3(Jan):993–1022. 2.2.2
- [17] Bonnabel, S. 2013. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control* 58(9):2217–2229. 6.2.4, 6.2.4
- [18] Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75. 2.2.2
- [19] Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, 380–388. ACM. 4.4
- [20] Chechik, G.; Sharma, V.; Shalit, U.; and Bengio, S. 2010. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research* 11(Mar):1109–1135. 2.2.1
- [21] Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262. ACM. 4.4
- [22] der Maaten, L. V., and Hinton, G. 2008. Visualizing data using t-SNE. *JMLR* 9. 3.1
- [23] Desrosiers, C., and Karypis, G. 2011. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*. 1.1.1
- [24] Dorow, B.; Laws, F.; Michelbacher, L.; Scheible, C.; and Utt, J. 2009. A graph-theoretic algorithm for automatic extension of translation lexicons. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, 91–95. Association for Computational Linguistics. 2.2.2
- [25] Edgar, R. C. 2004. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics* 5(1):113. 5.1
- [26] Ekstrand, M. D.; Riedl, J. T.; and Konstan, J. A. 2011. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction* 4(2):81–173. 3.3
- [27] Estévez, P. A.; Tesmer, M.; Perez, C. A.; and Zurada, J. M. 2009. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks* 20(2):189–201. 5.3.3, 6.3.3
- [28] Ferreira, O.; Iusem, A.; and Németh, S. 2014. Concepts and techniques of optimization on the sphere. *TOP* 22(3):1148–1170. 6.2.3, 6.2.4
- [29] Fraccaro, M.; Paquet, U.; and Winther, O. 2016. Indexable probabilistic matrix factorization for maximum inner product search. In *AAAI*, 1554–1560. 1.1.1, 2, 2.1.1, 3.3.2, 4.1, 4.2, 4.3,

4.3.1, 4.3.2

- [30] Geerts, F.; Mannila, H.; and Terzi, E. 2004. Relational link-based ranking. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 552–563. VLDB Endowment. 2.2.2
- [31] Gilkey, P. B., et al. 1975. The spectral geometry of a riemannian manifold. *Journal of Differential Geometry* 10(4):601–618. 6.2.2
- [32] Globerson, A.; Chechik, G.; Pereira, F.; and Tishby, N. 2007. Euclidean embedding of co-occurrence data. *JMLR* 8:2047–2076. 3.3, 3.3.1
- [33] Gomes, R. G.; Welinder, P.; Krause, A.; and Perona, P. 2011. Crowdclustering. In *NIPS*, 558–566. 2.2.1, 6.1
- [34] Guo, R.; Kumar, S.; Choromanski, K.; and Simcha, D. 2016. Quantization based fast inner product search. In *AISTATS*, 482–490. 2.1.2
- [35] He, R., and McAuley, J. 2016. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*. 4.1
- [36] He, G.; Feng, H.; Li, C.; and Chen, H. 2010. Parallel simrank computation on large graphs with iterative aggregation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 543–552. ACM. 2.2.2
- [37] Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 3.2.1
- [38] Huang, Q.; Ma, G.; Feng, J.; Fang, Q.; and Tung, A. K. H. 2018. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *KDD*, 1561–1570. 1
- [39] Hughes, T., and Ramage, D. 2007. Lexical semantic relatedness with random graph walks. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2.2.2
- [40] Jeh, G., and Widom, J. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 538–543. ACM. 1.1.2, 2.2.2, 5.1, 5.2.1, 5.3.1
- [41] Ji, J.; Li, J.; Yan, S.; Zhang, B.; and Tian, Q. 2012. Super-bit locality-sensitive hashing. In *Advances in Neural Information Processing Systems*, 108–116. 4.4
- [42] Jolliffe, I. 2005. *Principal Component Analysis*. Wiley Online Library. 3.3.1
- [43] Keivani, O.; Sinha, K.; and Ram, P. 2018. Improved maximum inner product search with better theoretical guarantee using randomized partition trees. *Machine Learning* 1–26. 1
- [44] Kendall, M. G. 1948. *Rank correlation methods*. Griffin. 3.2.1
- [45] Khoshneshin, M., and Street, W. N. 2010a. Collaborative filtering via euclidean embedding. In *RecSys*. 3.3.1
- [46] Khoshneshin, M., and Street, W. N. 2010b. Collaborative filtering via euclidean embedding. In *Proceedings of the fourth ACM conference on Recommender systems*, 87–94. ACM. 1.1.1,

2, 2.1.1, 4.3

- [47] Koenigstein, N.; Ram, P.; and Shavitt, Y. 2012a. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 535–544. ACM. 1.1.1, 4.1
- [48] Koenigstein, N.; Ram, P.; and Shavitt, Y. 2012b. Efficient retrieval of recommendations in a matrix factorization framework. In *CIKM*, 535–544. 2.1.1
- [49] Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8). 1.1.1
- [50] Krohn-Grimberghe, A.; Drumond, L.; Freudenthaler, C.; and Schmidt-Thieme, L. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*, 173–182. ACM. 4.1
- [51] Kruskal, J. B. 1964a. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29(1). 1.1.1
- [52] Kruskal, J. B. 1964b. Nonmetric multidimensional scaling: a numerical method. *Psychometrika* 29(2). 3.1
- [53] Kusumoto, M.; Maehara, T.; and Kawarabayashi, K.-i. 2014. Scalable similarity search for simrank. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 325–336. ACM. 2.2.2
- [54] Le, D. D., and Lauw, H. W. 2016. Euclidean co-embedding of ordinal data for multi-type visualization. In *SDM*, 396–404. 1.1.1, 1.2, 4.3
- [55] Le, D. D., and Lauw, H. W. 2017. Indexable bayesian personalized ranking for efficient top-k recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1389–1398. ACM. 1.2
- [56] Le, D. D., and Lauw, H. W. 2018. Multiperspective graph-theoretic similarity measure. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1223–1232. ACM. 1.2
- [57] Lerche, L., and Jannach, D. 2014. Using graded implicit feedback for bayesian personalized ranking. In *RecSys*, 353–356. ACM. 4.1
- [58] Li, C.; Han, J.; He, G.; Jin, X.; Sun, Y.; Yu, Y.; and Wu, T. 2010a. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, 465–476. ACM. 2.2.2
- [59] Li, P.; Liu, H.; Yu, J. X.; He, J.; and Du, X. 2010b. Fast single-pair simrank computation. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, 571–582. SIAM. 2.2.2
- [60] Li, H.; Chan, T. N.; Yiu, M. L.; and Mamoulis, N. 2017. Fexipro: Fast and exact inner product retrieval in recommender systems. In *SIGMOD*, 835–850. 2.1
- [61] Li, Z. 2018. Towards the next generation of multi-criteria recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18*, 553–557. New

York, NY, USA: ACM. 1.1.2

- [62] Lian, D.; Liu, R.; Ge, Y.; Zheng, K.; Xie, X.; and Cao, L. 2017. Discrete content-aware matrix factorization. In *KDD*, 325–334. 2
- [63] Liu, X.; He, J.; Deng, C.; and Lang, B. 2014. Collaborative hashing. *2014 IEEE Conference on Computer Vision and Pattern Recognition* 2147–2154. 1
- [64] Liu, D. C.; Rogers, S.; Shiau, R.; Kislyuk, D.; Ma, K. C.; Zhong, Z.; Liu, J.; and Jing, Y. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 583–592. International World Wide Web Conferences Steering Committee. 1.1
- [65] Liu, H.; He, X.; Feng, F.; Nie, L.; Liu, R.; and Zhang, H. 2018. Discrete factorization machines for fast feature-based recommendation. In *IJCAI*. 2
- [66] Ma, G.; Lu, C.-T.; He, L.; Philip, S. Y.; and Ragin, A. B. 2017. Multi-view graph embedding with hub detection for brain network analysis. In *2017 IEEE International Conference on Data Mining (ICDM)*, 967–972. IEEE. 2.2.2
- [67] Maehara, T.; Kusumoto, M.; and Kawarabayashi, K.-i. 2014. Efficient simrank computation via linearization. *arXiv preprint arXiv:1411.7228*. 2.2.2
- [68] Magdy, W., and Jones, G. J. 2010. Pres: a score metric for evaluating recall-oriented information retrieval applications. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 611–618. ACM. 5.3.1
- [69] Majumder, G. S.; Dwivedi, P.; and Kant, V. 2018. Matrix factorization and regression-based approach for multi-criteria recommender system. In Satapathy, S. C., and Joshi, A., eds., *Information and Communication Technology for Intelligent Systems (ICTIS 2017) - Volume 1*, 103–110. Cham: Springer International Publishing. 1.1.2
- [70] Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to information retrieval*. 3.1
- [71] Mardia, K. V. 1975. Distribution theory for the von Mises-Fisher distribution and its application. In *A Modern Course on Statistical Distributions in Scientific Work*. Springer. 6.2.3
- [72] McFee, B., and Lanckriet, G. 2011a. Learning multi-modal similarity. *JMLR* 12:491–523. 2.2.1
- [73] McFee, B., and Lanckriet, G. R. G. 2011b. Large-scale music similarity search with spatial trees. In *ISMIR*. 4.3.2
- [74] Morozov, S., and Babenko, A. 2018. Non-metric similarity graphs for maximum inner product search. In *Advances in Neural Information Processing Systems*, 4722–4731. 2.1.1
- [75] Muja, M., and Lowe, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP'09)*, 331–340. INSTICC Press. 3.3.2, 4.3.2
- [76] Neyshabur, B., and Srebro, N. 2015a. On symmetric and asymmetric lshs for inner product search. In *ICML*, 1926–1934. 1

- [77] Neyshabur, B., and Srebro, N. 2015b. On symmetric and asymmetric lshs for inner product search. In *ICML*. 2.1.1, 4.3.1
- [78] Pan, W., and Chen, L. 2013. GBPR: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*, volume 13, 2691–2697. 4.1
- [79] Parameswaran, S., and Weinberger, K. Q. 2010. Large margin multi-task metric learning. In *NIPS*, 1867–1875. 2.2.2
- [80] Qu, M.; Tang, J.; Shang, J.; Ren, X.; Zhang, M.; and Han, J. 2017. An attention-based collaboration framework for multi-view network representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*, 1767–1776. New York, NY, USA: ACM. 2.2.1
- [81] Radlinski, F., and Joachims, T. 2005. Query chains: learning to rank from implicit feedback. In *KDD*. 3.2.1
- [82] Ram, P., and Gray, A. G. 2012. Maximum inner-product search using cone trees. In *KDD*, 931–939. 2.1.1
- [83] Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 2.1.1, 3.2.1, 3.3, 3.3.1, 4.1, 4.2, 4.3
- [84] Rothe, S., and Schütze, H. 2014. Cosimrank: A flexible & efficient graph-theoretic similarity measure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1392–1402. 2.2.2
- [85] Roussopoulos, N.; Kelley, S.; and Vincent, F. 1995. Nearest neighbor queries. In *SIGMOD*. 3.1
- [86] Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290. 3.1
- [87] Salakhutdinov, R., and Mnih, A. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, 880–887. ACM. 1.1.1, 1.1.1, 4.1
- [88] Sarwar, B. M.; Karypis, G.; Konstan, J. A.; Riedl, J.; et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* 1:285–295. 1.1.1
- [89] Shepard, R. N. 1962. The analysis of proximities: Multidimensional scaling with an unknown distance function. i. *Psychometrika* 27(2). 3.1
- [90] Shi, Y.; Han, F.; He, X.; He, X.; Yang, C.; Luo, J.; and Han, J. 2018. mvn2vec: Preservation and collaboration in multi-view network embedding. *arXiv preprint arXiv:1801.06597*. 2.2.2
- [91] Shrivastava, A., and Li, P. 2014a. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, 2321–2329. 1.1.1, 4.2
- [92] Shrivastava, A., and Li, P. 2014b. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*, 2321–2329. 2.1.1, 1
- [93] Shrivastava, A., and Li, P. 2015a. Improved asymmetric locality sensitive hashing (alsh) for

- maximum inner product search (mips). In *UAI*, 812–821. 1
- [94] Shrivastava, A., and Li, P. 2015b. Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips). In *UAI*. 4.3.1
- [95] Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4(11):992–1003. 2.2.2
- [96] Sun, Y.; Bui, N.; Hsieh, T.-Y.; and Honavar, V. 2018. Multi-view network embedding via graph factorization clustering and co-regularized multi-view agreement. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 1006–1013. IEEE. 2.2.2
- [97] Tamuz, O.; Liu, C.; Shamir, O.; Kalai, A.; and Belongie, S. J. 2011. Adaptively learning the crowd kernel. In *ICML*, 673–680. 2.2.2
- [98] Teflioudi, C., and Gemulla, R. 2016. Exact and approximate maximum inner product search with lemp. *TODS* 42(1):5:1–5:49. 2.1
- [99] Teflioudi, C.; Gemulla, R.; and Mykytiuk, O. 2015. Lemp: Fast retrieval of large entries in a matrix product. In *SIGMOD*. 2.1
- [100] Tenenbaum, J. B.; Silva, V. D.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290. 3.1
- [101] Terada, Y., and Luxburg, U. V. 2014. Local ordinal embedding. In *ICML*. 2.2.2, 3.1, 3.2.3, 6.1, 6.3.1
- [102] Tian, B., and Xiao, X. 2016. Sling: A near-optimal index structure for simrank. In *SIGMOD Conference*. 2.2.2
- [103] Tong, H.; Faloutsos, C.; and Pan, J. Y. 2006. Fast random walk with restart and its applications. In *6th International Conference on Data Mining, ICDM 2006*. 2.2.2
- [104] Van der Maaten, L., and Hinton, G. 2012. Visualizing non-metric similarities in multiple maps. *Machine Learning* 87(1):33–55. 2.2.2
- [105] Van der Maaten, L., and Weinberger, K. 2012. Stochastic triplet embedding. In *MLSP*, 1–6. 2.2.2, 6.1, 6.3.1
- [106] Veit, A.; Belongie, S.; and Karaletsos, T. 2017. Conditional similarity networks. *Computer Vision and Pattern Recognition (CVPR 2017)*. 2.2.1
- [107] Vucetic, S., and Obradovic, Z. 2005. Collaborative filtering using a regression-based approach. *Knowledge and Information Systems* 7(1):1–22. 1.1.1
- [108] Weimer, M.; Karatzoglou, A.; Le, Q. V.; and Smola, A. J. 2007. Cofi rank - maximum margin matrix factorization for collaborative ranking. In *NIPS*. 1.1.1
- [109] Weinberger, K. Q., and Saul, L. K. 2009. Distance metric learning for large margin nearest neighbor classification. *JMLR* 10:207–244. 2.2.1
- [110] Wilber, M. J.; Kwak, I. S.; and Belongie, S. J. 2014. Cost-effective hits for relative similarity comparisons. In *HCOMP*. 6.1

- [111] Wills, J.; Agarwal, S.; Kriegman, D.; and Belongie, S. 2009. Toward a perceptual space for gloss. *TOG* 28(4):103. 6.1
- [112] Yang, P.; Huang, K.; and Liu, C.-L. 2013. Geometry preserving multi-task metric learning. *Machine Learning* 92(1):133–175. 2.2.2
- [113] Yang, L. 2007. An overview of distance metric learning. In *CVPR*. 2.2.1
- [114] Yu, H.-F.; Hsieh, C.-J.; Lei, Q.; and Dhillon, I. S. 2017. A greedy approach for budgeted maximum inner product search. In *NIPS*, 5453–5462. 2.1.1
- [115] Yu, W.; Lin, X.; Zhang, W.; Pei, J.; and McCann, J. A. 2018. Simrank*: effective and scalable pairwise similarity search based on graph topology. *The VLDB Journal* 1–26. 2.2.2
- [116] Yue, Y.; Wang, C.; El-Arini, K.; and Guestrin, C. 2014. Personalized collaborative clustering. In *WWW*, 75–84. 1.1.2, 2.2.1, 2.2.2, 5.1, 5.3.1, 6.1, 6.3.1
- [117] Zhang, Z.; Wang, Q.; Ruan, L.; and Si, L. 2014. Preference preserving hashing for efficient recommendation. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 183–192. ACM. 1
- [118] Zhang, H.; Shen, F.; Liu, W.; He, X.; Luan, H.; and Chua, T.-S. 2016. Discrete collaborative filtering. In *Proc. of SIGIR*, volume 16. 2.1, 2.1.2, 2, 2.1.2
- [119] Zhang, Y.; Wang, H.; Lian, D.; Tsang, I. W.; Yin, H.; and Yang, G. 2018a. Discrete ranking-based matrix factorization with self-paced learning. In *KDD*, 2758–2767. 2
- [120] Zhang, Y.; Yin, H.; Huang, Z.; Du, X.; Yang, G.; and Lian, D. 2018b. Discrete deep learning for fast content-aware recommendation. In *WSDM*, 717–726. 2
- [121] Zhang, S.; Yao, L.; Sun, A.; and Tay, Y. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.* 52(1):5:1–5:38. 1.1.1
- [122] Zhang, Y.; Lian, D.; and Yang, G. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *AAAI*, 1669–1675. 2
- [123] Zhou, K., and Zha, H. 2012. Learning binary codes for collaborative filtering. In *KDD*, 498–506. 1