University of Cincinnati

Date: 7/19/2016

I, Yi Zhang, hereby submit this original work as part of the requirements for the degree of Doctor of Philosophy in Mechanical Engineering.

It is entitled:
Groupwise Distance Learning Algorithm for User Recommendation Systems

Student's name: Yi Zhang

This work and its defense approved by:

Committee chair: Hongdao Huang, Ph.D.

Committee member: Jing Shi, Ph.D.

Committee member: David Thompson, Ph.D.

Committee member: Xuefu Zhou, Ph.D.
Groupwise Distance Learning Algorithm for User Recommendation Systems

A dissertation submitted to the

Graduate School

of the University of Cincinnati

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Department of Mechanical and Materials Engineering

of the College of Engineering and Applied Science

by

Yi Zhang

M.S. University of Cincinnati

B.S. University of Electronic Science and Technology of China

July 2016

Committee Chair: Samuel H. Huang, Ph.D.
Abstract

The dissertation study focuses on addressing some challenges in user-to-user recommendation area. As results, a new algorithm, Group-wise Distance Learning (GDL), is proposed in this study as a backbone to build user recommendation systems with an approximate personalization effect. By analyzing the established connections among users, the recommendation system empowered by GDL can induce users’ social preference. Instead of learning individual user’s social preference separately, GDL is designed to adopt an efficient strategy by clustering users into a number of groups. Users are grouped together in accordance with their similar social preference, which is extracted by GDL and encoded as feature-wise weights. Thanks for the approximate personalization with learned group-wise social preference, a GDL-empowered user recommendation system could help users more effectively expand their social networking. With the assistance of this new user recommendation system, a social network is expected to evolve faster and more connected in a collective manner.

The performance of a user recommendation system empowered by GDL was compared with the recommendation system empowered by Nearest Neighbours (NN) in a series of simulation experiments. The simulation experiment was designed to mimic the evolution of a social network fueled by the interactions between users and the recommendation system. According to simulation experiment results based on two distinct synthetic data sets, the proposed GDL-empowered user recommendation system outperformed NN-empowered counterparts. Because of the cold-start issue, we recommend building a hybrid user recommendation system of these two methods. For a new user with zero user connections,
use a NN-based method to generate recommendation initially. For users having a solid social profile, a GDL-empowered system can serve them better. In combination, a superior recommendation system can be built with benefits of two different methods.
Acknowledgements

At first, I want to acknowledge that I am deeply indebted to my advisor, Dr. Samuel H. Huang for countless reasons. His mentorship for me had extended significantly beyond the academic scope. With his influence had I been encouraged to be developing toward a more mature person. And, I want to express my appreciation to the members of my dissertation committee for the patience and willingness in a face of my prompt asking for help. Additionally, I would like to thank fellow students in Dr. Huang’s laboratory for their suggestions and friendships. And, I also want to thank the open-source community for their generous contributions to the world. Without their works, it is nearly impossible for me to complete this dissertation. At last, I want to thank my family for their unconditional support.
Contents

1 Overview ................................................. 10
  1.1 Introduction ........................................ 10
  1.2 Problem Statement ................................. 12

2 Literature Review ...................................... 15
  2.1 Overview ............................................. 15
  2.2 Algorithms .......................................... 17
    2.2.1 Collaborative Filtering ....................... 17
    2.2.2 Content-based Methods ....................... 19
    2.2.3 Factorization Matrix ......................... 21
    2.2.4 Hybrid Methodology ........................... 24
  2.3 Distance and Similarity Metrics .................. 26
    2.3.1 Euclidean Distance ............................ 27
    2.3.2 Cosine Similarity ............................... 27
    2.3.3 Pearson Correlation ............................ 28
    2.3.4 Mahalanobis Distance ......................... 28
  2.4 Evaluation of Recommendation Systems .......... 29
    2.4.1 Accuracy ........................................ 30
    2.4.2 Coverage ........................................ 35
    2.4.3 Learning Rate ................................... 35
5.3.1 Overview of Experiment Data Set .................................. 75
5.3.2 Experiment Configuration ......................................... 75
5.3.3 Results ................................................................. 76
5.3.4 Conclusion ............................................................. 79

6 Software Implementation .................................................. 82

6.1 Implementation .......................................................... 82

6.2 Introduction to Python Implementation ............................... 83

6.2.1 API Design .......................................................... 85

6.2.2 Summary ............................................................. 87

7 Conclusion and Future Plan ................................................ 88

7.1 Future Plan ............................................................... 89

7.2 Final Words .............................................................. 90
List of Figures

3.1 Group-wise Distance Learning (GDL) Algorithm ....................................... 41
3.2 Bar Plot of Feature-wise Difference between Users .............................. 48
3.3 Social network visualization ................................................................. 49
3.4 Empirical Cumulative Density Functions of Two Samples, D and S ......... 52

4.1 Cosine Distance between true weights vs. learned weights with various mixture percentage of users whose preference is represented by the true weights (red: 20 users, purple: 30 users, blue: 50 users, green: 60 users) ............. 59
4.2 Density Distribution Plots of P-values of KS-tests. ............................... 61
4.3 Learning Curve: Fit Score vs. Iteration ............................................... 63

5.1 Simulation Experiment Workflow ......................................................... 66
5.2 Common Edge Ratio (CER) Plot (Simulation Experiment 01) .................. 71
5.3 Average No. of New User Connections per Iteration (Simulation Experiment 01) ................................................................. 72
5.4 No. of New User Connections per Iteration (Simulation Experiment 01) . . 72
5.5 Common Edge Ratio (CER) Plot (Simulation Experiment 02) .................. 77
5.6 Average No. of New User Connections per Iteration (Simulation Experiment 02) ................................................................. 78
5.7 No. of New User Connections per Iteration (Simulation Experiment 02) . . 79

6.1 Implementation Workflow of Group-wise Learning Distance Algorithm . . . 84
List of Tables

2.1 Confusion Matrix for Classification Task ............................................. 31

4.1 Basic Properties of Synthetic Data Sets ................................................. 57

5.1 Median of CER per Iteration (Sample Rate = 0.2) .................................. 73

5.2 Standard Deviation of CER per Iteration (Sample Rate = 0.2) .................. 74

5.3 Median of CER per Iteration (Sample Rate = 0.3) .................................. 80

5.4 Standard Deviation of CER per Iteration (Sample Rate = 0.3) .................. 81
Chapter 1

Overview

1.1 Introduction

The advancement of mobile and web technology leads to the emergence of various online social networks (e.g. Facebook.com, Google+, etc.). The primary driver of a company operating those social networks is to maximize the number of people connected. The recommendation system is the most important device employed to facilitate exploration of unknown space and develop new connections among users. In general, recommendation system was invented to address information overload issue by proactively conducting information retrieval for users. Over last decades, numerous companies had blossomed with critical contributions of their recommendation systems. Amazon.com suggests products based on customer’s shopping history; Netflix recommends the next movie to a consumer based on her taste. There is a broad spectrum of algorithms/methods to develop item-to-user recommendation system. For example, collaborative filtering algorithm generates new items to a user with the items liked users having similar tastes. Content-based recommendation framework, as a mainstream alternative method, models every individual user’s preference separately by extracting the patterns from the historically consumed items. There are many more methods worthy of mentioning in this item-to-user recommendation arena. The consensus of various
researches concludes that the key feature of the recommendation system is to achieve full personalization recommendation.

To attempt to address some challenges in user-to-user recommendation area, a new algorithm, Group-wise Distance Learning (GDL), is proposed in this study as a backbone to build user recommendation systems. By analyzing the established connections among users, the recommendation system empowered by GDL can induce users’ social preference. Instead of learning individual user’s social preference separately, GDL is designed to adopt a more effective strategy by clustering users into some groups. Users are grouped with accordance to their common social preference, which is encoded by GDL with feature weights as a part of outputs of GDL’s learning. Thanks for the approximate personalization with encoded group-wise social preference, a GDL-empowered user recommendation system could help users more effectively expand their social network. In return, the entire social network evolves more connected in a collective manner.

The performance of user recommendation system empowered by GDL was compared with the recommendation system empowered by Nearest Neighbours (NN) in a series of simulation experiments. The simulation experiment was designed to mimic the evolution of a social network fueled by the interactions between users and recommendation systems. According to simulation experiment results based on two distinct synthetic data sets, the proposed GDL-empowered user recommendation system outperforms NN-empowered counterparts. Because of the cold-start issue, we recommend building a hybrid user recommendation system of these two methods. For a new user with zero user connections, use a NN-based method to generate recommendation initially. For users having a solid social profile, a GDL-empowered system can serve them better. In combination, a superior recommendation system can be built with benefits of two different methods.

In this dissertation, it starts with reviewing a selection of algorithms and evaluation methods for recommendation system development. After that, it is a detailed statement to describe the main challenge to tackle in this dissertation study. The proposed algorithm
is introduced thoroughly in the subsequent section. To examine the individual behavior of main modules of the proposed algorithm, the results of a set of experiments with simulation data were discussed. After the examination of each module, a series of simulation experiments with two different data sets were conducted to compare the recommendation systems empowered by the proposed algorithm versus the counterparts empowered by Nearest Neighbours algorithm. Finally, we discuss the performance of the proposed algorithm in conjunction with the benefits and drawbacks.

1.2 Problem Statement

Numerous endeavors of academic researchers and industrial practitioners had been focused on improving the overall user experience of recommendation system. Personalization and novelty are two of the vital avenues to improve the usability of a recommendation system. Personalization means generating more suggestions based on the user’s personal taste. Novelty measures how surprising a suggestion is to a user and liked by the user too. However, the established methods are suffering numerous issues and unable to achieve a satisfied balance between those two competing interests: personalization and novelty. Particularly, for user recommendation systems, users are suffering no personalization in recommendations with systems with Nearest Neighbor algorithm, which help all of the users find their new friends based on the same similarity score. On the other hand, recommendation systems based suggesting friends’ friends suffers novelty issue. In this dissertation study, a new method is proposed with concentrated efforts on improving the personalization experience while introducing completely unknown users.

The personalization is needed to build user recommendations system due to two factors: the context of social network and personal social preference. On context of a social network, there are various types of social networks. For instance, Facebook is the one of the largest social network based on general friendship purpose. Linkedin is based upon the concept to
facilitate professional networking. And, there are no many another social networks specializing on specific walks of lives. Consider personal social preference, the users’ preference toward connections depends on some factors, personal preference, the purpose for networking and the context of a social network. In a context of a given social network, users are expected to behavior differently regarding who they want to connect with. Within a given social network, a fraction of differences could be explained by the difference in personal preference and purposes for networking. It is also the common observation that the same user can act differently regarding whom she would connect with from one social network to the other one. In the classical framework of finding user recommendation, a user’s preference can be expressed by the importance of each characteristic in user’s profile. However, currently, mainstream methods are built upon assuming all users’ difference can be measured by a single metric. This treatment ignores the factors contributing into user’s personal viewpoint of peer users. Therefore, the recommendation systems built upon this concept lacks the important traits, personalization, in their recommendations.

The proposed method in this dissertation is developed as an attempt to promote the development of personalized user recommendation system. Within the context of its initial proposed application area, building a user recommendation system for an online social network, this goal is achieved by dividing user populations into multiple groups. And, a set of weighted distance metrics which act as the core of recommendation system is learned for each of those user divisions. The weights of a learned distance metrics encode the preferences of all users within the corresponding user group. The weights of the distance function reflect how important factors are when a user decides if she/he want to friend someone.

In addition to their most important benefit of allowing personalization, the another important advantage of this proposed algorithm is a balance between generalization and personalization. On either extreme of over-generalization or over-personalization, the performance of recommendation system is jeopardized. For over-generalized recommendation system, an same route is followed to generate suggestions for all users. On the other side, with over-
personalized recommendation system, it encounters a different set of challenges, like cold start issues for new users. The proposed algorithm is devised to alleviate the aforementioned issues associated with the two mainstream recommendation algorithm frameworks.

The balanced performance is accomplished with the ability of the proposed algorithm to capture a set of different patterns in numerous user groups. Therefore, similar users are grouped together, and system learned the same strategy when generating suggestions for members of the same group. For users who are significantly different regarding their tastes, the algorithm will assign them into different groups and generate recommendations via different paths. As results, it achieves some level of generalization and, at the same time, creates room for personalization. In the context of user recommendation, the algorithm yields some user groups to cluster users who are similar on their preference for connections. Subsequently, a pattern extracted from a user group is generalized overall group members and applicable for them. By differentiating user groups, it promotes the personalization for users with different taste. Therefore, it is expected that this proposed algorithm can develop a superior recommendation system to the single-metric-based system.

In the dissertation study, the proposed algorithm is initially developed and studied with a focused application to develop a user recommendation system on a social network. However, the algorithm itself can be used to develop a general recommendation system which integrates both user-to-user social network information and traditional user-to-item consumption information. Additionally, the algorithm can be extended to apply in a different scenario to address similar challenges.
Chapter 2

Literature Review

2.1 Overview

Recommendation Systems are techniques or algorithms capable of providing suggestions on items of interests to users [28, 27, 29]. Since its invention in the mid-1990s, Recommendation System had drawn much attention in both academia and industry [28]. Recommendation System is coined to aid in guiding human beings to deal with information overloads by actively suggesting subjects based on a variety of background information. After decades of development, recommendation system is considered as a well researched academic area. However, there are still some issues left at least partially unsolved. On its application-specific nature, it demands continuous efforts to address new issues introduced by innovative applications. Nonetheless, numerous forms of recommendation systems had been implemented by a large number of companies to serve people in a different domain. The e-commerce giants, Amazon.com help a customer find a right product out of thousands similar products in the same category [1, 19]. Netflix predicts the next movie which a user may like to watch to keep them entertained [4, 18, 43]. Facebook, the most popular social network, built a user recommendation system to promote connections between users with shared characteristics. Spotify’s recommendation is able of generating a new music playlist comprised of music
featuring user’s music tastes 933.43.

There are a collection of state-of-arts algorithms for building recommendation systems. Collaborative filtering predicts a user’s preference for a particular item by aggregating the ratings of the item given by other users who have similar tastes with the target user. A variant of Collaborative Filtering, an item-to-item collaborative filtering algorithm groups similar items for which had been purchased by a common group of customers. Whenever a user express her interest in an item, the similar ones of this item of interests will be recommended for the user. Content-based algorithms use the inherent attributes of items to match items against the taste of users. A collection of articles considered closely matching a user’s interest would be recommended. Regardless of its decent performance based on empirical application, collaborative filtering methods are subject to cold-start or long-tail issues and sparsity problems. Content-based methods are at least less suffering the issues associated with collaborative filtering. But, according to empirical implementations, a pure content-based recommendation system usually is outperformed by its counterpart of collaborative filtering 35. Hence, there are an array of hybrid methods proposed as attempts to exploit benefits of collaborative filtering and content-based algorithms 7. In addition to aforementioned methods, factorization matrix had been discussed and introduced to approach Netflix Prize challenge. Within the particular context, factorization matrix is proven as the superior solution to another classic method for generating recommendations 17. The basic concept of matrix factorization is to project both of items and users into the same space consisting of a collection of latent factors. Following this convention, users are initially represented by items associated with them and profiled by encoding those associated items with the latent factors. The inner product of user-matrix and item-matrix provides a proxy to item-to-user preference. Rather than the direct interpretation, the interaction is decomposed into four components: global average, item bias, user bias and user-item interaction 1718.37. In literature review section, it starts with the introduction to mainstream algorithms, collaborative filtering, model-based method, factorization matrix and hybrid strategy. At the second
part of this section, distance and similarity metrics are discussed. At last, it focuses on definitions of various properties of recommendation and evaluation metrics.

2.2 Algorithms

2.2.1 Collaborative Filtering

Collaborative filtering methods help people to find item or information of interests by suggesting items historically enjoyed by others. The strategy is based on the assumption that people tend to have similar attitudes toward an item if they have large overlaps in preferences in the past. The major benefit of collaborative filtering is its not requiring a model to explain a user preference on an item. A recommendation system rates an item based on how much a user would like it. A 1-5 rating scheme was introduced in the open architecture of recommendation system in Groupon project \cite{28}. A rating is a numeric value given by a user to describe how much she likes an item. However, it should be noted that the rating might be determined by user’s preference, assessment of quality, influence cast by temporal advertisement etc. When interpreting data, it should be noted that some users might only give ratings which span over a subset of rating ranges. For examples, user A may only gave scores between 2 to 4. While, user B had given scores between 1 and 3. Therefore, a collaborative filtering based on rating should be robust to those issues.

In a rating matrix, rows represent items and columns represent users. The cells of this matrix store the ratings given by users on columns to the items on rows. In reality, it is very common to find that major portion of cells is filled with no data. For collaborative filtering method, it doest needs to use imputation methods to predict the missing value. The heart of collaborative filtering is to use the weighted combinations of ratings to an item given by other users to predict the rating which one of the users could grant to the item.

When predicting a rating of user $A$ to item $i$, $r_{A,i}$, the first step is to find the weight of each user should bear in term of their similarity to the given user. There are multiple methods to
calculate the similarities between a pair of users, and the details will be discussed in section 2.3. As one of most widely adopted metrics, correlation coefficient is used to quantify a similarity between user A and user B by considering their ratings over a basket of same items. As one of popular correlation-based measurement, the Pearson correlation coefficient is applied to measure the similarity between two users’ taste and its formula is shown in Equation 2.1.

\[
S_{A,B} = \frac{\text{cov}(r_A, r_B)}{\rho_A \cdot \rho_B} = \frac{\sum_{i \in \Omega}(r_{A,i} - \overline{r}_A)(r_{B,i} - \overline{r}_B)}{\sqrt{\sum_{i \in \Omega}(r_{A,i} - \overline{r}_A)^2} \cdot \sqrt{\sum_{i \in \Omega}(r_{B,i} - \overline{r}_B)^2}}
\]  

(2.1)

Where, \(\Omega\) is the collection of items which are rated by both user A and user B. Secondly, for given item i which is not rated by user A, the estimated rating of user A to item i, \(r_{A,i}\) is computed in 2.2.

\[
r_{A,i} = \overline{r}_A + \frac{\sum_{j \in U}(r_{j,i} - \overline{r}_j)S_{j,A}}{\sum_{j \in U}|S_{j,A}|}
\]

(2.2)

Where, \(U\) represents the set of users whose ratings to item i are considered. And, \(S_{(j,A)}\) is the similarity score between user j and user A. \((r_A)\) is the average of all ratings given by user A. Hence, the estimated rating is a combination of personal rating average and the weighted sum of others’ ratings on their taste similarities.

The collaborative filtering introduced above is the user-based version. Its typical application is to proactively suggest items to a user based on ranked ratings. However, the steps of the user-based collaborative filtering can be easily modified to do item-used collaborative filtering. The modification is realized by switching the roles of users and items involved in the computations of user-based. The purpose is to group similar items concerning their common user base [32]. In contrast, the typical usage of item-based collaborative filtering is to recommend an additional item associated with an item which is liked / consumed by a
As indicated in rating estimation equations, collaborative filtering might not provide a reliable estimate for a user who has a few number of rated items. Besides, for a user who has a unique taste, the strategy based on gathering peers’ knowledge can not predict her individual preference. And, in real-world, applying collaborative filtering concerns the scalability issues.

2.2.2 Content-based Methods

A content-based recommendation system is to find items whose attributes matching closely against the profile of a user’s taste [25]. Content-based recommendation systems have roots in information retrieval community and employs many methods pertinent to this domain [2]. A high-level architecture of content-based recommendation system includes three separate modules [21].

1. Content Analyzer is a process of summarizing the properties of an item with a selection of features. This task normally requires the domain-specific knowledge about items for a recommendation to create a representative feature collection. For instance, an online article recommendation system could characterize an article of unstructured texts with a binary vector to indicate the presences of a group of selected keywords. However, for a music recommendation system, a keyword representation of music lyrics could not provide enough information about a song. In this scenario, to create a fair representation of a song item, temporal-frequency representation of the audio signal, Mel spectrograms and Mel-frequency cepstrum (MFCC) are extracted [14, 38].

2. Profile Learner is to use machine learning techniques to generalize interests of users based on their historical preference. There are a large number of algorithms which can be used to accomplish this task. Depending on a system specification, the preference of the user to an item could be stored either explicitly or implicitly. Within an explicit
feedback system, a user could directly rate an item with a numeric value or label it with either 'like' or 'dislike'. In contrasts, the preference might be revealed by a user’s interactions within an implicit feedback system. For instances, bookmarking an article, keeping an item on wish-list, viewing an article for a long period could arguably indicate a user’s favorable attitudes toward the corresponding item. The task of modeling a user preference can be considered as a classification problem. A personalized preference classifier is trained by separating items liked by a user from the items disliked by a user. There is some content-based recommendation system empowered by varieties of algorithms. As introduced by Raymond Mooney, a Naive Bayesian probabilistic models are experimented to profile user interests about books [24, 30]. Guodong Guo and Stan Z. Li discussed the implementation of Support Vector Machine (SVM) as a classifier to categorize music item [10].

3. Filtering Component is the module to implement some strategies to generate a recommendation to a user. Typically, the profile of a given user is provided by the profile learner. For a new item, the profile learner processes the item’s representation to tell how likely the user would like it. After it, the filtering component sorts all evaluated items on their probabilities. At last, the list of ranked items will be recommended to the user.

In contrast to collaborative filtering recommendation system, the content-based recommendation systems have mainly three advantages. First, data independence, content-based recommendation system suggests items to a user without dependency on peer users’ preference information; Second, preference of users can be explained by the profile learning and content analyzer by listing the used features representing items and the logic of trained models. However, not all of machine learning algorithms applied to this problem has equal transparency in their encoded knowledge. Third, it is also possible for a new item to be recommended to a user if its properties match the preference of a user closely.

But, in addition to being outperformed by collaborative filtering, content-based ones, in
general, facing the following two issues: a. Serendipity issue, only items which are close to the revealed interests of users, can be recommended. Therefore, the recommended items only have a limited degree of novelty in comparison to consumed items. b. new user issue, new users are less likely to be recommended with interesting items due to lack of knowledge about her taste.

2.2.3 Factorization Matrix

The basic concept of Factorization Matrix is to create a mapping of both user and item profiles into a joint space formed by f latent factors. The idea behind this concept is that some unobserved factors determine the preferences or attitudes of users toward items [31]. Within the joint space with f dimensionality, user-item interactions, a proxy for the preference of a user to an item or rating of a user on an item, are modeled with the inner product of users and items matrix [17]. To model the personalization effect, the user-item interaction calculated in the previous step is formulated as a linear combination of four components to account for various resources of variations in ratings. It is because of that the observed variance in ratings is in large part due to either a user attitude or an item profile alone, independent of the user-item interaction. While prioritizing recommendations for items based on their preferences to a user, the measure of positive user-item interaction is used instead of the absolute value of rating estimates. Besides the ability to calculate the exact user-item interaction, the learning framework of factorization matrix allows for modeling the temporal dynamics. In the reality, a user’s taste on a collection of items is changing in response to numerous factors over time. By considering the component-wise user-item interaction as functions of time, the learning framework introduces the flexibility to consider the temporal effects of user preference.

To map the item into the joint space, ith item encoded with a vector \( q_i \in \mathbb{R}^f \). Each of f elements of the vector \( q_i \) indicates a measure on how well the corresponding factor describes
the $i$th item. Similarly, the $j$th user is associated with a vector $p_j \in R^f$. For the given $j$th user, firstly, the collection of all items associated with $j$th user are found and mapped into the joint space. After that, profiles of all $j$th user’s items with $f$ factors are aggregated to become the projection of $j$th user in the joint space. As the result, the inner product, $p_j \cdot q_i^T$, modeling the user-item interaction can be used to explain the rating of $j$ user to $i$ item, $r_{j,i}$, as shown in the below equation:

$$ r_{i,j} = p_j \cdot q_i^T $$ (2.3)

As the relationship discussed above, the matrix of all ratings, $[r_{j,i}]$ of the dimension (m,n), is the inner product of user profile matrix, $[p_j]$ of the dimension (m,f), and item profile matrix, $[q_i]$ of the dimension (n,f). Where, $f(x) = x^T$ indicates a transpose operation. Or, it can be formulated in the equation listed below:

$$ [r_{i,j}] = [p_j] \cdot [q_i]^T $$ (2.4)

The low-rank approximation, $[r_{j,i}] = [p_j] \cdot [q_i]^T$ can be found by using Singular Value Decomposition (SVD) based on the sum of squared distances [31]. It is a challenge to factorize both the entire set of users and items separately when considering the computational expense. According to the empirical application, the sparsity problem raised due to lacking knowledge on user-item interactions. Some early-stage solution to this issue is to fill the missing interaction with imputed value and make the rating matrix dense. However, the arguments also follow the imputation strategy for its increasing the amount of data for computation or distort the information by inappropriate imputation methods. In contrasts, the more recent methods propose to include only the observed user-item combinations into modeling. And, adopting regularization term to manage over-fitting. Within this regularized learning approach, the factor vectors, $p_j$ and $q_i$, can be found by minimizing the sum of
squared error and penalty term, which is formulated in 2.5.

\[
\min_{p^*,q^*} \sum_{(j,i) \in \Omega} (r_{j,i} - p_j^T \cdot q_i)^2 + \lambda (\|p_j\|^2 + \|q_i\|^2)
\] (2.5)

Where \( \Omega \) is the set of all observed user-item combinations whose ratings are known. And, the magnitude of the constant \( \lambda \) term defines the level of penalization. By fitting to the observed data with regularization, the model is trained to be able to predict ratings of unknown user-item pairs [18]. It is suggested to decompose the relationship between \( r_{j,i} \), the rating of user \( j \) to item \( i \), and their modeled interaction, \( p_j^T q_i \), with the belief that there are multiple resources contributing to the variations in ratings. The relationship is formulated as shown in 2.6, where the rating is a function of its four components: a global rating average \( u \), user bias \( b_j \), item bias \( b_i \) and user-item interaction \( p_j^T \cdot q_i \).

\[
r_{j,i} = u + b_j + b_i + p_j^T \cdot q_i
\] (2.6)

The model learns the relationship by minimizing the error function below:

\[
\min_{p^*,q^*,b^*} \sum_{i,j \in \Omega} (r_{i,j} - u - b_j - b_i - p_j^T q_i)^2 - \lambda (p_j^2 + q_i^2 + b_j^2 + b_i^2)
\] (2.7)

Here, \( \lambda \) is the parameter controlling the magnitude of penalty. Additionally, alternative strategies of modeling the biases had been discussed in [16]. However, to deal with the cold start issue, which is a common problem in building recommendation system, Yehuda Koren explained the mechanism to utilize implicit feedback of users on an item to boost the available information [18]. Besides, by allowing the bias terms to vary over time, the matrix factorization framework can be extended by modeling the temporal effects. Temporal effects are all varieties of changing preference caused by evolutions of users’ tastes in relationship to the updated collection of available items over time. Its detail is introduced in [16]. In real-world applications, not all observed ratings or feedback can be considered with equal confidences. Therefore, a method incorporating the uncertainty and its implementation is
discussed in [13].

2.2.4 Hybrid Methodology

As discussed in the previous sections, there are various approaches to building a recommendations systems. In preceding sections, a subset of them including collaborative filtering, content-based and matrix-factorization are introduced. Advantages and disadvantages of classical approach had been well-discussed [25, 34]. In contrast to methods from different categories, the methods of the same categories commonly suffer same issues and enjoy similar advantages. For examples, by considering peers’ preference, the collaborative filtering method is more than capable of suggesting an item bearing innovative interests to users. Meanwhile, it suffers the cold-start problem for having a difficulty of linking unrated items to users. On a contrary, the cold-start issue had been addressed by content-based recommendation system. Since, without referring to the item’s interactions with peer users, the method bases recommendation on matching the properties of items against the modeled preference of a user. Therefore, one trend in the recommendation systems research community is to design a hybrid system with combined recommendation techniques. In this section, we will introduce some of the variant strategies of building hybrid recommendation systems which are discussed in [7].

1. Weighted: weighted recommendation systems generate a composite score which is a linear combination of all scores provided by all available recommendation modules. The final rating of each item reflects the summarized knowledge of all assembled techniques. The weight itself assigned to a score of a particular recommendation System can be tuned through maximizing the empirical performance. Published in 1997, Marko Balabanovic and Yoav Shoham proposed a hybrid recommendation system named Fab [2]. The system maintains user profiles composed by on content-based methods. And, the content-based user profiles are directly used to determine the similar users of each
user for collaborative filtering. Items can be recommended to users when they match closely both against the user’s profile and members of the similar user group.

2. Switching: a hybrid system employing switching scheme picks one of the recommendations provided by all underlying recommendation system to a user by following pre-programmed strategies. For example, Daily Learner is a hybrid system consisted of content-based and collaborative filtering techniques \[5\]. It employs the switching strategy which prioritizes recommendations of the content-based module over the collaborative filtering module. Only when the content-based module fails to generate suggestions, the item recommended by the collaborative filtering module will be delivered to a user.

3. Mixed: this type of recommendation system delivery recommendations generated from some recommendation system. PTV systems, a TV show recommendation system, showcases the mixed approach which aggregates the suggestions from both content-based and collaborative recommendation system \[36\]. The content-based recommendation system is developed upon textual information of TV programs. And, collaborative filtering recommendation system captures the preference of other users. The final recommendation is a combination of advice from both techniques.

4. Feature Combination: a recommendation system of this kind is a single recommendation system induced with a broad range of features which used to be utilized exclusively for a particular type of systems. As reported by Basu, Hirsh, and Cohen, a collaborative filtering system is developed by learning both ratings from other users and content features describing the recommended objects. The trained recommendation system was reported to achieve significant precision over purely collaborative system \[3\].

5. Cascading: this hybridization framework defines a stream-like process where a rating computed by a previous recommendation system is subject to refinement by its subsequent recommendation sub-systems. As an example, Entree is a restaurant recommen-
dation system which cascades a content-based recommendation system and collaborative system. The content-based techniques compute a rough ranking of restaurants. Then, collaborative filtering will re-adjust the ranking of items for a user based on the preference of similar users.

In summary, a hybrid system can alleviate some issues inherent in systems of a particular type. For example, by augmenting suggestions from content-based technique with collaborative filtering techniques, the hybrid system can advise users on interesting items which differ from user’s past preference. And, the user of distinct preference can be captured by content-based techniques and receive suggestions.

2.3 Distance and Similarity Metrics

The performance of many machine learning algorithms is largely based on the choice of distance metrics. For instance, Nearest Neighbor, K-means clustering, and collaborative filtering are popular algorithms empowering recommendation systems. All of them need a good distance metrics with which to retain the relationships among data points. There is a wide range of distance metrics which had been used in recommendation systems. For instance, Euclidean Distance, Cosine Similarity, Pearson Correlation and Mahalanobis Distance are implemented within various recommendation system and discussed thoroughly. The distance metrics out of the classical collection is not necessary to perform well in all their application context. In general, the distance metrics of this category treats every feature of a subject with equal importance. Unfortunately, not all of features cast equally strong influences into the aggregate difference between subjects in real-world situations. Alternatively, an improvement introduces feature weighting into a traditional distance metric to incorporate practitioner’s understanding the variety of feature importance. However, this manual weighting strategy is subject to human bias. To address the arbitrary style of a decision process, distance metric learning method was introduced by updating weights of selected
features in a supervised fashion. In this section, it starts with the selective introduction on a subset of classical distance metrics. The following section addresses a framework which allows distance metrics evolve by learning pattern revealed by data.

2.3.1 Euclidean Distance

Euclidean distance might be the most fundamental distance metrics for its well-defined geometric meaning. Let assume, $X$ and $Y$, two random instances of $R^p$ with $p$ dimensionality. The Euclidean distance between $X$ and $Y$ is defined in the equation below:

$$D_{euclidean}(X,Y) = \sqrt{\sum_{i=0}^{p} (x_i - y_i)^2} \quad (2.8)$$

Here, $x_i$ and $y_i$ is the $i$th features of $X$ and $Y$, respectively. Euclidean metrics penalizes the larger deviations. A larger value of this metrics implies that those two instances are more different than two instances having a Euclidean distance of a smaller value.

2.3.2 Cosine Similarity

Cosine similarity measures the cosine of an angle formed by two vectors, $X$, and $Y$, which originates from a common point, representations of two instances. The Cosine Similarity spans over the maximum, 1, to the minimum, -1. Cosine Similarity returns one if the formed angle is 0 and two compared instances are considered identical. On the opposite extreme, the same metrics turns a minus one for a pair of completely different instances. The computation of Cosine Similarity of two vectors can be found in the equation:

$$\text{Cosine Similarity} = cos(A, B) = 1 - \frac{A \cdot B}{\|A\| \|B\|} \quad (2.9)$$
2.3.3 Pearson Correlation

Pearson Correlation Coefficient was proposed to measure the similarity of two users with respects to their commonly rated items in collaborative filtering. In that setting, the users’ tastes were implicitly coded by their ratings of items. Pearson Correlation coefficient aims to capture the linear relationship between X and Y. The calculation of Pearson Correlation Coefficient can be found in the equation below:

\[ \rho_{X,Y} = \frac{\text{cov}(X,Y)}{\rho_x \rho_y} \]  
\[ = \frac{\sqrt{\sum_{i=0}^{p}(X_i - \bar{X}_i)(Y_i - \bar{Y}_i)}}{\sqrt{\sum_{i=0}^{p}(X_i - \bar{X}_i)^2} \sqrt{\sum_{i=0}^{p}(Y_i - \bar{Y}_i)^2}} \]  
\[ (2.10) \]
\[ (2.11) \]

2.3.4 Mahalanobis Distance

Mahalanobis Distance (MD) is a multivariate distance metric considering correlations of features associated with objects [8]. The computation of Mahalanobis Distance is based on the inverse of the variance-covariance matrix of data sets comprised by interested instances. The calculation of Mahalanobis Distances starts with finding the variance-covariance matrix as described in 2.12.

\[ C_x = \frac{1}{n} (X - \bar{X})^T \cdot (X - \bar{X}) \]  
\[ (2.12) \]

Where \( X \) is data sets and \( \bar{X} \) is a matrix of its columnar means. As results, \( X - \bar{X} \) is standardized version of the original matrix \( X \). Let assume that \( X \) consists of two columns or variables, the variance-covariance matrix, \( C_x \), can be written in the form shown in 2.13.

\[ X = \begin{bmatrix} \sigma_1^2, & \rho_{1,2} \sigma_1 \sigma_2 \\ \rho_{1,2} \sigma_1 \sigma_2, & \sigma_2^2 \end{bmatrix} \]  
\[ (2.13) \]
Here, $\sigma_1^2$ and $\sigma_2^2$ are the variances of two variables respectively. And, $\rho_{1,2}$ is the correlation coefficient of those two variables. And, the product of $\rho_{1,2}$, $\sigma_1$ and $\sigma_2$ turns out to be the covariance between those two variables. For any two objects, $X_i$, and $X_j$, their Mahalanobis Distance is formulated in (2.14)

$$MD(X_i, X_j) = \sqrt{(X_i - X_j)^T C_x^{-1} (X_i - X_j)}$$ (2.14)

Where, $C_x^{-1}$ is the inverse of variance-covariance matrix, $C_x$, whose computation is shown in (2.15)

$$C_x^{-1} = \begin{bmatrix} \sigma_2^2 / \det(C_x), & -\rho_{1,2} \cdot \sigma_1 \sigma_2 / \det(C_x) \\ \rho_{1,2} \cdot \sigma_1 \sigma_2 / \det(C_x), & \sigma_1^2 / \det(C_x) \end{bmatrix}$$ (2.15)

with $\det(C_x) = \sigma_1 \sigma_2 (1 - \rho_{1,2}^2)$ as the determinant of the variance-covariance matrix, $C_x$. When the variance-covariance is an identity matrix, the Mahalanobis Distance of two objects is equal to their euclidean distance. If the variance-covariance matrix is an diagonal matrix, it means there are no correlation between variables. Therefore, the expression of Mahalanobis Distance of two objects can be written as formula shown in (2.16). Here, $MD(X_i, X_j)$ is equal to normalized euclidean distances.

$$MD(X_i, X_j) = \sqrt{\sum_{i=0}^{p} \frac{(X_i - X_j)^2}{\sigma_i^2}}$$ (2.16)

### 2.4 Evaluation of Recommendation Systems

In this section, we will discuss the issues and properties of a recommendation system. In previous sections, building a recommendation system had been framed as a machine learning challenge, with predictor variables, surrogates for user taste and attributes of items, and a response variable, user attitude toward new items in the form of either rating or like/dislike. Within this framework, an item assigned with the highest rating score would be delivered at
highest priority to users. Therefore, the prediction accuracy of the recommendation system is widely considered as the main concern for practitioner and researcher in this domain. However, some empirical applications revealed that the recommendation system tuned solely on prediction usually end up with unsatisfying user experiences. The strategy of emphasizing the prediction accuracy tends to select a method which recommends items which are very similar to what user had been familiar with. And, the system chosen in this manner is reluctant in recommending novel items and active in recommending familiar items at a higher confidence. In general, a good recommendation system should focus on usefulness by providing useful suggestions rather than accuracy. Recently, there is an emerging agreement on a set of criteria beyond accuracy measurement to assess the recommendation system. In addition to accuracy, there are a wide range of aspects about recommendation system which should be concerned with. In this section, we started to discuss the popular metrics to assess the prediction accuracy of a recommendation system in various contexts. In addition to accuracy, the rest of section is used to cover the introduction of other properties: Coverage, Novelty, Scalability and Robustness.

2.4.1 Accuracy

Rating

Depending on the context of an application, user preference on items collapses into three categories: rating, classification, and ranking. For a numeric rating, Root Mean Squared Error is one of the most popular metrics to evaluate the prediction accuracy. \( r_{u,i} \) is the rating score given by a user \( u \) to the item \( i \). \( \Omega \) represents the test set of observed \( n \) user-item ratings. The following equation defines it:

\[
\text{RMSE} = \sqrt{\frac{1}{n} \sum_{(u,i) \in \Omega} (r_{u,i} - \hat{r}_{u,i})^2} \quad (2.17)
\]

Alternatively, MAE is also a popular choice when measuring the prediction accuracy.
Table 2.1: Confusion Matrix for Classification Task

<table>
<thead>
<tr>
<th>Liked</th>
<th>Recommended</th>
<th>True Positive (TP)</th>
<th>Not Recommended</th>
<th>False Negative (FN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disliked</td>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

And, MAE is defined by the equation listed below:

\[
\text{MAE} = \frac{1}{n} \sum_{(u,i) \in \Omega} |r_{u,i} - \hat{r}_{u,i}|
\]  

(2.18)

In contrast to MAE, which treats errors of different magnitudes equally, RMSE penalizes the larger errors. Average RMSE and Average MAE are suggested to evaluate systems in unbalanced test sets. In an unbalanced test, the overall value of RMSE and MSE could be dictated by prediction errors on a small collection of test items which have the large frequency in the test sets. By aggregating MSEs or MAEs computed collectively per item groups, the prediction accuracy based on this test sets could be better evaluated by Average RMSE and Average MAE. Average RMSE and Average MAE of the test samples can be better generalized to represent the overall accuracy.

**Classification**

In some context, the system is only able to collect implicit feedback from users preference. For instance, Instagram, a popular photo-sharing mobile application, allows users leave ”like”s to express their preference. In this context, the photo bearing likes left by the user can be considered favorable by her. On a contrary, the photo with no likes from the user could be marked as unfavorable. For recommendation systems operating in the similar setting, their recommendation based on predicting the likelihood of an item would be liked by a user. In this sense, this recommendation task can be safely interpreted as a classification problem. Therefore, metrics which had been used in evaluating classifier system can be applied in this assessment. The recommendations per user can be categorized in the manner shown in the below table, with respects to the actual user preference and recommendation status.
Based on the table listed above, the following metrics can be used for evaluating prediction accuracy:

\[
\text{Precision} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}} \tag{2.19}
\]

\[
\text{Recall} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{2.20}
\]

\[
\text{False Positive Rate} = 1 - \text{Specificity} = \frac{\#\text{FP}}{\#\text{FP} + \#\text{TN}} \tag{2.21}
\]

\[
F1 = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} \tag{2.22}
\]

As summarized value of precision and recalls, F1 metrics is a single measurement of prediction accuracy by blending the recall and precision. Receiver Operating Curves (ROC) depicts the varying precision and recalls in response to a different threshold. The size of the area under the curve (AUC) is widely used to measure the prediction accuracy in the similar setting. Based on the different strategy of grouping the prediction, Global ROC (GROC) curve and Customer ROC (CROC), two variants of ROC, had been developed for evaluation recommendation systems of this category.

**Ranking**

It is not uncommon that a recommendation system presenting a list of candidate options from which a user makes choices, rather than a single recommendation. In front of the list, users are expected to search from the top to the bottom. By following the impression order, the likelihood that a user can view an item would decay significantly along the depth of the list. In some applications, it is proposed that the probability at which an item could be regarded follows an exponentially decaying function. Considering this effect, a vertical list positioning higher an item which a user actually likes is superior to the counterpart which lists the same item to the same user at a lower position. Preference estimations determine the placement of recommended items by prioritizing the item of a higher confidence. In applications of this kind, the absolute value of a preference prediction of items is less important than relative
values among them. An accuracy measurement which suits this situation is the Normalized Distance-based Performance Measure (NDPM) which was coined by Yao [42].

The Normalized Distance-based Performance Measure (NDPM) was introduced to measure a distance of a target ranking from a reference ranking in term of their implied pair-wise relationships. Let the reference ranking to be $r_{u,i}$ and the rankings computed by a recommendation system $r_{u,i}$. Specifically, $r_{u,i}$ represents the ranking of $i$th item for the $u$th user. And, $n_u$ is the number of users in the system. The calculation of the following equations is based on iterating over all the combinations of ranked items for a given user.

$$C^+ = \sum_{i,j} \text{sign}(r_{u,i} - r_{u,j}) \cdot \text{sign}({\hat{r}}_{u,i} - r_{u,j})$$ (2.23)

$$C^- = \sum_{i,j} \text{sign}(r_{u,i} - r_{u,j}) \cdot \text{sign}({\hat{r}}_{u,j} - r_{u,i})$$ (2.24)

$$C^u = \sum_{i,j} \text{sign}^2(r_{u,i} - r_{u,j})$$ (2.25)

$$C^s = \sum_{i,j} \text{sign}^2({\hat{r}}_{u,i} - {\hat{r}}_{u,j})$$ (2.26)

$$C^{u_0} = C^u - (C^+ - C^-)$$ (2.27)

$$\text{NDPM} = \frac{C^- + 0.5C^{u_0}}{C^u}$$ (2.28)

Where, $C^+$ is the number of item pairs whose relationships in the reference ranking are same in the predicted relationship. On the other hand, $C^-$ is the number of pairs whose relationships are not agreed by the reference and predicted ranking. $C^u$ is the number of pairs of no ties in the reference ranking. Similarly, $Cs$ is the number of item pairs of no ties in the prediction ranking. $C^{u_0}$ stores the counts of pairs which are ties in the prediction ranking but not in the reference ranking. Hence, NPDM scores 0 if a prediction ranking gives the same pair-wise relationships as the reference ranking indicates. On the other extreme, NPDM returns 1 for a prediction reference reverses all the pairwise relationships given by
the reference ranking.

The NDPM evenly penalizes all incorrect ordering of item pairs at different places in a ranking. However, as the case introduced at the beginning of this section, the penalty for an incorrect order near the end of a list should be discounted due to its low likelihood of casting the influence on users. One popular alternative measure is based on the assumption that the total of a list of ranked recommendations is additively determined by the aggregation of the utilities of individual recommendations. The utility of a single suggestion in the ranked list is defined as a utility of an individual recommendation discounted by its position in the list. For an instance, an item \(i\), its utility is discounted more heavily toward a position near the end of a list. By assuming that the discount rate depending on locations follows an exponential decaying function, the expected utility of a recommendation list for a given user, \(u\) is formulated in \ref{eq:2.29}.

\[
EU_u = \sum_i^N \frac{\max(r_{u,i} - d, 0)}{2^{(i-1)/(\alpha-1)}}
\]  \hspace{1cm} (2.29)

Here, \(r_{u,i}\) is the rank of item \(j\) in the recommendation list for the user \(u\). \(d\), is a default value in this domain. \(N\) denotes the length of list and also the end position of a list. To assess the performance of entire recommendation system, a final score based the user-wise expected utility is calculated by aggregating over all users, \(m\), in the manner defined by the equation shown in Equation 2.4.1.3-3.

\[
EU = 100 \cdot \frac{\sum_{u=1}^m EU_u}{\sum_{u=1}^m EU_u^*}
\]  \hspace{1cm} (2.30)

\(\sum_{u=1}^m mEU_u^*\) denotes the score of best possible ranking for a user \(u\) generated by the system.
2.4.2 Coverage

The coverage of a recommendation system considers the array of items from which a system can yield recommendations. A recommendation system with a limited coverage may be less useful for users for being unable to help users discover some options. Some methods can measure coverage \[12\]. The most common approach is to calculate the percentage of items out of the complete set of choices, after randomly selecting a collection of item-user pairs. In practice, the coverage is better to be measured in conjunction with accuracy measurements. By considering both of metrics, a more useful assessment can be concluded.

2.4.3 Learning Rate

Learning rate is the rate at which a recommendation system can provide "acceptable" recommendations. The underlying machine learning models incorporated into recommendation systems can gradually improve their performance by accumulating data. It is desirable that a recommendation system can generate "acceptable" recommendations at a higher pace. However, the criteria for "acceptable" recommendations is largely vague in the context of applications. For measuring the learning rate, there are very limited discussions in literature. In general, learning rates are non-linear and asymptotic. "Asymptotic" means that learning rate could not improve forever. To compare the learning rates of multiple algorithms, the most commonly used method is to examine the quality of recommendations versus the number of ratings.

2.4.4 Novelty/Serendipity

Novelty and Serendipity are related and distinguished properties. For users, the recommended items which they are not familiar with are novel. Recommending items which users are familiar with can boost user confidence and trust. However, too many familiar items jeopardize the usefulness of the recommendation system. On the contrary, too many sugges-
tions for novel items impedes user’s trust about the quality of recommendations, particular for new users. Therefore, an effective recommendation system should balance the percentage of familiar items and novel items in their recommendation in the context of the applications. The distinction between novelty and serendipity is discussed in [32].

2.4.5 Scalability

Modern recommendation systems are designed to serve a large population of users through an equally large collection of items. In the real-world application, the recommendation system must be able to scale up to handle massive data set. Scalability does not only mean that a recommendation system can be re-trained to adapt to the massive amount of data but also must be responsive enough to respond a large number of users within a second. Since users frequently query a recommendation system. It is expected to give rapid recommendations before a user lost patience. Therefore, it is common practice to developers of recommendation systems would trade the accuracy for the scalability. For instance, according to the content report of the famous Netflix Prize, it is reported by Netflix that one of the prize-winning algorithms had been adopted in the production environment. The two of prize-winning models are all ensembles of many of underlying methods which make them not possible to be induced for production with actual Netflix data. Another example is memory-based recommendation system which is extremely difficult to scale up. Since it requires carrying a large number of neighborhoods to efficiently provide recommendations.

2.4.6 Robustness

The robustness of recommendation systems mainly concerns how stable the recommendations are in response to the injection of fake information. It had become profitable by submitting fake information to manipulate the likelihood of an item’s being recommended. The attempts to influence recommendation system via submitting fake information are labeled as attacks [23]. In reality, it is not practical to build a recommendation system which is immune to the
malicious influence of this kind. Therefore, it is necessary to measure the vulnerability of the system to attacks regarding cost of recommendations’ being manipulated. Unfortunately, this is no theory to give an analytical method to estimate such cost. Lam and Reidel introduced a method to provide an empirical estimation of the cost by launching simulated fake information.
Chapter 3

Methodology

3.1 Introduction to User Recommendation Systems

The majority of user recommendation systems are based on the assumption that users tend to get connected on a social network if they share some common features, e.g. growing up in the same town, having similar education, working in the same industry, have same hobbies, etc. Therefore, the task of suggesting users for friendship is equated to finding users who are similar to a target user. However, people have different preference on what attributes matters to them, when assessing others for the sake of connections. The preference of a user would change based on the context of the social network. Hence, if this were a method which can correctly learn the each user’s personal preference by analyzing a user’s established connections, it would be able to generate a better recommendation for users with their preferences.

In this study, the new method is proposed to build a recommendation system which can deliver an approximate personalization experience. It means that the recommendation system can generate suggestions for users based on their personal social preference. The personal social preference is extracted from a user’s established social connections. The proposed algorithm is based on two assumptions listed below:
• A user is significantly more likely to get connected with a user who is similar to her than a user who is not, particularly from her viewpoint.

• There are an infinite number of distinct preferences which can be extracted from a user population. However, the number of different preferences is smaller than the number of users. And, the frequency distribution of the preferences approximately follows Benford’s law. Therefore, a few number of preference types can account for a majority of users in term of preferences.

The first assumption is the foundation to equate the task of generating a recommendation to the task of finding similar users. The second assumption supports that an effective recommendation system can be developed if there is a method which can find a small collection of most frequent preferences and serves well to the majority of users.

The proposed algorithm is designed to delivery capabilities listed below to be used as the core recommendation machine.

• Find user groupings within which users have the same preference regarding influences of various attributes on their friendship decision;

• Extract user preference for each of groups found and express them as weight vectors. A weight vector consists of a series of elements which indicate the influence of each feature.

3.2 Methodology

3.2.1 Algorithm

Given a set of users, \( U = u_1, u_2, \ldots, u_n \in R^p \), a list of user pairs, \( S \), represents all of the user-pair connections, and a list of user pairs, \( D \), are all pairs of users who are not unconnected. In an implementation, \( D \) is an optional argument. If \( D \) not specified, it should be set to
complement $S$ to represent the entire set of all possible user pairs in $U$. $k$ is the number of
groups which are embedded in the user populations. And, $\alpha$ is the minimum of significant
improve in fit score.

1. Initiate $k$ user groups, $U_1, U_2, \ldots, U_k$, by populating them with randomly selected users.

2. Learn weight vector $w_i$ for every user group with learning distance metrics algorithm
with given set of information, $U_i$, $S_{ui}$ and $D_{ui}$.

3. Remove any users from its current group, $u_i$, and keep them in buffer group if being
detected as inappropriate for this group based on the results of KS-test with the group’s
distance weights, $w_i$.

4. Re-assign every user of the buffer group to a new user group by finding a user group
which provides the highest P-value of their user with its user group distance weights.
If the highest P-value of KS-test is less than predefined minimum threshold, the user
would remain in the buffer group and is available for re-assignment at the next iteration.

5. Calculate the fit score as a quantitative assessment of the overall goodness of learning
results regarding group composition and learned distance weights.

The workflow of executing the algorithm is illustrated in Figure 3.1.

3.2.2 Overview of the Proposed Algorithm for Recommendation
System

The core of proposed method for recommendation system development is group-wise distance
learning (GDL) algorithm which can segment population of users. And, it also simultane-
ously learns a set of unique distance metrics, feature-wise weights, for each of user groups.
As results, the entire population of users is divided into multiple groups. Users from the
same group share the same distance metrics/weights. The distance metrics is an abstract
Figure 3.1: Group-wise Distance Learning (GDL) Algorithm
representation of user’s social preference in term of the importance of each feature or characteristics of users. And for a particular user, a user recommendation system can utilize her/his personalized distance metrics to find suggestions on candidates. A distance metrics of a group is learned by the underlying learning distance metrics algorithm with information considering both user profile and all relationships involving any users of the group \cite{41}.

The algorithm is designed to accomplish two tasks: finding the optimal segmentation of user population and learning a unique distance metrics for each of groups. To accomplish that, the entire learning procedure consists of a number of iterations. Initially, the algorithm considers the data as a homogeneous population and yield a distance metrics for the single group. Then, the initial distance metrics is used to examine its goodness of fit to users’ existing relationships. The existing relationships are defined with the established connections with users, friends, and, missing connections with others, non-friends. If a distance metrics is defined as a good fit to a user, it must be able to separate a user’s friends and non-friends based on their mutual distances calculated with it. After examining the goodness of the distance metrics against their all corresponding users, the users whose relationships are not adequately explained are separated from the others within that group and organized into a new one. At the next iteration, distance metrics learning are conducted separately for each cluster. After the generation of a new set of distance metrics, the goodness examination will be performed again to regroup the users. This process is repeated until a specified stopping criterion is satisfied. The detail of the algorithm is discussed in the following sections. And, how many of groups will be eventually generated is determined a parameter for customization.

### 3.2.3 Introduction to Learning Algorithm

A user, \( u_i \), and is characterized with a set of profile features, \( u_i = [x_1, x_2, ..., x_p] \in \mathbb{R}^p \). The user population is denoted with \( U \). \( S_i \) is the set of users who are friends with \( u_i \). And, \( D_i \) denotes the set of users who are not befriended with \( u_i \). The proposed algorithm comprises
three main components listed below:

1. Learning Distance Metrics, an algorithm can learn a weight vector of Euclidean distance to minimize the ratio of aggregate distances between all connected user pairs to the aggregate distances between all non-connected user pairs. Therefore, weighted Euclidean distance is much more predictive of a user pair’s connection status;

2. Kolmogorov-Smirnov (KS) test-based goodness of fit, this KS test is used to verify if a learned distance is a good encoding of user preference regarding friendship on the social network.

3. A learning framework is a set of iterative guidelines defining the procedure of integrating a series of operations, functions, and logic to realize the objectives of the proposed algorithm.

The subsequent section is organized by above order to discuss the details of those three components.

3.2.4 Distance Learning Algorithm

In recommendation system community, the problem of recommending users to connect had been widely framed as a task to find users who share most common characteristics. A user i’s profile is denoted by \([x_1, x_2, \ldots], \in R^p\), which can be visualized as a data point in feature space. According to visual representation, users who are similar in profiles are clustering within a proximity in this feature space of p-dimensionality. While users were showing little or no common characteristics, they are locating remotely in the space. Therefore, the recommendation task is equated to finding neighbors of the target user in the feature space when trying to generate recommendations. In this sense, a user is more likely to friend with someone who is similar than whom is more different. The overall probability of being friends depends on the overall difference between two users. The overall distance is the aggregating
of the marginal difference in each feature. Marginal contributions of each feature to the overall difference differs concerning user preferences. The variable influence of each feature is caused by the variety of user’s preference or motivations. The established distance metrics, which considers all features equally, could not serve the recommendation task properly with this regard. As being discovered by empirical research studies, it is commonly present that multiple interaction patterns co-exist on a same social network and explain those established relationships in a complementary manner.

To introduce the mechanism how a specialized distance metrics is learned to represent a kind of user preference, we start with discussing the distance learning algorithm which is introduced by Eric Xing in 2003 [41]. The Learning Distance Metrics algorithm was proposed to learn a distance metrics respecting the known neighborhoods in a supervised context.

The classical Distance Metrics usually face the challenge of separating similar objects from dissimilar objects by assuming the equal importance of every feature of objects of interests. The distance metrics learning algorithm is designed to learn a transformation matrix which then be applied to distort the original feature space to adjust their relative influences. Within the transformed feature space, the distances among users of the same class are minimized, and the separations of users of different classes are enhanced. A particular version of this algorithm is to set the transformation matrix to be diagonal, by assuming there is no significant correlation between features. Therefore, the effect of applying the diagonal transformation matrix is equivalent to rescaling each feature differently. Accordingly, values on the diagonal of the transformation matrix represent influences of their corresponding features in term of determining the likelihood of use p|s being connected. This is based on the theory that users having greater similarity are likely to be related. The different similarity/distance metrics encapsulated different user motivations of making connections by varying the importance of each feature. The learned distance metrics is the concatenation
of the learned transformation matrix and the distance metrics of choice.

To automate the learning process, Learning Distance Metrics optimizes the distance function by minimizing the distances between similar objects, if the similar objects and dissimilar objects are known. In the subsequent paragraph, the details of Learning Distance Metrics is laid out.

Let \( u_i, u_j \in R^p \) to be profiles of user \( i \) and \( j \). And, be told with a list of pairs of similar objects: \( S : (u_i, u_j) \in S, \) if \( u_i \) and \( u_j \) are friends. The basic form of distance is defined as the equation listed below:

\[
D(u_i, u_j) = D_A(u_i, u_j) = \| u_i - u_j \|_A = \sqrt{(u_i - u_j)^T A (u_i - u_j)}
\]  

(3.1)

In Equation 3.3.2.2, the expression of the distance function is very similar to the mathematical expression of Mahalanobis Distance. To ensure the distance metrics meet the fundamental properties of distance metrics: non-negativity and the triangle inequality, \( A \) is required to be semi-definite \( A \geq 0 \).

There are three particular states of \( A \) where each relates a general form of distance metrics, \( A \), to a specific type of standard distance metrics. Firstly, if \( A \) is an identity matrix, it is the Euclidean distance. If \( A \) is a diagonal matrix, it turns out to be the weighted version of Euclidean distance. If \( A \) is in more general form with non-zero value on non-diagonal positions, it turns out to be generalized to Mahalanobis Distance. The objective of the learning process is finding a transformation matrix, \( A \), which encode the preference of a specified group of users.

The objective function and restraints for optimization of the learning process are defined in Equation 3.2.
\[
f(A) = \sum_{(u_i, u_j) \in S} \|u_i - u_j\|_A^2 - \sum_{(u_i, u_j) \in D} \|u_i - u_j\|_A \\
\text{s.t. } \sum_{(u_i, u_j) \in D} \|u_i - u_j\|_A \geq 1
\]

Where, \(D\) is set of the pairs of objects known dissimilar if this information is explicitly available. Otherwise, the set should include all pairs of objects which not belong to \(S\).

The transformation matrix, \(A\), can be found by minimizing the value of \(f(A)\). The resulted \(A\) must be a non-trivial solution or non-zeros. The restriction of the objective function is introduced to avoid the space collapsing into a single point. And, the constant of 1 is an arbitrary choice, which can be replaced by any positive numbers. However, the choice of 1 does bring some computational benefits which had been discussed in its initial publication [41]. In the proposed algorithm for computational efficiency, the \(A\) transformation matrix is set to be any of diagonal matrix. Therefore, the mathematical operation of the diagonal transformation and distance metrics calculation is equal to a weighted Euclidean distance. Therefore, the transformation matrix, \(A\), can be represented by a vector of the numbers on the diagonal. In this condition, the general form of transformation matrix \(A\) degraded to a weight vector, \(w\). The simplification ignores feature-feature interaction.

### 3.2.5 Kolmogorov–Smirnov Test: Goodness of Fit

Kolmogorov-Smirnov test (KS test) is a non-parametric hypothesis test comparing the difference of two distributions [34, 35]. In the proposed algorithm, KS test is utilized as a device to validate whether an estimated weight vector for distance calculating is an appropriate encoding of a user’s preference. With a proper weight vector, the distances of a user to her friends are expected to be smaller than counterparts of the user and her non-friends, accord-
ing to the first assumption made in Section 3.1. And, it is anticipated that the distribution of user-to-friends distances is probabilistically larger than the distribution of user-to-non-friends distances. Therefore, a version of KS test with the one-sided hypothesis is used to validate if a given weight vector can enhance the separation between friends and non-friends for a user. If a weight vector amplifies separation, the weight vector is in line with user’s preference. Otherwise, the particular weight vector is not appropriate for the user. The justifications of using KS test in a goodness-of-fit test device are discussed in the latter of this section.

To better understand the effect of an appropriate weight vector, let us create an example social network of a minimal structure. In this example, the analysis is focused on user A. On this social network, user A have two friends, user B and user C. There are another two users, D and E, who are not connected to user A and considered as non-friends for user A. In this example, each user is profiled with six features, \( x_p \in \mathbb{R}^6 \). For user A, only three out of all features cast an influence on her friendship decision on the social network. It means a user is more similar to user A when considering that three important feature only, the more likely to be connected to user A.

Then, profile features can be split into two categories, important features or unimportant features. This categorization is based on feature influence on user’s friendship decision. With regards to possible profile differences between any two users, the situations can be generalized into 4 different types: a) differences are low in both important features and unimportant features; b) differences are low in important features and high in unimportant features; c) difference are high in important features and low in unimportant features; d) difference are high in both important features and unimportant features, as shown in 3.2.

\( w_0 \) is a weight vector which assigns the same weight to every feature. Using \( w_0 \) to calculated other users distances to user A, it is straightforward that the resulting distances must follow this order, \( D_{A,B} < D_{A,C} \) and \( D_{A,D} < D_{A,E} \). In this case, both of the user A’s friend, user C, and her non-friend, user D, have approximately equal distances to her. On
Figure 3.2: Bar Plot of Feature-wise Difference between Users

By the clockwise order, the 1st bar plot illustrates the difference between user A and user B. The height of bar represents the absolute value of difference of the corresponding feature between user A and user B. In sequence, 2nd, 3rd and 4th plots show comparison of user A against user C, user D and user E, respectively.

contrast, $w_a$ is a weight vector which assigns heavier weights only on important features to appropriately reflect the preference of user A. And, the resulting distance calculated with $w_a$ follows this order: $D_{A,B} < D_{A,C} < D_{A,D} < D_{A,E}$. Friends of user A, are clearly separated from her non-friends according to Euclidean distance weighted with $w_a$. In this sense, an inappropriate weight which falsely emphasizes unimportant features has the effect of blurring the separation of friends and non-friends in the space.

For a user, given a weight vector, $w_k$, $D_{w_k}(F)$ is the set of all weighted Euclidean distances of the user and her friends. While $D_{w_k}(NF)$ is the set of all the weighted Euclidean distances between the user and her non-friends. With regards to the previous discussion, if $w_k$ is the appropriate encoding of the user’s preference, the distribution of $D_{w_k}(F)$ are more concentrated on lower value range in comparison to the distribution of $D_{w_k}(NF)$. Since the distances between a user and her friends tend to be smaller than the counterparts of her and
her non-friends. Therefore, KS-test, which tells how different a pair of distributions, $D(w_k)_F$ and $D(w_k)_N F$ are is used to validate whether a weight vector, $w_k$, can explain target user’s established connections.

On the left plot, users are positioned with their mutual distances based on Euclidean distances weighted with $w_0$, an even weight vector. On the right plot, the user is positioned with their mutual distances based on Euclidean distance weighted with $w_a$, an appropriate weight vector.

Kolmogorov-Smirnov test is a non-parametric statistical test which is invented to compare a set of data samples against a specific distribution or against another set of data samples. The test statistics is derived from the difference between cumulative density functions (CDF) or empirical cumulative density function (ECDF), depending on available information and hypothesis. In the context of our study, we are concerned with comparing two sets of data regarding the difference of their underlying distributions. Our discussion is focused on the introduction of comparing two sets of data samples. In this setting, there are three different hypotheses which can be examined with KS-test. Let assume $S$ and $D$ are two sets of data samples. And, $F_S(x)$ and $F_D(x)$ are empirical cumulative density functions for two sample sets, $S$ and $D$, respectively. Regarding hypothesis of interests, the KS tests are divided into three different types which are listed below.
1. Two-sided test:

\[ H_0 : F_S(x) = F_D(x) \quad \text{for all } x \]
\[ H_1 : F_S(x) \neq F_D(x) \quad \text{for at least one } x \]

This is used to test whether two sets of samples are drawn from a same unknown distribution. The test statistics is shown as below.

\[ T = \sup_x | F_S(X) - F_D(x) | \quad (3.5) \]

2. One-sided test:

\[ H_0 : F_S(x) \geq F_D(x) \quad \text{for all } x \]
\[ H_1 : F_S(x) < F_D(x) \quad \text{for at least one } x \]

As an one-sided test, this is used when it is suspected that x values of \( F_S(x) \) tends to be smaller than x values of \( F_D(x) \). In other words, \( F_S(x) \) is probabilistically larger than \( F_D(x) \). The test statistics is shown as below.

\[ T = \sup_x | F_S(X) - F_D(x) | \quad (3.6) \]
3. One-sided test:

\[ H_0 : F_S(x) \leq F_D(x) \quad \text{for all } x \]
\[ H_a : F_S(x) > F_D(x) \quad \text{for at least one } x \]

This test is used whether \( F_S(x) \) is probabilistically smaller than \( F_D(x) \) is of interest. The test statistics is shown as the below formula.

\[ T^+ = \sup_x | F_S(x) - F_D(x) | \quad (3.7) \]

In our study, it has been discussed that being weighted with an appropriate weight vector, \( w \), distances of a user to her friends intends to be smaller than distances to non-friends. Since a good weight vector only factor in important features and ignores unimportant features. As a complementary to similarity, the distance metrics calculated with a good weight vector became an effective proxy for the likelihood that a pair of users gets connected. On other hands, this relationship between two sets of distances could be reversed by using incorrect weight. It means that the distances to friends could tend to be larger than distances to non-friends. This is caused by falsely putting more weights on unimportant features, meanwhile overlooking the influence of important features. Additionally, for some users, the equal weight vector can clear separate user’s friends and non-friends based on their distances to the target user. In this sense, it is more important to identify an inappropriate weight vector than to confirm that a weight vector is appropriate.

In our study, the KS-test is used to detect whether a weight vector is not a good fit for a user’s existing relationships. The null hypothesis is that distances between a user and her friends are probabilistically no less than the distances between the user and her non-friends. Therefore, the hypothesis and test statistics can be formulated as below.
In the context of this study, the KS-test is used to detect whether a weight vector is not a good fit for a user’s existing relationships. The null hypothesis is that distances between a user and her friends are probabilistically no less than the distances between the user and her non-friends. Therefore, the hypothesis and test statistics can be formulated as below.

\[ H_0 : F_S(x) \geq F_D(x), \quad \text{for all } x \]

\[ H_a : F_S(x) < F_D(x), \quad \text{for at least one } x \]

Test Statistics: \[ T = \sup_x |F_D(x) - F_S(x)| \]

\( F_S(x) \) and \( F_D(x) \) are Empirical Cumulative Density Function for distances to friends and non-friends, respectively. In this sense, if P-value of KS-test for a user, \( u_i \), and a weight vector, \( w_j \) is smaller than a pre-defined threshold \( \alpha \), \( H_a \) the alternative hypothesis will be accepted. Therefore, it suggests the conclusion that \( w_j \) is not a good weight vector to reflect \( u_i \)'s relationship preference. Otherwise, the null hypothesis would hold. And, the weight
vector, $w_0$ is considered as a good one to encode the preference of the user, $u_i$.

### 3.2.6 Fit Score

The process of finding optimal groupings and associated weight vectors is iterative. The overall learning process is introduced in section [3.2.1](#). As being detailed previously, the process starts with learning an initial weight vector by analyzing the entire set of user profiles and their connection structures. Following the initial learning for weight vectors, it is the step of splitting users into different groups based on the results of KS-test. The objective of KS-test is to detect whether a user’s existing connections can be explained by a weight vector. Consequently, users whose tastes are represented properly by the weight vector can be grouped together. The next step is to learn weight vectors for every of new user groups, which are considered to be more homogeneous than the previous groups regarding user preference. Sequentially, the newly learned weight vector is used to re-examine their goodness of fit to each member of a group. If null hypothesis of KS-test is rejected for a user, the user will be re-assigned into a different group. Otherwise, the user would remain in the original group. This weight vector learning and member shuffling keep repeating in the same order until a certain stopping criterion being satisfied.

The stopping criteria are defined as the point where the additional iteration could not significantly improve the overall quality of groupings and associated weight vectors. To measure the overall quality of learning results, a metric called Fit Score is proposed and detailed in [3.8](#).

$$\text{Fit Score} = \frac{\sum_{j=1}^{k} \sum_{i \in U_j} \text{P-value}(u_i, w_j)}{\sum_{j=1}^{k} \|U_j\|} - C \cdot \frac{\|U_{\text{buffer}}\|}{\|U_{\text{buffer}}\| + \sum_{j=1}^{k} \|U_j\|}$$

(3.8)

Where, $k$ is the number of user groups. $U_j$ is the set of users in the $j$th group. $u_i$ is user $i$, who is in $U_j$. $w_j$ is the weight vector associated with the user group, $U_j$. $|U_j|$ is the number of users in $U_j$. $n$ is the total of users provided. And, P-value($u_i, w_j$) is the P-value.
of the proposed KS-test conducted for $u_i$ with the weight vector $w_j$. $U_{\text{buffer}}$ is the collection of users who are not considered suitable for any of user groups with respect to results of KS-test. $C$ is the term which governs the strength of penalty. Therefore, the fit score favors a higher average P-value over user groups, while being subjects to regularization of the size of buffer group.

A larger value of Fit Score indicates that overall quality of user groups and their weight vectors are more favorable. Since considering the proposed configuration of KS-test for goodness-of-fit, a larger P-value implied a less likelihood of rejecting the null hypothesis. The null hypothesis is supporting evidence for that $w_j$ is an appropriate encoding of $u_i$'s preference. According to the proposed definition of Fit Score, a learning process resulting with a larger size group is more favorable by incorporating the squared value of group size into metrics.
Chapter 4

Module Test

4.1 Validation Test of Algorithm Modules

The focus of this section is on validating the proposed algorithm with regards to its capability to accomplish its proposed tasks: a) learn distance weights to represent user preference, b) test the goodness of a learned weight vector with respect to separate user’s friend to non-friends, and c) cluster users of same preference. The validation is based on the application of the proposed algorithm to synthetic data sets of known patterns. The proposed algorithm is implemented in Python and open-sourced with an appropriate license for future modification and distribution. This attempt is based on three intentions. At the first, it facilitates the broader range of applications of this algorithm used by peer academic researcher and industrial practitioners. And, at the second, it promotes the re-use of the code to validate the results presented in this dissertation and build an innovative algorithm upon this implementation; The third is to allow further improvement made by more experienced developers who is interested in this algorithm by making the algorithm more efficient in computation.
4.1.1 Introduction to Synthetic Data Sets

In this study, there are two distinct synthetic data sets are used at the implementation and concept validation stage. All of those two synthetic data sets are created in the context of a social network. It means that each of data set includes user profile information and user-user relationships.

The data set \( A \) includes the total of 100 users who has seven features in profile. Out of seven features, only three of them are influential on user’s friendship intention. There is only one group of users having a definitive social preference in the network. Since 80 out of 100 users are known to be subjective to the collection of three influential features. Their friendship is based on distance only accounting for those three influential features. In contrast, the another 20 users have no pattern to follow when determining intention for friendships. Their user connections are randomly generated. The characteristics of the simulation data set can be found in Table 4.1.

The data-set \( B \) includes two original user groups. The entire data set includes 100 users. Each of users has six features to characterize the profile. For the 1st user group of 70 users, they follow a rule that the first three feature are relevant to user friendship decision. And, its preference can be coded with a weight vector \([0.5, 0.4, 0.1, 0, 0, 0]\). On the other hand, the second group has 40 users. Their tastes is coded with a weight vector, \([0, 0, 0, 0.3, 0.2, 0.2]\). Connections between users in this simulated social network are generated according to the stochastic process defined as rules listed below:

- a pair of users are connected if both of involved users were interested in the formation of friendship.
- the probability that a user \( i \) is interested in connecting with a user \( j \) is determined by \( D_{u_i}(u_i, u_j) \), which is the distance between \( u_i \) and \( u_j \) with accordance to \( u_i \)’s weights.
The decision rule is described in Equation 4.1

\[
P(u_i \text{ is interested in } u_j | D_{u_i}(u_i, u_j) \leq 0.3) = 0.6 \quad (4.1)
\]

\[
P(u_i \text{ is interested in } u_j | D_{u_i}(u_i, u_j) > 0.3) = 0.05 \quad (4.2)
\]

The basic information about those two synthetic data sets can be found in Table 4.1.

### 4.1.2 Module Test I: Learning Correct Weights

The purpose of this experiment is to assess the learning distance metrics algorithm in various scenarios. The experiment is designed to address the following two aspects of this algorithm: a) the relationships of the amount of provided information and the accuracy of the learned weight vector; b) the performance of the learning distance algorithm when being supplied with a mixture of users.

In this experiment, cosine distance is used to measure the differences between the correct weight vector, which is used in generating the simulation data, and the weight vectors, learned by distance learning algorithm. The mathematical definition of cosine distance is introduced in 4.3. The cosine distance ranges from 0 to 1. The largest possible value, 1, implies that the two measured vector are very different. In space, those two vectors are orthogonal. On the other hand, the smallest value, 0, indicates that two vectors are exactly same in term of direction in space. In this situation, one vector can be transformed to an other by multiplying with a scalar.

\[
\text{Cosine Distance} = 1 - \cos(A, B) = 1 - \frac{A \cdot B}{\|A\|\|B\|} \quad (4.3)
\]

<table>
<thead>
<tr>
<th>Index</th>
<th>No. of Users</th>
<th>No. of Features</th>
<th>No. of Groups</th>
<th>Group Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>7</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>B</td>
<td>100</td>
<td>6</td>
<td>2</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 4.1: Basic Properties of Synthetic Data Sets
According to the objectives of this experiment, there are two factors introduced to vary in various tests. They are the number of users provided to the distance learning algorithm and the percentage of target users out of the provided users. The target users are users whose preference are correctly represented by the baseline weight vector. The test settings are varying with respects to different combinations of values of those two factors of interests. The possible values of the first factor, the number of given users, are among either of 20, 30, 50 or 60. And, the value range of the second factor, the percentage of target users, is from 0% to 100% at an increment of 10%. The test is repeated ten times for each of a particular setting. And, the synthetic data set B, which includes 100 distinct users, introduced in section 3-1 is utilized for all the tests. During each of tests, a subset of users is randomly selected from the data set B to meet the specified experiment setting.

The results of the entire experiment is illustrated in 4.1. The y-axis represents the cosine-distance between the learned weight vector and the baseline weight vector. The x-axis represents the second factor, the percentage of target users out of all provided users. Each of test results is illustrated as a dot in the figure. Those dots are color-coded by the number of provided users. For each of user sizes in the experiment, a set of trend lines is formed by linking through the mean of 10 repeated tests at each given target user percentage. Assume the test results follow the normal distribution. The upper and lower bands of trend lines are created to represent 95% Confidence Interval of values of cosine distance for given distances. The current calculation of confidence interval ignores the boundaries of possible values of cosine distance.

Based on the experiment results shown in 4.1, the learning distance algorithm yield weight vectors which are very different from the baseline weight vector when the percentage of the target user is 0%. As the percentage of target users increases, values of cosine distance keep declines regardless of the total number of users provided. When the percentage is 100%, the learning algorithm can extract the correct weight vector out of the provided information. By comparing the four trend lines which represent test groups of different user sizes, it is
found that algorithm performs worse when the user size is smaller. Particularly, the test results of user size of 20 are significant all counterparts of the another three groups. But, we also notice that there are no obvious differences among the mean value of cosine distance for another user sizes.

Therefore, we can conclude that the learning distance algorithm can learn the correct weight to represent the social preference of provided users. The learning distance algorithm is sensitive to the homogeneity of users in term of social preference.

4.1.3 Module Test II: KS-test Goodness of Fit

The proposed algorithm incorporates a one-sided version of KS-test as the device to quantify the goodness of a weight vector in term of representing a user’s social connection preference. We should validate the actual performance of this key component before conducting any further developments based upon it. With those regards, an experiment is designed to
answer the following two questions: a) whether KS-test can effectively determine that a weight vector is appropriate to explain a user’s established social connections; b) understand the relationships between the performance of KS-test and varying dimensions of user profiles.

In this particular study, it sets primary to examine the distributions of P-value of the specified KS-test with different types of weight vectors. By adding the factor of the variable number of dimension of the user profile, the experiment is conducted four times at various scenarios: dimension=3, dimension=6, dimension=10, and dimension=20. Under each scenario, similar tests are repeated 100 times to retrieve more reliable results. Each single test consists of three major steps which are described as below:

1. generate 100 users being profiled with a specified number of dimensions. Each of those profile features follows a uniform distribution with a fixed range. The profiles of each user are randomly and independently created.

2. generate the network relationships among users. The same strategy used for creating the synthetic data set B is adopted. For each experiment scenario, a “true” weights vector are defined before creating users’ relationships. And, the same “true” weight vector is applied to the repeated tests in the same scenario.

3. calculate P-values of KS-test for every user with three different weight vectors: The ”true” weight vector, which is used in step 02 to create users’ relationships. And, the equal weight vector, which is full of equal numbers. The ”wrong” weight vector, which is a series of an arbitrarily chosen numbers in order to emphasize the features having lighter weights and ignore the features having heavier weights in ”true” weight vector.

Therefore, each of test is to calculate P-value of KS-test for a particular user within a randomly generated social network, which consists of 100 users. Any conclusion on a particular scenario is based on the results of those 100 independent tests.

Separate the inappropriate weight vector from both of equal weight vector and the correct weight vector. As shown in red, the density distributions of P-value of KS-test spread
Figure 4.2: Density Distribution Plots of P-values of KS-tests.
Red curves are the densities of P-value of KS-test with wrong weight vector; Green curves are the densities of P-value of KS-test with equal weight vector; Blue curves are the densities of P-value of KS-test with correct weight vector, which is the weight vector used in generating the users’ social connections. The top-left plot is for the scenario when user profile dimension=3; The top-right plot is for the scenario where user profile dimension=6; The bottom-left is the scenario where user profile dimension=10; And, the bottom-right is the scenario where user profile dimension=20.
widely and tends to have higher densities in the range of lower values. It aligns with the expected results that the KS-test result is more likely to reject the null hypothesis given an inappropriate weight vector. On the contrary, the counterparts of equal weight vectors and correct weight vectors are concentrated on a narrower range of higher values. Therefore, with either of a proper weight vector or an equal weight vector, the null hypothesis of the KS-test is more likely to hold than being rejected. When comparing the difference in distributions of equal weight vectors and correct weight vectors, the P-value’s distribution of equal weight vector has a more negatively skewed tail than the proper weight vector’s. The difference is merely noticeable when user profile dimension is small. It becomes more profound as the profile dimension grows. Moreover, as the user profile dimension grows large, the overlap area between P-value distributions of inappropriate weight vector and the counterparts of equal weight vector or correct weight vector becomes enlarged. Accordingly, it is more challenging to find a threshold to separate one clearly from the other.

According to the experiment introduced above, the proposed KS-test can tell whether a weight vector is an appropriate encoding of user preference. And, the effectiveness of KS-test will increase as the user profile dimension grows.

4.1.4 Module Test III: Application of Group-wise Distance Learning

In this section, the implementation of Group-wise Distance Learning (GDL) is applied to simulation data set B. The primary objective of this experiment is to collect information to characterize the learning process. To characterize the learning process, the sequence of fit score per iteration had been recorded. The simulation data set B’s detail can be found in 4.1.1.1

The progress of learning process is visualized in Figure 4.3. In this study, GDL algorithm was configured with following parameters: the penalty term \((C)\) is 0.5, the number of groups \((K)\) is 2 and update interval of threshold \((\alpha)\) is every ten iteration. According to Equation
3.8. GDL has a tendency of choosing user groupings which are accompanied with fewer users in buffer group, if given a stronger penalty terms. $K$ determined that the outcome of GDL includes two user groups having dedicated weights for distance calculation and a buffer group whose users are not associated with specific weights. Considering the impact of the interval for updating threshold in KS-test, the longer period improves the likelihood that GDL can find optimal user groups and their associated distance weights. In the meantime, the longer period also lengthens the procedure to the ultimate optimal user groupings.

According to Figure 4.3, the overall trend of learning curve indicates that fit score, which reflects the quality of learned results, increases over iterations at the initial phase. It is followed by the phase wherein the GDL is settling by finding a relative optimal to maximize the fit-score. Specifically, during the phase, the momentum of an improving fit scores halts and the fit score fluctuates within a small range. After a certain number of iterations in the settling phase, GDL stops and output the user groupings and their group-wise distance metrics at the iteration which fit score is the maximum of the learning process. The quality of the outcome of GDL is visited in the context of applying its results in user recommendation settings 5.3.
Chapter 5

Simulation Experiment

5.1 Simulation Experiment Introduction

In this section, we introduce the design of simulation experimentation framework which is used to evaluate user recommendation systems. In the previous section, the mechanism of building a user recommendation system with GDL algorithm had been introduced. The main objective of the simulation study is to compare a user recommendation system empowered by GDL algorithm with the counterpart empowered by Nearest Neighbour (NN) algorithm. A user recommendation system is designed in principal to facilitate a user’s exploration in unknown social space. To promote the growth of social relationships, the core value of a user recommendation system is to present unknown users to a target user and lead to new relationships between them. Therefore, we designed a simulation experiment framework to capture the influence of using a social network on the evolution of a social network.

The simulation study serves as the primary means of evaluation of a user recommendation system. The effectiveness of a user recommendation system is considered with respects to its ability to promote the development of a social network. With interaction between users and its user recommendation system, the social network will keep growing continuously until a certain point where no more new relationships can be formed. In the real-life situation,
a user would only get connected with interesting users if he/she given the opportunity to know the existence of interesting ones. To emulate the user’s behavior, we decided that, regardless of positions in the recommendation list, a target user will accept all the users who are truly interesting to him or her. Therefore, even in the theoretical scenario where every user pairs have the chance to be introduced to the counterpart, not every of pairwise connections can be bridged due to users’ social preference. So, in our experiment framework, given a social network of a set of established user connections as reference data, a simulation study is initiated with all users profile data and a subset of known user connections. This initial status represents the starting status of a social network in the simulation. At every iteration of a simulation study, each user in the social network queries the deployed user recommendation system once. Given the list of recommended users for new connections, the querying users decide whether or not to form a new connection according to the reference data. Every candidate user can be only recommended one time to the same user. Along the process of a simulation, a simulated social network will be compared against the reference social network to determine a simulation had reached the stable state.

Based on this simulation framework, a more effective user recommendation system is the one who helps the simulated social network reach the stable status after fewer iterations. The stable status is reached when all known user connections had been recovered with the assistance of user recommendation system. The simulation flow is illustrated in Figure 5.1.

The detailed description of work-flow of the simulation experiment is below:

1. An experiment is configured with a fixed user population and an initial set of user connections. The profiles of all users are known. The initial set of user connections represents the initial state of a social network.

2. Each of users is programmed to query the user recommendation system to retrieve suggestions for users to form new connections. A user reacts to the list of suggestions by following a Zipf distribution. Therefore, the probability of a recommended user to be accepted is determined by its ranking in a recommendation query. Hence, an
Figure 5.1: Simulation Experiment Workflow
effective user recommendation system is expected to be consistently able to suggest interesting peers and rank them in a way which most satisfy a user’s interest.

3. The social network is updated, after step 2, and compared to the known state of the social network, which has consisted of the reference user connections. In general, a superior user recommendation system is the one who can recover a higher percentage of text user connections. With this regards, the common edge ratio is coined and calculating to compare the similarity between the simulated social network and the unknown state of the social network represented by the reference set of user connections.

4. It repeats both of step 02 and steps 03 in sequence for a definitiveumber of times. The evaluation results collected over the simulation iterations is analyzed to provides insights on a tested user recommendation system with respects to its effects on facilitating social connections among a fixed user population.

Specifically, in this simulation study, three recommendation systems were compared. They are: 1) nearest neighbor-based user recommendation system with generic Euclidean distance; 2) GDL-empowered user recommendation system, which labelled as ”GDL once” since it learns group-wise distances only at beginning of the simulation; and 3) GDL-empowered user recommendation, which is marked as ”GDL frequent” since it keeps updating group-wise distance per each iteration of a simulation. For the GDL-empowered algorithm, both of system were configured to cluster users into two sets.

Regarding initialization setting, to get more general results, a set similar simulation was conducted several times with various initialization setting with respects to initial user connections. Each of recommendation systems would be deployed in simulation environments which were initialized with user connection subset of different sizes. These subsets of user connections were sampled and account for 30%, 40% and 50% of true user connections, respectively.

To evaluate a performance of various user recommendation systems in comparison, there
are two performance metrics which could be considered. First, Common Edge Ratio (CER) measure the similarity between the recovered user social network of the simulation experiment and the known user connections. Its definition is formulated in Equation 5.1.

\[
\text{Common Edge Ratio(CER)} = \frac{S \cap T}{S \cup T}
\]  \hspace{1cm} (5.1)

\(S\) denotes the set of social connections of a simulated social network. \(T\) denotes the set of social connections of the true social network. Secondly, the number of iterations taken by user recommendation system to recover the full set of known user connections in the simulation environment. A recommendation system which took fewer iterations to accomplish that is considered more effective. In the real-life application, this recommendation system arguably can be more efficient in expanding social network. In the context of this simulation study, a recommendation is considered a superior method if it either achieve a higher maximum similarity or reaching a certain level of similarity with the true social network at a faster pace. This superior performance can be achieved by that true interesting users were consistently suggested by the recommendation system at a higher probability.

5.2 Experiment 01: Simulation Study with Synthetic Social Network 01

5.2.1 Overview of Experiment Data Set

A simulation experiment of this study needs a social network data with user profiles and established user connections. In this particular experiment, the data set was fabricated through random generation by following a set of pre-defined rules. This data set includes 100 users and 1,494 user connections. There are two user communities developed among these 100 users. For one group, user connections of these group members are randomly
created according to closest distances with users’ social preference. The social preference for this particular group can be encoded as a weight vector, \([0, 1, 0, 1, 0, 1, 0]\) corresponding the influences of 7 user profile features: Gender, Age, Region, Education, Income, Hobby and Duration. On the other hand, the members of the other group whose connections were generated randomly. The detail information about this synthetic data set can be found in section 4.1.1.

5.2.2 Experiment Configuration

As introduced in above section, the purpose of the simulation experiment is to evaluate the proposed method in comparison to the widely adopted Nearest Neighbor algorithm. According to the specifics of experiment design, an algorithm which can recover full known user connections with a fewer iteration has a higher effectiveness as a user recommendation system.

To examine the possible variety of performance of user recommendation systems on different initialization, four sets of simulation experiments had been conducted by triggering the simulation studies with four different sample rate: 0.20, 0.30, 0.40 and 0.50. For example, for sample rate is 0.20, it means that 299 user connections were given as start point of a simulation.

The candidate user recommendations in this comparative simulation study are listed below. To generalize the results of this simulation study, the simulation is repeated 30 times with randomly re-sampled user connections applied to each of candidate system.

- **Group-wise Distance Learning (once)**: the GDL algorithm as the core of the user recommendation system would only learn user’s social group once at the beginning of simulation study. The learned weights and member groupings are utilized for generating suggestions at every iteration.

- **Group-wise Distance Learning (frequent)**: in contrast to the previous version of
GDL, this GDL algorithm will re-learn group-wise social preference and user groupings with updated user connections as a whole during the process of the simulation study.

- **Nearest Neighbour Algorithm:** user recommendation system empowered by NN algorithm will find candidates who have smallest distances to a querying user. NN algorithm would only use the user profile information for recommendation.

### 5.2.3 Results

In Figure 5.2, each line represents a single simulation run, with x-axis marks the iteration and y-axis, indicate CER measuring recovered user connections against the full set of known user connections. Lines are colored by the user recommendation systems. Blue is experiment results based on NN algorithm. Green is for GDL-once and Red for GDL-frequent. The CER line increases monotonically as the simulation progresses. It is because of that a simulated user would only accept new suggestions for users who were connected to the full set of data. At each iteration, a various number of new connections were formed. In Figure 5.3, the average number of new connections per iteration is illustrated. In Figure 5.4, the number of new connections per iteration of each experiment run were created to visualize the dynamics of the simulation process.

In addition to visualized experiment results summary, a table which describes the median CER per iteration is shown in Table 5.1.

And, the standard deviation of CER per iteration is shown in Table 5.2.

### 5.2.4 Conclusion

Based on the experiment results, the both version GDL-empowered user recommendation system performs better than the counterpart built with NN algorithm. Among GDL-once vs. GDL-frequent, there is no significant difference on their performance. This is because of that the relatively simple structure of user’s social preference can be accurately learned.
Figure 5.2: Common Edge Ratio (CER) Plot (Simulation Experiment 01) with a few established user connections.
Figure 5.3: Average No. of New User Connections per Iteration (Simulation Experiment 01)

Figure 5.4: No. of New User Connections per Iteration (Simulation Experiment 01)
<table>
<thead>
<tr>
<th>Iteration</th>
<th>GDL (Frequent)</th>
<th>GDL (Once)</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>1</td>
<td>0.39</td>
<td>0.37</td>
<td>0.28</td>
</tr>
<tr>
<td>2</td>
<td>0.49</td>
<td>0.48</td>
<td>0.36</td>
</tr>
<tr>
<td>3</td>
<td>0.56</td>
<td>0.56</td>
<td>0.43</td>
</tr>
<tr>
<td>4</td>
<td>0.62</td>
<td>0.62</td>
<td>0.49</td>
</tr>
<tr>
<td>5</td>
<td>0.67</td>
<td>0.66</td>
<td>0.56</td>
</tr>
<tr>
<td>6</td>
<td>0.72</td>
<td>0.71</td>
<td>0.63</td>
</tr>
<tr>
<td>7</td>
<td>0.77</td>
<td>0.75</td>
<td>0.70</td>
</tr>
<tr>
<td>8</td>
<td>0.81</td>
<td>0.79</td>
<td>0.75</td>
</tr>
<tr>
<td>9</td>
<td>0.85</td>
<td>0.84</td>
<td>0.80</td>
</tr>
<tr>
<td>10</td>
<td>0.88</td>
<td>0.87</td>
<td>0.84</td>
</tr>
<tr>
<td>11</td>
<td>0.91</td>
<td>0.91</td>
<td>0.88</td>
</tr>
<tr>
<td>12</td>
<td>0.94</td>
<td>0.94</td>
<td>0.91</td>
</tr>
<tr>
<td>13</td>
<td>0.96</td>
<td>0.96</td>
<td>0.94</td>
</tr>
<tr>
<td>14</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td>15</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>16</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>17</td>
<td>1.00</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>18</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>19</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>20</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 5.1: Median of CER per Iteration (Sample Rate = 0.2)
<table>
<thead>
<tr>
<th>Iteration</th>
<th>GDL (Frequent)</th>
<th>GDL (Once)</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1</td>
<td>0.01</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>0.02</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>0.03</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>0.03</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>0.03</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>0.03</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>0.03</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>0.03</td>
<td>0.04</td>
<td>0.00</td>
</tr>
<tr>
<td>9</td>
<td>0.02</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>0.02</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>11</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>12</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>13</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>14</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>16</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>17</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>18</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>19</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 5.2: Standard Deviation of CER per Iteration (Sample Rate = 0.2)
5.3 Experiment 02: Simulation Study with Synthetic Social Network 02

5.3.1 Overview of Experiment Data Set

This simulation study is conducted with a well-understood simulation data set formed in a controlled environment to facilitate the investigation on how GDL-empowered recommendations system works. The simulation data set includes two random generated groups which were associated with distinct distance metrics. And, the social connections of members from either of groups were fabricated with randomness. In this sense, a pair of users is connected to the bilateral agreement. It means that a connection can only be formed if users on both ends of the connection express interests. Whether a user likes to connect to a peer user is determined by their mutual distances. The shorter distance leads to a higher probability. The detail on how the simulation data set was generated can be found in section 4.1.1.

5.3.2 Experiment Configuration

As introduced in above section, the purpose of the simulation experiment is to evaluate the proposed method about the widely adopted Nearest Neighbor algorithm. According to the specifics of experiment design, an algorithm which can recover full known user connections with a fewer iteration has a higher effectiveness as a user recommendation system.

To examine the possible variety of performance of user recommendation systems on different initialization, four sets of simulation experiments had been conducted by triggering the simulation studies with four different sample rate: 0.20, 0.30, 0.40 and 0.50. For example, for sample rate is 0.20, it means that 299 user connections were given as start point of a simulation.

The candidate user recommendations in this comparative simulation study are listed below. To generalize the results of this simulation study, simulation is repeated 30 times
with randomly re-sampled user connections applied to each of candidate system.

- **Group-wise Distance Learning (once):** the GDL algorithm as a core of the user recommendation system would only learn user’s social group once at the beginning of simulation study. The learned weights and member groupings are utilized for generating suggestions at every iteration.

- **Group-wise Distance Learning (frequent):** in contrast to the previous version of GDL, this GDL algorithm will re-learn group-wise social preference and user groupings with updated user connections as a whole during the process of the simulation study.

- **Nearest Neighbour Algorithm:** user recommendation system empowered by NN algorithm will find candidates who have smallest distances to a querying user. NN algorithm would only use the user profile information for a recommendation.

### 5.3.3 Results

In Figure 5.5, it illustrates the similarity between simulated social networks at each iteration of simulation vs. ‘true’ social network. Within the figure, there are three sub-plots which represent simulations of different initial user connection size: 30%, 40% and 50% of known user connections respectively. The same set of random initial user connections were applied to all of the recommendation system candidates to make experiment results more comparable. All of the simulations, the recommendation size is five. It means that a recommendation system generates five suggestions per user’s request.

In addition to visualized experiment results summary, a table which describes the median CER per iteration is shown in Table 5.3. And, the standard deviation of CER per iteration is shown in Table 5.4.

GDL-empowered recommendation systems outperform NN-empowered recommendation system with respects to the number of iterations were required to reach a same level of CER. Specifically, according to data shown in Table 5.3, the GDL-empowered method had
recorded maximum margin of 0.05 CER in comparison to NN-empowered at some iteration. It is noted that the advantage of the GDL-empowered system is diminishing after that overall CER reached a high level. Hence, it almost took the same number of iterations for all of those methods to recover all known user connections and reported 1.00 CER.

In Table 5.4, GDL-empowered recommendation system vary wider than NN-empowered recommendation systems. Simulation on NN-empowered recommendation system states that standard deviation of CER are all 0.00. According to the setting of simulation experiment, there are two resources for variation in results. The first is the randomness introduced by user behaviors. The second is due to randomness derived from the information of train data set by the underlying algorithms. Considering NN, it utilizes a single distance metrics to serve all of users. Consequently, there is very minimal variance in its performance. On contrast, the GDL family reported a larger variance in its performance. It is because fo that the larger variance also indicates a greater flexibility of the algorithm. Fortunately, this undesired
Figure 5.6: Average No. of New User Connections per Iteration (Simulation Experiment 02)

variance can be mitigated by choosing suitable parameters (e.g. the number of groups, strength of penalty and etc.) to achieve a balance between variance and generalization.

Comparing two variants of the GDL-empowered recommendation system, recommendation systems empowered by GDL-once can achieve slightly higher similarity score than the counterparts empowered by GDL-frequent version. It seems to contradict the theory that GDL-frequent version has a better utilization of social network structure information, which facilitates frequent learning the evolving social network. However, due to the relatively simple pattern in the simulation data. And, an initial data set which includes 30% of known connections suffices to allow GDL-once to learn a reliable user preference. Therefore, this feature of GDL-frequent did not translate to be steady superior performance. On contrast, it reveals a wider variation in CER than all another two methods. However, the anticipated benefits will be tested thoroughly in subsequent studies.

In summary, GDL-empowered user recommendation systems are superior alternative
methods to NN-empowered counterparts regarding its flexibility to capture multiple social preferences. However, the greater flexibility of modeling is associated with its price. To build a more robust user recommendation system, the process of choosing optimal parameters for GDL algorithm is needed to mitigate the unnecessary variance.

5.3.4 Conclusion

Based on the simulation study with simulated data which have two inherent groups, recommendation system empowered by GDL algorithm outperform the counterparts empowered by Nearest Neighbor algorithm. It is proven that the GDL-empowered user recommendation system is a possible superior alternative to Nearest Neighbor-empowered as an algorithm to build a user recommendation system regarding the greater flexibility. The user recommendation system empowered GDL-frequent reported the best performance among these three
<table>
<thead>
<tr>
<th>Iteration</th>
<th>GDL (Frequent)</th>
<th>GDL (Once)</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
</tr>
<tr>
<td>1</td>
<td>0.54</td>
<td>0.54</td>
<td>0.53</td>
</tr>
<tr>
<td>2</td>
<td>0.62</td>
<td>0.63</td>
<td>0.61</td>
</tr>
<tr>
<td>3</td>
<td>0.70</td>
<td>0.71</td>
<td>0.68</td>
</tr>
<tr>
<td>4</td>
<td>0.77</td>
<td>0.77</td>
<td>0.73</td>
</tr>
<tr>
<td>5</td>
<td>0.82</td>
<td>0.83</td>
<td>0.78</td>
</tr>
<tr>
<td>6</td>
<td>0.87</td>
<td>0.88</td>
<td>0.83</td>
</tr>
<tr>
<td>7</td>
<td>0.90</td>
<td>0.91</td>
<td>0.86</td>
</tr>
<tr>
<td>8</td>
<td>0.93</td>
<td>0.94</td>
<td>0.89</td>
</tr>
<tr>
<td>9</td>
<td>0.95</td>
<td>0.95</td>
<td>0.92</td>
</tr>
<tr>
<td>10</td>
<td>0.97</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>11</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
</tr>
<tr>
<td>12</td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>13</td>
<td>0.99</td>
<td>1.00</td>
<td>0.99</td>
</tr>
<tr>
<td>14</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>15</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 5.3: Median of CER per Iteration (Sample Rate = 0.3)

candidates based on the observation that its curves are above others. This is because that, by triggering GDL algorithm to learn new social network per iteration of simulation, user recommendation system of this type can be adaptive to an ever-evolving social network. However, the performance difference between GDL-frequent and GDL-once diminished in setting where a relatively large percentage of initial user connections were provided. And, it also helps in reducing the performance variation if a larger set of initial user connections was provided.
<table>
<thead>
<tr>
<th>Iteration</th>
<th>GDL (Frequent)</th>
<th>GDL (Once)</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>0.02</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>0.02</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>0.03</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>0.03</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>0.03</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>0.02</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>0.02</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>9</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>11</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>13</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>14</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 5.4: Standard Deviation of CER per Iteration (Sample Rate = 0.3)
Chapter 6

Software Implementation

6.1 Implementation

Some implementation details of the group-wise distance learning algorithm are discussed in this chapter. In previous chapters, the content is focused on introducing the theoretical concepts of the proposed algorithm and various experiment studies with it for evaluation purpose. However, this dissertation study could not be conducted without the help of a collection of open source projects. The core implementation of the proposed algorithm is built upon a number of Python open-source libraries: Numpy, Scipy, Pandas, Networkx and etc [15, 22, 11]. And, some of results are analyzed in R, with help of some open source libraries: ggplot2, dplyr and etc. [26, 39, 40]. Being inspired by the generous contributors in the open-source community, the proposed algorithm had been implemented in Python and open-sourced to facilitate the further development and various applications in both academic and industrial domains. In this chapter, the overview structure of software is introduced.
6.2 Introduction to Python Implementation

In this section, it is concentrated on the discussion of some implementation details of Group-wise Distance Learning (GDL). GDL is the core component of the framework which is proposed in this dissertation study. In figure 6.1, it depicts the conceptual structure of GDL implementation based on description laid out in section 3.2.1.

Fit Group is a Python dictionary which serves as a container to store $K$ groups of user IDs. $K$ is an input argument defining the number of groups which the supplied user population will be divided into. Within a Fit Group, user IDs of the same group is kept in a Python list. Such list of user IDs is a value of a Python dictionary and can be retrieved easily with its Key. A Key denotes a user group. At an iteration, Fit Group houses all user IDs of users who can pass KS-test for goodness-of-fit with respects to at least one of group-wise distance metrics. At the beginning of each iteration, the member of a group of Fit Group will be examined by KS-test to tell if it is still a good fit for the group with respects to the updated distance metrics/weights of the particular group. If a user were rejected by KS-test, the user ID would be re-distributed to Unfit Group for KS-test goodness-of-fit test against other groups.

Unfit Group is also a Python dictionary which organizes user into a different group. Within this container, a user is stored in the value of the group whose group-wise distance metrics with which the user can be rejected by KS-test. Therefore, by specifying by which group a user had been rejected at the previous step, a user will be examined by KS-test with distance metrics of all another group but the one to which the user had belongs. If the user can be considered to bear a good fit to any of other groups, the user will be assigned to the group which the user’s established social connections can be explained best. However, if a user could be considered a good fit to none of other groups, the user will be assigned to Buffer Group.

Buffer Group is a container which houses the users who could be considered to a good fit to any of user groups in the previous iteration. Since there is no need to search an
Figure 6.1: Implementation Workflow of Group-wise Learning Distance Algorithm
alternative group for users in this container, the Buffer Group is implemented as a Python
list. At every new iteration, users in Buffer Group will be examined with KS-test against
updated group-wise distance metrics to see if any of them can be re-distributed to Fit Group.

At the end of each iteration, a fit score is computed to record the overall quality of learning
results. The learning results include the set of group-wise distance metrics, member compo-
sition in Fit Group, P-value scores of users in Fit Group and the number of users in Buffer
Group. A new iteration starts by computing group-wise distance metrics to accommodate
the updated member composition in Fit Group.

6.2.1 API Design

Application Programming Interface plays a critical role in determining how easy a software
is for users. A well-designed API provides an effective interface to represent logical flow of
accomplishing a solid task with the software. The API design of algorithms and framework
in this dissertation follows the API protocols of scikit-learn because of its popularity[^6]. The
scikit-learn is a popular machine learning library in Python community. Its well-designed
API had been widely adopted by many academic researchers and industry practitioners.

The below Python scripts are used to apply GDL to a social network, which is specified
by a list of user friends and a table of a user profile.

```
from groupwise_distance_learning.groupwise_distance_learner import GroupwiseDistLearner

# initiate a GDL learner, configured to cluster user population
# into two groups, requiring each groups having minimal 10 users
gwd_learner = GroupwiseDistLearner(n_group=2, min_group_size=10)

# feed social network information to learn groupings and distance metrics
# gwd_learner.fit(user_ids, user_profiles, user_friendships)

# retrieve distance metrics
learned_dist_metrics = gwd_learner.get_groupwise_weights()
```
In addition to GDL, a Python library had been developed to facilitate a simulation-based evaluation of user recommendation system. The following Python scripts describe the process of running a simulation experiment. The target user recommendation system is empowered by Nearest Neighbour algorithm.

```python
from user_recommender_framework.network_simulator import *
from user_recommender_framework.user_recommender import *

# define a recommendation system as subject for evaluation
nnu_recommender = NNUserRecommender(user_ids, user_profiles, init_user_connections)

# define evaluation system
evaluator = SocialNetworkEvaluator()

# load reference social network, which represent true status of social network
evaluator.load_ref_user_connections(user_connections)

# define a user behavior simulator
user_clicker = UserClickSimulator()

# setup experiment
experimentor = UserRecSysExpSimulator(name="MyExperiment")
experimentor.load_recommender(nnu_recommender)
experimentor.load_evaluator(evaluator)
experimentor.load_clicker(user_clicker)

# set the number of suggestions for each user at each iteration
experimentor.set_recommendation_size(5)

# start experiment, experiment results will be exported automatically
experimentor.run()
```
6.2.2 Summary

The proposed algorithm is implemented in Python and open-sourced with an appropriate license for future modification and distribution. This attempt is based on three intentions: a) facilitate the broader range of applications of this algorithm used by peer academic researcher and industrial practitioners; b) promote the re-use of the code to validate the results presented in this dissertation and build innovative algorithm upon this implementation; c) allow further improvement made by more experienced developer who is interested in this algorithm to make the algorithm more efficient in computation. As results, there are three Python libraries open-sourced for reuse. The first is the implementation of simulation-based evaluation framework of user recommendation system[1]. The second is the implementation of the proposed Group-wise distance learning algorithm[2]. The third one is the implementation of distance learning algorithm[3].

[1] https://github.com/beingzy/user_recommender_framework
Chapter 7

Conclusion and Future Plan

In this dissertation, a new algorithm called Group-wise Distance Learning (GDL) is introduced. It is proposed as a method to build a user recommendation system for online social network platforms. In Chapter 02, we collectively reviewed the current progress of methodologies in the research area of general recommendation systems. In Chapter 03, it layouts the main challenges and problems which driven the creation of a GDL-empowered user recommendation system. In Chapter 04, the detailed design of GDL algorithm and its application framework to empower a user recommendation system are illustrated. In Chapter 05, a set of experiments each of which focuses testing one main module of GDL is discussed. In Chapter 06, two simulation experiment are conducted to evaluate the effectiveness of GDL-empowered user recommendation system about Nearest Neighbor-based counterparts. Based on discussion and experiment results, we had drawn the conclusion that GDL algorithm could be a superior alternative method to Nearest Neighbor (NN) algorithm to build user recommendation systems.

Rather than suggesting to utilize the proposed GDL algorithm solely to build a user recommendation system, it is advised to create a hybrid user recommendation system by using both Nearest Neighbour and GDL jointly. Since Group-wise Distance Learning algorithm implicitly requires a user having a substantial number of user connections to understand her/his
social preference. For new users, they were facing the cold-start issue due to their often none of minimal amount of user connections. In this situation, by being complemented with NN, the hybrid user recommendation’s cold start issue can be mitigated. NN-empowered depend on user’s personal demographic information to serve their social exploration needs.

7.1 Future Plan

- Conduct the simulation experiment with real-life social network data. We had conducted two simulation experiment with synthetic social network data to provide evidence of the potential superiority of the proposed algorithm. These studies serve well as the stepstone to reveal the potential of the proposed algorithm. However, a simulation study with real-life social network data is needed to confirm the practical value of the proposed algorithm.

- Research on methodologies to improve KS-test to deal with heavily imbalanced social profiles. There are much more users in $D$, the set of users who are either not connected to or explicitly disconcerted by the studied users, than the number of users in $S$, which represents of users who are connected to those studied users. The test power of proposed one-sided KS-test would be diminishing due to the bias caused by the overwhelming size of one particular set. Therefore, some sampling strategies should be exploited to create relatively balanced data between $S$ and $D$ when evaluating the goodness-of-fit with KS-test.

- Develop more sophisticated hybrid user recommendation system. We had discussed that, by combining NN (Nearest Neighbor) and GDL (Groupwise Distance Learning) together, a user recommendation with better usability could be developed. And, it is also worthy to explore the opportunity to incorporate the recommendation strategy of suggesting friends’ friends in the user recommendation system. If the hybrid system can be equipped with a suitable cascading strategy, it alternates among a variety of
underline recommendation engines to serve on various scenarios. And, the hybrid system of this type could have the potential to provide superior performance.

7.2 Final Words

This dissertation presented a new method to build a user recommendation system. As part of the output of dissertation study, the proposed algorithm and a framework for conducting the simulation study to evaluate user recommendation system had been implemented and packaged as Python libraries. The tool is open-sourced for either academic researcher or industrial users to advance the method further. As results, there are three Python libraries open-sourced for reuse. The first is the implementation of simulation-based evaluation framework of user recommendation system

1. https://github.com/beingzy/user_recommender_framework
3. https://github.com/beingzy/learning_dist_metrics

The second is the implementation of the proposed Group-wise distance learning algorithm and The third one is the implementation of distance learning algorithm.
Bibliography


Hadley Wickham and Romain Francois. dplyr: A Grammar of Data Manipulation. R package version 0.4.3. 2015. URL: [https://CRAN.R-project.org/package=dplyr](https://CRAN.R-project.org/package=dplyr).

