ACADEMIC PERFORMANCE PREDICTION WITH MACHINE LEARNING TECHNIQUES

by

Zhiyun Ren

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

_____	Dr. Huzefa Rangwala, Dissertation Director
_____	Dr. Carlotta Domeniconi, Committee Member
_____	Dr. Jessica Lin, Committee Member
_____	Dr. Aditya Johri, Committee Member
_____	Dr. Xia Ning, Committee Member
_____	Dr. Sanjeev Setia, Department Chair
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____ Spring 2019
George Mason University
Fairfax, VA

Academic Performance Prediction with Machine Learning Techniques

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Zhiyun Ren
Master of Science
Beihang Uiversity, 2013
Bachelor of Science
Dalian University of Technology, 2010

Director: Dr. Huzefa Rangwala, Professor
Department of Computer Science

Spring 2019
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my parents who have always supported me in all my efforts.

Acknowledgments

I would like to thank my dissertation director, Dr. Huzefa Rangwala for the great help throughout the project. During my PhD program, Dr. Rangwala always guided me with inspiring ideas that help me power through many tough problems. I am very grateful for his countless hours of reading, encouraging and patience throughout the entire process.

I would also like to extend my thanks to the committee member, Dr. Xia Ning, Dr. Carlotta Domeniconi, Dr. Jessica Lin and Dr. Aditya Johri for their valuable guidance and precious time in reviewing. I am also thankful to lab members for the fruitful discussion and suggestion during the course of my dissertation work.

Finally, I would like to thank my parents, relatives, friends and supporters who made this happen.

This research was supported by the National Science Foundation (NSF) Grant #1447489.

Table of Contents

List of Tables

List of Figures

Abstract

ACADEMIC PERFORMANCE PREDICTION WITH MACHINE LEARNING TECHNIQUES

Zhiyun Ren, PhD

George Mason University, 2019

Dissertation Director: Dr. Huzefa Rangwala

Nationally, the six year graduation rate for four year degree programs at universities and colleges in the United States has remained approximately 60% over the past decade. One of the main reasons for poor retention (and ultimately training) of students has been lack of proper advising and planning. Recently, there has been the prevalence of educational technologies driven by data analytics in educational environments for assisting students in selecting courses, acquiring feedback and improving learning outcomes based on past academic performance and behaviors.

Grade prediction methods seek to estimate a grade that a student may achieve in a course/task that she may take in the future (e.g., next term, next assessment). Existing grade prediction methods are mainly based on matrix factorization (MF) approaches, and overlook important factors that could greatly influence student's performance. In this thesis, I present developed several methods for performance estimation for students within a traditional brick-and-mortar university and online courses.

Specifically, I model the evolution of a student's knowledge while studying a sequence of courses within a matrix factorization framework. I provide a flexible framework that allows for incorporation of course-related and student-related factors like instructor, academic level and effort within a latent factor model. I also incorporate the influence of multiple co-taken courses within a semester along with student's cumulative knowledge.

I also present a deep learning based recommender system approach for predicting the grade a student will earn in a course that he/she plans to take in the next-term. The deep learning inspired approach provides added flexibility in learning the latent spaces in comparison to MF approaches. The proposed approach also incorporates instructor information besides student and course information. In addition, I also engineer student learning and engagement features from the server logs of students enrolled in a Massive Open Online Course (MOOCs). These features are incorporated within a Personalized Linear Multi-Regression model to predict within-class student's performance in an online education environment.

This thesis demonstrates the strengths of academic performance prediction on multiple benchmarks. Incorporating these within Early Warning Systems to identify students who are at risk of dropping out can lead to timely help from human advisors in helping students succeed within their academic programs. Accurate and timely prediction of students' academic grades holds the promise for better student degree planning, personalized advising and providing timely feedback/interventions to ensure that students stay on track in their chosen degree program and graduate on time.

Chapter 1: Introduction

1.1 Motivation

Data mining technologies have been in high demand in educational environment as they provide technicality analysis on students' past academic performance and behaviors for assisting them in selecting courses, acquiring feedback and improving performance in the future. Grade prediction methods seek to estimate a grade that a student may achieve in a course/task that he/she may take in the future (e.g., next term, next assessment). Accurate and timely prediction of students' academic grades holds the promise for better student degree planning, personalized advising and automated interventions to ensure that students stay on track in their chosen degree program and graduate on time.

In the past decades, low graduation and retention rate has been one of the most severe problems in higher education institutions in America [2]. A report from The National Center for Education Statistics shows that in recent years, around 59 percent of students who start a four-year college to pursue a bachelor's degree are able to complete the degree in six years [1]. It turns out that government, education institutions and students all spend a great amount of money and energy on the education which a considerable part of has no valuable outcomes [3]. Undoubtedly, there is a critical need to develop new Educational Technologies (EdTech) [4] which are able to provide students successful degree pathways and ensure they graduate in a timely fashion (4 to 6 years) and are well prepared for jobs in their respective fields of study. Among many popular EdTech applications, data analytics always play a dominant rule. To increase student graduation rates, several Educational Data Mining (EDM) techniques have been developed and deployed at many institutions to help students pass towards successful graduation [5]. For example, *degree planners* [2] assist students in

[1] https://nces.ed.gov/programs/coe/indicator_ctr.asp

[2] http://www.blackboard.com/mobile-learning/planner.aspx

deciding their majors or fields of study, choosing the sequence of courses within their chosen major and providing advice for achieving career and learning objectives. *Early warning systems* [5] inform advisors/students of progress, and additionally provide cues for intervention when students are at the risk of failing one or more courses and dropping out of their program of study. An effective way to assist and improve degree planning and advising is via modeling the student's knowledge and foreseeing their future academic performance [2].

Among several EDM tasks, accurate and timely grade prediction is very important since it holds the promise for developing effective degree planners and early warning systems, and ultimately improving educational outcomes.

1.2 Problem Statement

In this thesis, I have worked on two different problems. The first problem is future performance prediction for students within a traditional brick-and-mortar university. Specifically, given a student's history academic activities (e.g., chosen courses) and performance (e.g., obtained grade on a course), I will predict the student's performance on a course in the future (e.g., next term). The student's history academic activities will be modeled as a sequence. Each term will be considered as a time step where students' performance will be stored in a student-course interaction matrix. Each row in the interaction matrix represents a student, and each column in the interaction matrix represents a course. The entry value is the corresponding student's grade on a course. To predict any student's performance in the next term, all the previous terms will be considered as training set. And the target term (i.e., the next term) is the test set. The goal is to provide accurate future performance prediction for students in order to build solid foundation to detect at-risk courses, build course recommendation system, plan academic pathway, etc. The second problem is in-class performance prediction in MOOCs. Usually each MOOC has several assignments, videos, quizzes, midterm and final exams. Given a student's activities in MOOCs, such as the number of videos she watches, the performance on assignments, the login frequency to the MOOC, etc., I will predict her performance for final exam, or her performance for assignments based on the different course settings.

1.3 Contribution

In the past few years, several algorithms have been developed to analyze educational data. Matrix factorization (MF) based approaches which are inspired from recommender system research [6] have been widely used for solving the grade prediction problems [7,8]. MF methods decompose the student-course (or student-task) grade matrix into two low-rank matrices, indicating student latent factors and course latent factors, respectively. Then the prediction of the grade for a student on an untaken course is calculated as the product of the corresponding vectors in the two decomposed matrices [9, 10]. Traditional MF methods have limited strength to deal with data sparsity which is common in educational data, due to the fact that students always choose a small set of courses comparing to the number of courses provided by the university. Previous work has been focusing on different modifications of MF methods to improve grade prediction performance [11–13].

In this thesis, I have developed several grade prediction models based on MF framework that take different factors into account, and finally achieve remarkable prediction results. Specifically, I consider that a student's knowledge is continuously being enriched while taking a sequence of courses and propose a model named **M**atrix **F**actorization with **T**emporal **C**ourse-wise **I**nfluence (MFTCI). In this model, students and courses are represented in a latent "knowledge" space. The grade of a student on a course is modeled as the similarity of their latent representation in the "knowledge" space. Course-wise influence is considered as an additional factor in the grade prediction. The experimental results show that the proposed method outperforms several baseline approaches and infer meaningful patterns between pairs of courses within academic programs. Furthermore, student's latent factors are substituted with accumulated knowledge of a sequence of courses taken by the student, jointly with the grade for each course. And I incorporate course instructor and student academic level effects along with student global latent factor to complete grade prediction. This model is named **A**dditive **L**atent **E**ffect (ALE). Moreover, I present next-term grade prediction models based on students' cumulative knowledge and co-taken courses. The proposed models are based on a matrix factorization framework and incorporate a co-taken course interaction function to learn the influence from the co-taken courses on the target course. The co-taken course interaction function is formed by a neural network, which takes the knowledge difference between

the co-taken courses and the target course as input, and outputs an influence value that will be used to predict students' grades on the target course. I compared the new models with several state-of-the-art methods on students of various characteristics (e.g., whether a student transferred in or not). The experimental results demonstrate that the proposed methods significantly outperform the baselines on grade prediction problem. Moreover, I perform a thorough analysis on the importance of different factors and how these factors can practically assist students in course selection, and finally improve their academic performance. Other than MF methods, deep learning (DL) has been in its blossom as it is widely used across many data mining fields, including computer vision (CV), natural language processing (NLP), recommender system (RS) and etc. [14–16]. In this thesis, I present a deep learning based recommender system approach called Neural Collaborative Filtering (NCF) for predicting the grade a student will earn in a course that he/she plans to take in the next-term. The deep learning inspired approach provides added flexibility in learning the latent spaces in comparison to MF approaches. The proposed approach also incorporates instructor information besides student and course information. In addition, I also apply a **P**ersonalized **L**inear **M**ulti-**R**egression (PLMR) model to predict student's performance on online education environment, i.e., Massive Open Online Courses (MOOCs), and gain great results.

The main contributions of my work are as follows:

1. I model and incorporate temporal course-wise influence in addition to matrix factorization for grade prediction. The proposed approach learns pairwise relationships between courses that can help in understanding pre-requisite structures within programs and tuning academic program chains [17].

2. I propose additive latent effect models that incorporate the information of course instructors, student's academic level and student global latent factor for the next-term grade prediction problem. The strengths of the proposed framework include the ability to enhance the standard MF methods with additional student and course-specific content information that may not be contained within the student-course grade matrix [18].

3. To model the influence of the co-taken courses on students' performance I introduce a deep

learning based co-taken course interaction function. This takes the knowledge difference between the co-taken courses and the target course as input, and outputs an influence value from the co-taken courses on the target course. Integrated with cumulative knowledge acquired by a student, this is the first work that learns and explicitly incorporates influences from co-taken courses for grade prediction.

4. I extend deep learning based Neural Collaborative Filtering (NCF) on next-term grade prediction problem. Beyond student-course interaction pairs, the proposed model effectively incorporates instructor level information.

5. I also develop a personalized multiple learning regression model for estimating within class performance. This models engineers features from back end server logs to capture student learning habits within a Massive Open Online Course (MOOC) to predict student's performance [8, 19].

Chapter 2: Background

2.1 Preliminaries and Notations

Formally, student-course grades will be represented by a series of matrices $\{\mathrm{G}_1, \mathrm{G}_2, ..., \mathrm{G}_T\}$ for T terms. Each row of G_t represents a student, each column of G_t represents a course, and each value in G_t, denoted as $g^t_{s,c}$, represents a grade that student s got on course c in term t ($g^t_{s,c} \in (0,4]$, $g^t_{s,c} = 0$ indicates that student s did not take the course c in term t. I add a small value to failing grade to distinguish 0 score from such situation.). Student-course grades up to the t_{th} term will be represented by $\mathrm{G}^t = \sum_{i=1}^{t} \mathrm{G}_i$ with size of $n \times m$, where n is the number of students and m is the number of courses. Given the database of (student, course, grade) up to term $(T-1)$ (i.e., G^{T-1}), the next-term grade prediction problem is to predict grades for each student on courses they might enroll in the next term T. To simplify the notations, if not specifically stated in this thesis, I will use $g_{s,c}$ to denote $g^t_{s,c}$. The testing set is then (student, course, grade) triples in the T_{th} term, represented by matrix G_T. Rows from the grade matrices representing a student s will simply be represented as $G(s,:)$ and the specific courses that student has a grade for in this row can be given by $c' \in G(s,:)$.

In this thesis, all vectors (e.g., $\mathbf{p}_s^\mathsf{T}$ and $\mathbf{q}_c$) are represented by bold lower-case letters and all matrices (e.g., A) are represented by upper-case letters. Column vectors are represented by having the transpose supscript$^\mathsf{T}$, otherwise by default they are row vectors. A predicted/approximated value is denoted by having a $\tilde{}$ head.

Given student-course grades up to term $(T-1)$, the objective of the next-term grade prediction problem is to predict grades for each student on courses that the student may consider for enrollment in the next term T.

Table 2.1 summarizes the key notations used in this thesis.

Table 2.1: Notations

Notation	Explanation
m	number of courses
n	number of students
k	the dimension of latent factors
G_t	student-course grades at term t
G^t	all the student-course grades up to term t
$g^t_{s,c}$	the grade of student s on course c at term t
$C_{s,t}$	the set of courses student s chooses at term t
C^t_s	the set of courses student s chooses up to term t
$G_{s,t}$	all the grades student s obtains at term t
G^t_s	all the grades student s obtains up to term t
$t_{s,c}$	the academic term when student s takes course c

2.2 Related Work on Grade Prediction

Over the past few years, several methods have been developed to model student behavior and academic performance [20,21], and they gain improvement of learning outcomes [22]. Methods influenced by Recommender System (RS) research [23], including Collaborative Filtering (CF) [24] and Matrix Factorization [25], have attracted increasing attention in educational mining applications which relate to student grade prediction [26] and in-class assessment prediction [27].

Matrix factorization from RS [28] can be applied for the next-term grade prediction problem, when the student-course grade matrix is considered as the user-item rating matrix. Two low-rank matrices containing latent factors of courses and students in a common knowledge space can be learned from such a student-course grade matrix [7]. Thus, the grade of a student s on a course c can be predicted as

$$\tilde{g}_{s,c} = \mathbf{p}_s^\top \mathbf{q}_c, \tag{2.1}$$

where $\mathbf{p}_s$ ($\mathbf{p}_s \in \mathbb{R}^k$) and $\mathbf{q}_c$ ($\mathbf{q}_c \in \mathbb{R}^k$) are the two vectors containing latent factors of k dimensions for student s and course c, respectively. This method is denoted as MF. Including the bias terms within the MF formulation has shown to be effective in modeling systematical biases [25]. For the

grade prediction problem using MF, student and course biases can be included as follows:

$$\tilde{g}_{s,c} = \mathbf{p}_s^\top \mathbf{q}_c + b_s + b_c, \tag{2.2}$$

where b_s and b_c are bias terms for student s and course c, respectively. This method is referred to as MF with bias terms and denoted as MF-b.

Many other models have been developed based on MF framework. For example, Sahebi *et al.* [29] modeled student learning progress and predicted student performance using tensor decomposition based on the sequence of student attempts within course quizzes. Lan *et al.* [30] predicted student performance on different questions within the context of intelligent tutoring systems. Meier *et al.* [31] developed an online learning method that learns the best time to intervene based on past student performance in a course. Sweeney *et al.* [7, 32] performed an extensive study of several recommender system approaches including SVD, SVD-kNN and Factorization Machine (FM) to predict next-term grade performance.

Additionally, incorporation of biases has shown to be important for several educational data mining problems, following its success in RS [25]. Elbadrawy *et al.* [11] developed a domain-aware grade prediction method with student/course-group based biases. To predict student s' grade on course c, this method groups students and courses in different ways based on student majors, academic levels and course subjects, and introduces group-based biases. In this method, the grade for student s on course c is predicted as:

$$\tilde{g}_{s,c} = \mathbf{p}_s^\top \mathbf{q}_c + b_s^{\varphi(c)} + b_c^{\varphi(s)}, \tag{2.3}$$

where $\mathbf{p}_s$ ($\mathbf{p}_s \in \mathbb{R}^k$) and $\mathbf{q}_c$ ($\mathbf{q}_c \in \mathbb{R}^k$) are the latent factors for student s and course c, respectively. $\varphi()$ denotes the grouping information. $b_c^{\varphi(s)}$ is the bias term for course c. This method models course bias based on the performance of students who are in the same group of student s (i.e., $\varphi(s)$) and have taken course c before. Similarly, $b_s^{\varphi(c)}$ is the student bias term modeled based on the grades student s has got on the courses which are in the same group as course c (i.e., $\varphi(c)$). The key intuition

of this method is that students who take the same course and can be grouped by domain information (e.g., student's major) may share a similar bias. This method is referred to as MF with domain-aware biases and denote it as MF-d in my future experiments. Moreover, inspired by content-based recommendation [33] approaches, Polyzou *et al.* [12] addressed the future course grade prediction problem with three approaches: course-specific regression, student-specific regression and course-specific matrix factorization. Moreover, neighborhood-based CF approaches [34–36] predict grades based on the student similarities, i.e., they first identify similar students and use their grades to estimate the grades of the students with similar profiles.

2.3 Related Work on Sequential Data Mining

In educational data mining problems, sequential information of students/courses over time is very common and thus methods that deal with sequential data can be beneficial. As a matter of fact, such methods have been extensively developed in RS research. For example, integrated methods of Markov Chains (MC) and MF have been popular in dealing with sequential data in RS. Rendle *et al.* [37] proposed the factorized personalized MC (FPMC) models. These models have personalized Markov chains that rely on transition matrices, and these methods use a factorization model to deal with the sparsity in the input data. Based on FPMC, He *et al.* [38] developed factorized sequential models with item similarities for sparse sequential recommendation. Their models consider both long-term and short-term dynamics among user-item data. Moreover, in their other work, He *et al.* [39] adopted a similar idea and developed large-scale recommender systems to model the preferences and short-term dynamics between both users and items. Morsy *et al.* [13], the proposed methods model each student's latent factors with accumulated knowledge of a sequence of courses taken by the student, jointly with the grade for each course. Morsy et. al. [13] consider the series of courses a student takes as a sequence and propose Cumulative-Knowledge Regression Models (CKRM). Specifically, to predict student s's performance on course c, CKRM represents student s with the series of courses she has taken in the past, and each course is represented by a vector which is expected to capture the latent knowledge components provided by the course. Moreover, CKRM represents course c with a vector which is expected to capture the latent knowledge components

required by the course. Consequently, given a student s in term t, $\mathbf{p}^t_{ck(s)}$ is the cumulative knowledge acquired until term t, and is given by:

$$\mathbf{p}^t_{ck(s)} = \sum_{g_{s,c'} \in G^{t-1}_s} (e^{-\lambda(t-t_{s,c'})} \mathbf{k}_{c'} \cdot g_{s,c'}), \tag{2.4}$$

where $t_{s,c'}$ is the term in which student s took course c', $e^{-\lambda(t-t_{s,c'})}$ is an exponential time decay function, $\mathbf{k}_{c'}$ contains the latent knowledge factors of course c', and $g_{s,c'}$ is the grade of student s on course c'. The grade of student s on course c is then predicted as follows:

$$\tilde{g}^t_{s,c} = {\mathbf{p}^t_{ck(s)}}^{\mathsf{T}} \mathbf{q}_c. \tag{2.5}$$

In this study, I average the results in Eq 2.5 with the sum of exponential time decay weight, that is:

$$\tilde{g}^t_{s,c} = \frac{1}{|G^{t-1}_s|} {\mathbf{p}^t_{ck(s)}}^{\mathsf{T}} \mathbf{q}_c \tag{2.6}$$

This method is referred to as averaged **CK**RM and denoted as CK. The preliminary experiments demonstrate that CK outperforms CKRM.

Other than FPMC models, in order to capture the changing of user dynamics over time in RS, various dynamic models have been developed. Many of such models are based on Matrix Factorization and state space models. Sun *et al.* [40, 41] model user preference change using a state space model on latent user factors, and estimate user factors over time using noncausal Kalman filters. Similarly, Chua *et.al.* [42] apply Linear Dynamical Systems (LDS) on Non-negative Matrix Factorization (NMF) to model user dynamics. Ju *et al.* [43] encapsulate the temporal relationships within a Non-negative matrix formulation. Zhang *et al.* [44] learn an explicit transition matrix over the latent factors for each user, and estimate the user and item latent factors and the transition matrices within a Bayesian framework. Other popular methods for dynamic modeling include time-weighting similarity decaying [45], tensor factorization [46] and point processes [47].

Another work related to sequential data mining is identifying course trajectories for college students [48,49]. Generally, students are partitioned into two groups by their grades: high-performance students and low-performance students. Then, sequential pattern mining algorithms, such as Apriori, are used to identify the course trajectories for the two groups of student, respectively. Such course trajectories can help detect course choices and degree planning that will affect student's academic performance. Moreover, given a student's course trajectory captured by effective algorithms, previous work has shown it can improve the classification accuracy when detecting high- or low-performance student. This groups of methods are different from my work. In my thesis, I'm working on a regression problem which predicts student's grade, instead of a classification problem which partitions students into two groups, high-performance student group and low-performance student group. Therefore, I'm working on numeral data instead of binary data.

2.4 Related Work on Online Learning System

As for MOOCs, several researchers have focused on the analysis of education data in this field, in an effort to understand the characteristics of student learning behaviors and motivation within this education model [50]. Brinton et. al. [51] developed an approach to predict if a student answers a question correct on the first attempt via click-stream information and social learning networks. Kennedy et. al. [52] analyzed the relationship between a student's prior knowledge on end-of-MOOC performance. Sunar et. al. [53] developed an approach to predict the possible interactions between peers participating in a MOOC. Elbadrawy et. al. [27] proposed the use of personalized linear multi-regression models to predict student performance in a traditional university by extracting data from course management systems (Moodle). My study focuses on MOOCs, which presents different assumptions, challenges and features in comparison to a traditional university environment.

Most similar to the proposed work, Pardos et. al. proposed a model "Item Difficulty Effect Model" (IDEM) that incorporates the difficulty levels of different questions and modifies Bayesian Knowledge Tracing (BKT) model [54] by adding an "Item" node to every question node. By identifying the challenges associated with modeling MOOC data, the IDEM approach and extensions

that involve splitting questions into several sub-parts and incorporating resource (knowledge) information [55] are considered state-of-the-art MOOC assessment prediction approaches and referred as KT-IDEM. However, this approach can only predict a binary value grade. In contrast, the model proposed in my thesis is able to predict both, a continuous and a binary grade.

2.5 Related Work on Deep Learning

2.5.1 Deep Learning in Educational Data Mining

Deep Learning (DL) techniques have been applied to solve many educational data mining problems. For example, Sharma *et al.* [56] proposed a composite deep neural network to predict human movement (e.g., walking, sitting) in educational videos. The proposed method first used a convolutional neural network to extract the video features, followed by a deep recurrent neural network to predict the human movement label. Klingler *et al.* [57] employed deep variational auto-encoders (VAE) on classification tasks in EDM. Specifically, the presented model makes effective use of unlabeled data to learn efficient feature embeddings for students, and significantly improved detection results of developmental dyscalculia (DD), compared to completely supervised training. Xiong *et al.* [58] introduced a recently developed model, Deep Knowledge Tracing (DKT), a pioneering algorithm that uses recurrent neural network to model student learning. The method is evaluated on various datasets compared with Factors Analysis Model and Knowledge Tracing models models, and the results show the proposed method outperforms the baselines on various datasets. Piech *et al.* [59] introduced Deep Knowledge Tracing (DKT) to model student learning with Recurrent Neural Networks. The authors provided experiments on how to use DKT to detect latent structure between the assessments in the dataset.

2.5.2 Deep Learning in Recommender Systems

Furthermore, DL has also been widely used in RS problems. Kim *et al.* [60] proposed a convolutional matrix factorization (ConvMF) model that integrates convolutional neural network (CNN) into probabilistic matrix factorization (PMF) for context-aware recommendation. Wang *et al.* [61]

first studied the editor article selection behavior, and then proposed a Dynamic Attention Deep Model (DADM) to automatically select a subset of articles from the large pool. DADM uses character-level text modeling and convolutional neural networks (CNNs) to learn the representation of each article, followed by an attention-based network architecture that aims to assign influence factors on recent models dynamically in order to improve the performance of the current recommender system. Salakhutdinov *et al.* [62] successfully applied Restricted Boltzmann Machines (RBMs) on the basic recommendation tasks. The paper also showed an efficient learning algorithm for RBM, named Contrastive Divergence (CD).

Feedforward Neural Network in Recommender Systems

He *et al.* [63] presented a general framework named Neural network-based Collaborative Filtering (NCF) in order to tackle the collaborative filtering problem in recommendation based on implicit feedback. NCF replaces the inner product with a neural architecture that could learn an arbitrary function from data, and has shown significant improvements over the state-of-the-art methods. Covington *et al.* [64] proposed a deep learning model for YouTube recommendation. The model comprises two parts: 1) the candidate generation network and the ranking network which are both fully connected neural network with concatenated embedded features and embedded categorical features as input, respectively. Yang *et al.* [65] developed a deep neural architecture called PACE (Preference And Context Embedding) for POI recommendation. PACE jointly learns the embeddings of users and POIs to predict both user preference over POIs and various context associated with users and POIs. Wang *et al.* [66] developed a hierarchical Bayesian model called collaborative deep learning (CDL) as a novel method for RS problem. Elkahky *et al.* [67] proposed a DL approach to map users and items to a latent space and learned the parameters by maximizing the similarity between users and their preferred items. Then the paper introduced a multi-view DL model to jointly learn features of items from different domains and user features. Cheng *et al.* [68] presented a DL model that combined wide and deep learning, i.e., combined the benefits of memorization and generalization for recommender system.

Autoencoder in Recommender Systems

Wu *et al.* [69] presented a Collaborative Denoising Autoencoder (CDAE) for top-N recommendation which adopts the idea of Denoising Auto-Encoders. Dong *et al.* [70] presented a hybrid model which combined deep learning techniques and collaborative filtering method in learning users and items' latent factors with side information and on the rating matrix, respectively. Wang *et al.* [71] developed a collaborative recurrent autoencoder (CRAE) that models the generation of content sequences. CRAE is based on the development of a hierarchical Bayesian model for the denoising recurrent autoencoder (DRAE) and is generalizated in collaborative filtering setting. Li *et al.* [72] proposed a general deep architecture for CF by combining probabilistic matrix factorization with marginalized denoising stacked auto-encoders. Li *et al.* [73] proposed a Bayesian generative model called collaborative variational autoencoder (CVAE) that considers both rating and content for various recommendation problems.

Recurrent Neural Network in Recommender Systems

Smirnova *et al.* [74] proposed a new class of Contextual Recurrent Neural Networks for Recommendation (CRNNs) that considered the contextual information both in the input and output layers. Wu *et al.* [75] built a deep RNN for personal recommendation. The proposed model tracked user browsing history with multiple hidden layers, and it only recorded a number of states, while the old states are represented by a single history state, in order to reduce the processing cost. Hidasi *et al.* [76] introduced a number of parallel RNN (p-RNN) architectures to model sessions based on the clicks and the features (images and text) of the clicked items for session-based recommendation.

2.6 Performance Evaluation

2.6.1 Experimental Protocol

To assess the performance of the next-term grade prediction models, I trained my models on data up to term $T-1$ and make predictions for term T. I evaluate my method for three test terms, i.e., Spring 2016, Fall 2015 and Spring 2015. As an example, for evaluating predictions for term Fall 2015, data from Fall 2009 to Spring 2015 is considered as training data and data from Fall 2015 is testing data. datasets. Figure 2.1 shows the three different train-test splits.

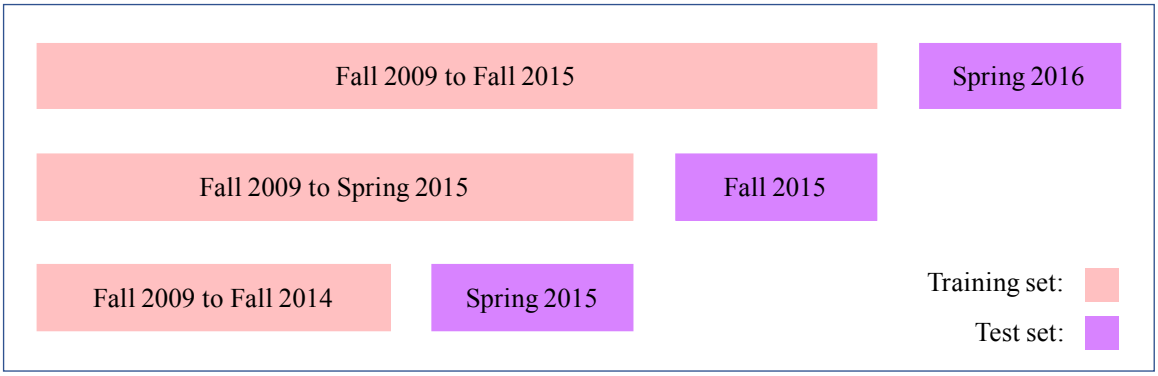


Figure 2.1: Different Experimental Protocols

2.6.2 Dataset Description

The data used in this thesis is obtained from George Mason University (GMU). The dataset contains two student groups: first-time freshmen (FTF; i.e., students who begin their study initially at this University), and transfer students (TR; i.e., students who transfer to this University from a different one). The dataset was extracted in the period of Fall 2009 to Spring 2018. It includes information of 23,435 FTF students and 28,470 TR students across 153 majors, who have enrolled in 5,431 courses.

2.6.3 Evaluation Metrics

I use Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Percentage of Tick Accuracy (PTA) as evaluation metrics. RMSE and MAE and are defined as follows:

$$RMSE = \sqrt{\frac{\sum_{s,c \in G_T} (g_{s,c} - \tilde{g}_{s,c})^2}{|G_T|}},$$

$$MAE = \frac{\sum_{s,c \in G_T} |g_{s,c} - \tilde{g}_{s,c}|}{|G_T|}$$

where $g_{s,c}$ and $\tilde{g}_{s,c}$ are the ground truth and predicted grade for student s on course c, and G_T is the testing set of (student, course, grade) triples in the T_{th} term. Normally, in next-term grade

prediction problem, MAE is more intuitive than RMSE since MAE is a straightforward method which calculates the deviation of errors directly while RMSE has implications such as penalizing large errors more.

PTA metrics is defined as follows. For the dataset, a student's grade can be a letter grade (i.e. A, A-, ..., F). As done previously by Polyzou et. al. [77], I define a tick to denote the difference between two consecutive letter grades (e.g., C+ vs C or C vs C-). To assess the performance of the proposed grade prediction method, I convert the predicted grades into their closest letter grades and compute the percentage of predicted grades with no error (or 0-ticks), within 1-tick and within 2-ticks denoted by Pct_0, Pct_1 and Pct_2, respectively. For the problem of course selection and degree planning, courses predicted within 2 ticks can be considered sufficiently correct.

Chapter 3: Grade Prediction with Temporal Course-wise Influence

In this chapter, I present a novel approach referred as **M**atrix **F**actorization with **T**emporal **C**ourse-wise **I**nfluence (MFTCI) model to predict next term student grades. MFTCI considers that a student's grade on a certain course is determined by two components: (i) the student's competence with respect to each course's topics, content and requirement, etc., and (ii) student's previous performance over other courses. I performed a comprehensive set of experiments on various datasets. The experimental results show that the proposed method outperforms several state-of-the-art methods. The work presented in this chapter has been published in International Conference on Educational Data Mining (EDM 2017).

3.1 Methods

3.1.1 MF with Temporal Course-wise Influence

I consider the student s' grade on a certain course c, denoted as $g_{s,c}$, as determined by two factors. The first factor is the student s' competence with respect to the course c's topics, content and requirement. This is modeled through a latent factor model, in which s' competence is captured using a size-k latent factor $\mathbf{p}_s$, c's topics and contents are captured using a size-k latent factor $\mathbf{q}_c$ in the same latent space as $\mathbf{p}_s$. Then the competence of s over c is modeled by the "similarity" between $\mathbf{p}_s$ and $\mathbf{q}_c$ via their dot product (i.e., $\mathbf{p}_s^\top \mathbf{q}_c$).

The second factor is the previous performance of student s over other courses. I hypothesize that if course c' has a positive influence on course c, and student s achieved a high grade on c', then s tends to have a high grade on c. Under this hypothesis, I model this second factor as a product between the performance of student on a previous "related" course where the pairwise course relationships are learned in the formulation. Note that I consider this pairwise course influence as time

independent, i.e., the influence of one course over another does not change over time. However, the impact from previous performance/grades can be modeled using a decay function over time. Taking these two factors, the estimated grade is given as follows:

$$\begin{aligned}\tilde{g}_{s,c} &= \mathbf{p}_s^\top \mathbf{q}_c \\ &+ \underbrace{e^{-\alpha} \frac{\sum_{c' \in G_{T-1}(s,:)} A(c',c) g_{s,c'}}{|G_{T-1}(s,:)|}}_{\Delta(T-1)} \\ &+ \underbrace{e^{-2\alpha} \frac{\sum_{c'' \in G_{T-2}(s,:)} A(c'',c) g_{s,c''}}{|G_{T-2}(s,:)|}}_{\Delta(T-2)},\end{aligned} \tag{3.1}$$

in which $A(c',c)$ is the influence of c' on c, $G_{T-1}(s,:)/G_{T-2}(s,:)$ is the subset of courses out of all courses that s has taken in the first/second previous terms, $|G_{T-1}(s,:)|/|G_{T-2}(s,:)|$ is the number of such taken courses. $e^{-\alpha}/e^{-2\alpha}$ denote the time-decay factors. In Equation 3.1, I consider previous two terms. More previous terms can be included with even stronger time-decay factors. Given the grade estimation as in Equation 3.1, I formulate the grade prediction problem for term T as the following optimization problem,

$$\begin{aligned}\min_{U,V,A} \; & \frac{1}{2} \sum_{s,c} (g_{s,c} - \tilde{g}_{s,c})^2 + \frac{\gamma}{2} (\|P\|_F^2 + \|Q\|_F^2) \\ & + \tau \|A\|_* + \lambda \|A\|_{\ell_1} \\ & \text{s.t., } A \geq 0\end{aligned}$$

where P and Q are the latent non-negative student factors and course factors, respectively; $\|A\|_*$ is the nuclear norm of A, which will induce an A of low rank; and $\|A\|_{\ell_1}$ is the ℓ_1 norm of A, which will introduce sparsity in A. In addition, the non-negativity constraint on A is to enforce only positive influence across courses.

Optimization Algorithm of MFTCI

I apply the ADMM [78] technique for Equation 3.2 by reformulating the optimization problem as follows,

$$\begin{aligned}
\min_{U,V,A,U_1,U_2,Z_1,Z_2} \quad & \frac{1}{2}\sum_{s,c}(g_{s,c}-\tilde{g}_{s,c})^2+\frac{\gamma}{2}(\|P\|_F^2+\|Q\|_F^2) \\
& +\tau\|Z_1\|_*+\lambda\|Z_2\|_{\ell_1} \\
& +\frac{\rho}{2}(\|A-Z_1\|_F^2+\|A-Z_2\|_F^2) \\
& +\rho(tr(U_1^\mathsf{T}(A-Z_1))) \\
& +\rho(tr(U_2^\mathsf{T}(A-Z_2))) \\
\text{s.t.,} \quad & A\geq 0
\end{aligned}$$

where Z_1 and Z_2 are two auxiliary variables, and U_1 and U_2 are two dual variables. All the variables are solved via an alternating approach as follows.

Step 1: Update P and Q Fixing all the other variables and solving for P and Q, the problem becomes a classical matrix factorization problem:

$$\min_{U,V}\frac{1}{2}\sum_{s,c}(f_{s,c}-\mathbf{p}_s^\mathsf{T}\mathbf{q}_c)^2+\frac{\gamma}{2}(\sum_s\|p_s\|_2^2+\sum_c\|q_c\|_2^2) \tag{3.2}$$

where $f_{s,c}=g_{s,c}-\Delta(T-1)-\Delta(T-2)$ (See Eq 3.1). The matrix factorization problem can be solved using alternating minimization.

Step 2: Update A Fixing all the other variables and solving for A, the problem becomes

$$\begin{aligned}\min_{A} \quad & \frac{1}{2}\sum_{s,c}(g_{s,c}-\tilde{g}_{s,c})^2+\frac{\rho}{2}(\|A-Z_1\|_F^2+\|A-Z_2\|_F^2) \\ & +\rho(tr(U_1^{\mathsf{T}}(A-Z_1)))+\rho(tr(U_2^{\mathsf{T}}(A-Z_2))) \\ \text{s.t.,} \quad & A\geq 0\end{aligned}$$

Using the gradient descent, the elements in A can be updated as follows.

$$\begin{aligned}A(c_i,c_j) = & A(c_i,c_j)-lr\times[\rho(A(c_i,c_j)-Z_1(c_i,c_j)) \\ & +\rho(A(c_i,c_j)-Z_2(c_i,c_j))+\rho U_1(c_i,c_j)+\rho U_2(c_i,c_j) \\ & -\sum_{s,c_j}(g_{s,c_j}-\tilde{g}_{s,c_j}) \\ & \times\begin{cases}\frac{e^{-\alpha}}{|G_{T-1}(s,:)|}g_{s,c_i} & \text{(if } c_i \text{ is taken in term } T-1\text{)} \\ \frac{e^{-2\alpha}}{|G_{T-2}(s,:)|}g_{s,c_i} & \text{(if } c_i \text{ is taken in term } T-2\text{)]}\end{cases}\end{aligned} \tag{3.3}$$

with projection into $[0,+\infty)$, where lr is a learning rate.

Step 3: Update Z_1 and Z_2 For Z_1, the problem becomes

$$\min_{Z_1}\ \tau\|Z_1\|_*+\frac{\rho}{2}\|A-Z_1\|_F^2+\rho(tr(U_1^{\mathsf{T}}(A-Z_1))) \tag{3.4}$$

The closed-form solution of this problem is

$$Z_1 = S_{\frac{\tau}{\rho}}(A+U_1) \tag{3.5}$$

where $S_\alpha(X)$ is a soft-thresholding function that shrinks the singular values of X with a threshold α, that is,

$$S_\alpha(X) = U\text{diag}((\Sigma - \alpha)_+)V^\mathsf{T} \tag{3.6}$$

where $X = U\Sigma V^\mathsf{T}$ is the singular value decomposition of X, and

$$(x)_+ = \max(x, 0). \tag{3.7}$$

For Z_2, the problem becomes

$$\min_{Z_2} \lambda \|Z_2\|_{\ell_1} + \frac{\rho}{2}\|A - Z_2\|_F^2 + \rho(tr(U_2^\mathsf{T})(A - Z_2)) \tag{3.8}$$

The closed-form solution is

$$Z_2 = E_{\frac{\lambda}{\rho}}(A + U_2) \tag{3.9}$$

where $E_\alpha(X)$ is a soft-thresholding function that shrinks the values in X with a threshold α, that is,

$$E_\alpha(X) = (X - \alpha, 0)_+ \tag{3.10}$$

where $()_+$ is defined as in Equation 3.7.

Step 4: Update U_1 and U_2 U_1 and U_2 are updated based on standard ADMM updates:

$$U_1 = U_1 + (A - Z_1); \qquad U_2 = U_2 + (A - Z_2) \tag{3.11}$$

In addition, I conduct computational complexity analysis of MFTCI and put it in Appendix.

3.1.2 Computational Complexity Analysis

The computational complexity of MFTCI is determined by the four steps in the alternating approach as described above. To update U and V as in Equation 3.2 using gradient descent method via

alternating minimization, the computational complexity is $O(\text{niter}_{uv}(k \times n_{s,c} + k \times m + k \times n)) = O(\text{niter}_{uv}(k \times n_{s,c}))$ (typically $n_{s,c} \geq \max(m,n)$), where $n_{s,c}$ is the total number of student-course dyads, n is the number of students, m is the number of courses, k is the latent dimensions of U and V, and niter_{uv} is the number of iterations. To update A as in Equation 3.3 using gradient descent method, the computational complexity is upper-bounded by $O(\text{niter}_a(n_{cc} \times \frac{n_{s,c}}{m}))$, where n_{cc} is the number of course pairs that have been taken by at least one student, $\frac{n_{s,c}}{m}$ is the average number of students for a course, which upper bounds the average number of students who co-take two courses, and niter_a is the number of iteractions. Essentially, to update A, I only need to update $A(c_i, c_j)$ where c_i and c_j have been co-taken by some students. For $A(c_i, c_j)$ where c_i and c_j have never been taken together, they will remain 0. To update Z_1 as in Equation 3.4, a singular value decomposition is involved and thus its computational complexity is upper bounded by $O(m^3)$. To update Z_2 as in Equation 3.8, the computational complexity is $O(m^2)$. To update U_1 and U_2 as in Equation 3.11, the computational complexity is $O(m^2)$. Thus, the computational complexity for MTFCI is $O(\text{niter}(\text{niter}_{uv}(k \times n_{s,c}) + \text{niter}_a(n_{cc} \times \frac{n_{s,c}}{m}) + m^3 + m^2)) = O(\text{niter}(\text{niter}_{uv}(k \times n_{s,c}) + \text{niter}_a(n_{cc} \times \frac{n_{s,c}}{m}) + m^3))$, where niter is the number of iterations for the four steps. Although the complexity is dominated by m^3 due to the SVD on $A + U_1$, since n (i.e., the number of courses) is typically not large, the run time will be more dominated by $n_{s,c}$ (i.e., the number of student-course dyads).

3.2 Experiments

3.2.1 Dataset Description

In this chapter, I use the data of six large and diverse majors for both non-transfer and transfer students from GMU. These majors include: (i) Applied Information Technology (AIT), (ii) Biology (BIOL), (iii) Civil, Environmental and Infrastructure Engineering (CEIE), (iv) Computer Engineering (CPE) (v) Computer Science (CS) and (vi) Psychology (PSYC). Table 3.1 provides more information about these datasets.

Table 3.1: Dataset Descriptions

Major	Non-Transfer Students			Transfer Students		
	#S	#C	#(S,C)	#S	#C	#(S,C)
AIT	239	453	5,739	982	465	14,396
BIOL	1,448	990	33,527	1,330	833	22,691
CEIE	393	642	9,812	227	305	4,538
CPE	340	649	7,710	91	219	1,614
CS	908	818	18,376	480	464	7,967
PSYC	911	874	22,598	1504	788	24,661
Total	4,239	1,115	97,762	4,614	1,019	75,867

#S, #C and #S-C are number of students, courses and student-course pairs in educational records across the 6 majors from Fall 2009 to Spring 2016, respectively.

3.2.2 Data Preprocessing

MFTCI predicts student s's grade on course c in next term based on the courses taken by student s in previous two terms. In effect, this will naturally raise the cold start problems [25] for next-term grade prediction. Accordingly, I conduct a data preprocessing step to exclude such problems. In detail, I exclude the students who enroll in the program for less than three terms. Moreover, if student s has taken the course pair c, c' or c, c'' which have not appeared before, I exclude such scenarios as well, i.e., I will not predict course c's grade for student s.

3.2.3 Baseline Methods

I compare the performance of the proposed method to several baseline approaches, including MF and MF-b which are described in Section 2.

Non-negative Matrix Factorization (NMF) [1]

I add non-negative constraints on matrix $\mathbf{P}$ and matrix $\mathbf{Q}$ in Equation 2.1. The non-negativity constraints allows MF approaches to have better interpretability and accuracy for non-negative data [79].

Table 3.2: Comparison Performance with PTA (%)

Methods	Spring 2016			Fall 2015			Spring 2015		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
MF-b	13.25	27.71	58.02	12.05	26.63	58.89	13.03	26.09	54.83
MF	16.52	31.65	57.46	15.51	30.03	55.64	15.53	29.53	54.94
NMF	13.21	27.04	57.18	15.33	30.12	56.15	15.56	29.23	54.93
MFTCI	**19.78**	**35.52**	**61.44**	**19.71**	**35.16**	**60.12**	**18.56**	**32.78**	**58.80**

i) "↑" indicates the higher the better. ii) Reported values of PTA_0, PTA_1 and PTA_2 are percentages. iii) Best performing methods are highlighted with bold.

3.3 Results and Discussion

3.3.1 Overall Performance

Table 3.2 presents the comparison of PTA_0, PTA_1 and PTA_2 for non-transfer students for the three terms considered as test: Spring 2016, Fall 2015 and Spring 2015. I observe that the MFTCI model outperforms the baselines across the different test sets. On average, MFTCI outperforms the MF-b, MF and NMF methods by 34.18%, 11.59% and 4.08% in terms of PTA_0, 16.64%, 7.96% and 4.03% in terms of PTA_1, and 2.10%, 3.00% and 1.98% in terms of PTA_2, respectively. I observe similar results for transfer students as well (not included here for brevity).

Table 3.3 presents the performance of the baselines and MFTCI model for the three different terms of both non-transfer and transfer students using RMSE and MAE as evaluation metrics. The MFTCI model consistently outperforms the baselines across the different datasets in terms of MAE. In addition, the results shows that MF, NMF and MFTCI tend to have better performance for Spring 2016 term than Fall 2015 term. Similar trend is observed between Fall 2015 term and Spring 2015 term. This suggests that MFTCI is likely to have better performance with more information in the training set.

Table 3.3: Comparison Performance with RMSE and MAE.

Methods	Non-Transfer Students					
	Spring 2016		Fall 2015		Spring 2015	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
MF-b	0.999	0.754	1.037	0.786	1.023	0.784
MF	0.929	0.714	0.977	0.752	1.014	0.778
NMF	1.020	0.769	**0.967**	0.746	**1.000**	0.771
MFTCI	**0.928**	**0.685**	0.982	**0.717**	1.012	**0.750**
Methods	Transfer Students					
	Spring 2016		Fall 2015		Spring 2015	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
MF-b	0.925	0.688	**0.921**	0.686	0.985	0.732
MF	0.893	0.668	0.944	0.705	1.011	0.765
NMF	0.906	0.683	0.932	0.701	**0.979**	0.746
MFTCI	**0.887**	**0.636**	0.927	**0.662**	1.000	**0.721**

Table 3.4: Comparison Performance for Different Majors

	Methods	AIT	BIOL	CEIE	CPE	CS	PSYC
PTA_0	MF-b	18.71	18.00	15.99	12.99	15.98	20.18
	MF	19.45	22.10	16.70	14.21	16.47	22.12
	NMF	19.77	22.16	**17.01**	14.32	16.61	22.17
	MFTCI	**22.30**	**24.24**	16.80	**14.32**	**17.32**	**25.83**
PTA_1	MF-b	37.95	35.43	31.47	27.86	31.53	39.41
	MF	37.21	39.68	31.87	**27.97**	30.51	39.63
	NMF	36.79	39.74	31.67	27.19	30.43	39.36
	MFTCI	**39.64**	**40.87**	**32.38**	27.53	**31.78**	**42.29**
PTA_2	MF-b	**67.02**	67.78	58.66	52.28	56.91	**71.01**
	MF	66.17	67.54	58.35	50.72	56.24	67.74
	NMF	66.70	67.54	58.55	51.17	56.17	67.79
	MFTCI	66.70	**68.25**	**58.76**	**52.94**	**58.18**	68.29

3.3.2 Analysis on Individual Majors

I divide non-transfer students based on their majors and test the baselines and MFTCI model on each major, separately. Table 3.4 shows the comparison of PTA_0, PTA_1 and PTA_2 on different majors. The results show that MFTCI has the best performance for almost all the majors. Among all the

results, MFTCI has the highest accuracy when predicting grades for PSYC and BIOL students for which I have more student-course pairs in the training set.

3.3.3 Effects from Previous Terms on MFTCI

In order to see the influence of number of previous terms considered in MFTCI, I run the proposed model with only $\Delta(T-1)$ in Equation 3.1. This method is represented as MFTCI$_{p1}$. Figure 3.1 shows the comparison results of MAE for six subsets of data which are reported in Table 3.3, where "NTR" stands for non-transfer students and "TR" stands for transfer students. The results show that MFTCI consistently outperforms MFTCI$_{p1}$ on all datasets. This suggests that considering two previous terms is necessary for achieving good prediciton results. Moreover, since I consider that the student's knowledge is modeled using an exponential decaying function over time, I do not include the influence from the third previous term in the model as its influence for the grade prediction is negligible in comparison to the previous two terms.

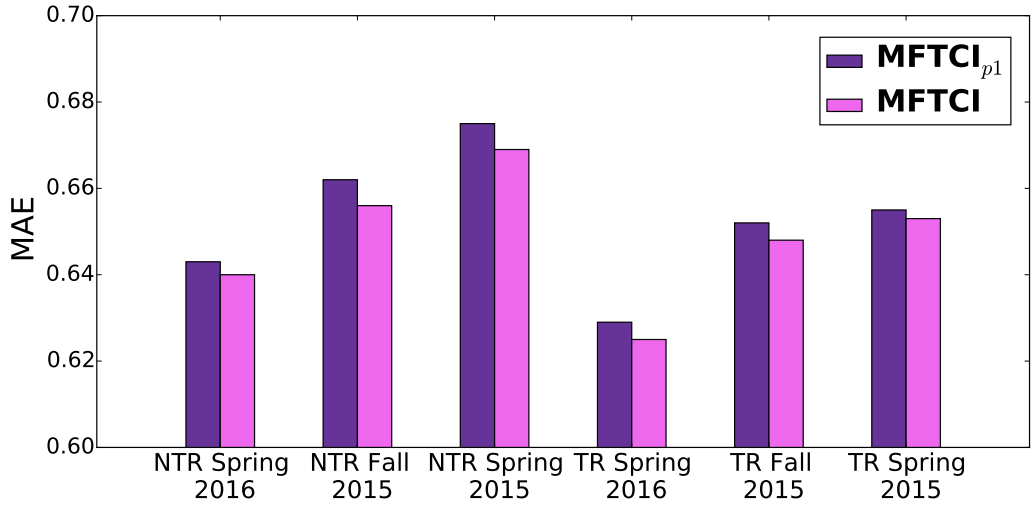


Figure 3.1: Comparison performance for MFTCI$_{p1}$ and MFTCI

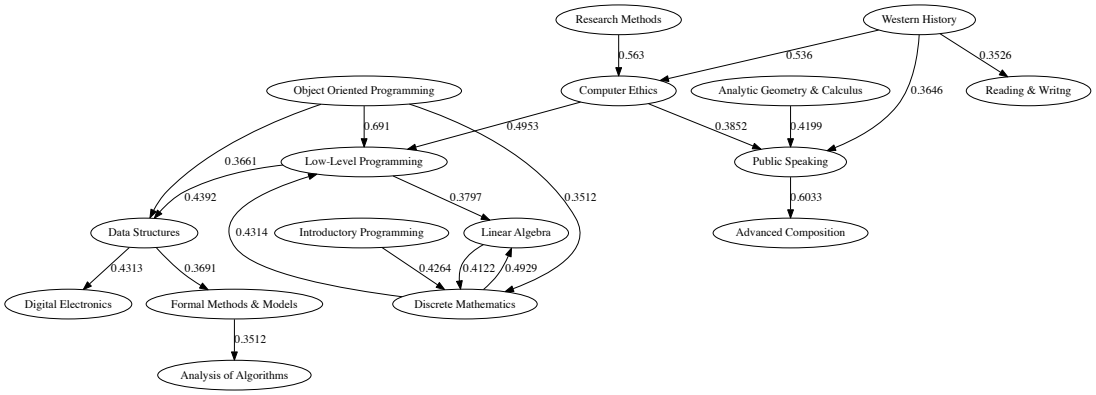


Figure 3.2: Identified course influences for CS major

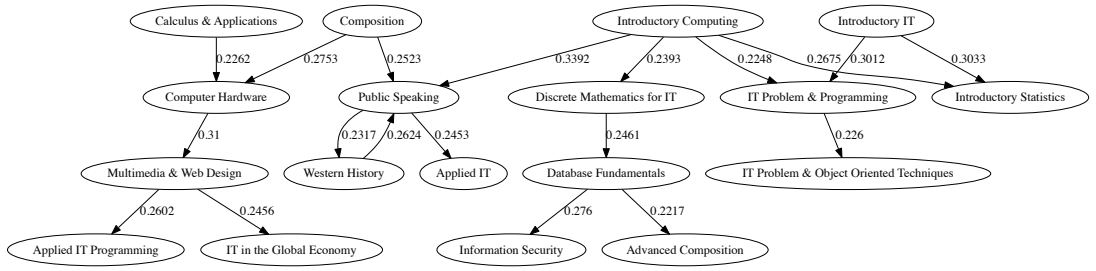


Figure 3.3: Identified course influences for AIT major

3.3.4 Visualization of Course Influence

To interpret what is captured in the course influence matrix A (See Eq 3.1), I extract the top 20 values with the corresponding course names (and topics) for analysis. Figure 3.2 and 3.3 show the captured pairwise course influences for CS and AIT majors, respectively. Each node corresponds to one course which is represented by the shortened course's name. I can notice from the figures that most influences reflect content dependency between courses. For example, in the CS major, "Object Oriented Programming" course has significant influence on performance of "Low-Level Programming" course (the former one is also the latter one's prerequisite course); "Linear Algebra" and "Discrete Mathematics" have influence on each other; "Formal Methods & Models" course

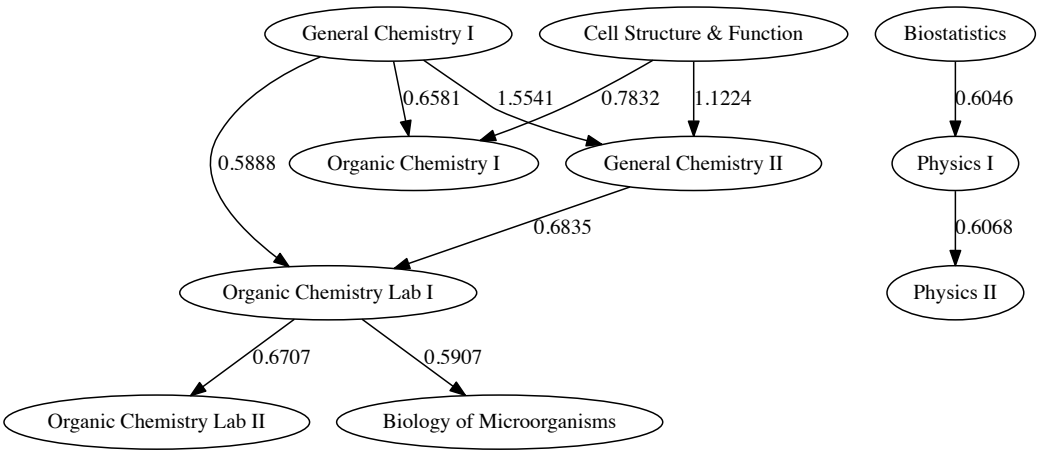


Figure 3.4: Identified course influences for BIOL major

has influence on "Analysis of Algorithms" course. In case of the AIT major, both "Introductory IT" course and "Introductory Computing" course have influence on "IT Problem & Programming" course; "Multimedia & Web Design" course has influence on both "Applied IT Programming" course and "IT in the Global Economy" course. GMU has a sample schedule of eight-term courses for each major in order to guide undergraduate students to finish their study step by step based on the level, content and difficulty of courses [1]. Among the identified relationships shown in Figures 3.2 and 3.3 I found 17 and 13 of the CS and AIT courses influences in the guide map, respectively. The rest of the identified influences are among other general electives but required courses (e.g., "Public Speaking" course), or specific electives pertaining to the major (e.g., "Research Methods" course). This shows that the proposed model learns meaningful course-wise influences and successfully uses it to improve MF model.

Figure 3.4 to 3.7 show the identified course influences for the BIOL, CEIE, CPE and PSYC majors. These identified course-wise influences seem to capture similarity of course content.

[1] http://catalog.gmu.edu

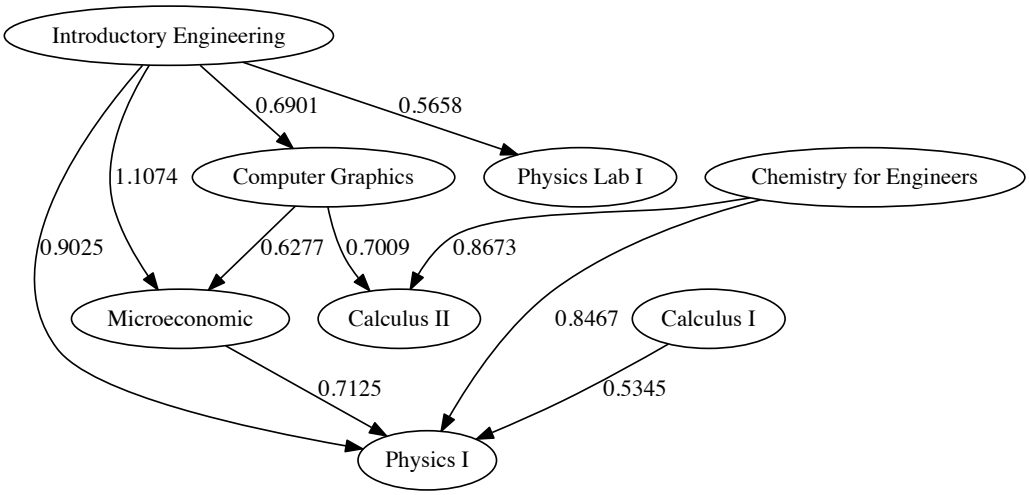


Figure 3.5: Identified course influences for CEIE major

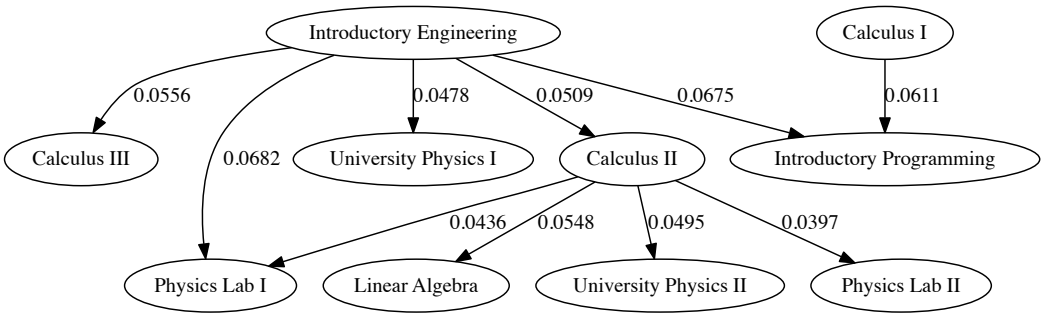


Figure 3.6: Identified course influences for CPE major

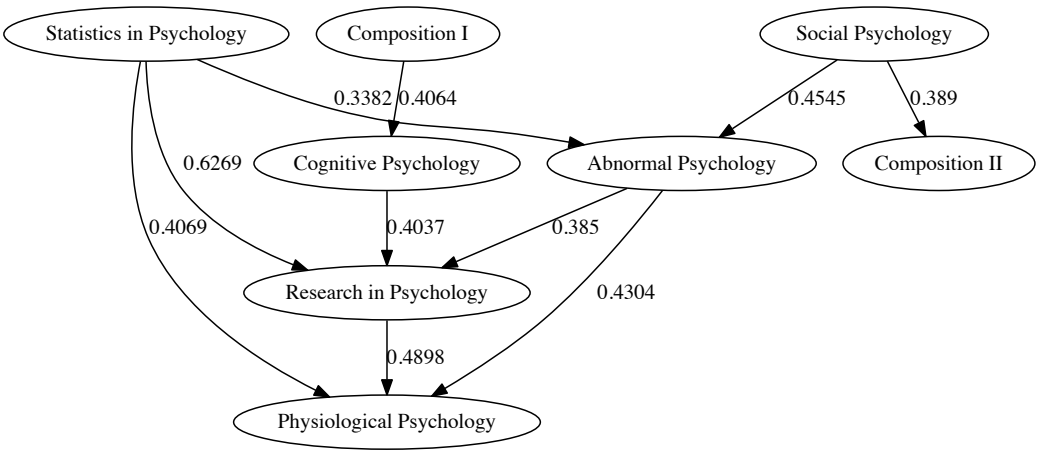


Figure 3.7: Identified course influences for PSYC major

3.4 Summary

I presented a Matrix Factorization with Temporal Course-wise Influence (MFTCI) model that integrates factorization models and the influence of courses taken in the preceding terms to predict student grades for the next term.

I evaluate the proposed model on the student educational records from Fall 2009 to Spring 2016 collected from George Mason University. The dataset in this study contains both non-transfer and transfer students from six different majors. The experimental evaluation shows that MFTCI consistently outperforms the different state-of-the-art methods. Moreover, I analyze the effects from previous terms on MFTCI, and I make the conclusion that it is necessary to consider two previous terms. In addition, I visualize the patterns learned between pairs of courses. The results strongly demonstrate that the learned course influences correlate with the course content within academic programs.

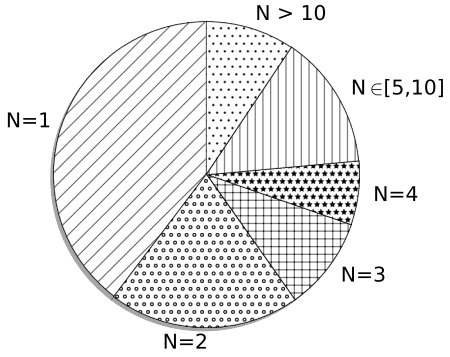
Chapter 4: ALE: Additive Latent Effect Models for Grade Prediction

Existing grade prediction methods often have a narrow focus on the potential influential factors. For example, course instructors, course difficulty, student's interest, capability and effort are rarely considered.

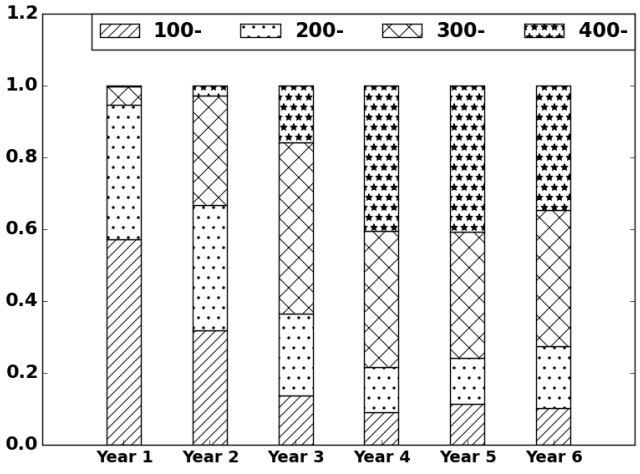
Data distribution: Fig. 4.1a shows the distribution of the number of instructors who teach the same course at George Mason University. More than 60% of the courses at this university have been taught by multiple instructors in a period of 18 terms. For a given course, different instructors differ in their course offerings with respect to coverage of course topics, pedagogy and grading criterion. All these factors impact a student's grade in a course. As such, I propose to model latent factors associated with each instructor in addition to the latent factors of the course she teaches. Fig. 4.1b shows the distribution of academic course levels at George Mason University (i.e., 100-,200-,300- and 400-level) offered to the students in different starting years. I assume that students in the same college terms (e.g., freshmen, sophomore, etc) tend to have similar learning behaviors, capabilities and expertise given the sequential aspects of most degree programs. For example, freshmen students may be undecided on their majors and mostly take courses with level 100, as shown in Fig.4.1b. Likewise, seniors tend to have an in-depth knowledge of study in a specific field, and mostly take higher level courses.

In this thesis, I propose **A**dditive **L**atent **E**ffect (**ALE**) models within the framework of MF to predict the grade that a student is expected to obtain in a course that she may enroll in the next term. Inspired by Morsy *et al.* [13], the proposed methods model each student's latent factors with accumulated knowledge of a sequence of courses taken by the student, jointly with the grade for each course. Furthermore, I incorporate course instructor and student academic level effects along with student global latent factor to enable accurate grade prediction.

Prior work in the RS literature that shares similarities with one of the proposed methods is from Koenigstein *et al.* [80]. In this work, the authors proposed a music rating recommendation system



(a) Distribution of #instructors over courses



(b) % of Students Enrolled in Different Course Levels based on their Current College Year

Figure 4.1: Course-Student Data Distributions

that models a user's music preferences based on her interest in a given music track, and the artist and album information associated with the specific track. Shared factor components were introduced to reflect the similar preference for music tracks of same artists (or genre, album). I have discussed the domain-aware grade prediction method [11] and cumulative knowledge-based regression model [13] in Section 2 in detail. These two models serve as foundations for the proposed formulation.

Finally, I conducted a comprehensive set of experiments on various datasets and provided a thorough analysis on the importance of different factors. The experimental results show that the proposed methods achieve superior prediction performance on various test datasets for next-term grade prediction. The work presented in this chapter has been published in SIAM International Conference on Data Mining (SDM18).

4.1 Preliminaries

Formally, student-course grades will be represented by $\{G_1, G_2, ..., G_N\}$ for N terms. G_t contains the set of tuples storing grade information for all students enrolled in courses within term t. Each tuple stores: (i) student identifier, (ii) course identifier, (iii) student academic level, (iv) course

Table 4.1: Notations

Notation	Explanation
$\mathbf{p}^t_{ck(s)}$	latent factors of accumulated knowledge of student s up to term t
$\mathbf{p}_{al(s)}$	latent factor of student s's academic level, $al(s) \in [1,12]$
$\mathbf{p}^t_{en(s)}$	integrated student latent factor
$\mathbf{q}_c$	latent factor of the knowledge components required by course c
$\mathbf{q}_{in(c)}$	latent factor of the instructor who teaches course c
$\mathbf{q}_{en(c)}$	integrated course latent factor
$\mathbf{p}_{g(s)}$	student s's global latent factor
$\mathbf{k}_c$	latent factor of the knowledge components provided by course c

instructor, and (v) grade obtained. Table 4.1 summarizes the key notations used in this chapter.

4.2 Additive Latent Effect Models (ALE)

I propose Additive Latent Effect (ALE) models to predict student s's performance on course c in term t. I will give a thorough presentation on how I model each effect in the following sections.

4.2.1 Student Academic Level Effect

Based on the assumption that students on a same academic level (i.e., freshmen, sophomore, junior and senior) have a similar level of academic maturity, experience, habits and knowledge, I model student s by integrating a factor associated to the college term she is attending, denoted as $\mathbf{p}_{al(s)^t}$, into the student accumulated knowledge factors. The integrated student latent factor is denoted as $\mathbf{p}^t_{en(s)}$, and is given as follows:

$$\mathbf{p}^t_{en(s)} = \mathbf{p}^t_{ck(s)} + \mathbf{p}_{al(s)^t}, \tag{4.1}$$

where $\mathbf{p}^t_{ck(s)}$ is calculated by Eq 2.6, and $al(s)^t$ represents the academic level of student s in term t, defined as $al(s)^t = t - (s\text{'s start term})$. Since most students finish college in 4-6 years (8-12 terms), $al(s)^t$ is in $[0,12)$. I include ℓ_1-norm regularization on $\mathbf{p}_{al(s)^t}$ to enforce sparsity on this representation. This is because $\mathbf{p}_{al(s)^t}$ aims to capture the academic factors (e.g., academic maturity), and student s is only able to hold a part of them on a particular academic level (e.g., student s cannot be both mature and immature at the same time).

4.2.2 Course Instructor Effect

Consider that a single course is often taught by multiple instructors who usually vary in their coverage of materials (topics), pedagogy, use of teaching technology, choice of assignments and grading criterion. I hypothesize that a student's performance on a specific course is greatly influenced by the instructor who teaches her the course. Specifically, for a course c, I add a factor associated with the specific instructor who teaches course c, denoted as $\mathbf{q}_{in(c)}$, to the original knowledge latent factors of course c. The integrated course latent factor is denoted by $\mathbf{q}_{en(c)}$, and is given as follows:

$$\mathbf{q}_{en(c)} = \mathbf{q}_c + \mathbf{q}_{in(c)}, \tag{4.2}$$

where $in(c)$ denotes the instructor who teaches course c. For $\mathbf{q}_{in(c)}$, I include ℓ_1-norm regularization to control its sparsity. I assume that an instructor is generally proficient only in certain topics (and knowledge components), but not all.

With the course instructor information and student academic level information as proposed above, the grade prediction for student s on course c is given as follows:

$$\tilde{g}^t_{s,c} = \mathbf{p}_{en(s)}{}^{\mathsf{T}} \mathbf{q}_{en(c)}. \tag{4.3}$$

4.2.3 Student Global Latent Factor

Eq. 4.3 captures student knowledge factors per term and captures the sequential dynamics in student's knowledge state over terms. This can be considered as a latent factor model localized by term. I propose to incorporate a term-agnostic global latent factor that captures the student-course performance interaction. I introduce an additional latent factor $\mathbf{p}_{g(s)}$ that captures the student's implicit information (e.g., student interest and subject matter mastery toward each knowledge component) in a common latent space as course knowledge components. The estimated grade of student s on course c at term t with this global latent factor is given as:

$$\tilde{g}^t_{s,c} = {\mathbf{p}^t_{en(s)}}^\top \mathbf{q}_{en(c)} + \mathbf{p}^\top_{g(s)} \mathbf{q}_c \tag{4.4}$$

Here, I compute the dot product of $\mathbf{p}_{g(s)}$ and $\mathbf{q}_c$ instead of $\mathbf{p}_{g(s)}$ and $\mathbf{q}_{en(c)}$ in this step. The exclusive l_1 norm for $\mathbf{p}_{g(s)}$ controls its sparsity since I assume most students have a tendency to perform well in a fraction of the represented knowledge states. I refer to this model as **A**dditive **L**atent **E**ffect (ALE).

4.2.4 Student and Course Bias Effect

Inspired by the success of MF methods with bias terms [25], I add student-specific and course-specific bias terms denoted by b_s and b_c within the CK and ALE formulation in Eq 2.6 and 4.4, respectively, as follows:

$$\tilde{g}^t_{s,c} = \frac{1}{|G^{t-1}_s|} {\mathbf{p}^t_{ck(s)}}^\top \mathbf{q}_c + b_s + b_c \tag{4.5}$$

and

$$\tilde{g}^t_{s,c} = {\mathbf{p}^t_{en(s)}}^\top \mathbf{q}_{en(c)} + \mathbf{p}^\top_{g(s)} \mathbf{q}_c + b_s + b_c. \tag{4.6}$$

Table 4.2: Method Summarization

	Method	Prediction Formulation	Property			
			ck	b_s	b_c	al,in,g
Baselines	MF	$\mathbf{p}_s^\top \mathbf{q}_c$ (Eq. 2.1)	✗	✗	✗	✗
	MF-b	$\mathbf{p}_s^\top \mathbf{q}_c + b_s + b_c$ (Eq. 2.2)	✗	✓	✓	✗
	MF-d	$\mathbf{p}_s^\top \mathbf{q}_c + b_s^{\varphi(c)} + b_c^{\varphi(s)}$ (Eq. 2.3)	✗	✓	✓	✗
	CK	$\frac{1}{\lvert G_s^{t-1}\rvert}{\mathbf{p}_{ck(s)}^t}^\top \cdot \mathbf{q}_c$ (Eq. 2.6)	✓	✗	✗	✗
	CK-b	$\frac{1}{\lvert G_s^{t-1}\rvert}{\mathbf{p}_{ck(s)}^t}^\top \cdot \mathbf{q}_c$ $+b_s + b_c$ (Eq. 4.5)	✓	✓	✓	✗
Proposed Methods	ALE	$(\mathbf{p}_{ck(s)}^t + \mathbf{p}_{al(s)})^\top (\mathbf{q}_c + \mathbf{q}_{in(c)})$ $+\mathbf{p}_{g(s)}^\top \mathbf{q}_c$ (Eq. 4.4)	✓	✗	✗	✓
	ALE-b	ALE $+b_s + b_c$ (Eq. 4.6)	✓	✓	✓	✓

"✓" indicates the method contains the corresponding property, and "✗" indicates the opposite. al, in, g indicate student academic level, course instructor and student global latent factor, respectively. b_s and b_c denote student and course bias terms.

I denote the CK with bias terms as CK-b, and ALE with bias terms as ALE-b.

Table 4.2 summarizes the proposed methods and comparative baselines in terms of their key features and effect-components considered in this study.

4.2.5 Optimization for ALE

The optimization problem for ALE can be formulated as follows:

$$\min_{\Theta} L(\Theta) + R(\Theta), \tag{4.7}$$

where Θ represents model parameters (i.e., the latent factors), $L(\Theta)$ is the loss function and $R(\Theta)$ is the regularization function. I use a squared error loss function in ALE:

$$L(\Theta) = \sum_{g_{s,c}^t \in G_s^{t-1}} (g_{s,c}^t - \tilde{g}_{s,c}^t(\Theta))^2 \tag{4.8}$$

Algorithm 1 *ALE*: Learn

procedure ALE_LEARN
  Initialize $\mathbf{k}_c$, $\mathbf{q}_c$ for each c, $\mathbf{p}_g$ for each student, $\mathbf{p}_{al}$ for each academic level and $\mathbf{q}_{in}$ for each instructor with random values in (0, 1)
  $\eta \leftarrow$ learning rate
  $\gamma, \alpha_1, \alpha_2 \leftarrow$ regularization weight
  $iter \leftarrow 0$
  while iter<maxIter and MAE decreases **do**
    for all $g^t_{s,c} \in \mathrm{G}^{t-1}_s$ **do**
      $\mathbf{p}_{ck(s)} \leftarrow 0$
      for all $c' \in C_s$ **do**
        $\mathbf{p}_{ck(s)} \leftarrow \mathbf{p}_{ck(s)} + e^{-\lambda(t_{s,c}-t_{s,c'})}\mathbf{k}_{c'} \cdot g^{t_{s,c'}}_{s,c'}$
      $\tilde{g}^t_{s,c} \leftarrow (\mathbf{p}_{ck(s)} + \mathbf{p}_{al(s)})^{\mathsf{T}}(\mathbf{q}_c + \mathbf{q}_{in(c)}) + \mathbf{p}^{\mathsf{T}}_{g(s)}\mathbf{q}_c$
      $e^t_{s,c} = g^t_{s,c} - \tilde{g}^t_{s,c}$
      for all $c' \in C_s$ **do**
        $\mathbf{k}_{c'} \leftarrow \mathbf{k}_{c'} +$
          $\eta((\mathbf{q}_c + \mathbf{q}_{in(c)}) \cdot e^{-\lambda(t_{s,c}-t_{s,c'})} \cdot g^{t_{s,c'}}_{s,c'} \cdot e^t_{s,c} - \gamma \cdot \mathbf{k}_{c'})$
      $\mathbf{p}_{al(s)} \leftarrow \mathbf{p}_{al(s)} + \eta((\mathbf{q}_c + \mathbf{q}_{in(c)}) \cdot e^t_{s,c} - \gamma \cdot \mathbf{p}_{al(s)} - \alpha_1)$
      $\mathbf{q}_c \leftarrow \mathbf{q}_c + \eta(((\mathbf{p}_{ck(s)} + \mathbf{p}_{al(s)}) + \mathbf{p}_{g(s)}) \cdot e^t_{s,c} - \gamma \cdot \mathbf{q}_c)$
      $\mathbf{q}_{in(c)} \leftarrow \mathbf{q}_{in(c)} +$
        $\eta((\mathbf{p}_{ck(s)} + \mathbf{p}_{al(s)}) \cdot e^t_{s,c} - \gamma \cdot \mathbf{q}_{in(c)} - \alpha_1)$
      $\mathbf{p}_{g(s)} \leftarrow \mathbf{p}_{g(s)} + \eta(\mathbf{q}_c \cdot e^t_{s,c} - \gamma \cdot \mathbf{p}_{g(s)} - \alpha_2)$
    $iter \leftarrow iter + 1$
return $\mathbf{k}_c$, $\mathbf{q}_c$, $\mathbf{p}_g$, $\mathbf{p}_{al}$ and $\mathbf{q}_{in}$

The $R(\Theta)$ is defined as follows:

$$
\begin{aligned}
R(\Theta) = \sum_{g^t_{s,c} \in G^{T-1}} \Bigg[& \frac{\gamma}{2}(||\mathbf{k}_c||_2 + ||\mathbf{q}_c||_2 \\
& + ||\mathbf{p}_{al(s)}||_2 + ||\mathbf{q}_{in(c)}||_2 + ||\mathbf{p}_{g(s)}||_2) \\
& + \alpha_1(||\mathbf{p}_{al(s)}||_1 + ||\mathbf{q}_{in(c)}||_1) + \alpha_2||\mathbf{p}_{g(s)}||_1 \Bigg]
\end{aligned}
\tag{4.9}
$$

I use stochastic gradient descend (SGD) to solve the optimization problem. The optimization algorithm is presented in Algorithm 1.

4.2.6 Computational Complexity Analysis

The computational complexity of ALE is determined by the steps from line 6 to line 19 in Algorithm 1. In detail, the computational complexity for line 9 and line 10 is upper-bounded by $O(m_c \times k)$, where m_c is the maximum number of courses that a student can take in college. For line 11 and line 12, the computational complexity is $O(k^2)$. Line 13 and line 14 have complexity $O(m_c \times k)$ as well. From line 15 to line 18, the total computational complexity is $O(4 \times k)$. Thus, the computational complexity for ALE is $O(n_{iter} \times n_g \times (2m_c \times k + k^2 + 4 \times k))$, where n_{iter} is the number of iterations, n_g is the total number of student-course grades, m_c is the maximum number of courses that a student can take, and k is the dimension of latent factors. Typically, $m_c > k$ and thus the complexity is $O(n_{iter} \times n_g \times m_c \times k)$.

4.3 Experiments

4.3.1 Dataset Description

In this chapter, I evaluated the proposed models on eight large and diverse majors from GMU including: (i) Mathematical Sciences (MATH), (ii) Physics (PHYS), (iii) Chemistry (CHEM) (iv) Computer Science (CS), (v) Civil, Environmental and Infrastructure Engineering (CEIE), (vi) Biology (BIOL), (vii) Psychology (PSYC), and (viii) Applied Information Technology (AIT). Table 4.3 presents the details about these majors.

4.3.2 Data Preprocessing

ALE predicts student s's grade on course c in next term based on the courses taken by student s in previous terms, student's academic levels, and course instructors. In order to prevent cold start problems [25], I conduct a data preprocessing step. In detail, I exclude the students who enroll in the program for less than two terms. Moreover, if student s has taken course c' before course c which have not appeared before, I exclude such scenarios as well, i.e., I will not predict course c's grade for student s. Furthermore, if the instructor for course c has not taught other courses before, I will not predict course c's grade for student s.

Table 4.3: Dataset Statistics

Major	FTF Students			TR Students		
	#S	#C	#S-C	#S	#C	#S-C
MATH	209	84	2,846	258	91	2,580
PHYS	127	53	1,830	74	48	854
CHEM	342	55	4,649	278	66	3,105
CS	988	76	13,809	554	68	7,028
CEIE	428	80	6,925	248	92	4,036
BIOL	1,629	109	21,519	1,525	115	16,615
PSYC	1,114	95	14,377	1,749	114	18,939
AIT	334	82	6088	1,170	90	15,060
Total	5,171	634	72,043	5,856	684	68,216

#S, #C and #S-C are number of students, courses and student-course grades from Fall 2009 to Spring 2016, respectively.

4.3.3 Parameter Learning

The parameters in the optimization problem (Eq 4.7) contain the number of latent dimensions (i.e., k), regularization weights (i.e., γ, α_1, α_2), and time decay parameter (i.e., λ). I use a validation set to select parameters. Specifically, for test term T, I have student-course grades up to term $T-1$ as the training set, i.e., G^{T-1}. Then I split the training set into two parts: G^{T-2} and G_{T-1}, the latter of which I consider as the validation set. I did a grid search over the parameters and selected the parameters that perform best on the validation set.

4.4 Results and Discussion

4.4.1 Overall Performance

Table 4.4 shows the comparison of MAE and PTA results for FTF and TR students across Spring 2016, Fall 2015 and Spring 2015 test terms. In Table 4.4, Columns under "parameters" indicate different model parameters for the corresponding methods. Specifically, for MF, MF-b and MF-d, the parameter is the dimension of latent factors, (k). For CK and CK-b, the parameters are the dimension of latent factors, (k), and time-decay coefficient, (λ). For ALE and ALE-b, the parameters are time-decay coefficient, (λ), regularization weight for $\mathbf{p}_{al(s)}$ and $\mathbf{q}^t_{in(c)}$, (α_1), and regularization

Table 4.4: Performance Comparison for All Methods

Method	FTF - Spring 2016							TR - Spring 2016						
	parameters			MAE	PTA_0	PTA_1	PTA_2	parameters			MAE	PTA_0	PTA_1	PTA_2
MF	10	–	–	0.723	0.188	0.338	0.580	10	–	–	0.706	0.188	0.341	0.601
MF-b	10	–	–	0.670	0.206	0.360	0.609	10	–	–	0.658	0.226	0.387	0.628
MF-d	5	–	–	0.661	0.221	0.381	0.621	10	–	–	0.683	0.216	0.366	0.614
CK	5	0.01	–	0.674	0.216	0.362	0.604	5	0.01	–	0.680	0.225	0.369	0.597
CK-b	5	0.01	–	0.647	0.218	0.379	0.625	5	0.01	–	0.658	0.227	0.387	0.627
ALE	0.01	0.01	0.1	**0.625**	**0.255**	**0.416**	**0.651**	0.01	0.001	0.1	0.645	**0.247**	**0.395**	**0.651**
ALE-b	0.1	0.01	0.1	**0.625**	0.225	0.389	0.648	0.01	0.1	0.01	**0.642**	0.231	0.389	0.637

Method	FTF - Fall 2015							TR - Fall 2015						
	parameters			MAE	PTA_0	PTA_1	PTA_2	parameters			MAE	PTA_0	PTA_1	PTA_2
MF	10	–	–	0.730	0.177	0.317	0.574	10	–	–	0.692	0.183	0.347	0.599
MF-b	10	–	–	0.691	0.205	0.360	0.605	10	–	–	0.653	0.213	0.378	0.631
MF-d	10	–	–	0.693	0.216	0.370	0.610	10	–	–	0.670	0.205	0.362	0.630
CK	5	0.01	–	0.706	0.193	0.347	0.585	5	0.01	–	0.665	0.210	0.372	0.616
CK-b	5	0.01	–	0.690	0.195	0.351	0.603	5	0.01	–	0.642	0.227	0.394	0.641
ALE	0.1	0.001	0.05	**0.654**	**0.251**	**0.400**	**0.638**	0.01	0.001	0.05	**0.615**	**0.243**	**0.418**	**0.670**
ALE-b	0.01	0.01	0.1	0.660	0.223	0.379	0.634	0.01	0.01	0.1	0.627	0.216	0.392	0.655

Method	FTF - Spring 2015							TR - Spring 2015						
	parameters			MAE	PTA_0	PTA_1	PTA_2	parameters			MAE	PTA_0	PTA_1	PTA_2
MF	10	–	–	0.760	0.168	0.306	0.547	10	–	–	0.743	0.169	0.316	0.559
MF-b	10	–	–	0.718	0.186	0.335	0.582	10	–	–	0.688	0.218	0.368	0.607
MF-d	10	–	–	0.716	0.215	0.358	0.595	10	–	–	0.693	0.229	0.383	0.618
CK	5	0.01	0.01	0.712	0.192	0.332	0.579	5	0.01	0.01	0.705	0.214	0.357	0.589
CK-b	5	0.01	0.01	0.690	0.203	0.354	0.599	5	0.01	0.01	0.688	0.207	0.354	0.606
ALE	0.01	0.001	0.1	**0.649**	**0.244**	**0.403**	**0.639**	0.01	0.001	0.1	**0.644**	**0.254**	**0.417**	**0.647**
ALE-b	0.01	0.01	0.1	0.657	0.214	0.372	0.618	0.1	0.01	0.01	0.653	0.226	0.383	0.631

weight for $\mathbf{p}_{g(s)}$, (α_2). Bold numbers are the best performing results. In the experiments, I select the best performance for all baseline methods. In the reported results, MF-d [11] has nine different combinations for student- and course-level groupings and has bias. I tried all the proposed combinations and report the best performance among all the results. The results show that ALE has the best performance on all the evaluation metrics (the only exception is in Spring 2016 on MAE). Specifically, ALE outperforms the baseline methods on PTA_0, PTA_1 and PTA_2 by 10.61%, 7.17% and 4.50%, respectively. I also observe that the improvement in performance of ALE over the baseline approaches is greater for Spring 2015 in comparison to Spring 2016, even though the training set for Spring 2015 is smaller than Spring 2016. This shows that ALE can overcome the scarcity issues in a dataset and yield good prediction performance.

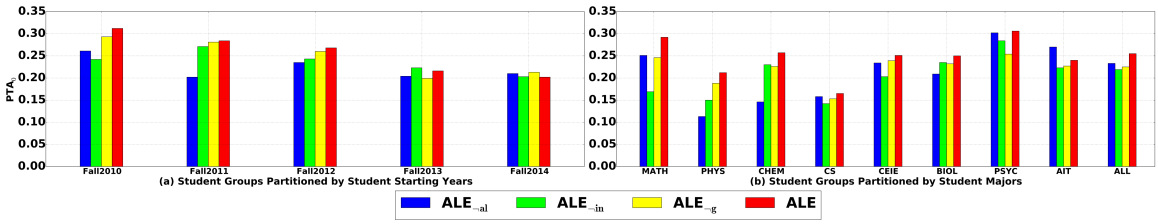


Figure 4.2: Comparison of PTA_0 with Each Effect Removed on Various Student Groups in ALE

4.4.2 Effects of Bias Terms

For all the datasets in Table 4.4, MF-b (i.e., MF with student/course-specific bias terms) and MF-d (i.e., MF with domain-aware biases) always outperform MF (i.e., MF without bias terms). In addition, MF-b achieves better PTA_0 on TR students, but worse PTA_0 on FTF students than MF-d. This is probably because the FTF students show consistent characteristics in comparison to TR students, who typically have more diverse backgrounds.

In Table 4.4, I also observe that CK-b consistently outperforms CK, similar to the comparison between MF-b and MF, but ALE always outperforms ALE-b. This may indicate that the additive latent effects in ALE have also captured the student and course bias information in ALE-b. The results in Table 4.4 demonstrate that ALE is able to achieve better prediction performance without explicitly modeling student and course biases.

4.4.3 Importance of Additive Latent Effects

In order to learn the importance of each additive latent effect, I perform a study to assess the prediction performance of different ALE models with a particular latent effect removed. Table 4.5 shows the details of the compared models in this experiment. Fig. 4.2 represents the PTA_0 performance of each ALE model variant. I test the results on various student groups partitioned by student starting years and student majors, as shown in Fig. 4.2a and Fig. 4.2b, respectively. I also implement the experiment for the whole test set, i.e. G_T, and present the results with label "ALL" in Fig. 4.2b.

Fig. 4.2 shows that for most student groups, ALE outperforms the other models, indicating that

Table 4.5: Comparison Method Summarization

Method	Prediction Formulation	Property		
		al	*in*	*g*
ALE$_{\neg al}$	$\mathbf{p}^{t}_{ck(s)}{}^{\mathsf{T}}(\mathbf{q}_c + \mathbf{q}_{in(c)}) + \mathbf{p}^{\mathsf{T}}_{g(s)}\mathbf{q}_c$	✗	✓	✓
ALE$_{\neg in}$	$(\mathbf{p}^{t}_{ck(s)} + \mathbf{p}_{al(s)})^{\mathsf{T}}\mathbf{q}_c + \mathbf{p}^{\mathsf{T}}_{g(s)}\mathbf{q}_c$	✓	✗	✓
ALE$_{\neg g}$	$(\mathbf{p}^{t}_{ck(s)} + \mathbf{p}_{al(s)})^{\mathsf{T}}(\mathbf{q}_c + \mathbf{q}_{in(c)})$	✓	✓	✗
ALE	$(\mathbf{p}^{t}_{ck(s)} + \mathbf{p}_{al(s)})^{\mathsf{T}}(\mathbf{q}_c + \mathbf{q}_{in(c)})$ $+\mathbf{p}^{\mathsf{T}}_{g(s)}\mathbf{q}_c$ (Eq. 4.4)	✓	✓	✓

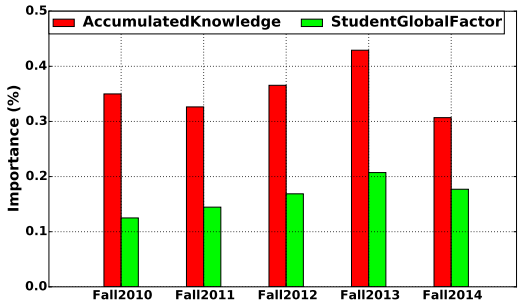
al, *in* and *g* indicate the property of student academic level, course instructor and student global latent factor, respectively. ✓ indicates the model contains the corresponding property, ✗ indicates the opposite.

each additive latent effect plays an important role in ALE. Specifically, Fig. 4.2a shows that for students who start school in Fall 2011, the PTA$_0$ of ALE$_{\neg al}$ (without the academic level) drops the most compared to other models. This shows that the student academic level is the most important effect for this student group. Moreover, for students who start school in Fall 2013 and Fall 2014, ALE does not outperform all the other models. This indicates that for these two student groups, it is not necessary to consider all the latent effects when predicting their grades.

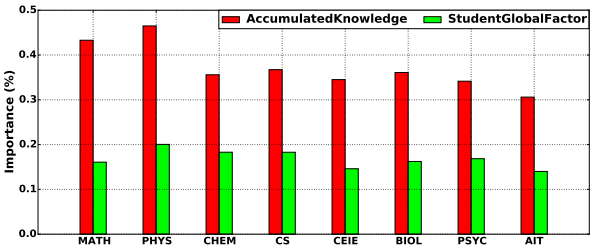
From Fig. 4.2b, I notice that for students in MATH, CS and AIT major, ALE$_{\neg in}$ has the worst PTA$_0$ results, indicating the course instructor associated latent factor is the most important for grade prediction. While, for students in PHYS, CHEM and BIOL majors, student academic level is the most important effect. Moreover, Fig. 4.2b also shows that for all the students ("ALL"), course instructor and student global latent factor are more important than student academic level effect. The ALE outperforms the other three variants. For students with different majors and academic levels, all three effects are important in providing an accurate grade prediction.

4.4.4 Importance of Accumulated Knowledge and Student Global Latent Factor

Students need help in course selections both in order to gain course credits and learn the knowledge and skills contained in the course. In ALE, accumulated knowledge and student global latent factor



(a) Student Groups Partitioned by Student Starting Years



(b) Student Groups Partitioned by Student Majors

Figure 4.3: The Importance of Student's Accumulated Knowledge and Student Global Latent Factor on Various Student Groups

are the two effects that are directly related to the students. Learning the importance of these two factors can assist students when they choose a course.

Specifically, I calculate the importance of each factor by averaging the proportion of its contribution in all the predicted grades within the test set as follows:

$$I_{ck} = \frac{1}{|G_T|} \sum_{g^T_{s,c} \in G_T} \frac{\mathbf{p}^T_{ck(s)}}{\tilde{g}^T_{s,c}} \tag{4.10}$$

and

$$I_g = \frac{1}{|G_T|} \sum_{g^T_{s,c} \in G_T} \frac{\mathbf{p}^{\mathsf{T}}_{g(s)} \mathbf{q}_c}{\tilde{g}^T_{s,c}} \tag{4.11}$$

where I_{ck} and I_g represent the importance of accumulated knowledge and student global latent factor, respectively.

I present this experiment on students partitioned by starting years and student majors in Fig. 4.3. For all student groups, accumulated knowledge is always more important than student global latent factor. Specifically, Fig. 4.3a shows that for students who start school in Fall 2013, the proportion of accumulated knowledge is the highest among all student groups and it is the lowest for students who start school in Fall 2014. Moreover, for students who start school in Fall 2010, the proportion of student global latent factor is the lowest among all student groups. I also notice that the difference between the two factors is the smallest for students who start school in Fall 2014. Fig. 4.3b shows the results for student groups partitioned by student majors. It shows that accumulated knowledge is more important than student global latent factor for MATH and PHYS majors. AIT has the smallest difference between accumulated knowledge and student global latent factor.

Based on the results of this experiment, students can balance the course knowledge and their own capabilities when selecting courses. For example, CS students who start school in Fall 2013 have the reference information that about 40% and 20% of their grades are influenced by the accumulated knowledge and student global latent factor, respectively.

4.5 Summary

This chapter presented additive latent effect models, which incorporate additive latent effects associated with students and courses to solve the next-term grade prediction problem. Specifically, I were able to highlight the improved performance of ALE with use of latent factors of course instructors, student academic levels and student global latent effect. The experimental results demonstrate that ALE outperforms all the state-of-the-art baselines in various experiments. Specifically, ALE model outperforms the best results among baselines for PTA_0, PTA_1 and PTA_2 by 10.61%, 7.17% and 4.50%, respectively. Moreover, I implemented different sets of experiments to analyze the importance of different effects contained in ALE.

Chapter 5: Grade Prediction Based on Cumulative Knowledge and Co-taken Courses

In this chapter, I propose grade prediction models that incorporate both **C**umulative **K**nowledge and **C**o-taken **C**ourses (CKCC) to predict students' performance in the next term. Inspired by Morsy *et al.* [13], the proposed methods model each student's latent factors by cumulating the knowledge provided by the sequence of courses the student has taken in the past terms. Furthermore, I introduce a co-taken course interaction function to model the influence of the co-taken courses on students' performance. The co-taken course interaction function is formed by a neural network which takes the knowledge difference between the co-taken courses and the target course as input, and outputs an influence value from the co-taken courses on the target course. I conduct comprehensive experiments on various datasets collected from a U.S. University and thorough analysis on the effect of co-taken courses. The experimental results show that CKCC significantly outperforms other competitive baselines methods for the task of grade prediction. I also provide detailed case study on how the model can help student in course selection for the next term.

To further present the motivation of the proposed model, I conduct a statistical analysis on a dataset collected from a U.S. University in order to demonstrate the effects of co-taken courses on students' performance. Figure 5.1 shows the true grade distribution of students' on a specific course *with* and *without* enrolling in another course in the same term. The course pairs I choose in this analysis are frequently co-occuring in the dataset. For each target course pair, I choose the students who take more than four courses in a term, including the corresponding course pairs. I keep the students if the other co-taken courses only share few topics/material as the target course pairs. Figure 5.1 shows that students who take BIOL311 (Genetics) with CHEM313 (Organic Chemistry) have fewer "F", "D" and "C" grades, and several more "B" grades than those students who only take BIOL311 in a term. Similar trend has been found for course pairs CS321 (Software Engineering) and ECE301 (Digital Electronics). Moreover, students who take MATH114 (Calculus) with CS211

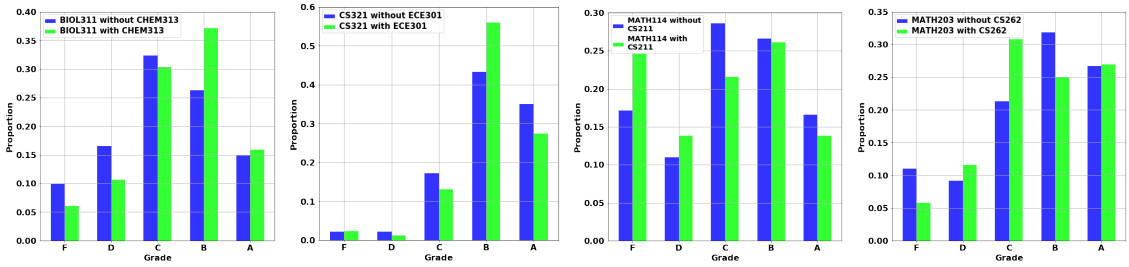


Figure 5.1: Students' Performance with Different Co-taken Course Pairs. Note: BIOL311 is course "General Genetics". CHEM313 is course "Organic Chemistry". CS321 is course "Software Engineering". ECE301 is course "Digital Electronics". MATH114 is course "Analytic Geometry and Calculus". CS211 is course "Object Oriented Programming". MATH203 is course "Linear Algebra". CS262 is course "Low-level Programming".

(Object Oriented Programmming) will have more "F" grades than those students who only take MATH114 in a term. Students who co-take MATH203 (Linear Algebra) and CS262 (Low-level programming) have more "C" grades than those students who only take MATH203 in a term. This shows that it can be challenging for students to take some courses together in a term (e.g., MATH114 and CS211, MATH203 and CS262), while it might not cause grade drop if taking other course pairs together (e.g., BIOL311 and CHEM313, CS321 and ECE301). Thus, I assume that co-taken courses can have substantial effect on student grades in different ways. In this chapter, I will discuss in detail how the proposed model incorporates the co-taken course influence, and improve the grade prediction performance. This work has been accepted by International Conference on Educational Data Mining (EDM 2019).

5.1 Methods

5.1.1 Model Overview

This chapter proposed grade prediction models that incorporate **C**umulative **K**nowledge and **C**o-taken **C**ourses (CKCC). To predict student s's grade on course c in term t, CKCC takes into account two factors: i) cumulative knowledge of student s up to term $t-1$, and ii) the other courses that will be taken together with course c in term t. To model the first factor, I adopt the CK model as

in Eq. 2.6, that is, I cumulate the provided knowledge of the courses which student s has taken in the past, denoted as c', to represent his/her cumulative knowledge, and use a latent factor to represent knowledge required by course c. To model the second factor, I introduce an co-taken course interaction function $f(\cdot)$ to learn the influence from co-taken courses, denoted as c'', on student s's grade on course c in term t.

Specifically, I use a latent vector $\mathbf{q}_c$ to represent the knowledge components that course c requires. I hypothesize that the difference of the required knowledge between two courses will cause the influence from one course on the other, as shown in Figure 5.1. Based on this hypothesis, the difference between $\mathbf{q}_c$ of course c and $\mathbf{q}_{c''}$ of a co-taken course c'' can be used in $f(\cdot)$ to learn the influence from c'' to c. I sum up the differences between each co-taken course c'' and c in order to aggregate the influence. Thus, the sum of the absolute values of the differences between each $\mathbf{q}_{c''}$ and $\mathbf{q}_c$, that is, $\sum_{c''\in C_{s,t}\setminus\{c\}} |q_{c''} - q_c|$, is used in $f(\cdot)$ to learn the influence from all co-taken courses. Note that the use of absolute values here is to avoid the scenarios in which the influences from different co-taken courses are canceled out. Thus, CKCC predicts student s's grade on course c in term t as follows:

$$\begin{aligned}\tilde{g}^t_{s,c} = &\frac{1}{|G^{t-1}_s|}\sum_{g_{s,c'}\in G^{t-1}_s}(e^{-\lambda(t-t_{s,c'})}\mathbf{k}_{c'}\cdot g_{s,c'})^{\mathsf{T}}\mathbf{q}_c + \\ &f(\sum_{c''\in C_{s,t}\setminus\{c\}}(|\mathbf{q}_{c''}-\mathbf{q}_c|)),\end{aligned} \tag{5.1}$$

where $|\mathbf{q}_{c''} - \mathbf{q}_c|$ is the vector of absolute values of entry-wise difference between latent vector $\mathbf{q}_{c''}$ and latent vector $\mathbf{q}_c$, $c'' \in C_{s,t} \setminus \{c\}$ indicates that course c'' is one of courses taken together with c in term t. Note that in Eq. 5.1, the two terms share a common latent vector $\mathbf{q}_c$.

5.1.2 Co-taken Course Interaction Function

In CKCC, the co-taken course interaction function $f(\cdot)$ learns the influence on student s's grade on course c from all the other co-taken courses in term t. I hypothesize that such influence can be

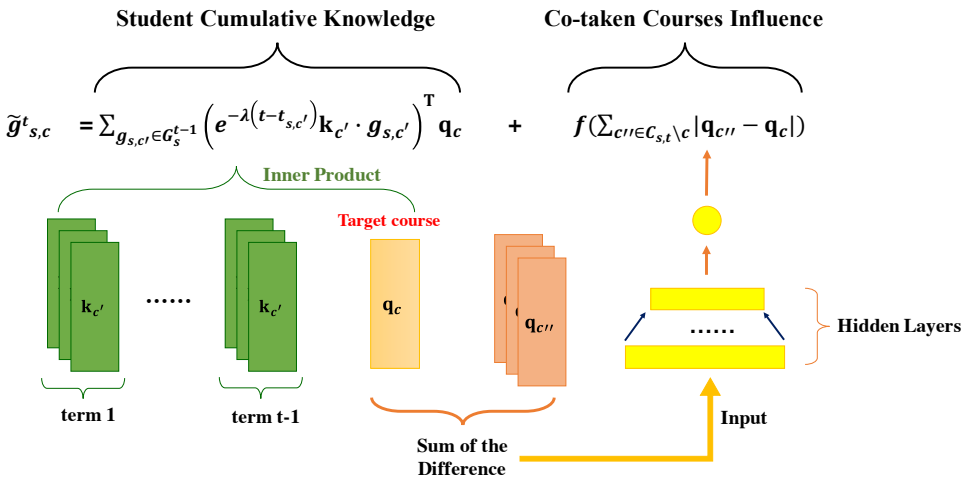


Figure 5.2: CKCC Model Structure

nonlinear in general. Therefore, I use a feedforward neural network (FNN) [81] as $f(\cdot)$ to model the influence. The FNN takes the input as described in last section, and outputs a scalar influence value on course c. I use hyperbolic tangent (Tanh) as the activation function in each layer of the FNN. Note that when there are no hidden layers and no nonlinearity, the FNN model learns the weights directly from the input layer (i.e., difference of courses) to the output layer (i.e., the influence), and the function $f(\cdot)$ becomes a simple inner product operation (parameterized by a vector). This simplified model is referred to as CKCC-*l*. Figure 5.2 shows the structure of the CKCC model.

5.1.3 Optimization of CKCC

Given the grade estimation as in Equation 5.1, I formulate the grade prediction problem for term T as the following optimization problem:

$$\begin{aligned} \underset{\Theta,\Theta_f}{\text{minimize}} \sum_{s} \sum_{t=1}^{T-1} \sum_{g^t_{s,c} \in G^t_s} (g^t_{s,c} - \tilde{g}^t_{s,c})^2 \\ + \alpha_1(|\mathbf{k}_c| + |\mathbf{q}_c|) + \alpha_2(\|\mathbf{k}_c\|_2^2 + \|\mathbf{q}_c\|_2^2) \\ + \alpha_3 \|\text{vec}(\Theta_f)\|_2^2, \end{aligned} \tag{5.2}$$

Algorithm 2 *CKCC*: Learn

procedure CKCC_LEARN
  Initialize $\mathbf{k}_c$, $\mathbf{q}_c$ for each c
  $\eta \leftarrow$ learning rate
  $T \leftarrow$ number of terms in training set
  $\lambda \leftarrow$ time decay parameter
  $\alpha_1, \alpha_2, \alpha_3 \leftarrow$ regularization weight
  $t \leftarrow 2$
  $iter \leftarrow 0$
  while iter<maxIter **do**
    for $t \leq T$ **do**
      for all $g^t_{s,c} \in \mathrm{G}^t_s$ **do** ▷ step 1
        $\hat{g}^t_{s,c} \leftarrow g^t_{s,c} - f(\sum_{c' \in C_{s,t} \setminus \{c\}}(|\mathbf{q}_{c''} - \mathbf{q}_c|))$
        $\mathbf{p}_{ck(s)} \leftarrow 0$
        for all $c' \in C^{t-1}_s$ **do**
          $\mathbf{p}_{ck(s)} \leftarrow \mathbf{p}_{ck(s)} + e^{-\lambda(t_{s,c} - t_{s,c'})} \mathbf{k}_{c'} \cdot g^{t_{s,c'}}_{s,c'}$
        $\tilde{g}^t_{s,c} \leftarrow \mathbf{p}^{\mathsf{T}}_{ck(s)} \mathbf{q}_c$
        $e^t_{s,c} = \hat{g}^t_{s,c} - \tilde{g}^t_{s,c}$
        for all $c' \in C^{t-1}_s$ **do**
          $\mathbf{k}_{c'} \leftarrow \mathbf{k}_{c'} +$
            $\eta(\mathbf{q}_c \cdot e^{-\lambda(t_{s,c} - t_{s,c'})} \cdot g_{s,c'} \cdot e^t_{s,c} - \alpha_1 \cdot \mathbf{k}_{c'})$
        $\mathbf{q}_c \leftarrow \mathbf{q}_c + \eta(\mathbf{p}_{ck(s)} \cdot e^t_{s,c} - \alpha_2 \cdot \mathbf{q}_c)$
      for all $g^t_{s,c} \in \mathrm{G}^t_s$ **do** ▷ step 2
        $\hat{g}^t_{s,c} \leftarrow g^t_{s,c} - \mathbf{p}_{ck(s)} \mathbf{q}_c$
        $\tilde{g}^t_{s,c} \leftarrow f(\sum_{c' \in C_{s,t} \setminus \{c\}}(|\mathbf{q}_{c''} - \mathbf{q}_c|))$
        $e^t_{s,c} = \hat{g}^t_{s,c} - \tilde{g}^t_{s,c}$
        Update Θ_f with Adam
    $iter \leftarrow iter + 1$
  return $\Theta = \{\{\mathbf{k}_c\}, \{\mathbf{q}_c\}\}$, Θ_f

where $\Theta = \{\{\mathbf{k}_c\}, \{\mathbf{q}_c\}\}$ represents the set of latent vectors, and Θ_f represents the parameters of $f(\cdot)$. α_1, α_2, and α_3 denote the nonnegative weights on the regularization terms to prevent overfitting.

The optimization process for CKCC is presented in Algorithm 2. It consists of two steps: The first step is to update the course parameters, i.e., Θ, using stochastic gradient descent. The second step is to update $f(\cdot)$ parameters, i.e., Θ_f, with the adaptive moment estimation (Adam) algorithm [82].

Table 5.1: Dataset Statistics

Major	FTF student group			TR student group		
	#S	#C	#S-C	#S	#C	#S-C
MATH	271	693	3,325	243	597	2,031
PHYS	144	488	2,044	73	286	905
CHEM	427	673	4,942	257	473	1,937
IT	430	473	5,984	1,163	487	10,302
CS	819	714	16,955	526	435	7,840
BIOL	1,951	1,197	22,065	1,481	980	10,851

#S, #C and #S-C are the number of students, courses and student-course pairs from Fall 2009 to Spring 2018, respectively.

5.2 Experiments

5.2.1 Dataset Description

For simplicity, I use students from six different majors in GMU to evaluate the proposed models. These majors have different numbers of enrolled students, courses, and different major syllabi. I will evaluate these majors on both FTF and TR student groups. The majors in my experiment include: (i) Mathematical Sciences (MATH), (ii) Physics (PHYS), (iii) Chemistry (CHEM), (iv) Information Technology (IT) , (v) Computer Science (CS) and (vi) Biology (BIOL). Table 5.1 shows the statistics across these majors.

5.2.2 Data Preprocessing

CKCC predicts student s's grade on course c in next term based on the courses taken by student s in previous terms and the co-taken courses in the same term. In order to prevent cold start problems [25], I conduct a data preprocessing step. In detail, I exclude the students who enroll in the program for less than two terms. Moreover, if student s has taken course c' before course c which have not appeared before, I exclude such scenarios as well, i.e., I will not predict course c's grade for student s. Furthermore, if student s only takes one course at a term, I exclude these terms as they do not match the situations I consider in CKCC.

5.2.3 Compared Methods

Since there is no prior research on the influence of co-taken courses within a same term, I use the two following methods and three other variants of CKCC as baselines in my experiments:

- **MF** The MF model is described as Eq. 2.1.
- **CK** The CK model is described as Eq. 2.6.
- **MFCC** I add the co-taken course influence to the MF model, and obtain the Matrix Factorization with Co-taken Courses (MFCC) model. Specifically, the predicted grade of student s on course c at term t is defined as

$$\tilde{g}^t_{s,c} = \mathbf{p}_s^\top \mathbf{q}_c + f(\sum_{c'' \in C^t_s \setminus c} (|\mathbf{q}_{c''} - \mathbf{q}_c|)), \tag{5.3}$$

  where $\mathbf{p}_s$ denotes the latent factors of for student s. Similar to the CKCC model, I optimize the MFCC model with two steps by alternately updating the latent factors and the model parameters in the mapping function $f(\cdot)$.
- **MFCC-l** The MFCC-l model is a special case of the MFCC model where $f(\cdot)$ is simply an inner product (parameterized by a vector) instead of an FNN.
- **CKCC-l** The CKCC-l model is described in Section 5.1.2.

5.2.4 Parameter Learning

The set of parameters in the optimization problem (Eq 5.2) includes the number of latent dimensions (i.e., k), regularization parameters (i.e., α_1, α_2, and α_3) and the decay rate (i.e., λ). I performed a grid search over all the parameters with $k \in \{5, 10, \ldots, 25\}$, and $\alpha_1, \alpha_2, \alpha_3, \lambda \in \{1e-3, 1e-2, 0.1\}$. Note that for the CKCC and MFCC models, the optimal neural network structure (e.g., number of layers, the size of each layer) depends on on the value of k. Thus, I swept different neural network structure parameters for every k value in my grid search. The neural network structures that consistently achieve good performance contain one hidden layer with 2 or 3 hidden units.

Table 5.2: Performance Comparison for All Methods on FTF students

Method	Spring 2018				Fall 2017				Spring 2017			
	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2
MF	0.762	0.172	0.303	0.549	0.759	0.168	0.303	0.556	0.772	0.162	0.306	0.540
MFCC-*l*	0.756	0.180	0.320	0.565	0.745	0.186	0.331	0.574	0.757	0.181	0.331	0.564
MFCC	0.763	0.175	0.317	0.573	0.753	0.188	0.322	0.573	0.760	0.173	0.317	0.565
CK	0.726	**0.190**	0.330	0.575	0.724	0.184	0.336	0.575	0.727	0.186	0.333	0.575
CKCC-*l*	**0.711**	0.189	**0.338**	0.589	0.712	0.191	**0.343**	**0.589**	0.717	0.182	0.332	0.587
CKCC	0.716	0.187	0.332	**0.593**	**0.709**	**0.195**	0.334	**0.588**	**0.710**	**0.196**	**0.339**	**0.594**

Table 5.3: Performance Comparison for All Methods on TR students

Method	Spring 2018				Fall 2017				Spring 2017			
	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2
MF	0.775	0.184	0.316	0.537	0.760	0.157	0.300	0.565	0.773	0.168	0.299	0.550
MFCC-*l*	0.763	0.178	0.315	0.543	0.748	0.187	0.326	0.571	0.755	0.185	0.328	0.563
MFCC	0.761	0.174	0.321	0.544	0.754	0.177	0.330	0.580	0.761	0.177	0.316	0.569
CK	0.753	**0.268**	**0.400**	**0.586**	0.770	**0.259**	**0.389**	0.570	0.750	**0.273**	**0.397**	**0.583**
CKCC-*l*	**0.733**	0.182	0.324	0.560	0.743	0.180	0.313	0.558	**0.739**	0.172	0.310	0.563
CKCC	0.735	0.181	0.323	0.562	**0.728**	0.175	0.335	**0.571**	0.740	0.169	0.318	0.553

5.3 Results and Discussion

5.3.1 Overall Performance

Table 5.2 and 5.3 shows the overall performance for all methods for both FTF and TR student groups, respectively.

Table 5.2 shows that for FTF students, CKCC and CKCC-*l* outperform the baseline methods over most datasets. Specifically, CKCC outperforms the other compared methods across different experimental protocols by 4.39%, 7.01%, 3.50%, 3.87% in terms of MAE, PTA_0, PTA_1, and PTA_2, respectively. Furthermore, CK based methods outperform MF based methods on all experimental protocols. This table also shows that co-taken course based methods (MFCC, MFCC-*l* and CKCC, CKCC-*l*) outperform their baseline methods (MF and CK) on all experimental protocols, respectively. This illustrates that for FTF students, both cumulative knowledge and co-taken courses have great influence on student's performance, and the proposed methods can capture such influence

Table 5.4: Performance Comparison for All Methods on FTF students on Different Majors

Method	MATH				PHYS				CHEM			
	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2
MF	0.762	0.234	0.336	0.523	1.099	0.106	0.206	0.383	0.684	0.262	0.399	0.601
MFCC-*l*	**0.758**	0.195	0.333	0.568	0.960	0.113	0.213	0.447	0.678	0.221	0.374	0.589
MFCC	**0.758**	0.206	0.322	0.559	0.998	0.163	0.248	0.433	**0.663**	0.249	0.380	0.592
CK	0.782	0.267	0.378	**0.569**	0.910	0.135	0.270	0.468	0.680	0.249	0.393	0.595
CKCC-*l*	0.784	0.184	0.316	0.535	0.978	0.238	0.294	0.437	0.734	**0.312**	**0.449**	0.611
CKCC	0.842	**0.309**	**0.413**	0.562	**0.842**	**0.254**	**0.373**	**0.508**	0.697	0.290	0.411	**0.620**

Method	IT				CS				BIOL			
	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2
MF	0.655	0.201	0.36	0.623	0.723	0.190	0.346	0.595	0.687	0.253	0.411	0.626
MFCC-*l*	0.664	0.181	0.365	0.630	0.715	0.177	0.326	0.603	0.777	0.317	0.439	0.599
MFCC	0.627	0.231	0.381	0.659	0.704	0.209	0.362	0.605	0.676	0.274	0.429	0.638
CK	0.606	0.299	**0.466**	0.681	0.722	0.244	**0.395**	0.597	**0.643**	0.316	0.464	**0.653**
CKCC-*l*	0.693	0.288	0.460	0.632	0.784	0.242	0.376	0.578	0.771	**0.341**	0.461	0.605
CKCC	**0.600**	**0.310**	0.465	**0.692**	**0.696**	**0.256**	**0.395**	**0.612**	0.660	0.329	**0.467**	0.649

Table 5.5: Performance Comparison for All Methods on TR students on Different Majors

Method	MATH				PHYS				CHEM			
	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2
MF	0.608	0.270	0.433	0.617	0.675	0.235	0.431	0.569	0.749	0.219	0.325	0.553
MFCC-*l*	0.637	0.270	0.418	0.610	0.669	0.216	0.353	0.588	**0.634**	0.281	0.412	0.649
MFCC	0.621	0.241	0.397	0.645	**0.577**	**0.353**	**0.471**	**0.667**	0.675	0.228	0.404	0.649
CK	**0.573**	0.394	0.545	0.677	0.741	0.200	0.275	0.550	0.679	0.368	0.491	0.623
CKCC-*l*	0.641	0.384	0.515	0.677	0.694	0.325	0.450	0.625	0.651	0.377	0.500	0.667
CKCC	0.613	**0.404**	**0.576**	**0.707**	0.805	0.200	0.350	0.600	0.642	**0.404**	**0.518**	**0.675**

Method	IT				CS				BIOL			
	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2	MAE	PTA_0	PTA_1	PTA_2
MF	0.614	0.217	0.405	0.662	0.836	0.175	0.302	0.538	0.711	0.200	0.341	0.559
MFCC-*l*	0.610	0.227	0.419	0.665	0.818	0.189	0.325	0.541	0.670	0.213	0.366	0.617
MFCC	0.608	**0.243**	0.415	0.658	0.796	0.193	0.333	0.578	0.674	0.206	0.367	0.604
CK	0.608	0.223	0.406	0.659	**0.737**	0.212	0.369	0.577	0.695	0.226	0.370	0.600
CKCC-*l*	**0.598**	0.235	**0.426**	0.659	0.756	0.184	0.343	**0.599**	0.679	0.228	0.384	0.600
CKCC	0.602	0.231	0.412	**0.672**	0.773	**0.234**	**0.371**	0.563	**0.643**	**0.260**	**0.393**	**0.629**

accurately.

Table 5.3 shows that CK has competitive results over TR students. Moreover, for MF based methods, MFCC and MFCC-*l* outperform MF for all the experimental protocols. This illustrates that co-taken courses are likely to have influence on student's performance, but the influence may

not be as strong as it is of cumulative knowledge for TR students.

5.3.2 Analysis on Individual Majors

In order to understand the proposed methods' performance on each major, I have tested all the aforementioned methods on different majors separately. I conducted this group of experiments for both FTF and TR students. And I use Spring 2018 as test set. I provide detailed experimental results in Table 5.4 and 5.5.

Table 5.4 shows that the CKCC model outperforms other compared methods for some majors (e.g., PHYS, CS) on all metrics, but has weak performance on some metrics for other majors (e.g., MATH, CHEM). Especially for MATH major, CKCC has the highest MAE result while MFCC and MFCC-*l* have the best MAE result. The reason might be that the performance of CKCC relies on the student historical information, and it tends to have good performance on the students with rich historical information. However, in the test set, some students in certain majors do not have much historical information and thus drag down the model performance. Table 5.5 shows that, for TR students, there is no method that consistently outperforms others across different metrics. The reason might be that the diversity in student characteristics (many TR students have different backgrounds) leads to diverse course selection plans among them. Such diversity greatly influences the performance of the different models.

5.3.3 Linear versus Nonlinear Mapping Function

As aforementioned, I have two forms of co-taken course interaction function: FNN model and linear model (parameterized by a vector). Specifically, I compare the results for MFCC versus MFCC-*l*, and CKCC versus CKCC-*l*, respectively, in order to understand how different mapping functions $f(\cdot)$ influence grade prediction performance. Table 5.2 shows that for FTF students, MFCC-*l* has slightly better performance than MFCC, and CKCC-*l* has competitive performance as CKCC across different experimental protocols. Same trend has shown in table 5.3 for TR students. Furthermore, table 5.4 shows that MFCC and CKCC consistently outperform MFCC-*l* and CKCC-*l* across different majors for FTF students. This illustrates that the influence of co-taken courses for FTF student

group can be better captured by a nonlinear model (i.e., FNN) than a simple linear model. Table 5.5 shows that for TR students, MFCC and CKCC don't always outperform MFCC-*l* and CKCC-*l* for different majors. The reason might be that some TR students will have fewer co-taken courses than those of FTF students, and the influence from co-taken courses can be well captured by a linear model.

5.3.4 Performance on Different Numbers of Co-taken Courses

In this section, I test the CKCC model on different data subgroups with different number of co-taken courses in a term. Specifically, I take the students in the test set and divide them into five groups: students who take $\{2,3,4,5,6+\}$ courses (6+ refers to six and more). I perform this experiment on each major for both FTF and TR students, respectively. For the sake of page limit, I only show the results for FTF students. Figure 5.3 shows the experimental results in terms of PTA_0, PTA_1 and PTA_2. The results show that different majors exhibit different trends when the number of co-taken courses varies. For example, for CHEM and BIOL majors, the performance of the CKCC model on PTA improves with more co-taken courses. This observation suggests that CKCC is able to leverage stronger influence of co-taken courses to improve its performance. However, for PHYS and CS majors, CKCC achieves better performance with 2, 3 or 6+ co-taken courses than with 4 or 5 co-taken courses. I postulate that this is due to the characteristics of courses chosen within a term and their content. These results also indicate that CKCC is able to model co-taken courses' influence despite of the number of the co-taken courses.

5.3.5 Performance on Different Numbers of Co-taken Course Subjects

In this section, I extract each course's subject and test the CKCC model on different data subgroups with different number of co-taken course subjects in a term. The reason I conduct this experiment is because I assume that courses with the same subject tend to have relative knowledge components. Students who have co-taken courses from many different subjects may have wide knowledge diversity. This experiment aims to test the performance of CKCC in terms of co-taken course subjects.

Specifically, I take the students in the test set and divide them into five groups: students who

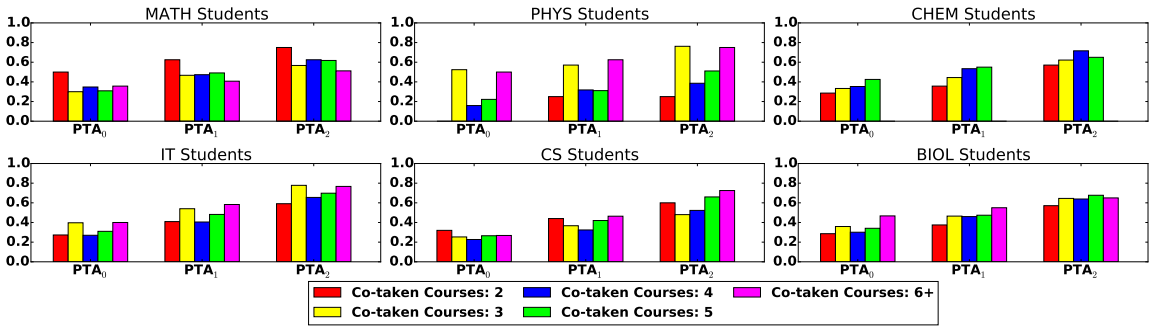


Figure 5.3: PTA Results for Different Number of Co-taken Courses on FTF students

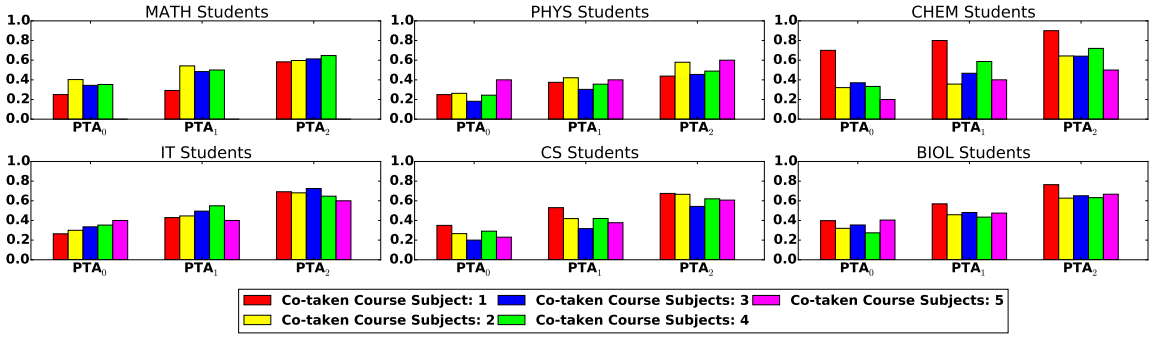


Figure 5.4: PTA Results for Different Number of Co-taken Course Subjects on FTF students

take courses from $\{1,2,3,4,5\}$ subjects in a term. Since there are few students co-taking courses from 6+ subjects, I exclude these students in the experiment. I perform this group of experiment on each major for both FTF and TR students, respectively. For the sake of page limit, I only show the results for FTF students. Figure 5.4 shows the experimental results in terms of PTA_0, PTA_1 and PTA_2. The results show that CKCC have different prediction results regarding the number of co-taken course subjects for different majors. For example, for CHEM, CS and BIOL majors, the performance of the CKCC model on PTA has the best performance with 1 co-taken course subject than other subgroups. This observation suggests that CKCC is able to model co-taken courses' influence better with less knowledge diversity in a term. However, for IT major, CKCC achieves better performance with more co-taken course subjects. And for MATH and PHYS majors, CKCC

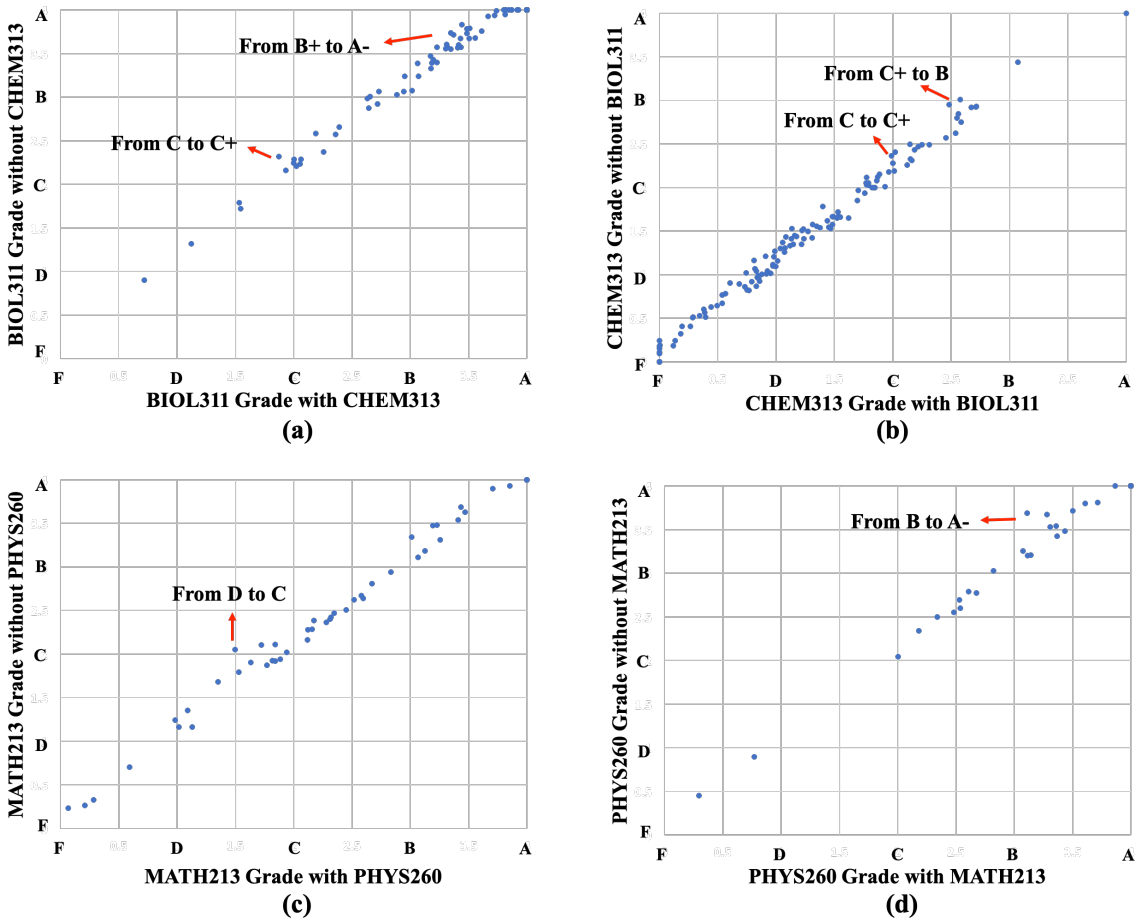


Figure 5.5: Comparison Results on the Co-taken Course Influence

has better performance on 2 or 5 co-taken course subjects than other subgroups. I assume that this is affected by the characteristics of different majors. Moreover, for MATH and IT major, the PTA results don't vary much comparing to CHEM and BIOL majors. This illustrates that for some majors, students may take courses from several subjects at a term, and the CKCC model can still well capture the co-taken courses' influence.

5.4 Significance and Impact

To highlight the use-case scenario of the developed next term grade prediction approach using co-taken courses, I ran a simulated case study. Having demonstrated the prediction accuracy of these

proposed models, the objective of this case study is to highlight the strengths of the proposed models in helping students to select courses in the future term. Implicitly I want to provide students information about their workload (or change in their overall grades) by addition of one or more courses within the next term.

Specifically, I extract two pairs of popular co-taken courses: BIOL311 ("General Genetics") and CHEM313 ("Organic Chemistry"), MATH213 ("Analytic Geometry and Calculus II") and PHYS260 ("University Physics"), and conduct a study to illustrate how the model can help plan students' course selections or allocate the necessary study time. Take the course pair BIOL311 and CHEM313 as an example. I extract the students who take course BIOL311 and CHEM313 together in a term. I predict students' performance on course BIOL311 using the CKCC model. I then eliminate course CHEM313 from the data set and predict the grade on course BIOL311 again using the CKCC model. Comparing the predicted grades helps determine if the two courses should be taken together within the same term or not. The sampled students have a total of five courses that they are enrolled in for the particular term. The comparison results are shown in Figure 5.5 (a). It is a scatter plot of predicted grades for a student where the x-axis shows the performance on course BIOL311 co-taken with the CHEM313 and the y-axis is the performance on course BIOL311 with course CHEM313 removed. I have conducted the same experiments for other course pairs using the same protocol and shown these results in Figure 5.5 (b), (c) and (d).

In general, students' performance will get better with the other course eliminated due to the reduction in workload. However, different students get affected differently by the additional course. For students who take BIOL311 and CHEM313, some of them will have improvement in BIOL311 grades if they do not enroll for CHEM313 in the same semester. On the other hand, some students will not have any change in their grades for BIOL311 based on course CHEM313 (the plotted results along the diagonal). Similar trends can be observed in Figure 5.5 (b), (c) and (d) as well. In the Figure 5.5, I also highlight different cases where students grade changes with the removal of the particular course. Using this information, students can plan the set of courses that they might enroll for in the next term, and allocate study time accordingly.

5.5 Summary

In this chapter, I propose grade prediction models that incorporate both cumulative knowledge and co-taken courses (CKCC) to predict students' performance in the next term. The proposed models consider both cumulative knowledge a student has acquired after taking a series of courses in the passing terms, and the co-taken courses the student plans to take in the next term. My experimental results on a dataset from a U.S. University shows that the proposed models significantly outperform other competitive baselines over most the datasets for the task of next-term grade prediction. Moreover, the experimental results show that the proposed model is able to capture strong influence of co-taken courses to improve its grade prediction performance. Furthermore, I ran a simulated case study to illustrate how the proposed model can help students in course selection for the future term.

In the future, I plan to take into account additive factors, such as instructor, student's academic level and course's difficulty level along with co-taken course information, in order to achieve more accurate grade prediction results. I hope such a grade prediction system can not only help students select courses, finish their study at college but also guide them in career planning in the future.

Chapter 6: Grade Prediction with Neural Collaborative Filtering

Over the past few years, several approaches inspired from the recommender systems literature have been adapted for predicting next-term student performance [7, 8]. MF approaches have been suitable for dealing with sparse datasets [6] and their extensions have incorporated temporal and dynamic information [43] that have shown an improvement in terms of model performance. However, when calculating inner product with MF, there are certain limitations [63]. For example, such inner products (linear combination of multiplication of latent factors) may not capture complex/nonlinear relations among data. To tackle this problem, He *et al.* [63] proposed a Neural Collaborative Filtering (NCF) model, which generalizes matrix factorization and learns non-linear relationships and aims to predict the rating a user would give to an item. Specifically, NCF uses two one-hot encoded vectors of users and items as input and learns embedding features for these two elements, followed by a fully connected feed-forward neural network to predict the user's rating on the item. Empirical results on several recommender system benchmarks demonstrates that the NCF approach greatly outperforms other methods [63].

I also extend NCF with non-negativity constraints and develop a non-negative neural collaborative filtering method, denoted as NCF_{nn}. I apply NCF and NCF_{nn} on next-term grade prediction problem. Similarly as in NCF model, I consider three elements as input: (i) students, (ii) courses and (iii) instructors. Adding of course instructor information has shown to improve the performance of prior grade prediction models [18]. Non-negative constraints on learned parameters provides for interpretability [27]. I propose NCF_{nn} by adding Rectified Linear Units (ReLU) [83] on the embedding layer. A ReLU is a unit that exploits the rectifier [84], which is an activation function that preserves the non-negative part of the argument and returns zero for the negative part.

The experimental results on a dataset from George Mason University (a large and diverse public university in Virginia, US) demonstrates that the proposed methods significantly outperform other

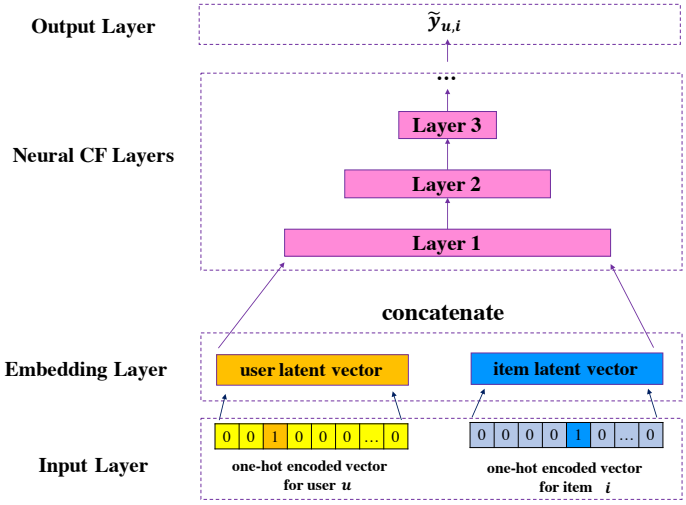


Figure 6.1: Model Structure of NCF on RS problem

competitive baselines for the task of grade prediction. In addition, I analyzed the model performance with respect to different embedding dimensions for students, courses and course instructors, respectively.

6.1 Background and Prior Methods

6.1.1 Neural Network-based Collaborative Filtering

Fig. 6.1 shows the structure of NCF for rating prediction problem. To predict user u's rating on item i, that is, $y_{u,i}$, NCF model takes two one-hot encoded vectors for user u and item i as input. Above the input layer is the embedding layer. It is a fully connected layer that projects the sparse representation to a dense vector. The embeddings of user and item are then concatenated and sent to a multi-layer fully connected neural network, which contains neural collaborative filtering (NCF) layers. Finally, the output of NCF layers is fed into the output layer and returns the predicted rating $y_{u,i}$. At each layer, different activation functions can be added, such as sigmoid function, hyperbolic tangent (tanh), and Rectifier (ReLU). The training of the model is performed by minimizing the point-wise loss between predicted ratings and the corresponding ground-truth ratings. NCF has shown to produce better recommendation results in comparison to MF methods. Unlike the traditional MF

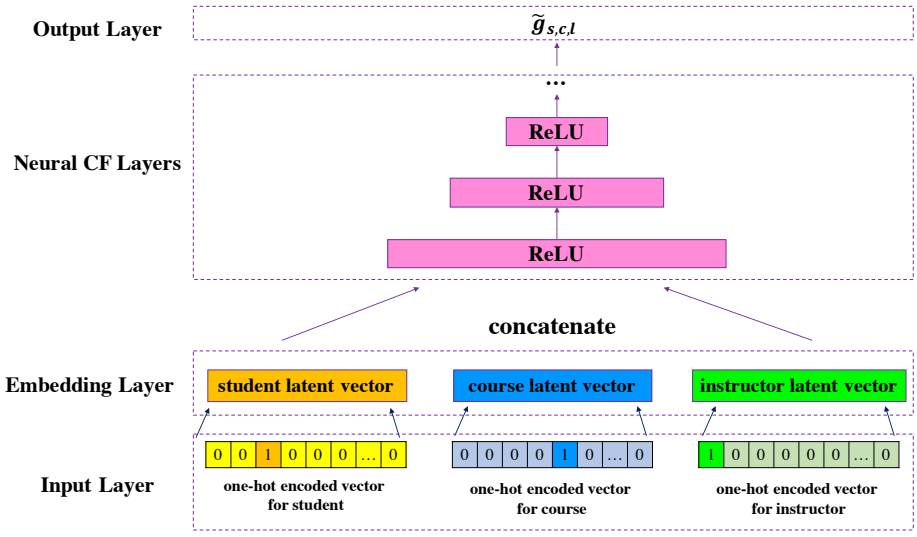


Figure 6.2: Model Structure of NCF on Grade Prediction

method, the NCF structure provides several advantages in terms of flexibility of input representation. The user/item latent factors can have varying number of dimensions and additional inputs beyond user/items can be easily incorporated within the NCF model.

6.2 Methods

6.2.1 NCF for Grade Prediction

In this thesis, I formulate the grade prediction problem within the NCF framework. Based on prior work [18] that course instructors can greatly influence student's grades, I consider three elements in grade prediction problem with NCF model, i.e., students, courses and the course instructors. Specifically, to predict student s's grade on course c taught by instructor l, I input three separate one-hot encoded vectors for the corresponding elements into the NCF model's input layer. Similar to the original NCF model, I have an embedding layer above the input layer, representing the latent factors for the three elements. Then the concatenated vector of the embeddings is fed into neural collaborative filtering layers to predict the final grade $g^l_{s,c}$. Different from original NCF, I specifically choose ReLU as activation function at each layer. As aforementioned, a ReLU exploits the activation

function rectifier which preserves the non-negative part of the argument and returns zero for the negative part. Therefore, I use ReLU to achieve non-negative constraint. The structure of this model is shown in Fig. 6.2.

Rectified Linear Unit

Rectifier is an activation function that is defined as follow:

$$f(x) = max(0,x). \tag{6.1}$$

where x is the input to a neuron. A ReLU is a unit that exploits the rectifier [84].

NCF with Non-Negativity Constraints

Since the embeddings can be interpreted in latent knowledge spaces, in order to get proper analysis of the model, I add non-negativity constraint on the embeddings by adding ReLU on the embedding layer. Since each one of NCF layers in the proposed model has ReLU as activation function, by adding ReLU on the embedding layer, the modified model has non-negativity values on all layers. I denote this NCF with non-negativity constraint as NCF$_{nn}$.

Parameter Learning

To predict grades in term t, the loss function for NCF and NCF$_{nn}$ can be formulated as follows:

$$L = \sum_{g^l_{s,c} \in G^{t-1}} (g^l_{s,c} - \tilde{g}^l_{s,c})^2 \tag{6.2}$$

I use Adaptive Moment Estimation (Adam) [82] method to learn model parameters.

Table 6.1: Dataset Statistics

Major	#S	#C	#S-C
MATH	209	84	2,846
PSYC	1,114	95	14,377
CHEM	342	55	4,649
CS	988	76	13,809
IT	334	82	6088
BIOL	1,629	109	21,519

#S, #C and #S-C are the number of students, courses and student-course grades from Fall 2009 to Spring 2016, respectively.

Table 6.2: #S-C for Different Terms

Major	Fall 2009 to Fall 2014	Spring 2015	Fall 2015	Spring 2016
MATH	942	100	78	168
PSYC	6,060	595	453	749
CHEM	1,139	86	106	103
CS	3,041	413	396	683
IT	1,492	474	439	599
BIOL	5,676	577	461	749

6.3 Experiments

6.3.1 Dataset Description

In this chapter, I evaluated the proposed models on the following six majors from GMU: (i) Mathematical Sciences (MATH), (ii) Psychology (PSYC), (iii) Chemistry (CHEM) (iv) Computer Science (CS), (v) Information Technology (IT), and (vi) Biology (BIOL). Table 6.1 presents the details of these majors. In the experiments, I only apply the models on FTF students as these students have more and complete data throughout college study than TR students. Table 6.2 shows the number of student-course grades of FTF students from different majors in different terms.

6.3.2 Data Preprocessing

I conduct a data preprocessing step to exclude the cold start problems [25] for next-term grade prediction. In detail, during testing, I exclude the students, courses and instructors that do not appeared in the training process.

6.3.3 Baseline Methods

I compare the proposed NCF methods with the following baseline methods.

Tensor Factorization

Tensor Factorization (TF) [85] has been successfully used in factorizing multi-way arrays and modeling relations among multiple types of elements. In my problem, since there are three elements, i.e., students, courses and course instructors, I use tensor factorization as one of the baseline methods to model their relations and make grade prediction.

I use $\mathscr{G}$ to represent a mode-3 student-course-instructor tensor, and each value in the tensor is the corresponding grade of a student in a course offered by an instructor; I use CANDECOMP/PARAFAC (CP) algorithm [86, 87], a very popular tensor decomposition algorithm, to decompose $\mathscr{G}$ into three matrices P, Q and R, where $P \in \mathbb{R}^{n\times k}$, $Q \in \mathbb{R}^{m\times k}$, $R \in \mathbb{R}^{l\times k}$ are the matrices containing length-k latent factors for students, courses and course instructors in the same latent space, respectively. Here k is the number of latent factors. Thus, student s's grade on course c taught by instructor l is the combination of the Hadamard product [88] of the corresponding vectors in P, Q and R, that is,

$$\tilde{g}^{l}_{s,c} = \sum_{i=1}^{k} p_{s,i} q_{c,i} r_{l,i}. \tag{6.3}$$

To predict student's grade at t_{th} term, the loss function of TF is as follows:

$$L = \sum_{g^l_{s,c} \in G^{t-1}} (g^l_{s,c} - \tilde{g}^l_{s,c})^2 + \alpha(\|P\|_2 + \|Q\|_2 + \|R\|_2) + \beta(|P|_1 + |Q|_1 + |R|_1) \tag{6.4}$$

I add both l_2 and l_1 norms on matrices P, Q and R to prevent overfitting. I use stochastic gradient descent algorithm (SGD) to solve the optimization problem.

Non-negative Tensor Factorization

I further extend the above TF method and introduce non-negativity constraint on matrix P, Q and R [89]. This TF method with non-negativity constraint is referred to as non-negative tensor factorization and denoted as TF_{nn}.

Additive Latent Effect Models

I have developed Additive Latent Effect Models (ALE) for next-term grade prediction in my earlier work [18], and here I use ALE as one of the baseline methods with compare with. ALE considers student's academic levels, student's global effect and course instructors in addition to student and course knowledge to tackle grade prediction problem. The experimental results showed that ALE achieved significant improvement over several state-of-the-art baseline methods. The predicted grade from ALE is calculated as follows:

$$\tilde{g}^l_{s,c} = \mathbf{p}_s{}^{\mathsf{T}}(\mathbf{q}_c + \mathbf{r}_l), \tag{6.5}$$

where $\mathbf{p}_s$ ($\mathbf{p}_s \in \mathbb{R}^k$), $\mathbf{q}_c$ ($\mathbf{q}_c \in \mathbb{R}^k$) and $\mathbf{r}_l$ ($\mathbf{r}_l \in \mathbb{R}^k$) are the size-$k$ latent factors for student s, course c and course instructor l, respectively.

I also include non-negativity constraint on latent factor matrix P, Q and R on ALE model. This non-negative ALE model is denoted as ALE_{nn}. To predict student's grade at t_{th} term, the loss

Table 6.3: Performance Comparison for All Methods

Test set: Spring 2016

Method	MATH			PSYC			CHEM		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
TF	0.149	0.315	0.589	0.148	0.307	0.602	0.117	0.223	0.534
TF_{nn}	0.173	0.321	0.559	0.154	0.300	0.619	0.136	0.214	0.544
ALE	0.256	0.387	0.625	0.279	0.425	0.585	0.214	0.359	0.573
ALE_{nn}	**0.268**	0.399	0.631	0.279	0.433	0.589	**0.282**	0.369	0.583
NCF	0.262	**0.435**	0.673	0.314	**0.506**	**0.737**	0.194	**0.408**	0.612
NCF_{nn}	0.238	0.429	**0.685**	**0.316**	0.487	0.726	0.184	**0.408**	**0.631**

Method	CS			IT			BIOL		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
TF	0.126	0.249	0.483	0.172	0.349	0.588	0.236	0.395	0.636
TF_{nn}	0.133	0.253	0.502	0.207	0.344	0.586	0.171	0.299	0.591
ALE	0.186	0.348	0.584	0.230	0.389	0.649	0.236	0.397	0.626
ALE_{nn}	0.184	0.337	0.616	**0.252**	**0.406**	0.613	0.234	0.394	0.644
NCF	**0.220**	**0.378**	0.616	0.235	0.397	0.691	**0.279**	**0.441**	**0.689**
NCF_{nn}	0.201	0.370	**0.619**	**0.235**	0.396	**0.698**	0.252	0.399	0.625

Test set: Fall 2015

Method	MATH			PSYC			CHEM		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
TF	0.128	0.231	0.449	0.161	0.287	0.607	0.226	0.358	0.585
TF_{nn}	0.090	0.218	0.423	0.166	0.313	0.598	0.226	0.358	0.594
ALE	**0.244**	0.321	0.436	0.331	0.455	0.583	**0.274**	**0.462**	**0.660**
ALE_{nn}	0.167	0.295	0.474	0.351	0.446	0.603	0.255	0.406	0.689
NCF	0.192	0.333	0.449	**0.366**	**0.550**	**0.728**	0.264	0.368	0.642
NCF_{nn}	0.218	**0.346**	**0.487**	0.322	0.497	0.706	0.245	0.368	0.642

Method	CS			IT			BIOL		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
TF	0.116	0.237	0.533	0.185	0.328	0.604	0.174	0.302	0.553
TF_{nn}	0.139	0.290	0.533	0.180	0.355	0.663	0.134	0.249	0.529
ALE	0.220	0.369	0.631	**0.257**	0.412	0.645	**0.269**	0.408	**0.696**
ALE_{nn}	**0.237**	**0.402**	**0.649**	0.241	0.412	0.667	0.262	**0.430**	0.681
NCF	0.139	0.283	0.604	**0.257**	**0.444**	**0.745**	0.254	0.412	0.664
NCF_{nn}	0.152	0.303	0.593	0.225	0.397	0.698	0.239	0.390	0.657

Test set: Spring 2015

Method	MATH			PSYC			CHEM		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
TF	0.150	0.310	0.560	0.180	0.333	0.634	0.116	0.279	0.651
TF_{nn}	0.180	0.340	0.550	0.210	0.336	0.610	0.163	0.314	0.581
ALE	**0.370**	**0.520**	**0.670**	**0.310**	**0.513**	**0.745**	0.186	**0.372**	0.558
ALE_{nn}	0.320	0.450	0.640	0.301	0.434	0.568	**0.233**	0.314	0.593
NCF	0.280	0.430	0.640	0.267	0.455	0.711	0.198	0.302	**0.628**
NCF_{nn}	0.270	0.440	0.660	0.259	0.461	0.713	0.198	0.291	0.616

Method	CS			IT			BIOL		
	PTA_0 (↑)	PTA_1 (↑)	PTA_2 (↑)	PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
TF	0.140	0.278	0.538	0.215	0.342	0.624	0.184	0.321	0.591
TF_{nn}	0.153	0.300	0.552	0.217	0.357	0.618	0.182	0.324	0.581
ALE	**0.218**	**0.378**	0.613	0.259	0.428	0.660	0.255	0.395	**0.669**
ALE_{nn}	0.211	0.366	0.608	0.262	0.454	0.656	**0.267**	**0.411**	0.659
NCF	0.211	0.363	0.620	**0.276**	**0.456**	0.703	0.210	0.371	0.645
NCF_{nn}	0.194	0.344	**0.622**	0.268	**0.456**	**0.719**	0.220	0.373	0.648

Table 6.4: Performance Comparison on MAE for All Methods

Test set: Spring 2016

	MATH	PSYC	CHEM	CS	IT	BIOL
TF	0.714	0.646	0.742	0.790	0.685	0.631
TF_{nn}	0.717	0.641	0.714	0.783	0.687	0.685
ALE	0.701	0.766	0.699	0.719	0.652	0.651
ALE_{nn}	0.689	0.742	0.652	0.716	0.669	0.650
NCF	**0.617**	**0.508**	0.617	0.691	0.604	**0.580**
NCF_{nn}	0.618	0.515	**0.616**	**0.690**	**0.602**	0.641

Test set: Fall 2015

	MATH	PSYC	CHEM	CS	IT	BIOL
TF	0.963	0.724	0.639	0.749	0.656	0.698
TF_{nn}	1.019	0.730	0.609	0.716	0.617	0.729
ALE	0.976	0.789	0.585	0.632	0.645	**0.570**
ALE_{nn}	0.953	0.754	**0.573**	**0.615**	0.595	0.574
NCF	0.895	**0.561**	0.586	0.672	**0.545**	**0.570**
NCF_{nn}	**0.883**	0.601	0.583	0.679	0.604	0.585

Test set: Spring 2015

	MATH	PSYC	CHEM	CS	IT	BIOL
TF	0.706	0.640	0.697	0.733	0.639	0.663
TF_{nn}	0.717	0.629	0.700	0.711	0.634	0.658
ALE	0.626	**0.518**	0.802	0.701	0.643	0.600
ALE_{nn}	0.596	0.770	0.742	0.701	0.609	**0.584**
NCF	0.596	0.543	0.672	**0.657**	0.574	0.600
NCF_{nn}	**0.595**	0.538	**0.660**	0.658	**0.570**	0.599

function of ALE is as follows

$$\begin{aligned} L = & \sum_{g^l_{s,c} \in G^{t-1}_s} (g^l_{s,c} - \tilde{g}^l_{s,c})^2 \\ & + \alpha(\|P\|_2 + \|Q\|_2 + \|R\|_2) + \beta(|P|_1 + |Q|_1 + |R|_1) \end{aligned} \tag{6.6}$$

Similarly, I add both l_2 and l_1 norms on matrices P, Q and R to prevent overfitting, and I use stochastic gradient descent algorithm (SGD) to solve the optimization problem.

Table 6.5: Performance Comparison for Different Embeddings Dimensions

Model	# StdEm	# CrsEm	# InstrEm	MATH			PSYC		
				PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
NCF_0	30	30	30	0.262	0.435	0.673	0.314	**0.506**	0.737
NCF_1	15	30	30	0.239	0.387	0.598	0.306	0.487	0.724
NCF_2	30	15	30	**0.299**	**0.461**	**0.705**	0.300	0.483	0.733
NCF_3	30	30	15	0.294	0.453	0.694	**0.319**	0.498	0.733
NCF_4	15	15	30	0.272	0.418	0.677	0.306	0.478	0.729
NCF_5	30	15	15	0.246	0.390	0.603	**0.319**	0.501	**0.740**
NCF_6	15	30	15	0.275	0.421	0.676	0.311	0.485	0.729
NCF_7	15	15	15	0.250	0.452	0.696	0.307	0.477	0.732

Model	# StdEm	# CrsEm	# InstrEm	CHEM			CS		
				PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
NCF_0	30	30	30	0.194	0.408	0.612	0.220	0.378	0.616
NCF_1	15	30	30	0.184	0.398	0.602	**0.227**	0.391	**0.635**
NCF_2	30	15	30	**0.243**	0.379	0.612	**0.227**	0.382	0.619
NCF_3	30	30	15	0.194	0.417	**0.621**	0.214	0.379	0.625
NCF_4	15	15	30	0.184	0.408	0.592	0.211	0.366	0.628
NCF_5	30	15	15	0.184	**0.437**	0.612	0.223	**0.394**	0.618
NCF_6	15	30	15	0.175	0.408	0.612	0.224	0.359	0.634
NCF_7	15	15	15	**0.243**	0.398	**0.621**	0.209	0.381	0.615

Model	# StdEm	# CrsEm	# InstrEm	IT			BIOL		
				PTA_0	PTA_1	PTA_2	PTA_0	PTA_1	PTA_2
NCF_0	30	30	30	0.235	0.397	0.691	0.279	0.441	0.689
NCF_1	15	30	30	0.237	0.397	0.691	0.215	0.372	0.602
NCF_2	30	15	30	0.229	0.397	0.686	**0.299**	**0.461**	**0.705**
NCF_3	30	30	15	0.220	0.407	**0.701**	0.294	0.453	0.694
NCF_4	15	15	30	0.227	0.392	0.683	0.272	0.418	0.677
NCF_5	30	15	15	0.239	**0.409**	0.694	0.210	0.378	0.614
NCF_6	15	30	15	**0.244**	**0.409**	0.693	0.275	0.421	0.676
NCF_7	15	15	15	0.230	0.394	0.699	0.264	0.427	0.677

StdEm, # CrsEm and # InstrEm indicate the dimensions of student embeddings, course embeddings and course instructor embeddings, respectively.

6.4 Results and Discussion

I now detail the experimental results and discuss the implications of these results.

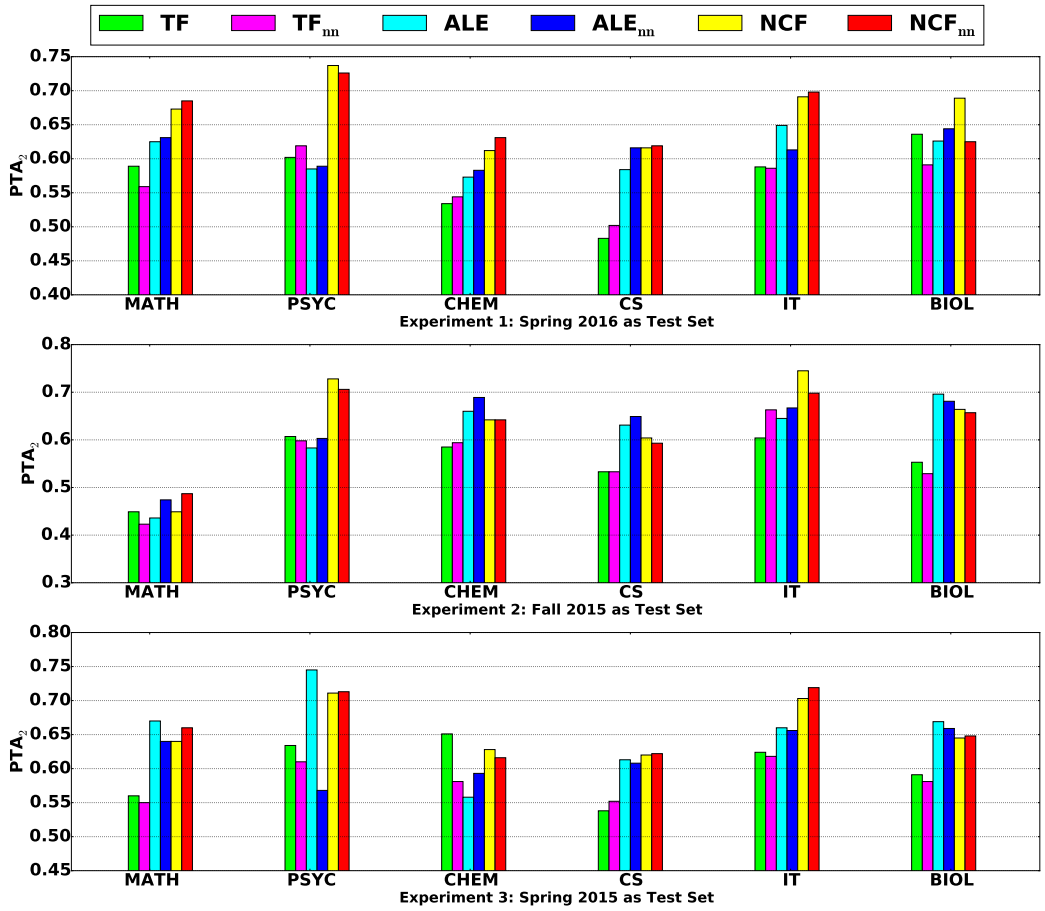


Figure 6.3: Analysis on the Effect of Non-Negativity Constraint

6.4.1 Overall Performance

Table 6.3 shows the results in terms of the PTA_0, PTA_1, and PTA_2 metrics. I use three terms: Spring 2016, Fall 2015, and Spring 2015 as test sets, and I implement comprehensive experiments on every major. In Tabel 6.3, "↓" indicates the lower the better, and ↑" indicates the higher the better. The best performing methods are highlighted with bold. I use a grid search method to sweep over the embedding dimensions, and choose value 30 for student, course and course instructor embeddings dimensions. (parameter study on embedding dimensions will be presented later in Section 6.4.2)

I observe that both NCF and NCF$_{nn}$ generally outperform the baselines across the different test

sets. Specifically, for the Spring 2016 term, NCF outperforms the TF, TF$_{nn}$, ALE and ALE$_{nn}$ baselines by 63.88%, 56.68%, 7.37% and 1.86% in terms of PTA$_0$, 43.85%, 51.19%, 11.15% and 9.72% in terms of PTA$_1$, and 17.45%, 18.19%, 10.41%, 9.41% in terms of PTA$_2$, respectively. However, I also notice that, for the Fall 2015 and Spring 2015 terms, the ALE baseline outperforms NCF and NCF$_{nn}$ on several majors. This result is likely due to the fact that neural network-based methods can often overfit and perform poorly with insufficient training data; in the experiments, when I use the Fall 2015 and Spring 2015 terms as test sets, the amount of training data is significantly less than using the Spring 2016 term as test set (see Table 6.2).

Table 6.4 shows the comparison on MAE for all methods. I note that when I use the Spring 2016 term as the test set, both NCF and NCF$_{nn}$ significantly outperform every baseline. When I use the Fall 2015 and Spring 2015 terms as test sets, ALE and ALE$_{nn}$ occasionally outperform NCF and NCF$_{nn}$. This observation agrees with that on the PTA metric. I also observe that when using the Spring 2016 term as test set, ALE$_{nn}$ outperforms on the PTA$_0$ metric but underperforms NCF$_{nn}$ on the PTA$_2$ metric, for the MATH, CHEM and IT majors. Meanwhile, NCF$_{nn}$ always outperforms ALE$_{nn}$ for all majors on the MAE metric. This observation shows that NCF$_{nn}$ outperforms ALE$_{nn}$ overall.

6.4.2 Effect of Embedding Dimensions

In order to get a deeper understanding on the impact of the embedding dimensions on model performance, I perform an experiment with different embedding dimensions for students, courses and course instructors. In total, I have eight different models, and I number each model NCF$_m$($m = 0, 1, ..., 7$); each model corresponds to a different set of embedding dimensions. NCF$_0$ corresponds to the NCF model in the previous experiments. Table 6.5 shows the experimental results on using the Spring 2016 term as test set. I observe that, on different test sets, the best-performing model differs. For example, for the MATH major, NCF$_2$ performs best in terms of PTA while NCF$_1$ performs worst. However, for the CS major, NCF$_1$ performs best in terms of PTA while NCF$_4$ performs worst. This observation shows that the flexibility to choose different dimensions for student, course, and instructor embeddings is crucial, since the best performing models often have different values

for the dimension of these embeddings. Therefore, this flexibility enables neural networks-based models to outperform matrix factorization-based models, which use the same dimension for every embedding.

6.4.3 Effect of Non-Negativity Constraint

For simplicity of exposition, I only show results on the PTA_2 metric for all methods and under all experimental protocols in Fig. 6.3, since the results on the other metrics are similar. I observe that for three head-to-head comparisons, i.e., TF versus TF_{nn}, ALE versus ALE_{nn} and NCF versus NCF_{nn}, in most circumstances, each pair of methods show similar results on different majors, such as the CS major. However, in some cases, adding the non-negativity constraint can lead to significant changes in model performance, e.g., ALE and ALE_{nn} on PSYC major in the experiments with Spring 2015 as test set. Further observing Fig 6.3, I notice that for the experiment with Spring 2015 as test set, NCF_{nn} is more likely to outperform NCF on different majors than the experiments with Spring 2016 and Fall 2015 as test sets, respectively. This shows that with insufficient training samples, non-negativity constraint can help NCF_{nn} model student, course and course instructors better than NCF, and finally gain better grade prediction results.

6.5 Summary

In this chapter, I develop a new deep learning inspired neural collaborative filtering approach for solving the next-term grade prediction problem. I consider three elements as input to the NCF model i.e., student, course and course instructor. Furthermore, the learned embeddings of these three input elements can be considered as the "hidden" factors within the classic latent factor model in collaborative filtering techniques. For proper analysis of the model, I also add non-negativity constraints on the embeddings by adding Rectified Linear Units (ReLU) on the embedding layer. The experimental results demonstrate that both NCF and NCF_{nn} significantly outperform the baselines on grade prediction problem over various test sets. In addition, I analyze the model performance with different embeddings dimensions for student, course and course instructor, respectively. The results

show that NCF provides flexibility in that, different from classic latent factor models in collaborative filtering techniques, different elements can have various dimensions and achieves better results than the one with the same embedding dimensions for all the elements. Finally, I provide in-depth analysis on the effect of non-negativity constraint. I have found that with insufficient training samples, non-negativity constraint can help NCF_{nn} model student, course and course instructors better than NCF, and finally gain better grade prediction results.

Chapter 7: Predicting Performance on MOOC Assessments using Multi-Regression Models

In this chapter, I will present models to predict a student's future performance for a certain assessment activity within a MOOC. Specifically, I develop an approach based on **P**ersonalized **L**inear **M**ulti-**R**egression (PLMR) to predict the performance of a student as they attempt various graded activities (assessments) within the MOOC. This approach was previously studied within the context of predicting a student's performance based on graded activities within a traditional university course with data extracted from a learning management system (Moodle) [27]. The developed model is real-time and tracks the participation of a student within a MOOC (via click-stream server logs) and predicts the performance of a student on the next assessment within the course offering. The proposed approach also allows us to capture the varying studying patterns associated with different students, and responsible for their performance. I evaluate the predictive model on two MOOCs offered using the OpenEdX platform and made available for learning analytics research via the Center for Advanced Research through Online Learning at Stanford University [1].

I extract features that seek to identify the learning behavior and study habits for different students. These features capture the various interactions that show engagement, effort, learning and behavior for a given student participating in studying; by viewing the various video and text-based materials available within the MOOC offering coupled with student attempts on graded and non-graded activities like quizzes and homeworks. The experimental evaluation shows accurate grade prediction for different types of homework assessments in comparison to baseline models. The approach also identifies the features found to be useful for predicting an accurate homework grade.

The work presented in this chapter has been published in International Conference on Educational Data Mining (EDM 2016).

[1] datastage.stanford.edu

7.1 Methods

7.1.1 Personal Linear Multi-Regression Models

I train a personalized linear multi-regression (PLMR) model [27] to predict student performance within a MOOC. Specifically, the grade $\hat{g}_{s,a}$ for a student s in an assessment activity a is predicted as follows:

$$\begin{aligned} \hat{g}_{s,a} &= b_s + p_s^t W f_{sa} \\ gg & \\ &= b_s + \sum_{d=1}^{l} (p_{s,d} \sum_{k=1}^{n_F} f_{sa,k} w_{d,k}), \end{aligned} \tag{7.1}$$

where b_s is bias term for student s, f_{sa} is the feature vector of an interaction between student s and activity a. The features extracted from the MOOC server logs are described in the next Section. n_F is the length of f_{sa}, indicating the dimension of the feature space. l is the number of linear regression models, W is the coefficient matrix of dimensions $l \times n_F$ that holds the coefficients of the l linear regression models, and p_s is a vector of length l that holds the memberships of student s within the l different regression models [27]. Using lasso [90], I solve the following optimization problem:

$$\underset{(W,P,B)}{minimize}\ L(W,P,B) + \gamma(\|P\|_F + \|W\|_F), \tag{7.2}$$

where W, P and B denote the feature weights, student memberships and bias terms, respectively. The loss function $L(\cdot)$ is the least square loss for regression problems. $\gamma(\|P\|_F + \|W\|_F)$ is a regularizer that controls the model complexity by controling the values of feature weights and student memberships. Tuning the scalar γ prevents model from over-fitting.

7.1.2 Feature Description

I extract features from MOOC server logs and formulate the PLMR model to predict real-time assessment grade for a given student. Figure 7.1 shows the various activities, generally available within a MOOC. Fig 7.1 (a) shows that each homework has corresponding quizzes, each of which

has its corresponding video as resources for learning. Fig 7.1 (b) shows that while watching a video, a student can have a series of actions. Fig 7.1 (c) shows that while studying using a MOOC, a student can have several login sessions. In order to capture the latent information behind the click-stream for each student, I extract six types of features: (i) session features, (ii) quiz related features, (iii) video related features, (iv) homework related features, (v) time related features and (vi) interval-based features. These features constitute the feature vector f_{sa} for a student and a homework assessment. The description of these features are as follows:

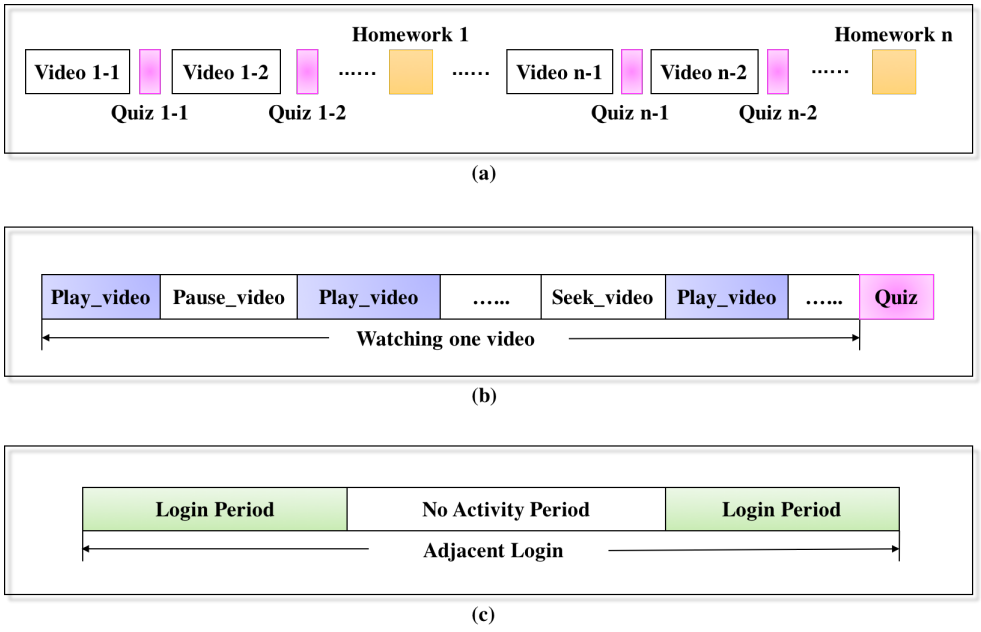


Figure 7.1: Different activities within a MOOC.

(i) Session features:

A single study session is defined by a student login combined with the various available study interactions that a student may partake in. Since, students do not always log out of a session, I assume that a "no activity" period of more than one hour constitutes a student logging out of a session. I then show a "no activity" period for a student between two consecutive sessions in Fig 7.1 (c).

- **NumSession** is the the average number of daily study sessions a student engages in, before a homework attempt.
- **AvgSessionLen** is the average length of each session in minutes. I calculate the average study time of a study session by

$$AvgSessionLen = \frac{Total\ study\ time}{NumSession}. \tag{7.3}$$

- **AvgNumLogin**. Students are free to choose when to login and study in a MOOC environment. I consider a day as a "work day" if a student logs into the study system; and a day as "rest day" if a student does not. The rate of "work" and "rest" can capture a student's learning habits and engagement characteristics.

$$AvgNumLogin = \frac{\#\ of\ \text{“}work\ day\text{”}}{\#\ of\ \text{“}work\ day\text{”} + \#\ of\ \text{“}rest\ day\text{”}}. \tag{7.4}$$

(ii) Quiz Related features:

- **NumQuiz** is the number of quizzes a student takes before a homework attempt. This feature reflects the student's dedication towards the course material and a factor towards performance in a homework.
- **AvgQuiz** is the average number of attempts for each quiz. The MOOCs studied in this chapter allow unlimited attempts on a quiz.

(iii) Video Related features:

- **VideoNum** denotes the number of distinct video sessions for a student before a homework attempt.

- **VideoNumPause** is the average number of pause actions per video. There are several actions associated with viewing videos, including "pause video", "play video", "seek video" and "load video". Tracking these actions allows for capturing a student's focus level and learning habits.
- **VideoViewTime** is the total video viewing time.
- **VideoPctWatch**. In a large amount of cases, students do not finish watching a full video. As such, I calculate the average percentage of the watched part of a video.

(iv) Homework Related features:

- **HWProblemSave** is the average number of "save answer" actions for each homework assessment. Before submitting answers for a homework, students are allowed to save their answer sheet and check as many times as they need. This feature is more valuable when the MOOC provides only one chance for a homework answer submission.

(v) Time Related features:

- **TimeHwQuiz** is the time between a homework answer submission and the last quiz attempt.
- **TimeHwVideo** is the time between a homework answer submission and the last video watching activity.
- **TimePlayVideo** is the percentage of study sessions with video watching activity over all the study sessions.
- **HwSessions** is the number of sessions that have homework related activities (save and submit).

(vi) Interval-Based features:

It is expected that there will be some changes in study activities once the students know the former homework's grade. They may study harder if they don't get a satisfactory score. The interval-based features are aiming to represent different activities between two consecutive homeworks.

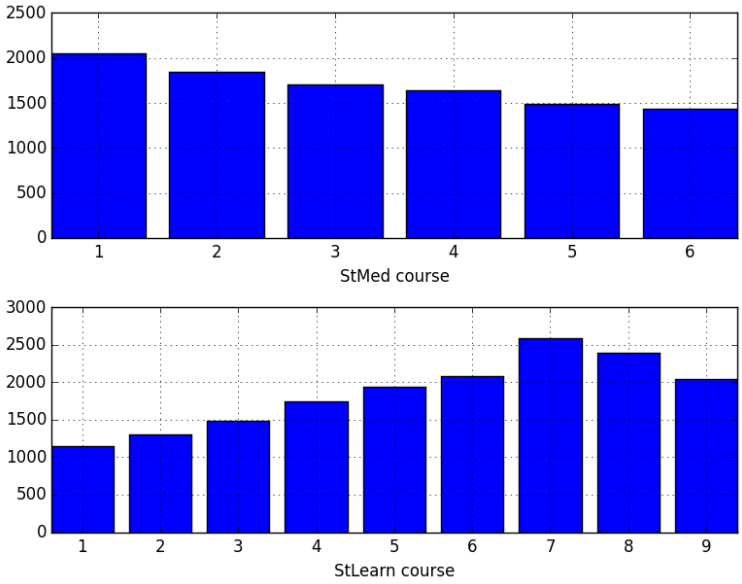


Figure 7.2: Distribution of Students Attempting Each Assessment.

- **IntervalNumQuiz**: denotes the number of quizzes the student takes between two homeworks.
- **IntervalQuizAttempt**: is the average number of quiz attempts between two homeworks.
- **IntervalVideo**: is the number of videos a student watches between two homeworks.
- **IntervalDailySession**: is the average number of sessions per day between two homeworks.
- **IntervalLogin**: is the percentage of login days between two homeworks.

I also use the cumulative grade (so-far) on quizzes and homeworks for a student as a feature and denote it by

Meanscore. For the baseline approach I only consider the averages computed on the previous homework.

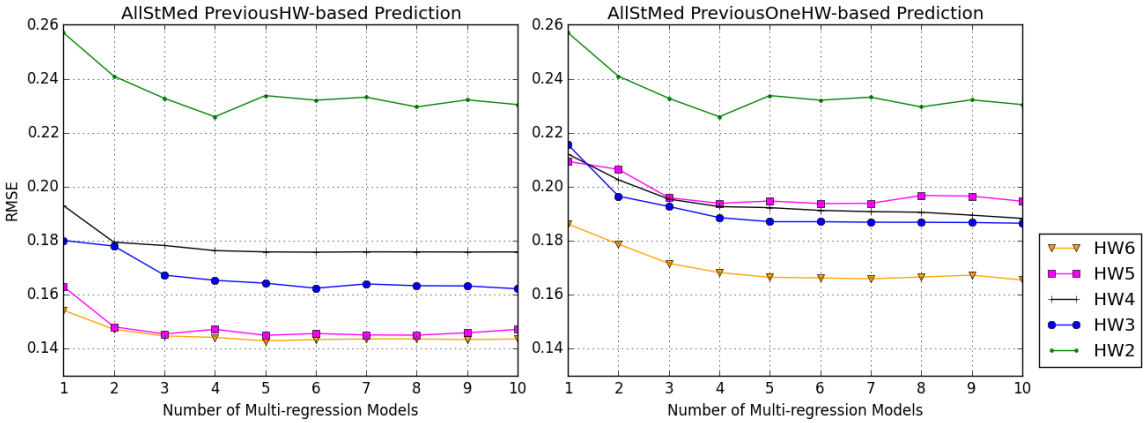


Figure 7.3: AllStMed Prediction Results. RMSE (↓ is better).

7.2 Experiments

7.2.1 Datasets

I evaluated the proposed methods on two MOOCs: "Statistics in Medicine" (represented as StMed in this chapter) taught in Summer 2014 and "Statistical Learning" (represented as StLearn in this chapter) taught in Winter 2015.

StMed: This dataset includes server logs tracking information about a student viewing video lectures, checking text/web articles, attempting quizzes and homework (which are graded). Specifically, this MOOC contains 9 learning units with 111 assessments, including 79 quizzes, 6 homework and 26 single questions. The course had 13,130 students enrolled, among which 4337 students submitted at least one assignment (quiz or homework) and had corresponding scores, 1262 students have completed part of the six homework and 1099 students have attempted all the homework. 193 students attempted all the 79 quizzes and six homework. This course had 131 videos and 6481 students had video related activity.

StLearn: This course had ten units. Except the first one, all units have quizzes and end of unit homework, which add up to 103 assessments in total. 52,821 students enrolled in this course, and 4987 students had assessment activities, 3509 students attempted a subsets of the available

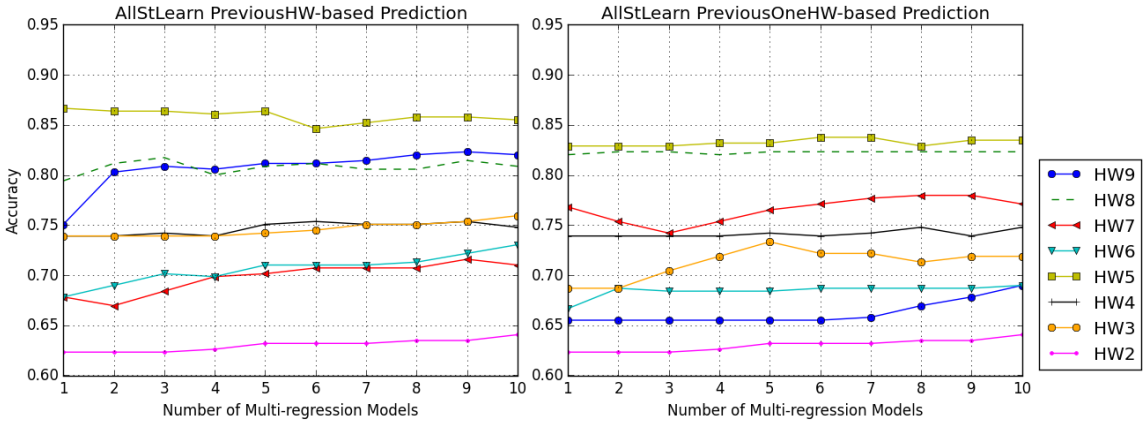


Figure 7.4: AllStLearn Prediction Results. Accuracy (↑ is better).

homework while 346 students attempted all the 9 homework, and 118 students attempted all the 103 assessments. The key difference between the homework in the StLearn in comparison to the StMed is that homework have only one question which a student can either get correct or incorrect. As such, scoring in this MOOC is binary instead of continuous. To predict whether a student answers a question correctly, I reformulate the regression problem as a classification problem using a logistic loss function. Figure 7.2 shows the distribution of students attempting the different assessments available across the two MOOCs studied here.

7.2.2 Experimental Protocol

In order to gain a deep insight of students' performance in a MOOC, I perform two types of experiments. Given n, homework assessments represented as $\{H_1, \ldots, H_n\}$ the objective is to predict the score a student achieves in each of the n homework. Depicting the most realistic setting, for the i-th homework, H_i I define the training set as all homework and student pairs who attempt and have a score for all homework up to the H_{i-1}. For predicting the score for H_i for a given student, I use all the features extracted just before attempting the target homework H_i. I refer to this as **PreviousHW-based Prediction**. Secondly, for the predicting i-th homework H_i's score, I use training data of student-homework pairs restricted from only the previous one homework i.e., H_{i-1}. This experiment is referred by **PreviousOneHW-based Prediction**. Note, in these cases I cannot make

Table 7.1: PreviousHW-based RMSE Performance (RMSE) comparison for AllStMed.

HW#	PLMR	Meanscore
2	**0.230**	0.248
3	**0.162**	0.176
4	**0.176**	0.196
5	**0.144**	0.156
6	**0.143**	0.150
Avg	**0.171**	0.185

any prediction for the first homework (H_1) since, I do not have any training information for a given student.

7.2.3 Data Partition

I partition the students for StLearn and StMed into two groups: the group of students who attempt *all* the requested homework, and the group of students who finish *few* of the homework. This allows us to consider the different motivations and expectations of students enrolling in a MOOC. For example, the students who aim to learn in a MOOC may choose watching videos over taking all homework. While, the students who want to achieve a degree certificate may focus on the homework completeness. I refer to the first group by "Partial homework accomplished group", and the second group by "All homework accomplished group". I evaluate the proposed models on the two groups for the **AllStMed** and **AllStLearn** datasets. Specifically, I name the four group of students as **AllStMed**, **AllStLearn**, **PartialStMed** and **PartialStLearn** based on their group and MOOC class.

7.2.4 Evaluation Metrics

StMed course has continuous scores for a homework, which are scaled between 0 and 1. However, the homework score is binary in the StLearn course, indicating whether the student answers a question correctly or incorrectly. For StLearn, I use a logistic loss and formulate a classification problem instead of the regression problem as done for the StMed course. To evaluate the performance of the

Table 7.2: PreviousHW-based prediction performance comparison for AllStLearn group.

HW#	Accuracy (↑)			F_1 (↑)		
	PLMR	Baseline		PLMR	Baseline	
		Meanscore	KT-IDEM		Meanscore	KT-IDEM
2	0.641	**0.646**	0.623	0.775	**0.777**	0.768
3	**0.760**	0.580	0.681	**0.821**	0.805	0.810
4	**0.754**	0.710	0.739	0.838	0.706	**0.850**
5	**0.867**	0.809	0.829	**0.920**	0.880	0.906
6	**0.730**	0.678	0.667	**0.808**	0.776	0.800
7	0.716	0.675	**0.730**	**0.887**	0.878	0.844
8	**0.817**	0.762	0.817	**0.903**	0.849	0.886
9	**0.823**	0.794	0.777	**0.864**	0.856	0.853
Avg	**0.764**	0.707	0.759	**0.852**	0.816	0.848

proposed approach, I use the root mean squared error (RMSE) as the metric of choice for regression problem. For classification problem, I use accuracy and the F1-score (harmonic mean of precision and recall), known to be a suitable metric for imbalanced datasets.

7.2.5 Comparative Approaches.

In this work, I compare the performance of the proposed proposed methods with two different competitive baseline approaches.

(i) Average grade of the previous homework I calculate the mean score of a given student's previous homework to predict their future performance and is denoted as Meanscore. I use this method to compare the prediction results on StMed.

(ii) KT-IDEM [91] KT-IDEM is a modified version of original BKT model. By adding an "item" node to every question node, the model is able to identify different difficulty levels of each question. Since this model can only predict a binary value grade, I use this model to compare the prediction results on StLearn.

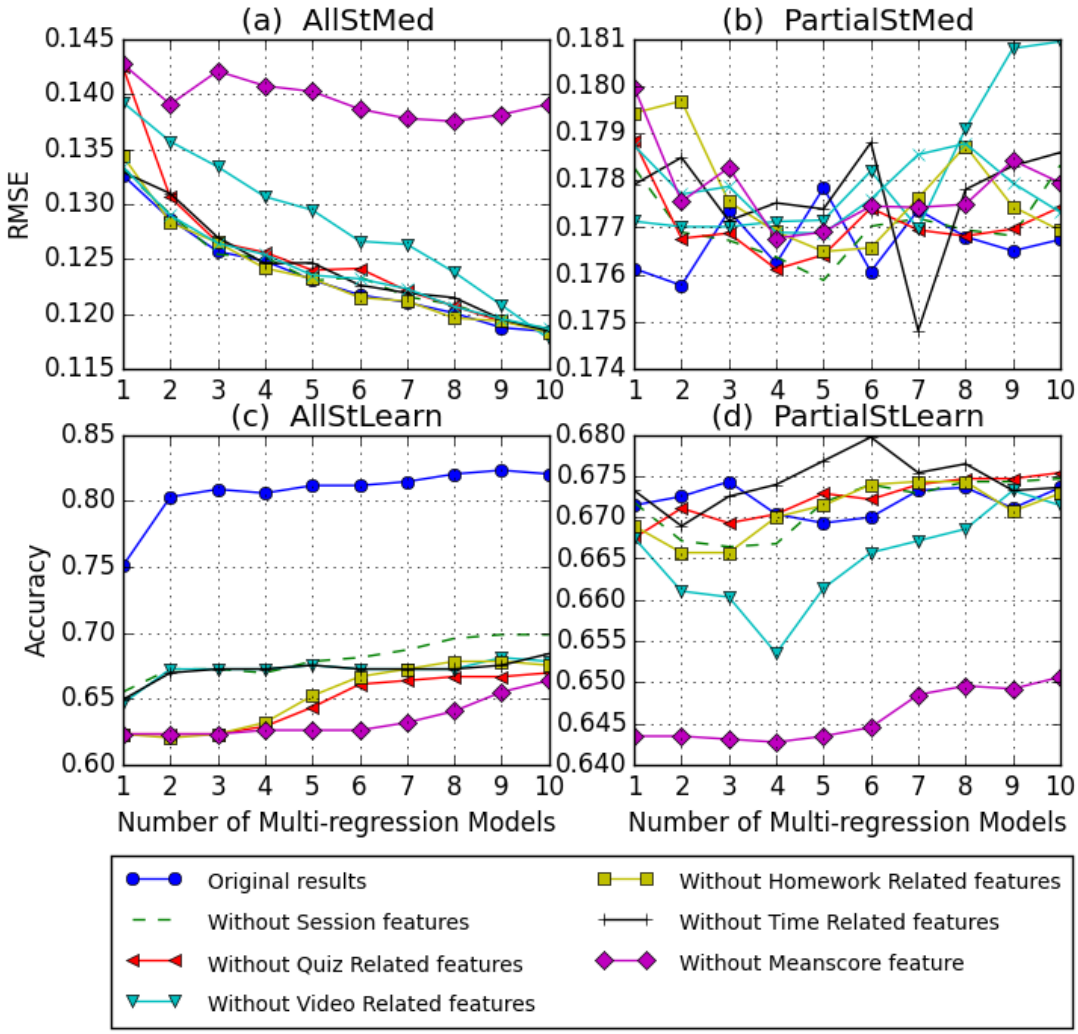


Figure 7.5: Predictive Performance with Removal of Feature Types.

7.3 Results and Discussion

7.3.1 Assessment Prediction Results

Figures 7.3 and 7.4 show the prediction results with varying number of regression models for the AllStMed and AllStLearn MOOCs, respectively. Figure 7.3 shows that as the number of regression models increases the RMSE metric goes lower and use of five models seems to be good choice for all the different homeworks. Comparing the PreviousHW- and PreviousOneHW-based results, it

shows that predictions for all the homeworks (HW3, HW4, HW5, and HW6) benefits from using all the available training data prior to those homeworks i.e., to predict grade for H_i it is better to use training information extracted from $H_1 \ldots H_{i-1}$ rather than just H_{i-1}. Similar observations can be made while analyzing the prediction results for the AllStLearn cohort which includes nine homework correct/incorrect binary assessments. Figure 7.4 shows the accuracy scores (higher is better) for the three experiments. For the PreviousOneHW- and PreviousHW-based experiments HW5 shows the best prediction results. This suggests that in the middle of a MOOC, students tend to have stable study activities and the performance is more predictable than other phases. Also, some homeworks thrive well with just using training data from the previous homework (PreviousOneHW-based, e.g. HW3).

Comparative Performance

Table 7.1 shows the comparison between baseline approach (Meanscore) and the predictive model for the PreviousHW-based experiments for the AllStMed group. I cannot report results for the KT-IDEM model since, it solves the binary classification problem only. Table 7.2 shows the comparison of the accuracy and F1 scores of the AllStLearn groups with baseline approaches. I notice that for predicting the second homework, which only uses the information from HW1, the predictive model is not as good as the mean baseline, which reflects that under the situation of lack of necessary amount of information, linear regression models cannot always outperform the baseline. But as the dataset gets larger, the proposed approach outperforms the baseline due to the availability of more training data. From Table 7.2, I also notice for some homework, KT-IDEM has better performance than PLMR (HW7 and HW4). This could be due to unstable academic activities during these two study periods, which can effect the performance of PLMR.

Feature Importance

I test the effect of each feature set in predicting the assessment scores by training the models under the absence of each feature group. For the StLearn course, since there is no limit on homework attempts, I do not add Interval-Based feature groups to the predictive model. Figure 7.5 shows the

comparison of each prediction result for AllStMed, PartialStMed, AllStLearn and PartialStLearn cohorts. Analyzing these results I observe that for the StLearn MOOC, meanscore is a significant feature and removing it leads to a substantial decrease in accuracy for both All and Partial- cohorts. For the AllStMed, the removal of video related features leads to the most decrease in performance (i.e., increased RMSE). This suggests that features related to the video watching are crucial for predicting the final homework scores. For the PartialStMed, the use of all feature types or a subset does not show a clear winner. This could be due to the varying characteristics of students within these group.

Another way to analyze feature importance is to exclude the influence of the dominant feature, which is meanscore in my study. The evaluation formula of the importance of the i_{th} feature (excluding meanscore feature) is as follows:

$$I_i = \frac{1}{N}\sum_{n=1}^{N}\frac{\sum_{d=1}^{l}|p_{n_S,d}f_{n_S,i}w_{d,i}|}{\sum_{d=1}^{l}|p_{n_S,d}\sum_{k=1}^{n_F}f_{n_S,k}w_{d,k}|}, \tag{7.5}$$

where N is number of test samples, n_S is the student number corresponding to the n_{th} test sample. $f_{n_S,i}$ is the feature value of an interaction between student n_S and activity i. n_F is the number of features. l is the number of linear regression models. $w_{d,i}$ is the coefficient of d_{th} linear regression model with i_{th} feature, and $p_{n_S,d}$ is the membership of student n_S with the d_{th} regression model. I calculate each feature's importance by calculating the percentage contribution of each feature to the overall grade prediction. Figure 7.6 shows the feature importance on the AllStMed group, excluding Meanscore feature. I can see **NumQuiz** and **VideoPctWatch** are the most important for AllStMed group besides **Meanscore** feature.

7.4 Summary

In this chapter, I formulated a personalized multiple linear regression model to predict the homework grades for a student enrolled and participating within a MOOC. My contributions include engineering features that capture a student's studying behavior and learning habits, derived solely from the

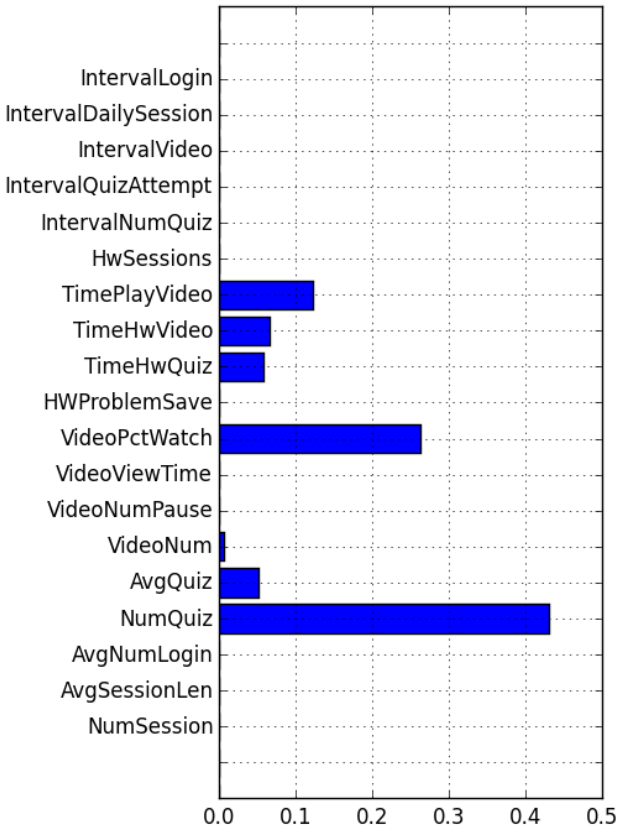


Figure 7.6: Feature importance for AllStMed.

server logs of MOOCs. I evaluated the proposed framework on two OpenEdX MOOC courses provided by an initiative at Stanford University. The experimental evaluation shows improved performance in terms of prediction of real time homework scores compared to baseline methods. I also studied on different groups of student participants due to their motivation. Features associated with engagement (logging multiple times), studying materials (viewing videos and attempting quizzes) were found to be important along with prior homework scores for this prediction problem.

Chapter 8: Conclusion and Future Work

8.1 Conclusion

To conclude, this thesis provides several algorithms for next-term grade prediction problem. First of all, I consider that a student's knowledge is continuously being enriched while taking a sequence of courses and substitute student's latent factors with accumulated knowledge of a sequence of courses taken by the student, jointly with the grade for each course. I propose a model named Matrix Factorization with Temporal Course-wise Influence. Following this method, I incorporate course instructor and student academic level effects along with student global latent factor to complete grade prediction. And I propose a model Additive Latent Effect. Moreover, I present next-term grade prediction models based on students' cumulative knowledge and co-taken courses. The proposed models are based on a matrix factorization framework and incorporate a co-taken course interaction function to learn the influence from the co-taken courses on the target course. The co-taken course interaction function is formed by a neural network, which takes the knowledge difference between the co-taken courses and the target course as input, and outputs an influence value that will be used to predict students' grades on the target course. Finally, I present a deep learning based recommender system approach called Neural Collaborative Filtering (NCF) for the next-term grade prediction problem. The deep learning inspired approach provides added flexibility in learning the latent spaces in comparison to MF approaches. The proposed approach also incorporates instructor information besides student and course information. Note that the baseline methods across different work may show different experimental results. The reasons for this include: (1) I have different datasets for the previous work, i.e., different training and testing sets as well as different majors; (2) I use different data preprocessing methods. In order to have a better understanding on all the proposed models, I have implemented all my methods on Spring 2016 for FTF students. Table 8.1 shows the comparison results. CKCC outperforms the other methods in terms of MAE and most

Table 8.1: Performance Comparison for All Models

	MAE	PTA_0	PTA_1	PTA_2
MF-b	0.687	0.159	0.324	0.632
MF	0.651	0.198	0.360	0.635
NMF	0.646	0.209	0.373	0.639
MFTCI	0.640	0.213	0.379	0.644
CK	0.621	0.236	0.395	0.638
ALE	0.623	**0.255**	0.394	0.635
CKCC	**0.607**	0.241	**0.402**	**0.654**
NCF	0.632	0.227	0.396	0.650

PTA metrics. ALE has the best PTA_0 result among all the methods. The results further show that in order to get accurate grade prediction for students, only considering course and student latent factors are far from enough. Many important factors have shown their immense influence on grade prediction results. Such important factors include student's previous enrollment information, course instructor information, student's global interests and co-taken courses. I have also presented how to use such information to better guide students in selecting courses in the future.

Figure 8.1 shows the diagram for my grade prediction tool. With the strength of the academic performance prediction, the grade prediction tool can be incorporated into course recommendation system, early warning system, degree planner and etc. For example, CKCC can help decide good course pairs to take in a new term. ALE can predict the grade for courses with different advisors, and help students to choose course sessions. Given the promising predicted grades, students can select courses, detect at-risk courses and finally form a practical pathway of the college study. However, there are some limitations in the current work. For example, even though I have achieved a better grade prediction results than the state-of-the-art methods, I haven't built a comprehensive system in assisting students to select courses, and guide students in their degree pathways. As such, I present some possible and thrilling future work in the next section.

Finally, I also apply a Personalized Linear Multi-Regression model to predict student's performance on online education environment, i.e., Massive Open Online Courses (MOOCs), and gain

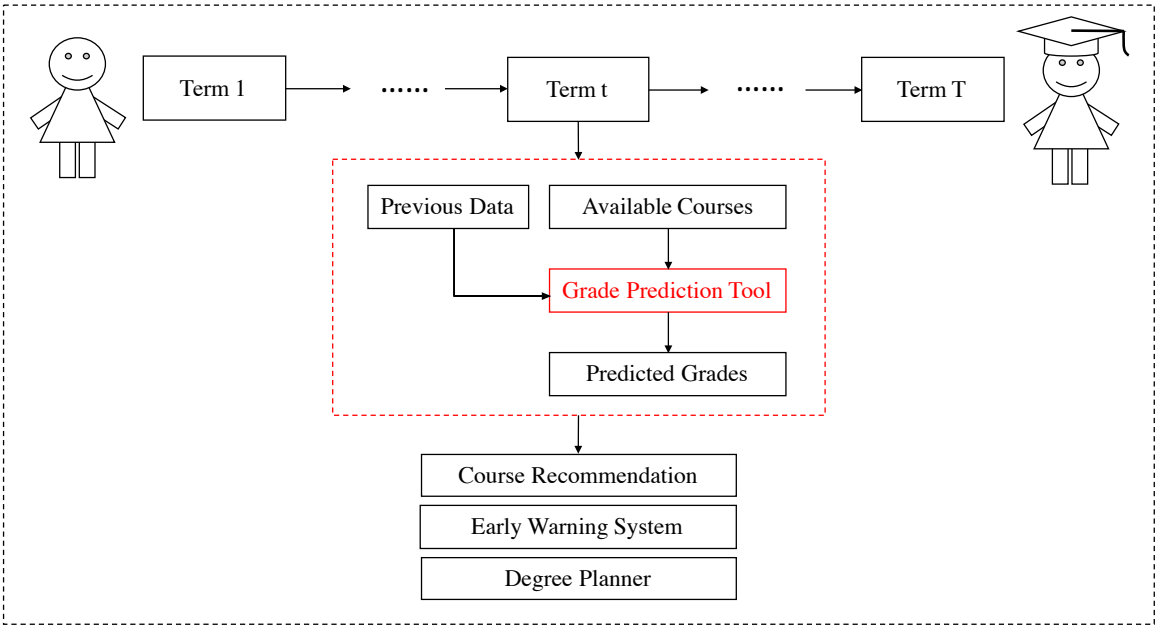


Figure 8.1: The Diagram for Grade Prediction Tool

great results.

8.2 Future Work

8.2.1 Personalized degree planner

In previous studies, prerequisite courses and co-taken courses are shown to have great influences on students' performance. However, students' personal interests and diligence which are likely to have high impact on students' performance are often neglected in grade prediction. It could be beneficial if I provide personalized degree planner for each student based on their own academic background as well as their personal interests. In the future, one of the interesting research directions would be studying students' personal learning behaviors. This will help researchers understand students' academic performance across different terms, and finally build a personalized degree planner to guide students individually in selecting courses, and eventually obtain a college degree.

8.2.2 Early warning system for instructor

While being in charge of a course that have a large number of students attended, instructors are often in need of students' feedback to discover how well the students learn the knowledge, and when to interrupt to assist. While explicit feedback is always hard and energy consuming to get, implicit feedback is believed to be feasible to help instructors to learn students' study status. In such case, it would be beneficial to build a system that can monitor students' in-class performance, and set up alarm if students have an obvious decrease in performance. The system is expected to understand the cause of the decrease in performance, such as the difficulty level of the assessment, and help instructors generate personalized assessments for students.

8.2.3 Course/material recommendation/generation for MOOCs

In MOOCs, there are large number of courses from different sources. Students on MOOCs always have uneven background and different demands. Course recommendation can be profitable for students to keep engaged in the course. However, there exists situations that the current courses or course materials are hardly suitable for students' demand. Therefore, generating new courses/materials based on the knowledge components which match students' needs can be beneficial. While this is a research direction which has rarely been worked on before, a few specific objectives would be interesting for further study: developing new methods for course representation by knowledge components, studying students' knowledge level and academic interests, and generating related courses/materials.

Bibliography

Bibliography

[1] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[2] M. Parker, "Advising for retention and graduation," 2015.

[3] N. Johnson, "The institutional costs of student attrition. research paper." *Delta Cost Project at American Institutes for Research*, 2012.

[4] R. Naqvi, "Data mining in educational settings," *Pakistan Journal of Engineering, Technology & Science*, vol. 4, no. 2, 2015.

[5] J. M. Simons, *A National Study of Student Early Alert Models at Four-Year Institutions of Higher Education.* ERIC, 2011.

[6] Y. Koren, R. Bell, C. Volinsky *et al.*, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[7] M. Sweeney, J. Lester, and H. Rangwala, "Next-term student grade prediction," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 970–975.

[8] A. Elbadrawy, A. Polyzou, Z. Ren, M. Sweeney, G. Karypis, and H. Rangwala, "Predicting student performance using personalized analytics," *Computer*, vol. 49, no. 4, pp. 61–69, 2016.

[9] Š. Pero and T. Horváth, "Comparison of collaborative-filtering techniques for small-scale student performance prediction task," in *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*. Springer, 2015, pp. 111–116.

[10] C.-S. Hwang and Y.-C. Su, "Unified clustering locality preserving matrix factorization for student performance prediction," *IAENG Int. J. Comput. Sci*, vol. 42, no. 3, pp. 245–253, 2015.

[11] A. Elbadrawy and G. Karypis, "Domain-aware grade prediction and top-n course recommendation," *Boston, MA, Sep*, 2016.

[12] A. Polyzou and G. Karypis, "Grade prediction with models specific to students and courses," *International Journal of Data Science and Analytics*, pp. 1–13, 2016.

[13] S. Morsy and G. Karypis, "Cumulative knowledge-based regression models for next-term grade prediction," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 552–560.

[14] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[15] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[16] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," *arXiv preprint arXiv:1707.07435*, 2017.

[17] Z. Ren, X. Ning, and H. Rangwala, "Grade prediction with temporal course-wise influence," *arXiv preprint arXiv:1709.05433*, 2017.

[18] ——, "Ale: Additive latent effect models for grade prediction," *arXiv preprint arXiv:1801.05535*, 2018.

[19] Z. Ren, H. Rangwala, and A. Johri, "Predicting performance on mooc assessments using multi-regression models," *arXiv preprint arXiv:1605.02269*, 2016.

[20] R. Baker *et al.*, "Data mining for education," *International encyclopedia of education*, vol. 7, pp. 112–118, 2010.

[21] W. He, "Examining students' online interaction in a live video streaming environment using data mining and text mining," *Computers in Human Behavior*, vol. 29, no. 1, pp. 90–102, 2013.

[22] A. Peña-Ayala, "Educational data mining: A survey and a data mining-based analysis of recent works," *Expert systems with applications*, vol. 41, no. 4, pp. 1432–1462, 2014.

[23] C. C. Aggarwal, *Recommender Systems: The Textbook*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[24] X. Ning, C. Desrosiers, and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender systems handbook*. Springer, 2015, pp. 37–76.

[25] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.263

[26] N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme, "Recommender system for predicting student performance," *Procedia Computer Science*, vol. 1, no. 2, pp. 2811–2819, 2010.

[27] A. Elbadrawy, S. Studham, and G. Karypis, "Personalized multi-regression models for predicting students performance in course activities," *UMN CS 14-011*, 2014.

[28] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, "Recommender systems handbook." 2011.

[29] S. Sahebi, Y.-R. Lin, and P. Brusilovsky, "Tensor factorization for student modeling and performance prediction in unstructured domain," in *Proceedings of the 9th International Conference on Educational Data Mining*. IEDMS, 2016, pp. 502–506.

[30] A. Lan, T. Goldstein, R. Baraniuk, and C. Studer, "Dealbreaker: A nonlinear latent variable model for educational data," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 266–275.

[31] Y. Meier, J. Xu, O. Atan, and M. van der Schaar, "Personalized grade prediction: A data mining approach," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 907–912.

[32] M. Sweeney, H. Rangwala, J. Lester, and A. Johri, "Next-term student performance prediction: A recommender systems approach," *arXiv preprint arXiv:1604.01840*, 2016.

[33] M. J. Pazzani and D. Billsus, "The adaptive web," P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. Content-based Recommendation Systems, pp. 325–341. [Online]. Available: http://dl.acm.org/citation.cfm?id=1768197.1768209

[34] S. Ray and A. Sharma, "A collaborative filtering based approach for recommending elective courses," in *International Conference on Information Intelligence, Systems, Technology and Management*. Springer, 2011, pp. 330–339.

[35] H. Bydžovská, "Are collaborative filtering methods suitable for student performance prediction?" in *Portuguese Conference on Artificial Intelligence*. Springer, 2015, pp. 425–430.

[36] T. Denley, "Course recommendation system and method," Jan. 10 2013, uS Patent App. 13/441,063.

[37] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 811–820.

[38] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," *arXiv preprint arXiv:1609.09152*, 2016.

[39] R. He, C. Fang, Z. Wang, and J. McAuley, "Vista: A visually, socially, and temporally-aware model for artistic recommendation," *arXiv preprint arXiv:1607.04373*, 2016.

[40] J. Z. Sun, D. Parthasarathy, and K. R. Varshney, "Collaborative kalman filtering for dynamic matrix factorization," *IEEE Transactions on Signal Processing*, vol. 62, no. 14, pp. 3499–3509, 2014.

[41] J. Z. Sun, K. R. Varshney, and K. Subbian, "Dynamic matrix factorization: A state space approach," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 1897–1900.

[42] F. C. T. Chua, R. J. Oentaryo, and E.-P. Lim, "Modeling temporal adoptions using dynamic matrix factorization," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 91–100.

[43] B. Ju, Y. Qian, M. Ye, R. Ni, and C. Zhu, "Using dynamic multi-task non-negative matrix factorization to detect the evolution of user preferences in collaborative filtering," *PloS one*, vol. 10, no. 8, p. e0135090, 2015.

[44] C. Zhang, K. Wang, H. Yu, J. Sun, and E.-P. Lim, "Latent factor transition for dynamic collaborative filtering." in *SDM*. SIAM, 2014, pp. 452–460.

[45] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, ser. CIKM '05. New York, NY, USA: ACM, 2005, pp. 485–492. [Online]. Available: http://doi.acm.org/10.1145/1099554.1099689

[46] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, *Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization*, 2010, pp. 211–222. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611972801.19

[47] D. Luo, H. Xu, Y. Zhen, X. Ning, H. Zha, X. Yang, and W. Zhang, "Multi-task multi-dimensional hawkes processes for modeling event sequences," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, pp. 3685–3691. [Online]. Available: http://dl.acm.org/citation.cfm?id=2832747.2832763

[48] H. R. Omaima Almatrafi, Aditya Johri and J. Lester, "Identifying course trajectories of high achieving engineering students through data analytics," in *American Society for Engineering Education*, 2016.

[49] A. Elbadrawy, "Prediction, ranking and mining methods for higher educational data," 2017.

[50] A. Peña-Ayala, "Educational data mining: A survey and a data mining-based analysis of recent works," *Expert systems with applications*, vol. 41, no. 4, pp. 1432–1462, 2014.

[51] C. G. Brinton and M. Chiang, "Mooc performance prediction via clickstream data and social learning networks," *To appear, 34th IEEE INFOCOM. IEEE*, 2015.

[52] G. Kennedy, C. Coffrin, P. de Barba, and L. Corrin, "Predicting success: how learners' prior knowledge, skills and activities predict mooc performance," in *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*. ACM, 2015, pp. 136–140.

[53] A. S. Sunar, N. A. Abdullah, S. White, and H. C. Davis, "Analysing and predicting recurrent interactions among learners during online discussions in a mooc," *Proceedings of the 11th International Conference on Knowledge Management*, 2015.

[54] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[55] Z. Pardos, Y. Bergner, D. Seaton, and D. Pritchard, "Adapting bayesian knowledge tracing to a massive open online course in edx," in *Educational Data Mining 2013*, 2013.

[56] A. Sharma, A. Biswas, A. Gandhi, S. Patil, and O. Deshmukh, "Livelinet: A multimodal deep recurrent neural network to predict liveliness in educational videos." in *EDM*, 2016, pp. 215–222.

[57] S. Klingler, R. Wampfler, T. Käser, B. Solenthaler, and M. Gross, "Efficient feature embeddings for student classification with variational auto-encoders."

[58] X. Xiong, S. Zhao, E. Van Inwegen, and J. Beck, "Going deeper with deep knowledge tracing," in *EDM*, 2016, pp. 545–550.

[59] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in neural information processing systems*, 2015, pp. 505–513.

[60] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 233–240.

[61] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 2051–2059.

[62] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[63] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[64] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

[65] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1245–1254.

[66] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[67] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 278–288.

[68] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 7–10.

[69] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016, pp. 153–162.

[70] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems." in *AAAI*, 2017, pp. 1309–1315.

[71] H. Wang, S. Xingjian, and D.-Y. Yeung, "Collaborative recurrent autoencoder: recommend while learning to fill in the blanks," in *Advances in Neural Information Processing Systems*, 2016, pp. 415–423.

[72] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 811–820.

[73] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 305–314.

[74] E. Smirnova and F. Vasile, "Contextual sequence modeling for recommendation with recurrent neural networks," *arXiv preprint arXiv:1706.07684*, 2017.

[75] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in netease," in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 1218–1229.

[76] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 241–248.

[77] A. Polyzou and G. Karypis, "Grade prediction with models specific to students and courses," *International Journal of Data Science and Analytics*, pp. 1–13, 2016.

[78] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[79] N.-D. Ho, "Nonnegative matrix factorization algorithms and applications," Ph.D. dissertation, ÉCOLE POLYTECHNIQUE, 2008.

[80] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy," in *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011, pp. 165–172.

[81] A. Zell, *Simulation neuronaler netze*. Addison-Wesley Bonn, 1994, vol. 1, no. 5.3.

[82] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[83] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[84] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[85] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[86] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[87] R. A. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[88] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 1990.

[89] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 792–799.

[90] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[91] Z. A. Pardos and N. T. Heffernan, "Kt-idem: Introducing item difficulty to the knowledge tracing model," in *User Modeling, Adaption and Personalization*. Springer, 2011, pp. 243–254.

Curriculum Vitae

Zhiyun Ren received her Bachelor's degree in Automation from Dalian University of Technology (DUT), Dalian, China in 2010. She received her Master's degree in Control Science and Engineering from Beihang University (BU), Beijing, China in 2013. She joined George Mason University (GMU) in Computer Science Department in 2014, and received a Doctor of Philosophy degree in Computer Science from GMU in 2019. She has worked as a research intern in Adobe for 2018 summer. Before joining GMU, she has worked as a Software Engineer for six months at VMware, Beijing, China in 2013.