Sequencing in Intelligent Tutoring Systems based on Online Learning Recommenders

A thesis submitted to the Faculty 4 - Mathematics, Natural Science, Economics and Computer Science for the degree of

Doctor of Natural Science (Dr. rer. nat.)
with core subject Computer Science

by

Carlotta Schatten, M.Eng.

Department of Computer Science

Information Systems and Machine Learning Lab (ISMLL)

University of Hildesheim, Germany

31. January 2017

Abstract

Intelligent Tutoring Systems (ITS) are computer programs used to teach students without the assistance of a human expert.
One of the most interesting aspect of ITS is the personalization and scheduling ability that Learning Analytics implements. In the past years, **Educational Data Mining** was used to analyze the data collected by ITS and proved that specific students' states can be recognized. **Learning Analytics** goes one step forward as it analyzes the data while the student is interacting with the system and takes scheduling decisions accordingly. Its main goals are ameliorating the learning experience and reduce the authoring efforts required when designing an ITS.

In this thesis we design and test Learning Analytics algorithms for **personalized tasks' sequencing** that suggests the next task to a student according to his/her specific needs. Our solution is based on a sequencing policy derived from the Vygotsky's Zone of Proximal Development (ZPD), which defines those tasks that are neither too easy not too difficult for the student. The sequencer, called Vygotsky Policy Sequencer (VPS), can identify tasks in the ZPD thanks to the information it receives from performance prediction algorithms able to estimate the knowledge of the student.

Under this context we describe hereafter the thesis contributions.

- A feasibility evaluation of domain independent Matrix Factorization applied in ITS for Performance Prediction.
- An adaption and the related evaluation of a domain independent update for online learning Matrix Factorization in ITS.
- A novel Matrix Factorization update method based on Kalman Filters approach. Two different updating functions are used: (1) a simple one considering the task just seen, and (2) one able to derive the skills' deficiency of the student.

- A new method for offline testing of machine learning controlled sequencers by modeling simulated environment composed by a simulated students and tasks with continuous knowledge and score representation and different difficulty levels.
- The design of a minimal invasive API for the lightweight integration of machine learning components in larger systems to minimize the risk of integration and the cost of expertise transfer.

Profiting from all these contributions, the VPS was integrated in a commercial system and evaluated with 100 children over a month.
The VPS showed comparable learning gains and perceived experience results with those of the ITS sequencer. Finally, thanks to its better modeling abilities, the students finish faster the assigned tasks.

Zusammenfassung

Intelligent Tutoring Systems (ITS) sind Computer Programme, die benutzt werden, um Schüler ohne den Beitrag von menschlichen Lehrern zu unterrichten. Eine der interessantesten Aspekte von ITS ist die Personalisierung und die Sequenzierung, die von Learning Analytics implementiert ist. In der Vergangenheit wurde Educational Data Mining benutzt, um die von ITS gesammelten Daten zu analysieren. Es wurde bewiesen, dass bestimmte Lernzustände von den Schülern erkannt werden können. Learning Analytics geht einen Schritt weiter indem es die Daten analysiert während der Schüler mit dem System interagiert. Dementsprechend ist Learning Analytics in der Lage Entscheidungen zu treffen, z.B. wird eine schwierigere (oder einfachere) Aufgabe vorgeschlagen, wenn es aus den Daten erkennt, dass der Schüler gelangweilt (oder überfördert) war. Das Hauptziel von Learning Analytics ist die Lernerfahrung zu verbessern und den Designaufwand von ITS zu reduzieren.

In dieser Arbeit entwickeln und testen wir Algorithmen für Learning Analytics, die die personalisierte Sequenzierung von Matheaufgaben erlauben. Die Sequenzierung schlägt die nächste Aufgabe einem Schüler vor, die seine Lernbedürfnisse entspricht. Unsere Lösung basiert auf Vygotskys "Zone of Proximal Development" (ZPD), das die weder zu einfachen noch zu schwierigen Aufgaben für den Schüler bestimmt. Der Sequenzer, auch Vygotsky Policy Sequencer (VPS) genannt, ist in der Lage Aufgaben im ZPD zu erkennen, dank die von einem Vorhersagealgorithmus geschätzte zukünftige Leistung des Schülers.

Die Arbeit enthält folgende Beiträge:

- Die Evaluation der Anwendbarkeit von Matrix Factorization als Inhaltsdomäne unabhängige Algorithmus für die Vorhersage der Leistung der Schüler.
- Anpassung und Evaluation eines Matrix Factorization basierenden Algorithmus, der die zeitliche Evolution der Schülerkenntnisse einbezieht.

- Entwicklung von zwei Ansätzen für die Aktualisierung von Matrix Factorization basierenden Modellen durch den Kalman Filter. Zwei Aktualisierungsfunktionen sind benutzt: (1) eine einfache, die nur die letzte vom Schüler gesehene Aufgabe betrachtet, und (2) eine, die in der Lage ist, seine fehlenden Kompetenzen einzuschätzen.
- Ein neues Verfahren von Machine Learning gesteuerte Sequenzer zu testen durch die Modellierung einer simulierten Umgebung, die aus simulierte Schülern und Aufgaben mit stetigen erzielten und gebrauchten Fähigkeiten und Schwierigkeitsgraden besteht.
- Die Entwicklung einer minimal eingreifenden API für die leichte Integration von Machine Learning basierende Komponente in größere Systeme, um das Integrationsrisiko und die Kosten vom Know-How-Transfer zu minimieren.

Dank all diesen Beiträgen, wurde der VPS in ein großes kommerzielles System integriert und mit 100 Kinder für einen Monat getestet. Der VPS zeigte Lerneffekte und wahrgenommene Erlebnisse, die mit den von den ITS Sequenzer vergleichbar sind. Infolge der besseren VPS Modellierfähigkeiten konnten die Schüler die Aufgaben schneller lösen.

Contents

1 Introduction **1**
- 1.1 Collaboration to the iTalk2Learn EU–Project . . . 2
- 1.2 Contributions . . . 4
- 1.3 Publications . . . 5
  - 1.3.1 First–author Publications . . . 5
  - 1.3.2 Coauthor Publications . . . 6
- 1.4 Chapters Overview . . . 7

2 Problem Formulation **11**
- 2.1 Student's Knowledge Estimation . . . 11
  - 2.1.1 Evaluation Framework for Static Algorithms . . . 13
  - 2.1.2 Evaluation Framework for Time Evolving Algorithms . . . 15
- 2.2 Sequencing . . . 16
  - 2.2.1 Sequencing Evaluation . . . 17
- 2.3 Data Requirements . . . 18
  - 2.3.1 Datasets for Performance Prediction . . . 18
  - 2.3.2 Exploratory Corpus . . . 21
- 2.4 iTalk2Learn Datasets . . . 21
  - 2.4.1 Large Commercial Dataset . . . 22
  - 2.4.2 Fraction Tutor Datasets . . . 23

3 State of the Art **25**
- 3.1 Performance Prediction . . . 26
  - 3.1.1 Domain Dependent Performance Prediction . . . 26
    - 3.1.1.1 Bayesian Knowledge Tracing . . . 26
    - 3.1.1.2 Performance Factor Analysis (PFA) . . . 29
  - 3.1.2 Domain Independent Performance Prediction . . . 30
    - 3.1.2.1 Matrix Factorization in Intelligent Tutoring Systems 30
    - 3.1.2.2 Time-aware Recommender Systems . . . 31
- 3.2 State Modeling Techniques . . . 32

3.3 Sequencing in Intelligent Tutoring Systems . . . 33
3.3.1 Rule-based Sequencers . . . 34
3.3.2 Adaptive Rule-Based Sequencers . . . 34
3.3.3 Policy–based Sequencers: Reinforcement Learning . . . 35

4 Online Learning Matrix Factorization for Performance Prediction 37
4.1 Static Matrix Factorization . . . 38
4.1.1 MF and BKT Comparison . . . 39
4.1.2 MF for Commercial ITS . . . 40
4.2 Updating Matrix Factorization . . . 41
4.2.1 Matrix Factorization Update . . . 41
4.2.2 Incremental Matrix Factorization . . . 43

5 Progress Modeling 45
5.1 Kalman Filter theory . . . 47
5.2 Kalman State Estimation for Matrix Factorization . . . 49
5.2.1 Simple previous/next State Mapping . . . 50
5.2.2 Skill Deficiency Aware KSEMF (KSEMF_SD) . . . 50
5.3 Experiments . . . 52
5.3.1 Dataset characteristics . . . 52
5.3.2 Hyperparameters' Selection . . . 54
5.3.3 State Variables' Initialization . . . 54
5.3.4 RMSE Evaluation . . . 54
5.3.5 Evaluation of the Cold Start Problem . . . 62
5.3.6 Modeling Student Progress . . . 62
5.3.7 Personalization . . . 65
5.3.7.1 Personalized state evolution . . . 65
5.3.7.2 Personalized update evolution . . . 65
5.4 Conclusions . . . 68

6 The Vygotsky Policy Sequencer 71
6.1 Content Sequencing Structure . . . 73
6.1.1 The Sequencer Structure . . . 74
6.1.2 Simulated Learning Process . . . 76
6.2 Experiment Session . . . 78
6.2.1 Experiments on the Simulated Learning Process . . . 80
6.2.2 Sensitivity Analysis on the Vygotsky Policy . . . 80
6.2.3 VPS Evaluation . . . 81
6.2.4 Advanced Experiments . . . 82
6.3 VPS Feasibility and Utility . . . 86

6.3.1 Sequencing VPS Feasibility 88
6.3.2 Sequencing VPS Utility 89
6.4 Conclusions 90

7 Large Scale Experiment **91**
7.1 Lightweight Integration of Machine Learning algorithms 93
7.1.1 Machine Learning Requirements 94
7.1.2 A novel Protocol for Machine Learning Integration 95
7.2 Sequencers' Integration in Commercial ITS 98
7.2.1 Commercial ITS Dataset Preprocessing 100
7.2.2 Online Update Integration 101
7.2.3 Vygotsky Policy Integration 102
7.2.4 Technical Integration 103
7.3 Experiment Session 103
7.3.1 Experiment Design 104
7.3.2 Results from Dataset Analysis 104
7.3.3 Post Test 105
7.3.4 Questionnaire 107
7.3.5 Integration 108
7.4 Computational Requirements 108
7.5 VPS with adaptive Threshold 109
7.6 Conclusions 111

8 Conclusions and Future Work **113**
8.1 Achieved Results 114
8.2 Future Work 116

References **119**

List of Figures

2.1 Cross validation, the dataset is split in k partitions and for each iteration a different set of data is used to train or test the model. In the figure the test set is highlighted for each iteration in light blue. The computed RSMEs for the k test partitions is then averaged to obtain the final RMSE. 14
2.2 Dataset split for static Performance Prediction. To select the test lines, the dataset is grouped by student and then ordered from the oldest to the newest entry. The last lines for each unit completed by a student, marked with the dark blue color, are excluded from the training set and used for evaluation. 15
2.3 Bridge and Algebra Intelligent Tutoring System 19
2.4 Algebra and Bridge snapshot 20
2.5 Example of questions posed within a task to the students 23

3.1 Table of scores given for each student on tasks (or interacting with generic contents) (left), completed table by the MF algorithm with predicted scores (right). 31
3.2 [32] cycle for switching between structured and exploratory tasks . . 35

4.1 Matrix decomposition 38

5.1 Kalman Cycle 47
5.2 RMSE sensitiveness analysis to latent features. 56
5.3 SlidingW_RMSE with window size $w = 5$ 58
5.4 SlidingW_RMSE with window size $w = 10$ 59
5.5 SlidingW_RMSE with window size $w = 15$ 60
5.6 Comparison between KSEMF and KSEMF_SD 61
5.7 D_{Test} **Total_RMSE behavior over time**: Models marked with "Cold" label are initialized with only 1 interaction in D_{Train} whereas the others with 10. 63

5.8 Actual score (left), KSEMF_SD predicted score (center), UpMF predicted score (left). 64

5.9 **x-Axis:** Number of tasks seen by the student or interactions. **y-Axis:** *(a)* state evolution according to KSEMF_SD with K=62. *(b)* state evolution according to UpMF with K=102. *(c) and (d):* knowledge evolution for KSEMF_SD and UpMF computed as in Eq. (5.9). *(e)* Total_RMSE of KSEMF_SD (blue), MF (green) and UpMF (black). *(f)* Actual performance of the student (blue), predicted performance by KSEMF_SD (green), MF (red). 66

5.10 **x-Axis:** Number of tasks seen by the student or interactions. **y-Axis:** *(a) and (d):* KSEMF_SD state evolution of two different students, K=62. *(b) and (e): kn* of KSEMF_SD latent features computed as in Eq. 5.9. *(c) and (f):* Total_RMSE of KSEMF_SD (blue), MF (green) and UpMF (black) of two different students. . . 67

5.11 **x-Axis:** Number of tasks seen by the student or interactions. **y-Axis:** *(a)* how the state evolves according to KSEMF_SD with K=62. *(b)* shows how the state evolves according to UpMF algorithm with K=102. *(c)* and *(d)* show the knowledge evolution, computed as in Eq. (5.9). *(e)* RMSE of KSEMF_SD (blue), RMSE of MF (green) and UpMF (black). *(f)* Actual Performance of the student (blue), predicted performance of the student by the KSEMF_SD (green), predicted performance by MF (red) and $\tilde{y}$ (turquoise). . . . 69

5.12 **Mean Update Over Time** Update behavior at each interaction on average for all students. 70

6.1 System structure in a block diagram. 73

6.2 Student with two skills, $s_i = [0.3, 0.5]$ represented with a red circle, could interact with the contents described in Tab. 6.1 represented as the light blue circles. The radius of the circles indicates difficulty for contents and ability level for the student. The red lines shows the ZPD according to the simulated learning environment. Contents whose center lies in the ZPD can be solved by the student. 79

6.3 Scenario: content number and difficulty level. 83

6.4 Comparison between RANGE and RND. Average skills sum, i.e. knowledge, over all the students with variance 83

6.5 Policy selection, i.e. the performance of the Vygotsky policy with different y_{th} at the same time step. Different groups of students learned with the Vygotsky policy with y_{th} values going from 0.1 to 0.9. As shown in the figure the knowledge levels change according to the y_{th} selected. 84
6.6 Effects of the different y_{th} on the final knowledge of the students. The learning curves of the student groups that learned with the different Vygotsky policies. 84
6.7 Average Total Knowledge. How the average learning curve of the students changes over time. 85
6.8 Average sequence selected by the GT and the VPS. The VPS approximate the optimal sequence that GT computes thanks to the real skills of the students. 85
6.9 Gain over RANGE policy varying n_k. The gain is measured at a specific time step in percentage, considering the average knowledge level of the two groups of students, one practicing with the RANGE sequencer and one with the VPS. 87
6.10 Gain over RANGE policy varying n_c. The gain is measured at a specific time step in percentage, considering the average knowledge of the two groups of students, one practicing with the RANGE sequencer and one with the VPS. 87
6.11 Effect of noise in the simulated learning process. Beta distribution noise with $\sigma^2 = 0.1$. 88

7.1 New Framework for lightweight ML integration. Structure and interaction between the ITS platform and the Learning Analytics Services platform 99
7.2 Two questions of the commercial ITS 100
7.3 Inter topic standard deviation, i.e. average of the standard deviation between Topic Math Age for students that interacted with 10-24, 25-49, 50-99, or 100-150 tasks. 106
7.4 Time required on a laptop with an Intel Core i5 CPU (2.6GHz) and 8GB RAM without DB accesses for updating one student's model by UpMF (blue) and KSEMF_SD (red). 110
7.5 Time required to extract one student's history form a DB with 926000 lines on a laptop with an Intel Core i5 CPU with 2.6GHz and 8GB RAM. 111
7.6 [18, 19, 20], VPS with adaptive threshold 112

List of Tables

4.1 In this table we report the RMSE obtained by [53] with MF step preprocessing. 39
4.2 Dataset Statistics 41
4.3 Performance Prediction Error 41

5.1 Baselines' names, reference and description. 53
5.2 Subset statistics of the commercial ITS dataset described in Sec. 2.4.1 53
5.3 Hyperarameters' ranges tested and selected values for the different algorithms. 55

6.1 Simulated learning process with two skills. A simulated student with $\boldsymbol{\varphi} = \{0.3, 0.5\}$ scores y and learning τ after interacting with different contents c_j. 78
6.2 Parameters MF 82
6.3 Sequencers Description 86

7.1 Parameter of the ML API 97
7.2 Trial Data Analysis. Values are indicated with ± standard deviation 105
7.3 Post Test Comparison. Values are indicated with ± standard deviation 107
7.4 Questionnaire comparison. 1: strong disagreement, 5: strong agreement. Values are indicated with ± standard deviation 108

Chapter 1

Introduction

Contents

1.1 Collaboration to the iTalk2Learn EU–Project 2

1.2 Contributions 4

1.3 Publications 5

1.3.1 First–author Publications 5

1.3.2 Coauthor Publications 6

1.4 Chapters Overview 7

Intelligent Tutoring Systems (ITS) are computer programs used to teach students without the assistance of a human expert. Such systems represent the opportunity for a great change, especially in those countries where the rate teachers/students is extremely low. Also in countries where this phenomenon does not occur, ITS have become extremely popular, as they reduce the tuition costs for students.
One of the most interesting aspect of ITS is the personalization and scheduling ability that Learning Analytics implements. In the past years, **Educational Data Mining** was used to mine a static snapshot of the data collected by ITS and proved that specific students' states can be recognized. According to the application and the available sensors' data, Machine Learning algorithms were used to recognize if the student was over– or under–challenged, if (s)he was frustrated, bored, surprised, etc.. Moreover, it was also possible to predict if the students were going to give a correct answer to the posed questions and if they required help to proceed

in the task.
Learning Analytics goes one step forward as it analyzes the data while the student is interacting with the system and takes scheduling decisions accordingly. Its main goals are ameliorating the learning experience and reduce the authoring efforts required when designing an ITS.

In this thesis we develop Learning Analytics algorithms for task Sequencing, i.e. we propose for each student a sequence of tasks adapted to his/her needs and that maximizes his/her performances. We therefore suggest a solution based on a Sequencing policy derived from the Vygotsky's Zone of Proximal Development (ZPD), which defines those tasks that are neither too easy not too difficult for the student. The sequencer, called Vygotsky Policy Sequencer (VPS), can sequence task in the ZPD thanks to the information it receives from Performance Prediction algorithms. It is also our aim is to reduce the authoring effort for sequencers integrated in large ITS to increase integrability.

To implement **Performance Prediction**, that is used by the VPS to predict the next student's score by means of his/her modeled knowledge, we will use Recommender Systems algorithms that have the advantage of being domain independent, i.e. they do not require an extensive contents analysis to be used. We then extend Performance Prediction developing **Progress Modeling**, that allows modeling the state of the student in a meaningful way over time.
Since this work is strongly connected to the EU iTalk2Learn project we first present its brief description. Then, we present the contributions and publications presented in this work and, finally, we briefly summarized the Chapters contents and the associated papers.

1.1 Collaboration to the iTalk2Learn EU–Project

Many of the results presented in this work were achieved thanks to the FP7 EU project called "Talk, Tutor, Explore, Learn: Intelligent Tutoring and Exploration for Robust Learning"–iTalk2Learn (grant no. 318051), where the University of Hildesheim was the coordinator.
The main goal of the project was to build an intelligent platform able to collect, analyze and adapt to student's data, with the goal to ameliorate current state of the

art of ITS. This platform combined key factors for the amelioration of personalized tuition such as (1) natural language production and recognition, (2) structured and exploratory learning activities and (3) several interventions methods.
To obtain the aforementioned three results the project individuated six objectives:

1. Design of new methods for automatic intervention selection, where with intervention feedback, hints, tasks' sequence or task's type is meant.

2. Enable different type of tasks, i.e. students should practice both on structured and exploratory tasks.

3. Integrate voice interaction, the platform should be able to process the students' utterances to analyze their behavior and also be able to use speech production to communicate with the user

4. Provide an open source platform

5. Develop rich structured and exploratory contents

6. Evaluate the previous results with a statistically significant number of students

Our research focused on the first aforementioned objective: "new methods for automatic intervention selection". In particular, in this thesis we present Sequencing as intervention method with Progress Modeling as a way to deliver such intervention without detrimental data collections or too extensive domain information. In achievement of Objective 6, the sequencer was evaluated with a large amount of students and was able to sequence the rich structured contents selected for Objective 5.
All articles written in collaboration with Ruth Janning are listed in Sec. 1.3.2 and represent the connection between Objective 1 and 3 of the iTalk2Learn project as we will better explain in the state of the art chapter (Chapter 3 and Chapter 7). More precisely, we will show how the analysis of students' speech production can deliver insights on their perceived difficulty level and therefore better adapt the VPS behavior. Another project contribution related to the VPS was already proposed in [32] as well as in several project reports[1]. Connections between [32] and the work presented in this thesis can be found in Chapter 5 and in [56].

[1] "Final report on methods and prototype for adaptive intelligence for robust learning support" `http://www.italk2learn.eu/deliverables-and-publications/deliverables/`

1.2 Contributions

This work delivers following contributions to the state of the art:

- A feasibility evaluation of domain independent Matrix Factorization applied in Intelligent Tutoring Systems for Performance Prediction. As a result, we show that, with Matrix Factorization, task IDs, student IDs, and scores can be used to obtain a prediction for datasets not possessing the level of detail that benchmark datasets of the area have.
- An adaption and the related evaluation of a domain independent update for online learning Matrix Factorization in Intelligent Tutoring Systems. The algorithms designed for Data Mining purposes are extended to work with online learning problems in a Learning Analytics context.
- A novel Matrix Factorization update method based on Kalman Filters approach. The model is presented in two variations with two different updating functions: (1) a simple one considering the task just seen, and (2) one able to derive the skills' deficiency of the student.
- A new method for offline testing of Machine Learning controlled sequencers by modeling simulated environment composed by simulated students and tasks with continuous knowledge and score representation and different difficulty levels.
- An alternative to Reinforcement Learning for task Sequencing called VPS approach that does not require detrimental data collection for users and extensive authoring effort. As shown in this work and the related derived papers, the method also allow easy integration with other Machine Learning state modeling techniques.
- The design of a minimal invasive API for the lightweight integration of Machine Learning components in larger systems to minimize the risk of integration and the cost of expertise transfer. The API allowed integration of the developed sequencer in a large commercial ITS, that could not allow the effort of a invasive integration.
- A large scale evaluation of the designed sequencer in a commercial system with 100 users over one month. The sequencer proved to have comparable learning gains and perceived experience results with those of the ITS sequencer, which was designed over the years by experts. In addition, the sequencer proved to have better modeling abilities, so that the students could proceed faster through the curriculum.

1.3 Publications

In this Section we list the papers that contain or are strongly related to the work presented in this thesis.

1.3.1 First–author Publications

The papers listed hereafter are part of this thesis.

1. Carlotta Schatten, Lars Schmidt-Thieme (2016):
   Hybrid Matrix Factorization Update for Progress Modeling in Intelligent Tutoring Systems, in Communications in Computer and Information Science 2016, Revised Selected Papers.

2. Carlotta Schatten, Lars Schmidt-Thieme (2016):
   Student Progress Modeling with skills deficiency aware Kalman Filters, in Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016).

3. Carlotta Schatten, Ruth Janning, Lars Schmidt-Thieme (2015):
   Integration and Evaluation of a Matrix Factorization Sequencer in Large Commercial ITS, in Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015).

4. Carlotta Schatten, Ruth Janning, Lars Schmidt-Thieme (2014):
   Vygotsky based Sequencing without Domain Information: A Matrix Factorization Approach, in Proceedings of the Computer Supported Education.

5. Carlotta Schatten, Manolis Mavrikis, Ruth Janning, Lars Schmidt-Thieme (2014):
   Matrix Factorization Feasibility for Sequencing and Adaptive Support in ITS, in Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014).

6. Carlotta Schatten, Martin Wistuba, Lars Schmidt-Thieme, Sergio Gutirrez-Santos (2014):
   Minimal Invasive Integration of Learning Analytics Services in Intelligent Tutoring Systems, in Proceedings of the 14th IEEE International Conference on Advanced Learning Technologies.

7. Carlotta Schatten, Lars Schmidt-Thieme (2014):
   Adaptive Content Sequencing without Domain Information, in Proceedings of the 6th International Conference on Computer Supported Education (CSEDU 2014).

1.3.2 Coauthor Publications

The work presented in this thesis was developed within the iTalk2Learn project in collaboration with other researchers. We therefore often refer to papers that are highly related to this thesis as they either deepened other aspects or proposed enhancements for the VPS.
These papers are:

1. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2016):
   Perceived task-difficulty recognition from log-file information for the use in adaptive intelligent tutoring systems, in International Journal of Artificial Intelligence in Education (JAIED).

2. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2015):
   Improving Automatic Affect Recognition on Low-Level Speech Features in Intelligent Tutoring Systems, in Proceedings of the 10th European Conference on Technology Enhanced Learning (EC-TEL 2015).

3. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2015):
   Recognising perceived task difficulty from speech and pause histograms, in Proceedings of the 17th International Conference on Artificial Intelligence in Education (AIED 2015).

4. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2015):
   How to aggregate multimodal features for perceived task difficulty recognition in intelligent tutoring systems, in Proceedings of the 8th International Conference on Educational Data Mining (EDM 2015).

5. Lydia Voss, Carlotta Schatten, Claudia Mazziotti, Lars Schmidt-Thieme (2015): A Transfer Learning approach for applying Matrix Factorization to small ITS datasets, in Proceedings of the 8th International Conference on Educational Data Mining (EDM 2015).

6. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2014):
   Automatic Subclasses Estimation for a Better Classification with HNNP, in Proceedings of the 21th International Symposium on Methodologies for Intelligent Systems (ISMIS 2014), in Lecture Notes in Artificial Intelligence.

7. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme, Gerhard Backfried, Norbert Pfannerer (2014): An SVM Plait for Improving Affect Recognition in Intelligent Tutoring Systems, in Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2014).

8. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2014):
Local Feature Extractors Accelerating HNNP for Phoneme Recognition, in Proceedings of the 37th German Conference on Artificial Intelligence (KI 2014).

9. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2014):
Feature Analysis for Affect Recognition Supporting Task Sequencing in Adaptive Intelligent Tutoring Systems, in Proceedings of the 9th European Conference on Technology Enhanced Learning (EC-TEL 2014).

10. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2014):
Multimodal Affect Recognition for Adaptive Intelligent Tutoring Systems, in Extended Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014).

11. Ruth Janning, Carlotta Schatten, Lars Schmidt-Thieme (2013):
HNNP - A Hybrid Neural Network Plait for Improving Image Classification with Additional Side Information, in Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2013).

1.4 Chapters Overview

Chapter 2 In this Chapter we formulate the problems of Performance Prediction, Progress Modeling and Sequencing from a Machine Learning perspective. The dataset requirements and iTalk2Learn datasets are presented to better explicate the challenges of applying specific algorithms in this context.

Chapter 3 In this Chapter we introduce the state of the art of Domain Dependent and Domain Independent Performance Prediction, Kalman Filters as state modeling technique and Sequencing in Intelligent Tutoring Systems.

Chapter 4 In this Chapter we explain in detail Matrix Factorization and its online updating versions, that are used in Chapter 5 as comparison to the developed Progress Modeling approach.
Contributions of the work presented here are:

- Feasibility of Domain Independent Matrix Factorization for Performance Prediction in ITS and
- Feasibility of Online Leaning Matrix Factorization for for Performance Prediction in ITS.

These contributions are also published in: [43, 44].

Chapter 5 In this Chapter we go a step forward with respect to domain independent Performance Prediction. From an approach informing only on the current/next state of the user, we move to Progress Modeling, where the students' state has to evolve in a meaningful, plausible and therefore interpretable way over time.
In this scenario three problems arise:

1. Domain information, like tagging involved skills in tasks, necessitates experts and thus is a time-consuming, costly, and, subjective. For large commercial ITS it is even unfeasible.
2. Progress Modeling requires to be able to interpret the model, i.e. to be able to associate the value of the model parameters with a specific user state.
3. The continuously changing student's state and the necessity of new data requires online updating algorithms, that refine their prediction after each interaction.

The method developed for Progress Modeling is described in this Chapter and its contributions are also published in: [46, 47]

Chapter 6 In this Chapter we propose a novel method of Sequencing based on Matrix Factorization Performance Prediction and Vygotsky's concept of Zone of Proximal Development. This approach represents a valid alternative to Reinforcement Learning and other domain dependent solutions. Sequencing contents, like tasks, hints, and feedbacks, is an open issue for Intelligent Tutoring Systems. The common approach is based on domain analysis by experts, who characterize each content with skills involved and a difficulty level. In addition, Machine Learning based sequencers require a specific dataset collection to create users' models and

a Sequencing policy, which needs to be tested online with strong ethical requirements and a high number of users. The contributions of this Chapter are also published in: [42, 45].

Chapter 7 In this Chapter we show how we adapted the Machine Learning based domain independent sequencer of Chapter 6, composed of a Performance Predictor and a score based task Sequencing policy, in order to be integrated in a large commercial online maths ITS.
Thanks to a minimal invasive API we could trial the sequencer with 100 students for a month and discuss the obtained online experiment's results from different perspectives. These contributions are also published in: [43, 48].

Chapter 2

Problem Formulation

Contents

2.1 Student's Knowledge Estimation 11
2.1.1 Evaluation Framework for Static Algorithms 13
2.1.2 Evaluation Framework for Time Evolving Algorithms . 15
2.2 Sequencing 16
2.2.1 Sequencing Evaluation 17
2.3 Data Requirements 18
2.3.1 Datasets for Performance Prediction 18
2.3.2 Exploratory Corpus 21
2.4 iTalk2Learn Datasets 21
2.4.1 Large Commercial Dataset 22
2.4.2 Fraction Tutor Datasets 23

In this Chapter we are going to define, from a machine learning perspective, the problems introduced in Chapter 1. Following the structure of this work and related chapters, the problems of Performance Prediction, progress modeling and sequencing are presented.

2.1 Student's Knowledge Estimation

In this work we start with addressing the problem of student's knowledge estimation for sequencing as a special case of the well known Performance Prediction

problem in ITS. Thanks to the ZPD concept we hypothesized that it is possible to indirectly estimate the students' knowledge over time by the achieved performances (e.g. scores) in structured activities with different levels of difficulty. Performance Prediction will be formalized here as a regression problem, where at each time step the score of the student in the different tasks must be predicted. Of course also a classification modeling is possible, where the task can be either answered correctly or incorrectly, but we will see later in the sequencing section, how the regression approach is fundamental to the working of the VPS.
Information available to the problem are not only task ID, student ID and students' performances. Domain information locates a task in a topic hierarchy where learning units are composed of sections, sections are composed of problems, that are finally subdivided in steps. This scaffolding is necessary to allow an easy automatic evaluation of steps that can be either correct or wrong. Steps are associated with skills or Knowledge Components (KC), so that the predicted probability of answering a task correctly can be associated with the amount of knowledge of the student. Further examples of domain information are: the curriculum structure (sum, subtraction, or multiplication of fractions, additions, etc.), number of skills required to solve the exercise and other information necessary in order to individuate an unique step. In Algebra and Bridge datasets examples of KC or skills are: Circle-Area, Rectangle-Area, Square-Area, etc. Domain in the area of ITS is difficult to obtain as tagging tasks with required skills and difficulties necessitate experts and thus is time-consuming, costly, and, especially for fine-grained skill levels, also potentially subjective. Several taxonomies exist, but nothing prevents ITS developers to use their own formalization [43].
In this work we want to use a completely data-driven approach represented by domain independent algorithms, that predict the students performances by means of latent unobservables parameters. Here only information such as task IDs, student IDs and obtained scores are used [53]. The prediction of a score by mean of only task ID and student ID matches the problem of rating prediction for Recommender Systems. By interpreting the student knowledge as the student state, we implicitly assume that this state needs to evolve in a meaningful and reasonable way over time. This brought to the introduction of Progress Modeling, that can be still formalized as a regression problem, where the model is updated in an iterative way. The machine learning algorithms designed to solve these two problems have high time constraints to be taken into account. In particular, the update should occur without damaging the experience with the system. Therefore, the goal is to update the model in 0.1s as requested by real time applications [34]. Considering Recommender Systems algorithms Time Aware algorithms for the rating problem have been developed. Here the problem of progress modeling differs in granularity

as the time required for user tastes to evolve is different from the abilities acquired from a task. This is also proven by the fact that Time Aware algorithms model time in slices, e.g. five slices for a period of years [55]. As such the problem of progress modeling could be associated to the problem of Time Aware Recommenders with the smallest possible size of time slice. Since learning session are composed of many tasks this work is highly related also to Online Updating Recommenders.

Domain Independent Performance Prediction Performance Prediction is a regression problem where, given a set $S = \{s_1, \ldots, s_i, \ldots, s_{|S|}\}$ of students and a set of tasks C with $C = \{c_1, \ldots, c_j, \ldots, c_{|C|}\}$, we want to predict the real score $y_{ij}^t \in [0, 1]$ that will be obtained by the i–th student s_i in the j–th task c_j, based on his previous performances. The predicted value will be computed by a function $\hat{y}_{ij}^t : \mathbb{S} \times \mathbb{C} \to [0, 1]$. The t index refers to the fact that the prediction changes over time.

Domain Independent Progress Modeling Progress Modeling differs from Performance Prediction since it implies that there is a meaningful evolution over time. As such it must be possible to derive a function $\tau(s_i, c_j)$ that, given the current student's abilities and the task (s)he interacting with, updates the student's model parameters.

2.1.1 Evaluation Framework for Static Algorithms

Performance Prediction algorithms aim at minimizing the Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i,j \in |D_{Test}|} (\hat{y}_{ij} - y_{ij})^2}{|D_{Test}|}}, \tag{2.1}$$

where $|D_{Test}|$ is the total number of data points in the test set. Cross validation is an evaluation protocol that involves partitioning a sample of data into complementary portions. The training of the model is done using one subset (called the training set), whereas validating is done on the other subset (called the validation set or test set) (See Fig. 2.1). To reduce variability, multiple iterations of cross validation are performed using different portions, and the validation results are averaged over the iterations. One of the main reasons for using cross validation instead of conventional validation, e.g. partitioning the dataset into two sets of 66% for training and 34% for testing, is to be sure that an ill posed dataset partitioning does not under- or overestimate the error of the model. The dataset is split in partitions and for each iteration a different set of data is used to train or

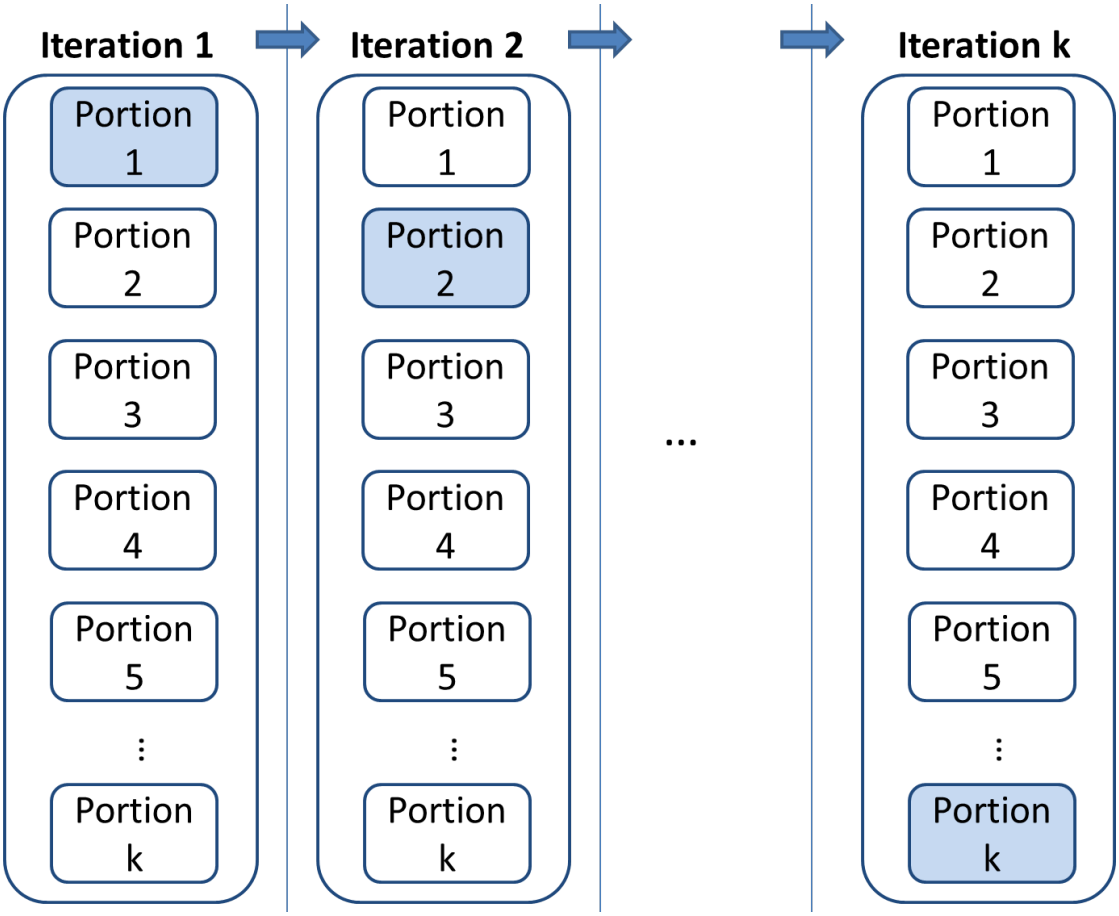

Figure 2.1: Cross validation, the dataset is split in k partitions and for each iteration a different set of data is used to train or test the model. In the figure the test set is highlighted for each iteration in light blue. The computed RSMEs for the k test partitions is then averaged to obtain the final RMSE.

test the model. In Fig. 2.1 the test set is highlighted for each iteration in light blue. The computed RSME for the k test portions is then averaged to obtain the final RMSE.

For ITS datasets the split used is generally the one presented in Fig. 2.2, where the dataset is grouped by student and then ordered from the oldest to the newest entry. The last lines of each student are excluded from the training set and used for evaluation. Generally the proportion 66% for train and 34% for test is mantained. This evaluation approach considers the fact that cross validation cannot be used because it destroys the temporal dependency of the data. The training and testing procedure is nevertheless repeated five to ten times and then the RMSEs resulting are averaged to avoid the influence of the random initialization of the

Figure 2.2: Dataset split for static Performance Prediction. To select the test lines, the dataset is grouped by student and then ordered from the oldest to the newest entry. The last lines for each unit completed by a student, marked with the dark blue color, are excluded from the training set and used for evaluation.

model parameters on the model performances.

For small datasets Leave One Out (LOO) is used as evaluation protocol for small datasets since excluding 34% of the data for test would not leave enough data and the model would not have enough data to generalize. The procedure simply takes the last available line of a student for using it as test sample.

2.1.2 Evaluation Framework for Time Evolving Algorithms

In [55] interesting considerations were made about the requirement of a new evaluation approach for online updating algorithms that consider streams of data.
First of all, as the sequence of the data is crucial in this kind of problems, datasets cannot be shuffled and have to be considered in their natural order. Moreover, shuffling could destroy the time-awareness of the algorithms that take ordered streams of data as input. Consequently, the evaluation of performances in the prediction of past ratings using future ones is rather uninteresting. [55] continues pointing out how online updates allow refining the model as soon as a new data point is available. We want to stress here how this can help reducing many of the issues related to personalized prediction [56]. [55] also informs how grouping

data points in time slices is computationally demanding as it implies, for instance, adding a hyperparameter, i.e. the time slice size. These slices are domain dependent as one could choose winter, spring, summer, and autumn seasons for clothes, months for movies, day of the week, etc., all such combinations needs to be tested and evaluated. Aggregating lines in time slices cannot always be done in ITS, since there is a continuous evolution of the student's state over time. Therefore we consider each time step as relevant and a more fine grained time representation is required. Finally, one should consider that the action of rating an object, or, as in our case, interacting with a task, influences the user. Such aspect can only be taken into consideration by online updating algorithms.
For the aforementioned reasons, a new evaluation approach must be designed. Since in cross validation the natural order of the data is destroyed, it cannot be used to evaluate the performance of the error for data streams. To evaluate the evolution of the error over time and avoid the cold start problem in personalized algorithms such as Matrix Factorization, a part of the dataset must be used for initialization and the other part is used to evaluate the online learning algorithm. Similarly, [55] selects 20% of the dataset for the training and the rest for the online testing.
In this work, we consider two kinds of error for the online testing, the Total_RMSE, i.e. the total RMSE evolution over time, and the SlidingW_RMSE, i.e. the RMSE of overlapping sliding windows of size w. So, while the Total_RMSE gives and overall performance of the algorithm, the SlidingW_RMSE shows the performances evolution over time. How these errors are computed is explained in detail in Alg. 1 for the Total_RMSE and Alg. 2 for the SlidingW_RMSE. The Total_RMSE is computed by taking the root of the by $N^{t=T+1}$ normalized squared error at time T. The SlidingW_RMSE evaluates the RMSE of the last w available data points, i.e. the points in a sliding window of size w over the T–interactions' period.

Algorithm 1 Total_RMSE, $CumErr^{t=T}$ cumulative squared error at time T, $N^{t=T}$ total number of interactions evaluated at time T

Input: $y^{t=T+1}$, $\hat{y}^{t=T+1}$, $CumErr^{t=T}$, $N^{t=T}$
$CumErr^{t=T+1} = CumErr^{t=T} + (y^{t=T+1} - \hat{y}^{t=T+1})^2$;
Total_RMSE$^{t=T+1} = \sqrt{CumErr^{t=T+1}/N^{t=T}}$;

2.2 Sequencing

Let $C \subseteq \mathbb{C}$ and $S \subseteq \mathbb{S}$ be sets of contents and students respectively, $y : \mathbb{S} \times \mathbb{C} \rightarrow [0, 1]$ is a function that computes the performance or the score of a student working

Algorithm 2 $SlidingW\_RMSE^{t=T-(w-1):T+1}$, $CumErr^{t=T-(w-1):T}$ cumulative squared error of the squared error computed from time $T-(w-1)$ to time T, w selected window size

Input: $y^{t=T+1}$, $\hat{y}^{t=T+1}$, $CumErr^{t=T-(w-1):T}$, w
$CumErr^{t=T-(w-1):T+1} = CumErr^{t=T-(w-1):T} + (y^{t=T+1} - \hat{y}^{t=T+1})^2$;
SlidingW_RMSE$^{t=T-(w-1):T+1} = \sqrt{CumErr^{t=T-(w-1):T+1}/w}$;

on a content, and T be the number of time steps assuming that the student is seeing one content every time step. The content sequencing problem consists in finding the optimal policy π^*:

$$\pi^* : (\mathbb{C} \times \hat{Y}_i) \to \mathbb{C}. \tag{2.2}$$

that selects the next content given the available contents and the predicted score on the contents $\hat{Y}_i$. In this work we consider a special kind of sequencing problem as we want to find the best sequence without any domain knowledge, i.e. without knowing the difficulties of the contents and the required skills to solve them.

2.2.1 Sequencing Evaluation

Sequencing problems requires a special approach to be evaluated. We are creating a model able to suggest a sequence given the past outcomes produced by a human being. Consequently, we are evaluating an adaptive and highly individualized set of sequenced actions, whose effect is distributed over time. This generally causes several difficulties in measuring differences in learning gains. Difference in sequencing performances were observed over longer period of time such as one entire semester. Therefore, several success parameters are considered at the same time to get an overview of the contribution of the sequencers implemented. Apart from learning gains, perceived experience, and indicators retrieved from exploratory data analyses are used. Examples of possible indicators are the time required for adapting the sequencer to a new ITS and a comparison between the knowledge of the student estimated or measured in different ways [10].
Further problems for sequencing algorithms evaluation involve the need to retrieve a policy in light of the possible interactions with the system and the derived subsequent state of the user. Being this subsequent state directly influenced by the action of the sequencer algorithm, the effectiveness of a sequencer can be evaluated only with an online experiment involving interacting users. Nevertheless, before having proof of the correct implementation of the sequencer, it is difficult to have access to users. This occurs especially in ITS, where technologies' issues could introduce further frustrating elements to the learning process and therefore jeopardize the experience with the system. Consequently, the algorithms' usual

assessment is a two-steps evaluation, that involves first an offline evaluation of the algorithms interacting with a simulated environment. The latter mimics the learning process and allow parallel developing and testing. If the offline tests are successful, the algorithm can be applied in a real environment. This phase is called online evaluation.

2.3 Data Requirements

In this Section we present the requirements of the data used for training Performance Prediction, Progress Modeling and Sequencing algorithms. By individuating key characteristics of benchmark datasets, we exemplify current research challenges for machine learning applied to ITS personalization tasks.

2.3.1 Datasets for Performance Prediction

To create Machine Learning based Performance Prediction and Progress Modeling, a dataset containing the performances over time of the students in the different tasks is required. In this Section we explain the algorithms' data requirements by considering the publicly available datasets. We later explain in Sec. 2.4, how the iTalk2Learn datasets differ from the commonly used benchmark datasets and what changes to the algorithms were consequently required.
Commonly available benchmark datasets in the area of Performance Prediction and Progress Modeling in ITS: ASSISTments [14], Bridge, and Algebra[1].
We consider the Bridge and Algebra datasets to discuss the typical structure of with ITS collected datasets. Bridge and Algebra were collected by the ITS displayed in Fig. 2.3[2].

Domain Information. As shown in the snapshot of Fig. 2.4[3], we can see that the data does not only contain the minimal information required for performance prediction, i.e. student ID, task ID and performance indicators, but also general information about the tasks. Such information split the tasks in a set of problems. Each problem is then further subdivided in steps that are associated to Knowledge Components (KC). This means, as shown in Fig. 2.2, that each row of Fig. 2.4 can be grouped or analyzed singularly according to the application needs as we will later explain. In Algebra and Bridge datasets examples of KC are: Circle-Area, Rectangle-Area, Square-Area, etc.

[1] `http://pslcdatashop.web.cmu.edu/KDDCup/rules_data_format.jsp`
[2] `http://pslcdatashop.web.cmu.edu/KDDCup/rules_data_format.jsp`
[3] `http://pslcdatashop.web.cmu.edu/KDDCup/rules_data_format.jsp`

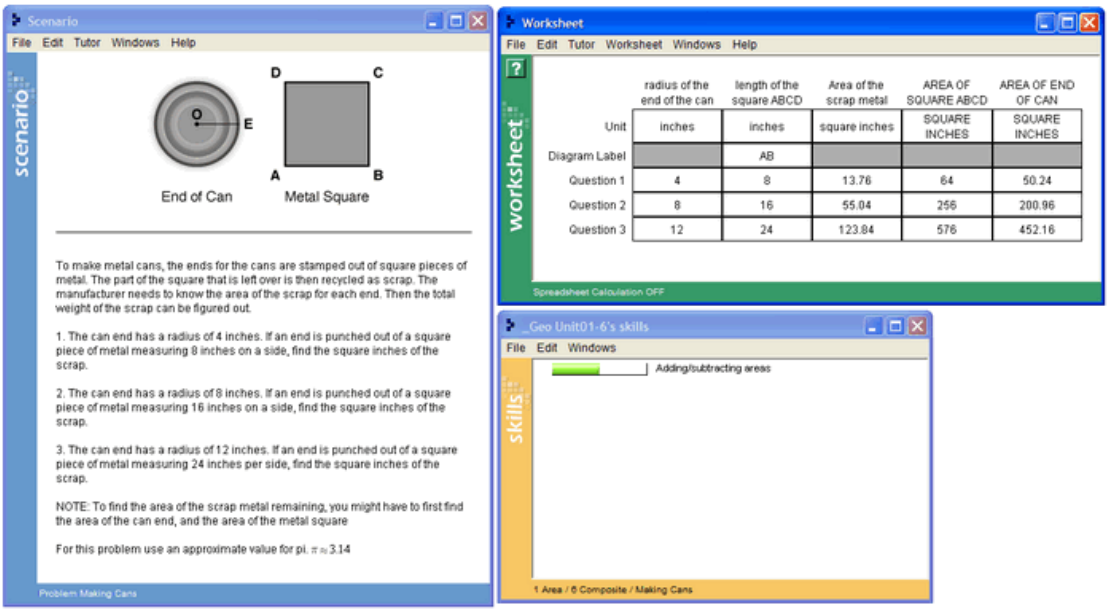

Figure 2.3: Bridge and Algebra Intelligent Tutoring System

Domain in the area of ITS is difficult to obtain as tagging tasks with required skills and difficulties necessitate experts and thus is time-consuming, costly, and, especially for fine-grained skill levels, also potentially subjective. Since this is a mandatory input for domain dependent Performance Prediction, understanding and being able to map such information is crucial as collecting the students' interactions with the system.

Performance Indicators. The log files record of the students, that interacted with the available tasks, contains several performance indicators such as if an incorrect answer was given, the error rate and the opportunity count. The binary performance measure used by BKT approach can be converted to a continuous one without loss in prediction performances [58]. Additional data such as the number of hints requested can be also collected.

Data Sequence. Generally the tasks are shown to the students with a fixed sequence, i.e. equal for everyone, and the data are ordered temporally and per student. This prevents the dataset usage for Reinforcement Learning algorithms [45] as we will explain in the next Section. In addition, the data will be highly unbalanced as more data will be available for the firstly seen tasks rather then the last ones. Generally a limited amount of time can be dedicated to the data collec-

Row	Student	Problem	Step	Incorrects	Hints	Error Rate	Knowledge component	Opportunity Count
1	S01	WATERING_VEGGIES	(WATERED-AREA Q1)	0	0	0	Circle-Area	1
2	S01	WATERING_VEGGIES	(TOTAL-GARDEN Q1)	2	1	1	Rectangle-Area	1
3	S01	WATERING_VEGGIES	(UNWATERED-AREA Q1)	0	0	0	Compose-Areas	1
4	S01	WATERING_VEGGIES	DONE	0	0	0	Determine-Done	1
5	S01	MAKING-CANS	(POG-RADIUS Q1)	0	0	0	Enter-Given	1
6	S01	MAKING-CANS	(SQUARE-BASE Q1)	0	0	0	Enter-Given	2
7	S01	MAKING-CANS	(SQUARE-AREA Q1)	0	0	0	Square-Area	1
8	S01	MAKING-CANS	(POG-AREA Q1)	0	0	0	Circle-Area	2
9	S01	MAKING-CANS	(SCRAP-METAL-AREA Q1)	2	0	1	Compose-Areas	2
10	S01	MAKING-CANS	(POG-RADIUS Q2)	0	0	0	Enter-Given	3

Figure 2.4: Algebra and Bridge snapshot

tion and sometimes it is not possible to allow students to complete the designed sequence by experts.

Sparseness. With sparseness we mean the percentage of tasks seen by each student. The lower the percentage the sparser the dataset and the lower the model performances especially if this is a personalized model. Sparseness is a well known problem in Recommender systems as users rates only a small percentage of the available items. [53] and [56] evaluated how this affected Recommenders applied to ITS datasets by trying out several preprocessing approaches.

Personalization and Cold Start Problem. In addition to the previously mentioned aspects, a very important issue in this context is the small data availability. Cold start problem arises when not enough interactions are available either for the students (students' cold start problem) or for the tasks (tasks' cold start problem). The cold start problem can be ignored when the student interacted with at least 10 tasks and the students practiced with a task at least 10 times [39]. Often domain information is used to reduce data sparsity [53] or cold start problems [56]. Although this problem is common to many applications, as reported in [56], in the case of performance prediction it is even more challenging as novel students are often the only ones available for testing. Moreover, schools generally give an availability of only a few hours to interact with the ITS. Being the prediction fully personalized for the tasks as well as for the students, it is strictly required to have at least some interactions for a student and for a task in the training set

to have a prediction. In the same paper, it is shown how the RMSE suffers from this issue. For these reasons it is important to have an updating model that can partially overcome this problem. More information about the relation between personalization and the cold start problem can be found in Sec. 4.1.1 and Sec. 6.3.2.

2.3.2 Exploratory Corpus

Machine Learning methods targeting at retrieving the correct sequence of actions were developed for robots or artificial agents. Therefore, issues regarding experimentation on human test subjects were not considered, in particular constraints on the number of data required for creating the state model were not set. In addition, for specific sequencing algorithms such as Reinforcement Learning, the dataset has to possess particular characteristics. For instance, with robots there is the possibility of recording different sequences of actions without considering the fatigue of the subject under test. It is needed to evaluate a good percentage of the possible combination of actions. Otherwise the algorithm has not enough information to find an optimal policy.
In order to collect this special kind of datasets, called exploratory corpus, in an ITS domain the tasks should be sequenced randomly. This makes an ethical question arise about the rights we have in suggesting difficult contents to novices or easy contents to experts [10].
In conclusion, the necessity arose to develop a novel approach for sequencing in ITS, which is presented in Chapter 6.

2.4 iTalk2Learn Datasets

Thanks to our collaboration in the iTalk2Learn project we could work with datasets with restricted access, e.g. data collected with several versions of Fraction Tutor [56], and data of a commercial ITS [44]. This implied solving additional challenges as we will describe hereafter.
One of the main challenge of the project was being able to analyze log files within the same framework. iTalk2Learn students practiced with very different ITS, with different performance indicators and domain information structure. One in particular was developed only for iTalk2Learn during the entire lifetime of the project [61][1]. As we will see in this Section, it was of crucial importance utilizing domain

[1] `http://www.italk2learn.eu/wp-content/uploads/2015/10/D1.2-report-on-learning- tasks-and-cognitive-models.pdf`

independent Performance Prediction, so that the large experiment described in Chapter 7 could run within the time period of the iTalk2Learn project.

2.4.1 Large Commercial Dataset

The large commercial dataset used in the experiments of this work was collected with an ITS with 20 topics about maths for children aged from 6 to 14, who can practice with over 2000 tasks at school or at home.
An example of questions proposed to the students can be found in Fig. 2.5. From these questions proposed in sets, that we call interactions or tasks, we do not know which ones precisely were answered correctly since the ITS aggregates the information in a single score.

Domain Information. For these multiple-skills interactions we do not possess the skills involved we cannot use classic domain dependent approaches. To have an idea of how much time is required for an extensive domain analysis for a set of tasks under study, one should consider that domain experts individuated in three years KCs or skills for fractions equivalence's tasks, which represent 2% of the whole datasets' tasks. Since just a small portion of the datasets' tasks was analyzed, it would have been impossible to run a large scale experiment in iTalk2Learn, reported in [43], unless domain independent algorithms were used.

Performance Indicators. As success indicator the ITS uses a score represented in a continuous interval which goes from 0 to 10.
A lesson can be shown in test or exercise mode. The exercise mode consists of approximately 10 questions on a topic and specific learning objectives. While trying to solve those exercises a student can consult several hints, one of those is the bottom-out hint, which displays the solution. The tests, instead, are composed of 5 questions without hints.

Data Sequence. The topics and new skills to be acquired are introduced following the curriculum of the country. The tasks are presented with a rule-based sequencer, which increases the difficulty of the tasks once the student completed and passed all the tasks of the difficulty level. If the tasks are not passed the student gets a regression exercise or can try again to solve the task. This sequencer also relies on the assumption that a student will be able to solve the exercises of the achieved difficulty level but not the more difficult ones without having completed all the lessons of the previous level.
The data granularity level is low if compared with benchmark systems, since we

Figure 2.5: Example of questions posed within a task to the students

possess a single score record for the 10 questions of the exercises and one record for the five test questions.

Cold Start, Personalization and Sparseness. Such aspects and their impact on performance prediction, Progress Modeling and sequencing are evaluated in Section 4.1.2 and in Sec. 6.3.2. In this work we show that it is possible to use this dataset for performance and progress prediction despite the reduced granularity and without using domain information (See Chapter 4). Moreover, the possibility to use this dataset allowed to implement the large scale experiment discussed in in Chapter 7 where we required a system with large dataset to test our fast updating algorithms and API with lightweight integrability.

2.4.2 Fraction Tutor Datasets

Faction Tutor datasets were used during the iTalk2Learn project to evaluate the effect that small datasets have on personalized models' performances. Moreover, they were used to show how little effort was required to adapt the domain independent sequencer to another system (see Sec. 6.3.2). As reported by [56] this was the smallest dataset ever used for Matrix Factorization performance prediction. In particular, the preprocessing of different versions resulted from a cultural adaption was discussed. Being curricula specific for each nation, several adjustments were required between the German and U.S. version [56] as the students age and prior knowledge changed. In this dataset we have then both students' and tasks' cold start problem. Therefore, we present an analysis of the effect of these problems and of sparseness and how they were analyzed solved in Sec. 6.3.2 and [32, 56].

Chapter 3

State of the Art

Contents

3.1 Performance Prediction 26

3.1.1 Domain Dependent Performance Prediction 26

3.1.2 Domain Independent Performance Prediction 30

3.2 State Modeling Techniques 32

3.3 Sequencing in Intelligent Tutoring Systems 33

3.3.1 Rule-based Sequencers 34

3.3.2 Adaptive Rule-Based Sequencers 34

3.3.3 Policy–based Sequencers: Reinforcement Learning . . . 35

'Machine Learning algorithms' refer to algorithms that can learn from the available data. They are able to distinguish between different situations, e.g. distinguish between picture of different objects, and predict an outcome, e.g. price prediction, score prediction, or fuel consumption prediction.
Many Machine Learning techniques have been used to ameliorate ITS, especially in order to extend learning potential for students and reduce engineering efforts for designing the ITS. In this Chapter we will present the state of the art of Performance Prediction, State Modeling, and Sequencing in ITS. For Performance Prediction we will distinguish between domain dependent and domain independent Performance Prediction, for state modeling we will present Kalman Filters that are later used to build Progress Modeling, whereas for Sequencing we will present rule based, adaptive and reinforcement learning based sequencers.

3.1 Performance Prediction

Different algorithms have been applied to model the knowledge acquisition process with the objective of Performance Prediction, i.e. to predict what is the probability of a correct answer or what will be the score obtained in a task. These algorithms are of primary importance when this models are used for task selection.
This Section will start with presenting domain dependent algorithms, which are used not only to predict the score of a student, but also to build a system internal representation of his or her state or knowledge. Afterwards, we will present domain independent algorithms, a valid alternative when the required domain information is not available.

3.1.1 Domain Dependent Performance Prediction

In this section we present the two most famous domain dependent implementations of Performance Prediction: Bayesian Knowledge Tracing and Performance Factor Analysis.

3.1.1.1 Bayesian Knowledge Tracing

The most widely used algorithm for Performance Prediction is Bayesian Knowledge Tracing (BKT), which was introduced by Corbett and Anderson in 1995 and extended and refined in subsequent years [12]. The first implementation consisted of a simple Hidden Markov Model where the performance prediction of all students was modeled by four variables, two representing the performance (probability of learning and probability of forgetting) and two representing the knowledge on a single skill of a student population (probability of guessing and of slipping). In this particular model the knowledge variables considered are called latent features because they are never observed directly. Moreover, knowledge and performance are represented as binary features (i.e., it is assumed that just two states are possible): for a skill learned/not learned and for an answer correct/wrong. Another important variable is the prior probability representing the prior knowledge of the student at the moment he or she starts to use the system. During the training phase, the four previously mentioned variables are estimated using a dataset. Then, the model evolves during the testing phase by increasing or decreasing the probability of learning of a skill according to the student's answers. This is done until the skills can be considered as learned. A skill is considered as such if the probability of a correct answer is greater than 0.95.
Hereafter, we report the formulas of two Bayesian implementation: [2] and [22]. These formulas show how the probability of knowledge, guessing, and slipping are

used to predict the probability of the students to answer correctly. Moreover, they also show how the model changes over time, i.e. how the probability of knowledge is updated. In particular, the equations report that, as aforementioned, CKT consider the probabilities of all involved skills for computing the performance and knowledge probability. The two model's parameters are the initial knowledge or prior knowledge probability, i.e. $P(k_j^0)$, and the transition probability, i.e. $P(T)$. The initial knowledge probability is the probability that a particular skill is already learned at time step zero. The transition probability, instead, is the probability that a skill from time t to time $t+1$ is learned. The two performance parameters are $P(G)$ and $P(S)$, which are respectively the guess and the slip rate.

t	is the current time step.
j	refers to the j–th skill or KC.
y^t	represents the performance at time step t, where $y^t = 0$ and $y^t = 1$ mean the students gave a wrong and a correct answer respectively.
k_j^t	represents the hidden state. It is the knowledge at time t on the j–th skill.
$P\left(k_j^t\right)$	is the learning probability, i.e. the probability that the student learned the j–th skill a time t.
$P\left(G\right)$	is the guessing probability for the j–th skill.
$P\left(S\right)$	is the slipping probability for the j–th skill.
$P\left(T\right)$	is the transition probability, i.e. $P\left(k_j^t = 1 \mid k_j^{t-1} = 0\right)\big)$.
$P\left(k_j^0\right)$	is the initial knowledge or prior knowledge probability, i.e. the probability that a particular skill is already learned at time step zero.
K_1	is the set of known skills, where known means $P\left(k_j^t\right) \geq 0.95$.

Standard BKT Prediction for correctly applying a skill:

$$P\left(y^t = 1\right) = P\left(k_j^t\right)\left(1 - P\left(S_j\right)\right) + \left(1 - P\left(k_j^t\right)\right) P\left(G_j\right) \tag{3.1}$$

Standard BKT skill update for successful steps:

$$P_{\text{posterior}}\left(k_j^t\right) = P\left(k_j^t \mid y^t = 1\right) = \frac{P\left(k_j^t\right)\left(1 - P\left(S_j\right)\right)}{P\left(k_j^t\right)\left(1 - P\left(S_j\right)\right) + \left(1 - P\left(k_j^t\right)\right) P\left(G_j\right)} \tag{3.2}$$

Standard BKT skill update for failed steps:

$$P_{\text{posterior}}\left(k_j^t\right) = P\left(k_j^t \mid y^t = 0\right) = \frac{P\left(k_j^t\right) P\left(S_j\right)}{P\left(k_j^t\right) P\left(S_j\right) + \left(1 - P\left(k_j^t\right)\right)\left(1 - P\left(G_j\right)\right)} \tag{3.3}$$

In the various extensions proposed the researchers have focused on different aspects separately, such as multiple-skill modeling, personalization, time, partial credit and difficulty [2, 35, 36, 59].
Multiple skill modeling has been one of the most important targets of BKT researchers. Creating a classic BKT model for each skill cannot infer properly on the score of multiple step exercises and consequently limit the use of BKT to simple structured exercises. Xu and Mostow (2012) [63] make a comparison between different approaches based upon both joint and not joint computation of multiple skills. The work of [6] and [15] is based on a single skill which is entirely responsible for the outcome of the considered step. [22], instead, suggests a first combination of skills' knowledge to predict students' performances. This method is called Conjunctive Knowledge Tracing (CKT) and modifies knowledge tracing by changing the equations that deal with updating the student model after a student error. The equations for updating after correct student responses are kept the same.

CKT Performance Prediction:

$$P\left(y^t = 1\right) = \prod_j P\left(k_j^t\right)\left(1 - P\left(S_j\right)\right) + \left(1 - P\left(k_j^t\right)\right) P\left(G_j\right) \tag{3.4}$$

CKT Skill Update for failed steps:

$$P_{\text{posterior}}\left(k_j^t \mid y^t = 0\right) = \tag{3.5}$$
$$= \frac{\left(P(S_j) + (1 - P(S_j)) \prod_{i \neq K_1} \left(P\left(k_i^t\right)(1 - P(S_i)) + \left(1 - P\left(k_i^t\right)\right) P(G_i)\right)\right) P\left(k_j^t\right)}{1 - \prod_j P\left(k_j^t\right)(1 - P(S_j)) + \left(1 - P\left(k_j^t\right)\right) P(G_j)}$$

Nevertheless, in CKT the students' knowledge and performance is still modeled as a binary variable. Another more recent approach is proposed by [63], where Item Response Theory is applied instead of Logistic Regression to refine knowledge tracing.
Regarding personalization, there are two papers that are of particular interest. The first, [35], proposes a multiple prior knowledge parameter. The algorithm will decide to which level a student appertains. As a consequence, different students' levels are defined. The second, [26], points out the necessity of modeling all the

variables differently for the students because the probability of knowledge is not equal for each student at a specific time step, therefore the authors suggest creating a model for each student. This information is then used to compute the necessary time each student requires to practice, i.e. the number of exercises to be solved before achieving proficiency.
Time modeling is a more recent advance of BKT and shows that training a model with data that is too old has a negative influence on the model accuracy. This happens because, from one learning session to the others, the student's behavior and knowledge can change [3]. This approach is similar to [31], that we will later describe in more detail.
Partial Credit was also introduced by [58, 59]. A simple equation is developed to create from a binary performance (the score can be either 0 or 1 in public datasets) a continuous one (a score defined between 0 and 1). This strategy proved to be effective for ameliorating the accuracy of BKT. This approach was used in several papers related to this work, like [20, 56], that used with Factions Tutor collected datasets, but required a continuous score measure for task Sequencing.

3.1.1.2 Performance Factor Analysis (PFA)

Another alternative to the aforementioned algorithms for domain dependent Performance Prediction is Performance Factors Analysis (PFA) based algorithms. As pointed out by [9, 15], the first related method to BKT is the Additive Factor Model (AFM) [6] and Performance Factor Analysis (PFA), where the subject ability on a skill, easiness of a skill and the learning rate for each skill are modeled. AFM has been earlier applied to multiple-skills [6, 27]. Although the model considers the frequency of failure and success, the outcome of the exercises performed is not considered. As a consequence, PFM is suggested [37]. Nevertheless, the two algorithms are still considered equivalent since, comparing different error and accuracy measures, one does not outperform the other in a statistically significant way [9]. [9] proposes a new Factors Analysis based algorithm in order to consider instructional interventions during the exercises. The algorithm used is called Instructional Factor Analysis (IFM). In Eq. (3.8) one can find how the probability of a correct response is computed by the different algorithms.

$$\textbf{AFM:} \quad \ln \tfrac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj} \left(\gamma_k N_{ik}\right) \tag{3.6}$$

$$\textbf{PFA:} \quad \ln \tfrac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj} \left(\mu_k S_{ik} + \rho_k F_{ik}\right) \tag{3.7}$$

$$\textbf{IFM:} \quad \ln \tfrac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k Q_{kj} \left(\mu_k S_{ik} + \rho_k F_{ik} + \nu_k T_{ik}\right) \tag{3.8}$$

Where: These algorithms model the Performance Prediction problem as a Machine

i	i-th student
j	j-th step
k	k-th KC or skill
p_{ij}	probability that student i performs step j correctly
θ_i	coefficient representing the proficiency of student i
β_k	coefficient representing the difficulty of the skill or KC k
Q_{kj}	matrix informing weather step j requires knowledge about skill k
γ_k	learning rate for KC or skill k
N_{ik}	number of times a student has practiced on a skill
μ_k	coefficient representing the benefit of previous successes
S_{ik}	number of previous successes of student i on skill k
ρ_k	coefficient representing the benefit of previous failure on skill k
F_{ik}	number of previous failures of student i on skill k
ν_k	coefficient representing the benefit of previous tells on skill k
T_{ik}	number of previous tells of student i on skill k

Learning regression. The main difference between them is based on the number of features considered.

3.1.2 Domain Independent Performance Prediction

Matrix Factorization (MF) has many applications like, for instance, dimensionality reduction, clustering and also classification [11], but its most famous application is for Recommender Systems [24], where the algorithm recommends items to a user by predicting the ratings (s)he would give to them. A matrix, whose elements represent the ratings users gave to some items, is decomposed to approximate the missing values.
When algorithms are used for novel applications, several adjustments and analysis are required. In this Section we refer first to work where static MF was adapted to Intelligent Tutoring Systems. Then we continue presenting Time-Aware MF and Online Updating MF. Online Updating MF are of particular interest for this work as they can be used for task Sequencing as we will show in Chapter 6.

3.1.2.1 Matrix Factorization in Intelligent Tutoring Systems

The first times that MF was applied to ITS, [51, 52, 53] associated users with students, items with tasks, and ratings to the probability of a correct answer at first attempt.

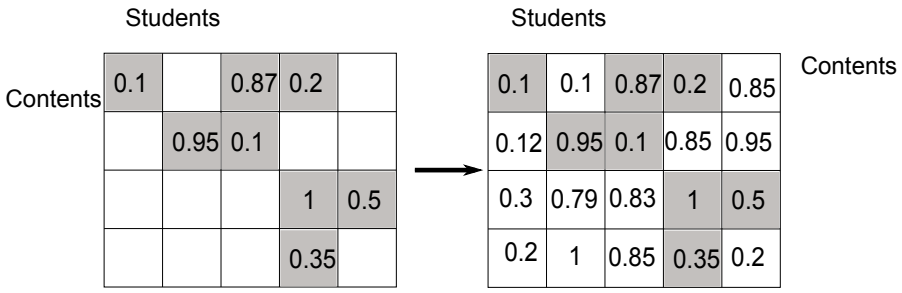

Figure 3.1: Table of scores given for each student on tasks (or interacting with generic contents) (left), completed table by the MF algorithm with predicted scores (right).

As these algorithms are the one we used as comparison for our developed method, we are going to explain them into detail in Sec. 4.1.

3.1.2.2 Time-aware Recommender Systems

Time-aware recommender system includes time as context information, that often is modeled by learning the prediction of an item by a user at a specific time. In papers predicting movie ratings or doing item recommendations, such as [62] and [28], the data of a user are divided in time slices. Within the time slice the user's model is constant, and differs between time slices. This works well for applications with day or seasonal influence and allows to have an implicit update of the model without increasing too much the model dimensions that grows with the number of slices. [16], instead, models the time in a two–step approach as its main tasks is ranking based on users' previous ratings. First item latent features are learned by a "continuous bag of word". A neural network predicts an item of a user's item sequence. This allow to code in the item latent features sequential information. Also in this case the time is modeled as slices. The so learned items' latent features are used to enrich the vanilla MF model by adding them as additional elements to the items' latent features. In the state of the art presented involving movie ratings and ranking time slices where aggregated in months for a total of 5 slices for the entire years-lasting datasets.
As aforementioned, the prediction of student's performances over time implies having fast-changing users' states. For this reason, a time-slices approach does not seem to be the more suitable way to model what is more a stream of data. Moreover, Time-aware Matrix Factorization not necessarily has to run online, i.e. while users are interacting with a system or with real time performances. At the same time, Time-aware MF methods do not have to be an update of already existing models, therefore more similar to our approach are Incremental MF or Online

Learning MF. The online update proposed in [40] learns for each new sample available the model forgetting the previous estimate. This is slightly different than the so-called "incremental matrix factorization" [1, 55], that update the model incrementally, i.e. they update the current model with the latest state obtained. However, as also shown and explained in Chapter 5, [40] is still better performing as it is a multiple-epoch SGD instead of a single-epoch one as [55] and [1]. Nevertheless, computational performances should be considered. As explained in Chapter 7, considering the entire history of a student requires heavyweight queries that extract this information from the database. As reported by [43] 6 seconds were required for an update, whereas real time performances should stay under the 0.1 seconds threshold [34].
Therefore, forgetting methods were proposed. [31] shows how one can shrink the data used to retrain the users' latent features as the more data are available the more time is require for the update. In [13] and [31] methods to reduce the number of data points is selected by a set of possible rules, that are dataset dependents and require context data. Nevertheless, such approach usually reduces the accuracy of the method. Moreover, how the algorithms forget is application specific and needs to be investigated for ITS.
The online update proposed by [40] is explained in detail in Sec. 4.2.1. The online update proposed by [1, 55] is explained in detail in Sec. 4.2.2.

3.2 State Modeling Techniques

Kalman Filters are one of the most used state estimation algorithms in operations research [21]. They constitute a valid approach to our Progress Modeling problem as they are able to model next state just considering the input, the observations, and the current estimate of the state. Moreover, in such algorithms the sequentiality of the measurements plays a major role. Classic Kalman Filters are used to model a linear relationship between the next and current state and the current observation. Nevertheless, several other extensions exists such as, for instance, the Extended, the Unscented and the Particle Kalman Filter. The Extended Kalman Filters (EKF) relax the assumption that a linear relationship between state and observation is required. In order to do so the non-linear state and measurement functions are approximated via Taylor expansion under the assumption of linearity of a function around one point. Despite its efficiency in comparison to other Kalman extensions, the EKF could still fail to deliver an acceptable prediction. This happens if the uncertainty is too high or if the functions do not have a local linearity. Therefore, the Unscented Kalman Filters (UKF) implement linearization in a different way. The unscented transform is used to model the state by means

of a derivativeless technique. The performances of EKF and UKF are comparable as the UKF are just a factor slower than EKF and UKF predictions are equal to those generated by the Kalman Filter for linear systems. For non-linear systems UKF produces equal or better results than EKF. However, if the distribution of the filter's belief is highly non-linear, the UKF performs poorly and Particle Filters represents a better choice [54]. Particle Filters partition the state space in many parts where particles are filled according to some probability measure. The higher the probability, the denser the particles are concentrated, that represents the evolving probability density function [8].
As classic Kalman Filters were the algorithm we used as proof of concept that Kalman Filters and Matrix Factorization can be combined to implement online Performance Prediction, in Sec. 5.1 we describe such algorithm in detail.

3.3 Sequencing in Intelligent Tutoring Systems

In this work we present algorithms used for task Sequencing, that respect the data requirements presented in Sec. 2.3. In this Section in particular, we distinguish between five kinds of sequencers:

1. **Fixed Sequencer:** This sequencers follow a fixed path designed by human experts.

2. **Self Choice Sequencer:** This sequencers suggest a fixed sequence or a set of alternatives, but it is the user that at long last decides with which task to practice.

3. **Rule–based Sequencers:** This sequencers use a set of "if...then..." rules to select the next task. Those rules are designed by human experts basing on automatically retrievable indicators, e.g. the score or the difficulty of the task. When a rule is triggered the corresponding previously decided action is performed. This could be to increase or decrease the difficulty or change skill to be learned.

4. **Adaptive Rule–based Sequencers:** These sequencers are an extension of rule–based ones. They use "if...then..." rules but the indicators of the student's state are provided by intelligent components, e.g. Performance Predictors or emotion recognition.

5. **Policy–based Sequencers:** This sequencers implement the sequencing rules in the form of an equation that takes as input the estimated students' state. This equation is called policy and it can be either learned as in the case

of reinforcement learning or inspired by pedagogical theories as the Vygotsky Policy Sequencer presented in Chapter 6.

Hereafter, we introduce the current state of the art on the topic of Sequencing in ITS. We start with rule–based sequencers, we continue with adaptive rule–based sequencers, and finally with policy based sequencers.

3.3.1 Rule-based Sequencers

Rule–based sequencers are used by ITS to schedule tasks in a way that imitates adaptivity. The users proceed through a main sequence designed by experts according to if (s)he pass or fail tasks. Depending how serious the failure is, different actions might be executed by the ITS. Generally, entire subsequences are designed, that consist of specific tasks or repetitions of the failed tasks. The design of a rule–based sequencer requires years of work for experts, as the sequences and subsequences need to take under consideration every possible scenario. A radical solution to this problem is to allow the students to select the contents they want to practice with. However, this approach cannot be applied in ITS since not every user knows what task is better for him/her [43].
In conclusion, despite the fact such sequencers have some adaptivity based on simple indicators, we do not refer to them as adaptive sequencers as they have a reduced set of alternative paths and therefore can be rather seen as an extension of fixed sequencers.

3.3.2 Adaptive Rule-Based Sequencers

Newly developed method for Sequencing in ITS is the so–called adaptive rule–based sequencer. This sequencer is characterized by the important role played by Machine Learning methods, which, with their predictions, produce more accurate indicators of the current state of the student.
BKT researchers have discussed the problem of Sequencing both in single and in multiple skill environment in [22]. However, as pointed out by the same authors, their policy was not able to order the tasks in order of difficulty as easy and difficult tasks were alternated. Moreover, [22] underline how multiple skill tasks are modeled as single skill ones in order to overcome current BKT limitations. We would like to stress that the Sequencing requires an internal skills representation and consequently, together with the Performance Prediction algorithm, is domain dependent.
Mazziotti et al. [32] suggest Sequencing approach using indicators computed by

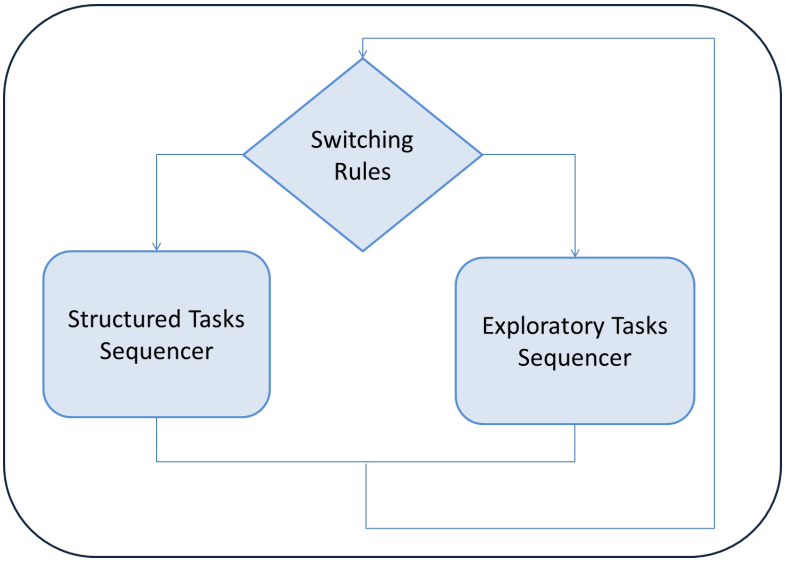

Figure 3.2: [32] cycle for switching between structured and exploratory tasks

intelligent components performing emotion recognition, speech, and log-files analysis. Such intelligent components can recognize if the student is under–, over–, or appropriately challenged. These indicators are used to define the rules that decide whether it is necessary to switch from structured to exploratory tasks or the other way around. The cycle shown in Fig. 3.2 is performed after each task is completed. Once it is decided which type of task the student must practice with, a type-specific sequencer performs the next action. The role of the sequencer used to schedule structured tasks could be taken by the sequencer presented in Chapter 6.

3.3.3 Policy–based Sequencers: Reinforcement Learning

Reinforcement Learning (RL) is a Machine Learning approach that computes the best sequence trying to maximize a previously defined reward function. Both model–free and model–based [4, 29] RL were tested for content Sequencing.
Unfortunately, the model–based RL necessitates of a special kind of datasets called exploratory corpus. Available datasets are log files of ITS which have a fixed Sequencing policy that teachers designed to grant learning. They explore a small part of the state–action space and yield to biased or limited information. For instance, since a novice student will never see an exercise of expert level, it is impossible to retrieve the probability of a novice student solving some contents. Without these probabilities the RL model cannot be built [10]. Model–free RL,

instead, assumes a high availability of students with which one can perform an online training. The model does not require an exploratory corpus but needs to be built while the users are playing with the designed system. Given the high cost of an experiment with humans, most authors exploit simulated single skill students based on different technologies like Artificial Neural Networks or self developed student models [29, 41] for an offline evaluation. Particularly similar to our appraoch is [29], where contents are sequenced with a particular model–free RL based on the actor critic algorithm [23], which was selected because of its faster convergence in comparison with the classic Q–Learning algorithm [50]. Nevertheless, RL algorithms still need many iterations to converge and will always need preliminary trainings on simulated students also after their evaluation.

Chapter 4

Online Learning Matrix Factorization for Performance Prediction

Contents

4.1 Static Matrix Factorization 38
4.1.1 MF and BKT Comparison 39
4.1.2 MF for Commercial ITS 40
4.2 Updating Matrix Factorization 41
4.2.1 Matrix Factorization Update 41
4.2.2 Incremental Matrix Factorization 43

In this Chapter we explain in detail Matrix Factorization and its online updating versions, that are used in Chapter 5 as comparison to the developed Progress Modeling approach.
Contributions of the work presented here are:

- Feasibility of Domain Independent Matrix Factorization for Performance Prediction in ITS and
- Feasibility of Online Leaning Matrix Factorization for for Performance Prediction in ITS.

These contributions are also published in: [43, 44].

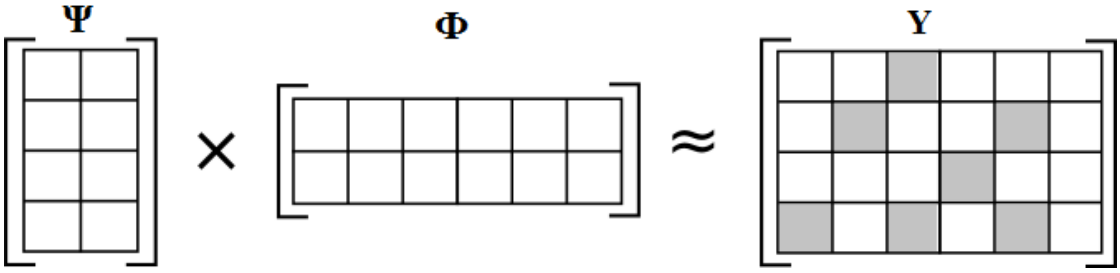

Figure 4.1: Matrix decomposition

4.1 Static Matrix Factorization

As described in Chapter 2, given are a set $S = \{s_1, \ldots, s_i, \ldots, s_{|S|}\}$ of students and a set of tasks C with $C = \{c_1, \ldots, c_j, \ldots, c_{|C|}\}$. We want to predict the real score $y_{ij}^t \in [0, 1]$ that will be obtained by the i–th student $s_i \in S$ in the j–th task, $c_j \in C$. s_i is modeled by Matrix Factorization (MF) as a vector $\boldsymbol{\varphi} \in \mathbb{S} := \mathbb{R}^K$, where K is the number of skills involved and $\mathbb{S}$ is the student's space. Also the j–th task is defined as a vector $\boldsymbol{\psi}_j \in \mathbb{C} := \mathbb{R}^K$ representing the K skills required to solve a task defined in the tasks' space $\mathbb{C}$. In this context we want to find a suitable prediction function able to compute the predicted performance $\hat{y}(\varphi_i, \psi_j)$ of a student s_i on a task c_j considering all his past interactions. For static MF we assume that $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ are constant over time.

In static MF, the matrix $Y \in \mathbb{R}^{n_s \times n_c}$ can be seen as a table of n_c total tasks and n_s students used to train the system, where for some tasks and students performance measures are given. MF decomposes the matrix Y in two other ones $\Psi \in \mathbb{R}^{n_c \times K}$ and $\Phi \in \mathbb{R}^{n_s \times K}$, so that $Y \approx \hat{Y} = \Psi\Phi^T$. Ψ and Φ are matrices of latent features, where the latent features of each task c_j and each student s_i are collected. Although these latent features cannot be mapped to an exact meaning as done in BKT technology, in [53] those values were associated with the skills involved in the tasks and the skills of the students. The latent features learned with stochastic gradient descent from the given performances allow computing the missing elements of Y for each student i in each task j of a dataset D (Fig. 3.1) without manually tagging the skills of the domain. For this reason this approach has been called domain independent in [45]. This approach is also called static, as the model parameters are not updated over time. The optimization function of MF is represented by:

$$\min_{\boldsymbol{\psi_j}, \boldsymbol{\varphi_i}} \sum_{i,j \in D} (y_{ij} - \hat{y}_{ij})^2 + \lambda(\|\Psi\|^2 + \|\Phi\|^2) \tag{4.1}$$

where one wants to minimize the regularized Root Mean Squared Error (RMSE) on the set of known scores. The prediction function is represented by:

$$\hat{y}_{ij} = \sum_{k=0}^{K} \varphi_{ik}\psi_{jk}, \tag{4.2}$$

A simple extension of Matrix Factorization is Biased Matrix Factorization (BMF), where μ, μ_c and μ_s are respectively the average performance of all tasks of all students, the learned average performance of a content, and learned average performance of a student. The two last mentioned parameters are also learned with the gradient descend algorithm.

$$min_{\boldsymbol{\psi_j},\boldsymbol{\varphi}_i,\mu_{cj},\mu_{si}} \sum_{i,j\in D} (y_{ij} - \hat{y}_{ij})^2 + \lambda(\|\Psi\|^2 + \|\Phi\|^2 + \|\mu_{cj}\|^2 + \|\mu_{si}\|^2) \tag{4.3}$$

where one wants to minimize the regularized Root Mean Squared Error (RMSE) on the set of known scores. The prediction function is represented by:

$$\hat{y}_{ij} = \mu + \mu_{cj} + \mu_{si} + \sum_{k=0}^{K} \varphi_{ik}\psi_{jk}, \tag{4.4}$$

4.1.1 MF and BKT Comparison

We report hereafter the RMSE obtained by [53], whose implementation we used to calibrate our. As one can see in Tab. 4.1, MF outperformed two BKT reference implementations [2] and [7]. In order to avoid sparseness in benchmark datasets [53] each line was abstracted to step level, i.e. the algorithm predicts if the student is going to answer correctly to a specific step. Having the authors shared the used hyperparameters we obtained equal results up to the third value after the comma. The discrepancies can be imputed to the random initialization of the model parameters.

Algorithm	**Algebra**	**Bridge**	**ASSISTments**
BKT - EM [7]	0.31098	N/A	0.48860
BKT - BF [2]	0.31308	0.30849	0.49353
MF [53]	0.29898	0.29446	0.46041
BMF [53]	0.29819	0.29385	0.45822

Table 4.1: In this table we report the RMSE obtained by [53] with MF step preprocessing.

4.1.2 MF for Commercial ITS

In this Section we discuss how MF can be applied in a commercial ITS, which records data such as those described in Sec. 2.4.1. As aforementioned, the data granularity level of this dataset is low if compared with benchmark systems, since we possess a single score record for the 10 questions of the exercises and one record for the five test questions. Having an aggregated test value for 5 different questions on different concepts a multiple skill representation of the lessons would be the most plausible. Nevertheless, this information is not available for all the lessons, which are summarized with a single learning objective description.
Given the differences with the other benchmark datasets we needed first to discuss whether these data could be used for Performance Prediction and think of an eventual preprocessing.
For the preprocessing, we started from the approach proposed in [53], who underlined the necessity to avoid sparsity to maximize MF performances. Since we did not have information about the performance of the students up to the step level, we predict the score on the single lessons. We then distinguish, if the lesson was solved in exercise or test mode. This was done, since the use of hints strongly influences the outcome of the exercise session and modifies the experiment modalities. Moreover, we removed the skipped lessons in order to have only completed ones.
We followed the standard approach in the field to divide the student history temporally in two thirds for training and one third for testing (see Sec. 2.1.1), evaluating the performances with the RMSE. The score, as in [45], is represented in a continuous interval which goes from zero to ten, normalized between zero and one.
In Tab. 4.2 we present the statistics of the dataset used for the experiments. In particular, out of the 30 Million lines recorded, approximately two thirds were seen in exercise mode, i.e. with a bottom out hint available. Approximately half of the tasks seen in exercise mode used the bottom out hint. For this reason, exercise scores must be considered carefully, especially for adaptive sequencing. In Tab. 4.3 we present Global Average, i.e. a worst case predictor that assumes students will always perform equally to the global score average computed on the training dataset. The Biased Student-Task predictor, instead, uses only the biases μ, μ_s, and μ_c of Eq. 4.4, i.e. the latent features number K is set to zero. Consequently, out of Tab. 4.3 one can see the contribution of the single components of Eq. 4.3 in ameliorating the prediction.
In conclusion, according to the results in Tab. 4.2, the dataset is suitable for the task of Performance Prediction despite the reduced granularity and MF is able to predict a continuous interval performance in a multiple-topic scenario. This is different to what was done in e.g. [12], where the main task was to predict if

Table 4.2: Dataset Statistics

Number of Tasks (Exercise and Test lessons)	2 035
Number of Students	258 391
Total Student-Task Interactions	30 813 070
Total Exercise sessions	17512972
Exercise passed (Score 70-99)	9 520 278 i.e. 54
Gaming the system (Score 100 + Bottom out hint)	3 988 891 i.e. 23%
Total Test sessions	13 300 098
Test session passed (Score 60-99)	4 378 461 i.e. 33%
Average score obtained	8.1

Table 4.3: Performance Prediction Error

Experiments, score range [0,1]	**RMSE**, ± SD over five experiments
Global average	0.3032796
Biased User-Item Exercise	$0.2639167 \pm 3.6989\ 10^{-5}$
Biased User-Item Topic	$0.26416832 \pm 3.36935\ 10^{-5}$
Task Preprocessing	$0.26061115 \pm 5.97504\ 10^{-5}$

the student was going to answer correctly at first attempt. Finally, thanks to the positive results of single tasks' score prediction, we can conclude that with this dataset it is possible to implement domain independent Performance Prediction.

4.2 Updating Matrix Factorization

Several Time-Aware Matrix Factorization approaches exist, in this work we will focus on online–updating ones that can be used for sequencing. In particular, we present two algorithms called hereafter Matrix Factorization Update [40] and Incremental Matrix Factorization [1, 55].

4.2.1 Matrix Factorization Update

One of the criticized problems of MF is that it does not deal with time, i.e. the latent features are constant after the first training. In order to keep the model up to date, [43] implemented, in a large commercial ITS, the online update proposed in [40]. The update, that we will call hereafter UpMF, consists in solving again the minimization problem of Eq. (4.5) optimizing only $\boldsymbol{\varphi}_i$ with stochastic gradient

Algorithm 3 UpMF [40], where β is the learning rate, λ is the regularization parameter, Ψ are the tasks' latent features, $iter_{Max}$ is the number of algorithm's iterations, $History_i$ are all the tasks IDs j the student i interacted with performance y. The algorithm computes $\boldsymbol{\varphi_i^t}$, i.e. the latent features of the student at time t.

Input: $History_i$, λ,Ψ, β, K, $iter_{Max}$
Output: $\boldsymbol{\varphi_i^t}$
$\boldsymbol{\varphi_i^t} \sim N(0, \sigma^2)$;
$iter_{Max}$= $History_i.length * iter_{Max}$;
for *iter = 1 to* $iter_{Max}$ **do**
    Select j randomly from $History_i$;
    $Err = y - \left(\sum_{k=0}^{K} \varphi_{ik}^t \psi_{jk}\right)$;
    for k *= 1 to* K **do**
        $\varphi_{ik}^t += \beta\,(err * \psi_{jk} - \lambda \varphi_{ik}^t)$;
    end
end

descent algorithm considering the interactions of the single student i.

$$\min_{\boldsymbol{\varphi}_i} \sum_{j \in D_i} (y_{ij} - \hat{y}_{ij})^2 + \lambda \left(\|\varphi_i\|^2\right) \tag{4.5}$$

This means the student's model is learned at each interaction from the beginning forgetting the previous estimate. [43] coherently with [40] noticed that after approximately 20 interactions the model update's error for UpMF was degenerating. [43] overcame the problem by retraining the model each night, assuming students would see approximately 10 tasks per day. This was of course imposing strong requirements on the machine where the application ran since the training is more demanding computationally in comparison to the prediction phase.
According to the pseudo-code Alg. 3 reported, several limitations of this algorithm can be identified as we will see in Sec. 5.3.4. The main one that we can identify in Alg. 3 is the dependency between the history length and the number of algorithm's iterations required to converge to a solution. The more student's interactions are available, the more iterations are needed by UpMF to converge. As a consequence the time required to update the model increases over time (See Sec. 7.4). To keep the update time constant one should select meaningful samples out of the given history. This problem was tried to be solved with the help of context data in [31] and is explained in Sec. 4.2.2. However, this approach affect the error performances which are reduced. This happens because less data points are used to train the model.

4.2.2 Incremental Matrix Factorization

The difference between Alg. 3 and Alg. 4 is that in the latter the update is performed in an incremental way, i.e. starting from the last estimation of $\boldsymbol{\varphi_i}^{t-1}$ and using j-th task's latent feature $\boldsymbol{\psi_j^t}$ and the score $y_{i,j}^t$ the i-th student obtained. [1, 55] online update (without context information) is a one-iteration SGD, therefore, as we will see in Chapter 5 it will have a faster computational performance but worse RMSE performances in comparison to UpMF, which is a multiple-iteration SGD and it can be seen as a retraining of students' latent features. The pseudocode of such algorithm is presented in Alg. 4. Alg. 4 was extended by [31] by

Algorithm 4 Abernathy [1, 55], where β is the learning rate, λ is the regularization parameter, ψ_j^t are the task's latent features of task j and $y_{i,j}^t$ is the score, the last task ID j the student i interacted with with performance $y_{i,j}^t$. The algorithm computes $\boldsymbol{\varphi_i^t}$, i.e. the latent features of the student at time t.

Input: $\boldsymbol{\varphi_i^{t-1}}$, λ,$\boldsymbol{\psi_j^t}$, $y_{i,j}^t$, β, K
Output: $\boldsymbol{\varphi_i^t}$
$Err = y_{i,j}^t - \left(\sum_{k=0}^{K} \varphi_{ik}\psi_{jk}\right)$;
for *k = 1 to K* **do**
  $\varphi_{ik}^t += \beta\left(err * \psi_{jk} - \lambda\varphi_{ik}^{t-1}\right)$;
end

adding a forgetting mechanism that reduces the error but unfortunately also slows down the algorithm as the forgetting procedure also requires some computational time.
The algorithm proposed is a multiple-iteration SGD like [40] but a subset of ratings is selected to retrain the model. The simplest feasible forgetting approach consists of a sliding window that selects the last N interaction of a user. Other more complicated approaches require to individuate the most popular items or scale the user latent feature by a factor.
There is a high difference in performances when these forgetting methods are applied to different datasets, moreover these forgetting methods require additional hyperparameters to be selected and therefore computational performances are affected. Finally, some forgetting approaches cannot be extended to all domains, for instance post popular item selection cannot be applied to tasks or hints selection in ITS.

Chapter 5

Progress Modeling

Contents

5.1 Kalman Filter theory 47

5.2 Kalman State Estimation for Matrix Factorization . . 49

5.2.1 Simple previous/next State Mapping 50

5.2.2 Skill Deficiency Aware KSEMF (KSEMF_SD) 50

5.3 Experiments 52

5.3.1 Dataset characteristics 52

5.3.2 Hyperparameters' Selection 54

5.3.3 State Variables' Initialization 54

5.3.4 RMSE Evaluation 54

5.3.5 Evaluation of the Cold Start Problem 62

5.3.6 Modeling Student Progress 62

5.3.7 Personalization 65

5.4 Conclusions 68

In this Chapter we want to go a step forward with respect to domain independent performance prediction. From an approach informing only on the current/next state of the user, we move to Progress Modeling, where the students' state has to evolve in a meaningful, plausible and therefore interpretable way over time.
In this scenario three problems arise:

1. Domain information, like tagging involved skills in tasks, necessitates experts and thus is time-consuming, costly, and, subjective.

2. Progress Modeling requires to be able to interpret the model, i.e. to be able to associate the value of the model parameters with a specific user state.

3. The continuously changing student's state and the necessity of new data requires online updating algorithms, that refine their prediction after each interaction.

Problem (1) involves domain dependent performance prediction algorithms: Bayesian Knowledge Tracing (BKT) [12] and Performance Factors Analysis (PFA)[37]. Therefore, [53] proposed Matrix Factorization (MF) as domain agnostic alternative. As such, MF does not require skills' tagging (see Sec. 4.1.2) and can be easily integrated in larger systems (See Chapter 7).
[30] stresses the fact that intelligent components' too high requirements often prevent them to be used in commercial environments. Therefore, domain independence and lightweight integrability are desirable properties as large ITS often do not possess the skills involved in the tasks and cannot invest large efforts either in tagging all their contents or in changing their systems'structure.
If MF algorithms represent a good solution to Problem (1), they unfortunately suffer from Problem (2) since the parameters of the model cannot be used to interpret the state of the user and its progress over time.
Finally, Problem (3) arises for MF applications in ITS. Item Recommendation is affected by time differently than task recommendation, since voting movies in different permuted orders will not affect the user's ratings unless a long time passes between ratings. For this reason it is possible to model user evolution and item characteristics in aggregated time slices, where more subsequent ratings are considered as generated from the same static model.
In ITS, instead, the students learn something according to their learning rate after each exercise. If exercises are shown in order or in reverse order of difficulty not only scores will change, but also the acquired knowledge will be different. Therefore the usage of an online updating model is mandatory for an accurate prediction. Best way to do so, would be to refine the prediction after each action to allow in the future also hints and feedback recommendation.
In this Chapter we first explain Kalman Filters, a state estimation algorithm used in operations research [21]. Then we continue explaining how MF and Kalman Filters were combined to find an updating function for the students' model. This is achieved in two different ways, one of which exploits equations of a student simulator (See Sec. 6.1.2) mimicking the learning process of a student.
We perform an extensive evaluation considering different error measures, cold start problem and interpretability of the model.
As a result, the model:

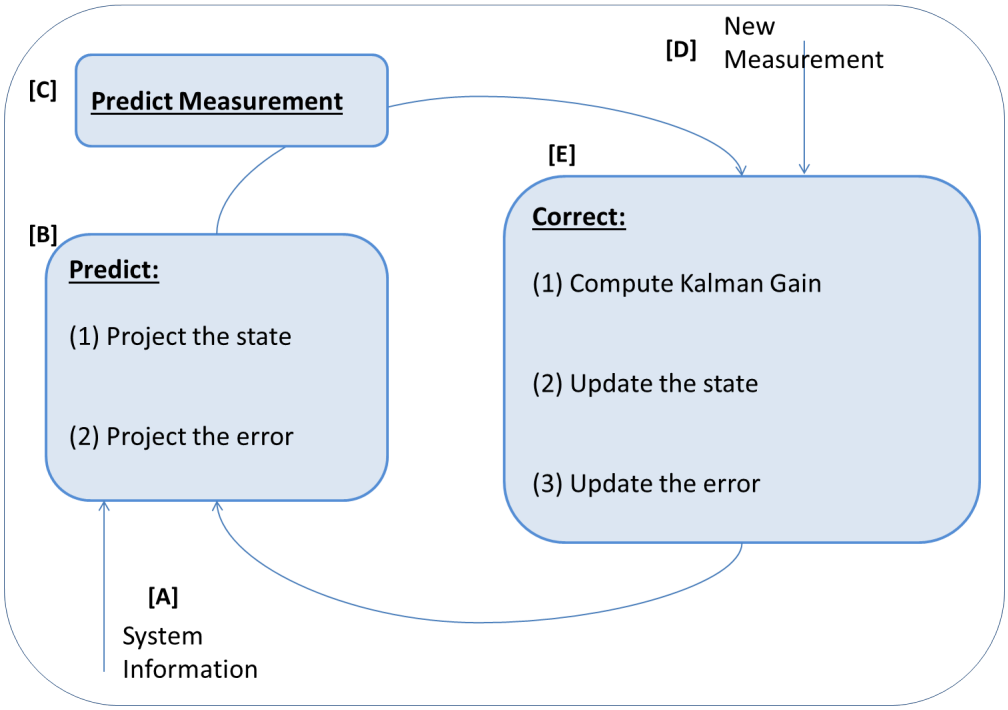

Figure 5.1: Kalman Cycle

1. has reduced computational requirements,
2. remains domain independent,
3. has a reduced prediction error,
4. is less sensitive to the lack of user data,
5. and is made interpretable.

These contributions are also published in: [46, 47].

5.1 Kalman Filter theory

Kalman Filters constitute a valid approach to our Progress Modeling problem as the sequentiality of the measurements plays a major role. For their recursive structure they do not require the load of the entire student's history to compute the update, so that the update time is constant.
Before explaining the Kalman's equations into detail, we briefly introduce the Kalman approach in modeling state's evolution. As shown in Fig. 5.1 the Kalman algorithm can be described as a two-steps cycle, where the alternating phases are called "predict phase" and "correct phase". During the predict phase the state and error estimation are projected, i.e. one tries to predict what influence a

certain input would have on the state's evolution. The input can be seen as the set of actions that could be performed and our goal is to find the best one for the specific situation modeled by the system. With a more actual state estimate we can try to predict the next observed measurement, where the measurement is what we can observe from the surrounding environment. When a new measurement is finally observed, the algorithm uses the information to correct its estimates. At this point, the cycle starts from the beginning. In order to implement the cycle equations, it is required to identify in the problem under analysis the state, input and observable measurement of the system as shown in Fig. 5.1 with the step "[A] System information". How this was done for student Progress Modeling is one of the key contributions of this Chapter.
The state $\boldsymbol{x}$ at time t is modeled as a linear combination of the state at time $t-1$ and a control input $\boldsymbol{u}$ at time $t-1$ with additive Gaussian noise $\mathbf{w}$ (Eq. (5.1)), where A and B are matrices of coefficients multiplying the state and control variables respectively.

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + w_t \tag{5.1}$$

In Eq. 5.2 the measurements of the environment are predicted adding the current state estimation multiplied by a coefficient matrix H to Gaussian noise $\mathbf{v}$.

$$y_{t+1} = H\boldsymbol{x}_t + v_t \tag{5.2}$$

Instead of learning from scratch the student's parameters after each interaction, the Kalman Filter updates its estimation at each time step with the predict (Eq. (5.3)) and correct (Eq. (5.4)) equations, integrating in its prediction the novel available information. Kalman Filters computes the belief on the current state modeling its mean $\hat{\boldsymbol{x}}_t^-$ and its error covariance matrix P_t^- with Eq. (5.3), where Q is the state noise covariance matrix derived from the Gaussian noise variance $\mathbf{w}$ of the state variables.

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1}^- &= A\hat{\boldsymbol{x}}_t + B\boldsymbol{u}_t \\ P_{t+1}^- &= AP_tA^T + Q \end{aligned} \tag{5.3}$$

Then, with a new measurement $\boldsymbol{y_t}$, state estimation $\hat{\boldsymbol{x}}_t$ and error covariance matrix P_t are corrected with Eqs. 5.4, where K_t is the so-called Kalman Gain and R the measurement noise covariance matrix derived from the variance of the measurement noise $\boldsymbol{v_t}$.

$$\begin{aligned} K_t &= P_{t+1}^- H^T \left(HP_{t+1}^- H^T + R\right)^{-1} \\ \hat{\boldsymbol{x}}_{t+1} &= \hat{\boldsymbol{x}}_{t+1}^- + K_t\left(y_t - H\hat{\boldsymbol{x}}_{t+1}^-\right) \\ P_t &= \left(I - K_tH\right)P_t^- \end{aligned} \tag{5.4}$$

R, Q and P_0 are all diagonal matrices whose values are treated as hyperparameters, e.g. $Q = diag(0.01)$ means that all Q values on the diagonal are assigned to 0.01 [54]. We want to use this approach to model the evolution over time of the MF's latent features and consequently show the students' progress over time.

5.2 Kalman State Estimation for Matrix Factorization

In this Section we present our novel method for Progress Modeling: the Kalman State Estimation for Matrix Factorization. In order to integrate the Kalman Filter and MF we first need to identify the state and the control of the system.
As described in Chapter 4, the i–th student s_i is modeled by MF as a vector $\boldsymbol{\varphi}^t \in \mathbb{S} := \mathbb{R}^K$, where K is the number of skills involved and $\mathbb{S}$ is the student's space and the j–th task c_j is defined as a vector $\boldsymbol{\psi}_j \in \mathbb{C} := \mathbb{R}^K$ representing the K skills required to solve a task defined in the tasks' space $\mathbb{C}$.
In this context we want to find a suitable prediction function able to compute the predicted performance $\hat{y}(\varphi_i, \psi_j)$ of a student s_i on a task c_j considering all his past interactions. Moreover, at each time step $\boldsymbol{\varphi}^t$ of student s_i needs to be updated to $\boldsymbol{\varphi}^{t+1}$ with a function $\tau : \mathbb{S} \times \mathbb{C} \to \mathbb{S}$.
Under this interpretation, $\boldsymbol{\varphi}_i^t$ should be the evolving state. The control over the system are the tasks' latent features $\boldsymbol{\psi}_j$ presented to the student, whereas the score y_t represents the measurement and its prediction $\hat{y}_t$ at time t (Eq. (5.5)). Since this algorithm is modeling the state and the interaction with the environment explicitly, a working Kalman Filter does not only show that the approach is valid for performance prediction, but also that (1) the students' latent features can be interpreted as the students' state and that (2) the tasks' latent features can be interpreted as the tasks' characteristics.

$$\begin{aligned} \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_k \end{bmatrix}_{t+1} &= A \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_k \end{bmatrix}_t + B \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_k \end{bmatrix}_t + \mathbf{w}_t \\ \hat{y}_{t+1} &= H \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_k \end{bmatrix}_t + \mathbf{v}_t \end{aligned} \tag{5.5}$$

In order to integrate the prediction function of MF (Eq. (4.2)) we formalized the relationship between state $\boldsymbol{\varphi}_i^t$ and predicted measurement $\hat{y}_t$ as in (5.6), having

then $H = \boldsymbol{\psi}^T$.

$$\hat{y}_t = \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_k \end{bmatrix}^T_{t-1} \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_k \end{bmatrix}_{t-1} + \mathbf{v}_{t-1} \tag{5.6}$$

Still missing is Eq. (5.1), i.e. the function τ mapping the state $\boldsymbol{\varphi_i}^t$ with the state at time $t+1$. We present in the following subsections two working ways to do this, but several other approaches could exist.

5.2.1 Simple previous/next State Mapping

For Eq. (5.1), we need to formulate an update function τ, which represents the learning from one task interaction. In order to do so we applied the simple reasoning that the student needs to learn proportionally from the task according to its learning rate ν and the tasks' required skills $\boldsymbol{\psi}$. Therefore we suggested following equations

$$\varphi_{ik}^{(t)} = \varphi_{ik}^{(t-1)} + \nu\psi_{jk},$$

i.e.

$$A = diag(1)$$

and

$$B = diag(\nu)$$

. We treated ν as a hyperparameter of the model as v, and w. In the future, however, ν could be learned in the initialization procedure. In our case the initialization of KSEMF consists in the computation of a first estimation of the Ψ and $\Phi^{(t=0)}$. In our experiments this was done with the training obtained by MF standard algorithm as we will explain in Sec. 7.3. Finally, in order to have a personalized model, each student has his/her own KSEMF instance updating according to his/her latent features values and performances.
How an update cycle of KSEMF algorithm work can be found in Alg. 5.

5.2.2 Skill Deficiency Aware KSEMF (KSEMF_SD)

In this Section we present an amelioration of KSEMF, by making it aware of the student's skills deficiencies. In order to do so we design the update function τ starting from the simulated student described in Sec. 6.1.2 that was able to simulate a learning process with continuous knowledge and score representation and tasks with multiple difficulty levels. Nevertheless, we do not exclude the possibility to use also other equations to model the relationship between $\boldsymbol{\varphi^t}$ and $\boldsymbol{\varphi^{t+1}}$. The

Algorithm 5 KSEMF update

Input: D_{Train}, D_{Test}, Q, R, P_0, ν
Use D_{Train} and Eq. (4.1) to obtain $\Phi^{(t=0)}$ and Ψ;
for *each* s_i c_j *interactions in* D_{test} N **do**
    $A = diag(1)$, $H = \boldsymbol{\psi_j}^T$;
    $B = diag(\nu)$
    $\hat{y}$=Predict, Eq. (5.3);
    Correct, Eq. (5.4);
end

simulator models a learning process defined by the Zone of Proximal Development (ZPD) [57], i.e. a student can learn from a task only if it is of the correct difficulty level. This is defined in the simulated environment as the difference $\boldsymbol{\alpha^{i,j}}$ between the skills of the student $\boldsymbol{\varphi_i^t}$ and those required to solve the task $\boldsymbol{\psi_j}$. As a consequence $\boldsymbol{\alpha^{i,j}}$ represents the skill deficiency of the student. In Eq. (5.7) $\tilde{y}$ represents the simulated score of the student and in [45] they were allowed to be only positive. Therefore $\boldsymbol{\varphi_{ik}} > \boldsymbol{\varphi_{gk}}$ means student i is more knowledgeable than student g in skill k and $\boldsymbol{\psi_{jk}} > \boldsymbol{\psi_{fk}}$ means that task j is more difficult than task f considering skill k. Finally $\psi_{jk} < \varphi_{ik}$ means of task j is too easy for student i and (s)he cannot learn skill k from it (See 6.1.2).
To develop the Skill Deficiency aware Kalman State Estimation for Matrix Factorization (KSEMF_SD) we interpreted the simulator modeled skills ψ_{jk} and φ_{ik}, for all i, j, and k as the from MF computed latent features. We then reformulated the equations modeling the process, Eq. (5.7), to fit Eq. (5.1) and work also with negative latent features. Therefore, we slightly modified Eq. (5.7) by considering the absolute value of ψ_{jk} and φ_{ik}. These changes allowed also negative latent features, but kept the ZPD properties of the simulator, i.e. a student cannot learn from too easy tasks and learns from a task proportionally to his knowledge and the skills required to solve the task. Also in this case each student has his/her own KSEMF_SD instance.

$$\begin{aligned} \tilde{y}(\varphi_i, \psi_j) &= \max(1 - \frac{||\boldsymbol{\alpha^{i,j}}||}{||\boldsymbol{\varphi_i}||}, 0) \\ \tau(\varphi_i, \psi_j)_k &= \tilde{y}(\varphi_{ik}, \psi_{jk})\alpha_k^{i,j} \\ \alpha_k^{i,j} &= \max(|\psi_{jk}| - |\varphi_{ik}|, 0) \end{aligned} \tag{5.7}$$

Under this interpretation $\varphi_{i,k} = 0$ means student i does not possess skill k and $|\varphi_{i,k}| > 0$ now means having some ability in skill k. Given Eq. (5.7) we obtained

$$\varphi_{ik}^{(t)} = \varphi_{ik}^{(t-1)} + (\tilde{y}(\varphi_{ik}, \psi_{jk})\gamma\delta(|\varphi_{ik}| < |\psi_{jk}|))\psi_{jk},$$

i.e.

$$A = diag(1)$$

and

$$B = diag\left((\tilde{y}(\varphi_{ik}, \psi_{jk})\tau_k^{i,j}\gamma\right), \tag{5.8}$$

where γ is a weight selected as an hyperparameter. Therefore, the pseudo-code for KSEMF_SD is described in Alg. 6.

Algorithm 6 Experiments' Framework

Input: D_{Train}, D_{Test}, Q, R, P_0
Use D_{Train} and Eq. (4.1) to obtain $\Phi^{(t=0)}$ and Ψ;
for *each* s_i c_j *interactions in* D_{test} N **do**
    $A = diag(1)$, $H = \boldsymbol{\psi_j}^T$;
    Compute B using Eq. (5.8);
    $\hat{y}$=Predict, Eq. (5.3);
    Correct, Eq. (5.4);
end

5.3 Experiments

In this Section we analyze different aspects of the algorithms KSEMF and KSEM_SD performances. First, we describe the dataset used for the experiments; then, we analyze the hyperparameters' selection, the performances, the model initialization, and the sensitivity to the cold start problem. Afterwards, we discuss the ability of the algorithm to model the student progress. This is done from different perspectives, which involve the personalization of the state modeling and the update rate.
First we present the comparison with the baseline algorithms presented in Tab. 5.1.

5.3.1 Dataset characteristics

Of the large dataset of the commercial ITS, described in Sec. 2.4.1, we selected a subset described in Tab. 5.2. According to the results presented in Sec. 6.3.2 having students with less than 10 interactions would damage considerably the performances, without the possibility to distinguish between the error due to the cold start problem and the error due to the inability of the model. Moreover, our purpose to derive an incremental and highly temporal dependent model requires

	Paper Reference	Section Reference	Acronym
Matrix Factorization	[53]	Sec. 4.1	MF
Incremental Matrix Factorization	[1, 55]	Sec. 4.2.2	Abernathy
Matrix Factorization Update	[40]	Sec. 4.2.1	UpMF

Table 5.1: Baselines' names, reference and description.

Table 5.2: Subset statistics of the commercial ITS dataset described in Sec. 2.4.1

	D_{Train}	D_{Test}
Number of Tasks	2035	2035
Number of Students	24288	713
Total Student-Task Interactions, N	751109	102038

students with long histories to verify that the prediction performances do not deteriorate over time.
We selected only the students i of one country with at least $N_i > 10$ interactions, where one interaction correspond to a student solving a task. The available students are then further divided in two groups. The group of those with $10 < N_i < 100$ is used to initialize the latent features of all algorithms (D_{Train}, Tab. 5.2), whereas the others with $N_i > 100$ are used to test the online updates of Abernathy, UpMF, KSEMF, and KSEMF_SD (D_{Test} Tab. 5.2). Since D_{Train} and D_{Test} have no overlapping students, the D_{Test} students' cold start problem is solved by including in D_{Train} data of their first interactions with the ITS, so that their latent features can be learned in a full training. The samples necessary to avoid the cold start problem, both for students and tasks, are generally 10. This amount was empirically defined by [39].
This subset selection should not influence the generalization ability of the model since it is safe to assume that all the students come from the same distribution. For instance, the fact of having or not having a long history in the system is not related to the general ability of the student. Generally, in experiments with small systems at prototype level, it is possible to distinguish good and not so good students by having a look at the number of tasks they completed in a specific amount of time [56]. On the contrary here, the commercial ITS possesses a sequence adaptivity, that reduces this problem. Moreover, since the ITS is used used mainly at school over the years it is unlikely that only good students completed ten tasks.

5.3.2 Hyperparameters' Selection

All the model hyperparameters of MF, Abernathy, UpMF, KSEMF, and KSEMF_SD were selected with a full Grid Search, i.e. the influence on the model error of different combinations of hyperparameters is analyzed in a brute force manner. First MF ones were evaluated considering the RSME obtained with a further split of D_{Train}. 66% of D_{Train} was used to train the MF model and its 34% was used to test the model with the different hyperparameters. Abernathy, UpMF, KSEMF, and KSEMF_SD best hyperparameters are then selected in the ranges presented in Tab. 5.3 according to the performances in D_{Test}. In particular we used the value $Total\_RMSE$ computed as in Alg. 1 to evaluate the performances of the algorithm.
UpMF, Abernathy, and MF hyperparameters are λ, β, $iter_{Max}$ and K. In addition to these, KSEMF, and KSEMF_SD possesses four more hyperparameters: Q, R, γ or ν, and P_0. The empirical approach is to model Q, R, and P_0 as diagonal matrices and test their diagonal values with a logarithmic scale. The selected hyperparameters are reported in Tab. 5.3.
In the future more efficient approaches to hyperparameters' selection could be used as the ones suggested by [60] and [49].

5.3.3 State Variables' Initialization

The next question to answer was how to initialize the latent features of the following online updates: UpMF, Abernathy, KSEMF, and KSEMF_SD. Since all algorithms are fully personalized they suffer from the so called cold-start problem, which occurs when no information is available about the students or the tasks. Therefore, a random initialization of the latent features would lead to very bad performances [56]. Usual approach to solve the problem is to train a model with the classic MF algorithm and use the computed tasks' latent features to initialize the online updates. These are then kept constant while the student's latent features are updated with the different algorithms, e.g. as in Alg. 5. So, the by MF computed students' and tasks' latent features are used to initialize respectively $\Phi^{(t=0)}$ and Ψ of all online updating algorithms. Initialization of D_{Test} students' is possible because data of their first interactions with the ITS is included in D_{Train}, so that their latent features was learned in a full training.

5.3.4 RMSE Evaluation

In this Section we evaluate the overall algorithm performances by computing the $Total\_RMSE$, as in Alg. 1. For MF, UpMF, Abernathy, KSEMF, and KSEMF_SD

Table 5.3: Hyperarameters' ranges tested and selected values for the different algorithms.

Parameters	**Range**	**Step**	**UpMF**	**Abernathy**	**KSEMF**	**KSEMF_SD**
Learning Rate β	0.01-0.1	0.01	0.01	0.01	0.01	0.01
Latent Features K	2-120	20	102	120	62	62
Regularization λ	0.01-0.1 0.001-0.01	0.01 0.001	0.01	0.009	0.01	0.01
Number of Iterations $Iter_{Max}$	10-200	10	100	100	25	25
State Noise Cov. Q	0.00001-1	logarithmic	-	-	0.001	0.00001
Error Noise Cov. P_0	0.00001-1	logarithmic	-	-	1	1
Measurement Noise Cov. R	0.00001-1	logarithmic	-	-	0.001	0.001
Weight γ	0.00001-1	logarithmic	-	-	-	0.001
Learning Rate ν	0.00001-1	logarithmic	-	-	0.001	-

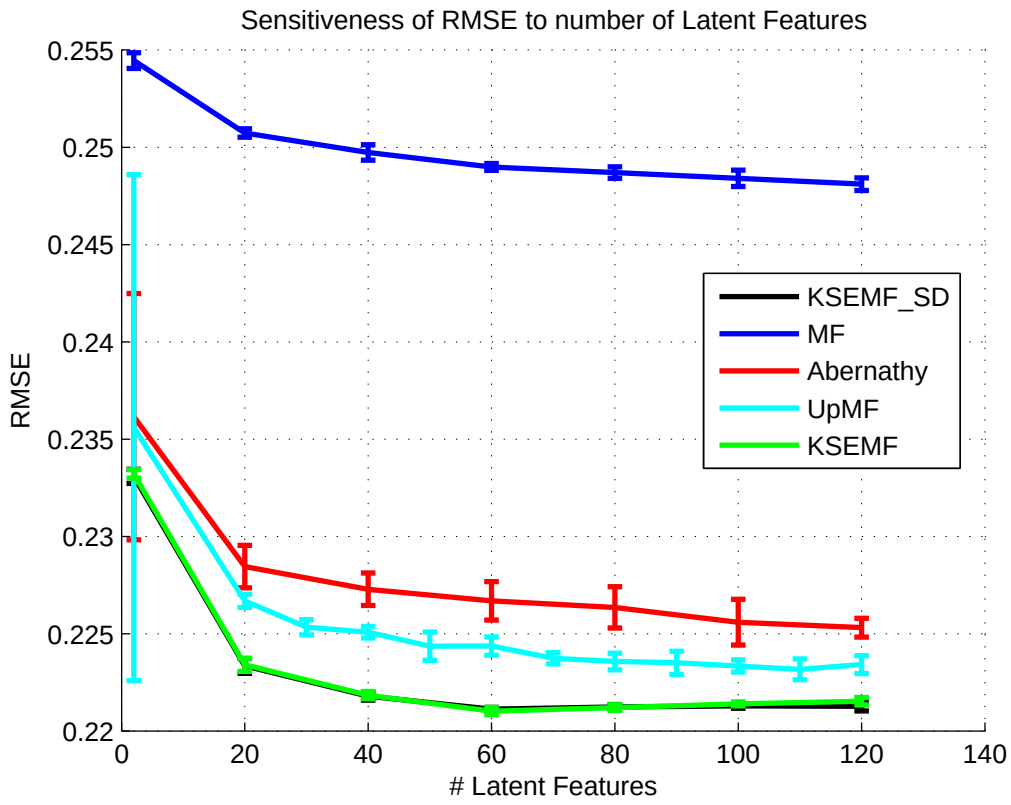

Figure 5.2: RMSE sensitiveness analysis to latent features.

we analyze the sensitiveness to the number of latent features. Moreover, we repeated the experiment five times to be able to exclude the variance influence due to the random initialization of the MF. As shown in Fig. 5.2 KSEMF and KSEMF_SD are able to outperform our reference baselines in all tried latent features configurations. As expected, any kind of update delivers a lift in the performances of the algorithms.

Abernathy shows worse performances than UpMF, this can be easily explained by considering that Abernathy is a single-epoch SGD whereas UpMF has a multi-epoch SGD based update. Therefore, UpMF not only could learn from the new data point, but also had more information available as it uses the entire student's history.

In comparison to KSEMF and KSEMF_SD, Abernathy has worse performances despite it uses the same amount of data information: KSEMF and KSEMF_SD, and Abernathy perform their update considering the last state estimate and the last observation. Nevertheless, KSEMF and derived algorithms model the uncer-

tainty on the state, measurements, and observations and therefore they can use these information to decide how to update the state. How much the filter should update its belief is coded in the Kalman Gain, $K_t = P_{t+1}^{-} H^T \left(H P_{t+1}^{-} H^T + R\right)^{-1}$, which controls the relation between the predicted state estimate and the measurement. The smaller R, the surer we are about our measurement. By considering the "correct" phase equation, $\hat{\boldsymbol{x}}_{t+1} = \hat{\boldsymbol{x}}_{t+1}^{-} + K_t \left(y_t - H\hat{\boldsymbol{x}}_{t+1}^{-}\right)$, jointly with that of the Kalman Gain, we see how the state update will rely mostly on the measurements if R is small. When the state is known accurately then P_{t+1}^{-} and consequently $HP_{t+1}^{-}H^T$ are smaller compared to R, causing that the filter mostly ignores the measurements and relies mostly on the previous state estimate. The same role of the Kalman Gain is played in Abernathy and UpMF by the regression hyperparameter, which is however constant over time. It is well known by experts in the area of MF algorithms that both the regression hyperparameter and the number of latent features should be adapted to the amount of data available, i.e. the history length. This means that the best hyperparameters combination changes over time and implies a trade off between optimizing for the first interactions or the last ones. This problem can be seen clearly in Figs. 5.3, 5.4, and 5.5, where we evaluated the RMSE evolution over time in overlapping sliding windows, SlidingW_RMSE. In the first interactions the algorithm has still good performances, but afterwards it deteriorates significantly. The problem is reported also in Chapter 7, and, to avoid this issue, we retrained the model each night. This is however a quite demanding computational requirement and, as we will see in Sec. 5.3.6, can affect the Progress Modeling approach we want to use. Moreover, UpMF requires the entire history of one student as input parameter for Alg. 3 that in case of database (DB) implementation will not only slow down the performances but also increase the complexity of the system. The other algorithms do not require demanding DB accesses to extract the entire student's history since it uses only information of the current time step to predict the next one. More difficult is to evaluate if KSEMF or KSEMF_SD performs better. In Fig. 5.6 one can see that the algorithms have almost comparable results. The slightly better performance of KSEMF_SD for $K = 122$ points out a less sensitivity to the increase of latent features' by KSEMF_SD. Nevertheless, this difference is not statistically significant. As shown by the confidence intervals plotted in the Figure. We can assume more safely that KSEMF_SD performs better than simple KSEMF from the SlidingW_RMSE results reported in Figs. 5.3, 5.4, and 5.5, were for specific interactions the KSEMF_SD error curve is clearly under the one of KSEMF.

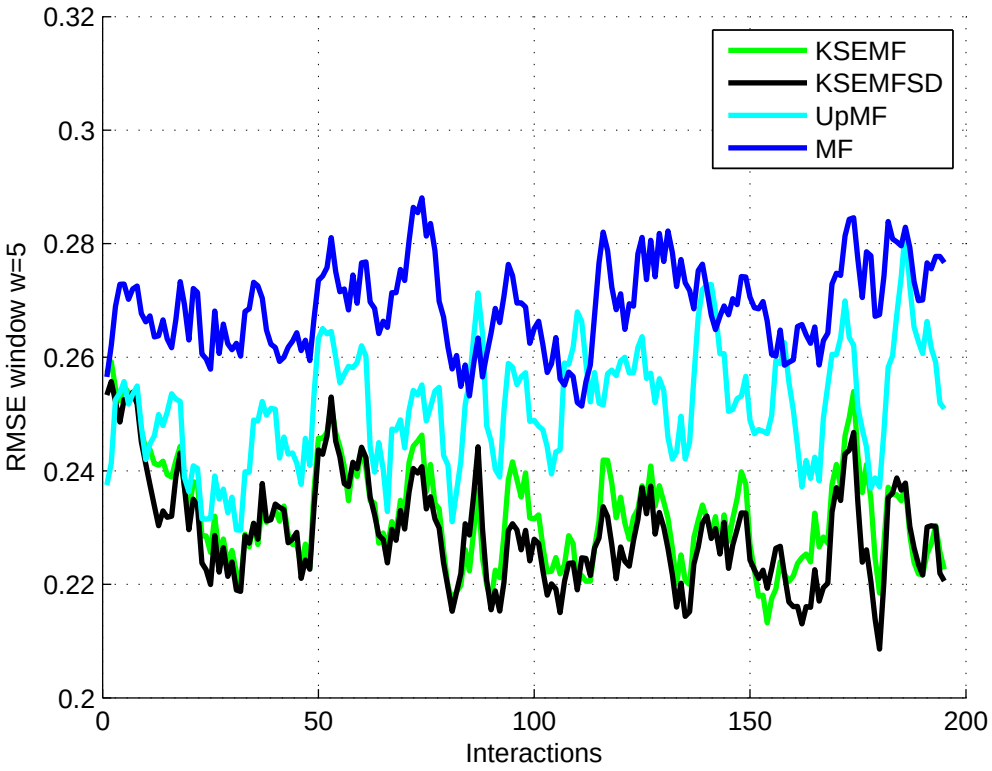

Figure 5.3: SlidingW_RMSE with window size $w = 5$

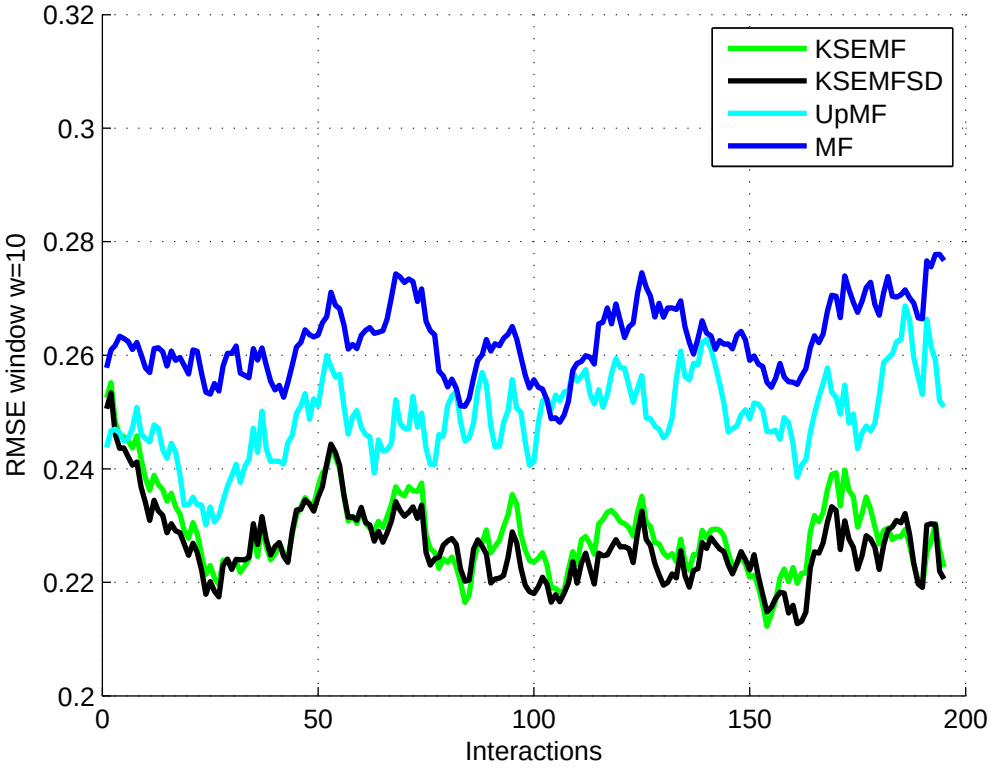

Figure 5.4: SlidingW_RMSE with window size $w = 10$

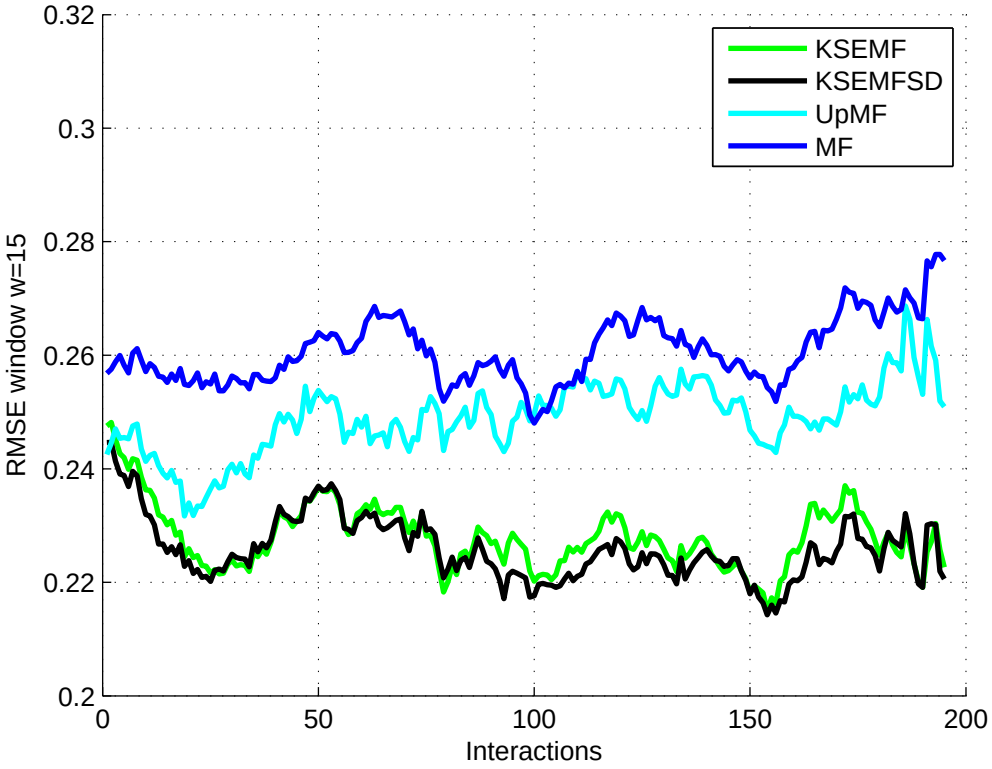

Figure 5.5: SlidingW_RMSE with window size $w = 15$

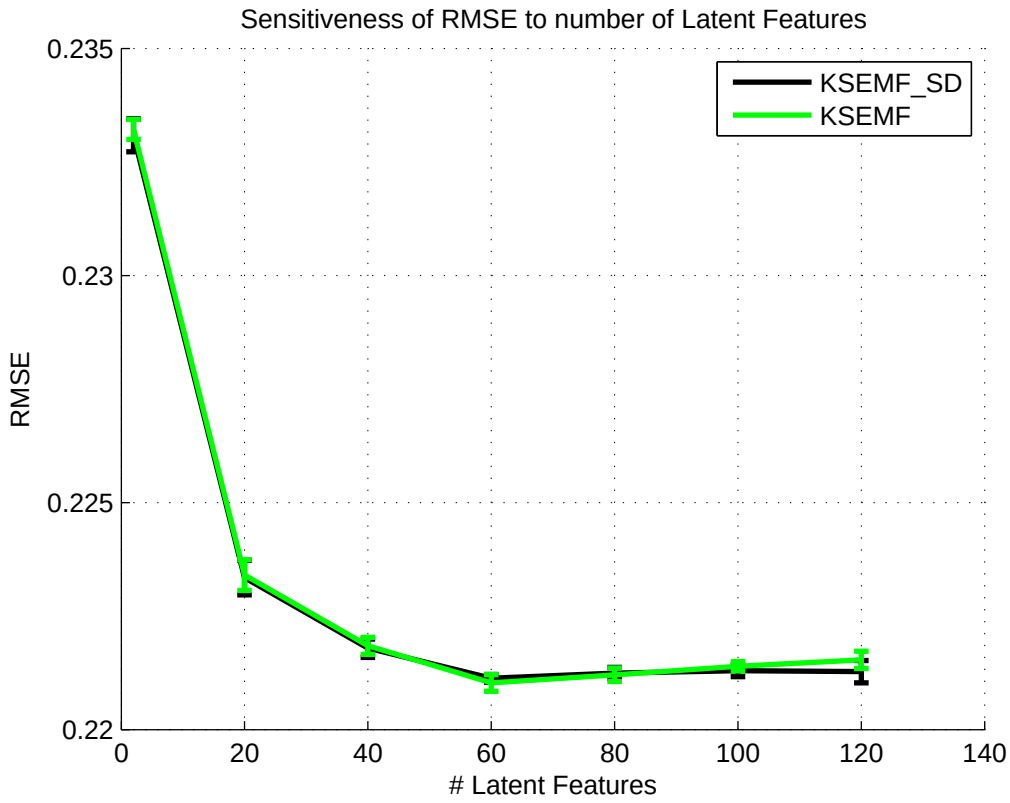

Figure 5.6: Comparison between KSEMF and KSEMF_SD

5.3.5 Evaluation of the Cold Start Problem

In this Section we discuss the impact of the cold start problem to the score prediction for new students. As already discussed in Sec. 6.3.2, often researchers cannot obtain enough time with children that have already experience with the system and therefore MF utility is dramatically reduced. So, since the 10 interactions required to avoid the student's cold start problem ([39]) are not always available, we show also results when just one interaction is included in D_{Train}.
In Fig. 5.7 we can see how the Total_RMSE, computed as in Alg. 1, evolves over time. Models marked with the "Cold" label are initialized with only 1 sample whereas the others are initialized with 10. MF Cold behaved like a random predictor with an error around 0.5 and is not shown in Fig. 5.7. As it is possible to see, the 10 samples substantially improved the error. Nevertheless, we believe this is still not an optimal initialization for KSEMF_SD, since for the first interactions KSEMF_SD is outperformed by UpMF and MF with 10 samples initialization. KSEMF_SD, initialized with 10 samples, has a similar behavior as MF because it inherits the error of MF tasks' latent features whereas KSEMF_SD error amelioration is due to the better students' latent features modeling.
If these 10 interactions are not available, KSEMF_SD Cold converges faster to smaller errors than UpMF Cold. In [56] it was discussed how the cold start problem limits the usage of MF in small ITS or for short experiments with new students. Therefore, a faster converging error is an appealing property, that could further reduce the requirements of MF.

5.3.6 Modeling Student Progress

In order to use the developed algorithm to model student progress, it is important to be able to use the performance predictor as model for the user state and take decisions accordingly. One of the claimed disadvantages of MF approaches in comparison to BKT and PFA is that the amount of knowledge of the student cannot be extracted directly from the latent features computed by the algorithms. For this reason in Chapter 6 we proposed a sequencer which uses only the information coming from the predicted score. In Fig. 5.9 it is shown (a) how the latent features evolve according to KSEMF_SD algorithm in a scenario with 62 latent features and (b) how the latent features evolve according to UpMF algorithm in a scenario with 102 latent features. Fig. 5.9 (f) shows the actual score of the student (blue) and the predicted performance of the student by KSEMF_SD (green) and MF (red). This figure can be found larger in Fig. 5.8. There, one can see how the actual score curve cannot be used to interpret the student progress. It must be considered that the 200 interactions involve a period of one year, where the student

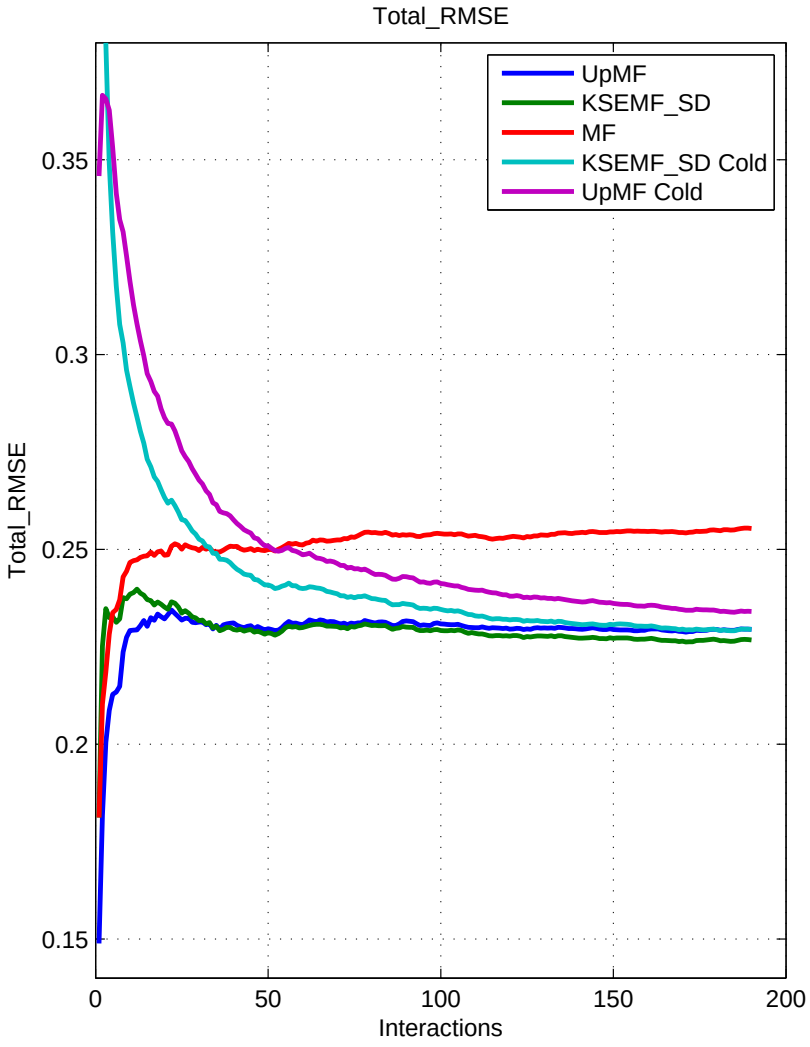

Figure 5.7: D_{Test} **Total_RMSE behavior over time**: Models marked with "Cold" label are initialized with only 1 interaction in D_{Train} whereas the others with 10.

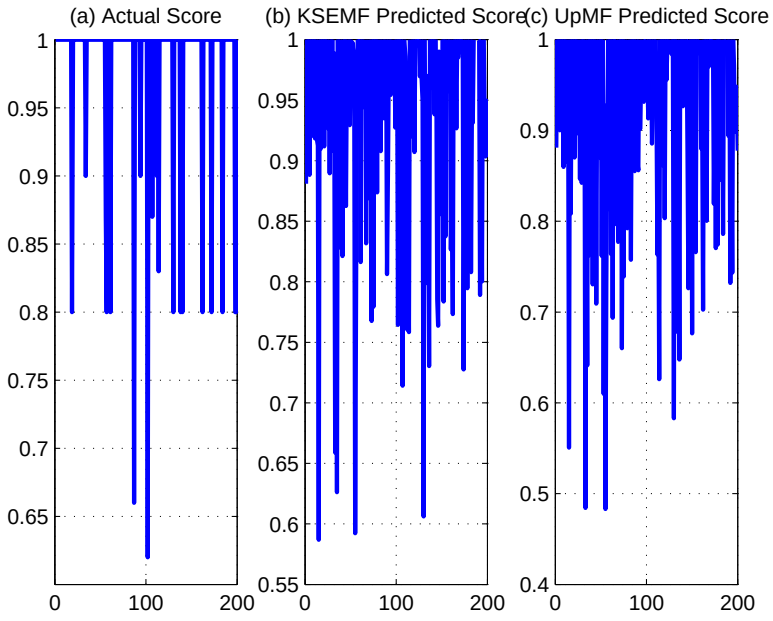

Figure 5.8: Actual score (left), KSEMF_SD predicted score (center), UpMF predicted score (left).

could have increased his knowledge in each of the 22 topics of several difficulty levels. Since both UpMF and KSEMF_SD mimic the actual score behavior, the same interpretability problem occurs. However, the predicting ability of KSEMF_SD let us suppose that the latent features have indeed a state meaning for the algorithm and consequently an evolution according to the student's performance should be monitored. Therefore, to monitor a meaningful trend, we aggregated the features computing the absolute value normalized for the number of latent features as in Eq. (5.9) and depicted the results in Fig. 5.9 (c) for KSEMF_SD and (d) for UpMF.

$$kn = \frac{1}{K} \sum_{k=0}^{K} |\varphi_k| \tag{5.9}$$

Under the interpretation that $\varphi_{i,k} = 0$ means student i does not possess skill k, whereas $|\varphi_{i,k}| > 0$ means having some ability in skill k, variable kn could be understood as the personalized knowledge evolution or the learning curve of the user. Although UpMF latent features are learned from scratch after each interaction one can notice in the figures an evolution trend, which is as plausible as the one of KSEMF_SD. This also confirms that the latent features in MF approaches represent the state of the user and their value could be used to retrieve the students

knowledge amount. We believe this works because the tasks' latent feature are kept constant. Therefore, in order to keep track of the current state of the students one cannot do a full retrain of the UpMF model, as done by [43], since this would reset the values of the tasks' latent features, that allow reconstructing at each interaction the state of the student by means of the student's history.

5.3.7 Personalization

One important aspect of Progress Modeling is personalization. MF creates an individualized model as well for tasks as for students. In order to do so also for KSEMF_SD, each student has his/her own KSEMF_SD equations updating according to his/her modeled state and performances. Since the simulator equations are based on the state variable, in this context, also the B matrix is personalized and change at each interaction. Therefore, the update equations of KSEMF_SD model personalization in two different ways. The B matrix represents the influence of the student on the update, i.e. what is his/her learning rate and its skills' deficiency. The control u, i.e. the tasks' latent features $\boldsymbol{\psi}$, represents the influence of the task on the knowledge acquisition of the student. Hereafter, we will see how the state as well as the update evolve over time in a personalized way.

5.3.7.1 Personalized state evolution

In Fig. 5.10 (a) and (d) we can see the personalized latent features' trends of KSEMF_SD. In Fig. 5.10 (c) and (f) and in Fig. 5.9 (e) we can see the Total_RMSEs of the models for three specific students. These are overall coherent with the results presented in Fig. 5.7. This information could be used in several ways, e.g. by later establishing the mapping between the computed kn trend and the actual knowledge acquisition of the users, we could design novel policies for sequencing tasks, feedback and hints. In addition, the relationship between kn and the model error should be further analyzed. This will allow also to monitor the performances of the performance predictor over time.

5.3.7.2 Personalized update evolution

In this Section we discuss the plausibility of the personalized update trend derived through Eqs. (5.7). For simplicity we considered $\tilde{y}$, which represents the update of the state, since it is later multiplied with constant γ to obtain B (See Eq. 5.1). In Fig. 5.11 (f) we show, for a student, how $\tilde{y}$ evolves over time. An almost constant update is plausible, since it mimics the learning rate of the student, which is related to his/her learning ability. However, its adaptive computation through

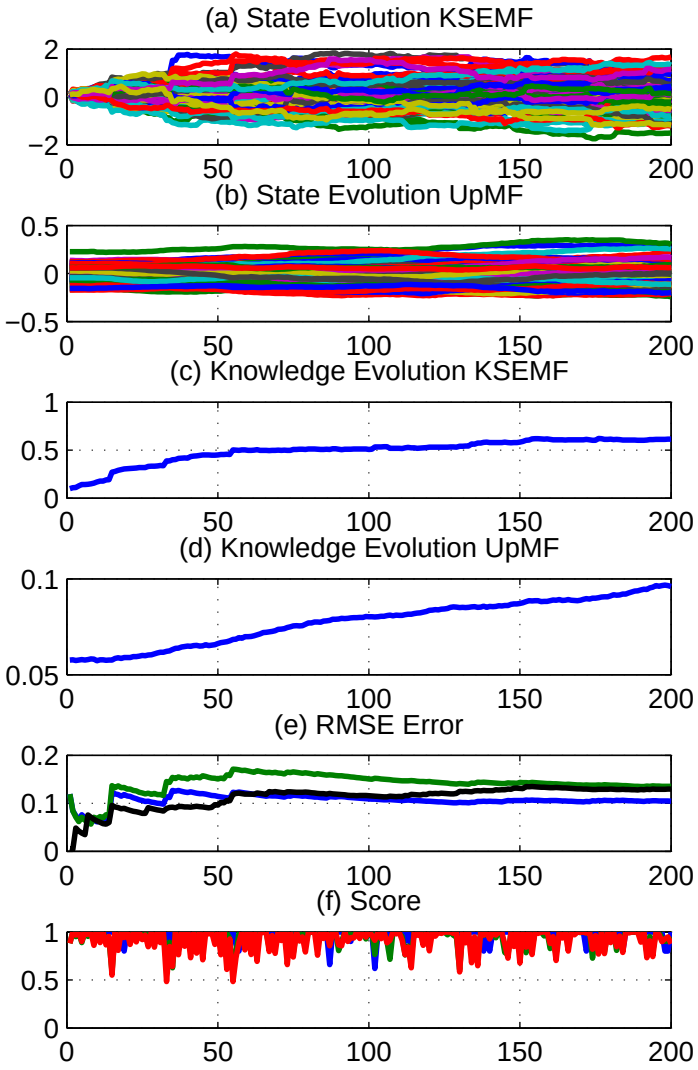

Figure 5.9: **x-Axis:** Number of tasks seen by the student or interactions.
y-Axis:
(a) state evolution according to KSEMF_SD with K=62.
(b) state evolution according to UpMF with K=102.
(c) and (d): knowledge evolution for KSEMF_SD and UpMF computed as in Eq. (5.9).
(e) Total_RMSE of KSEMF_SD (blue), MF (green) and UpMF (black).
(f) Actual performance of the student (blue), predicted performance by KSEMF_SD (green), MF (red).

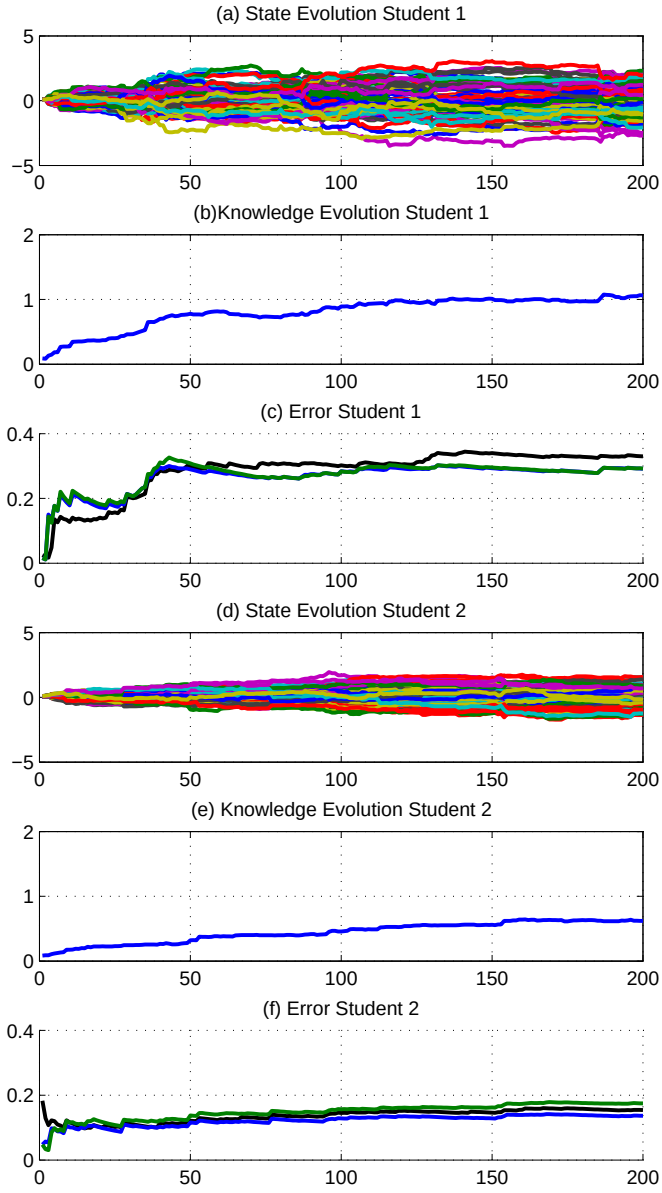

Figure 5.10: **x-Axis:** Number of tasks seen by the student or interactions.
y-Axis:
(a) and (d): KSEMF_SD state evolution of two different students, K=62.
(b) and (e): kn of KSEMF_SD latent features computed as in Eq. 5.9.
(c) and (f): Total_RMSE of KSEMF_SD (blue), MF (green) and UpMF (black) of two different students.

the state is of advantage, since it allows the model to faster adjust to the students' states changes. In Fig. 5.12 we see that the average update for all students evolves over time converging only in the last interactions to a constant value. This is explicable with the previously seen behavior of KSEMF_SD in the first interactions (see Fig. 5.7) and should be seen as another indicator that the initialization of the algorithm is not optimal.
Although we were not interested in keeping the simulation properties of the simulator from which we derived our equations, we briefly discuss why $\hat{y}$ is smaller than the actual performance. Since the variables of $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ are not clipped between 0 and 1 as in [45] $\left\|\alpha_k^{i,j}\right\|$ is consequently bigger on average and $\tilde{y}$ smaller than the actual performance. In conclusion, given the ameliorated results of KSEMF_SD over UpMF, reported in both Fig. 5.7 and Fig. 5.2, we overall showed that the designed equations for KSEMF_SD are suitable to update the students' latent features.

5.4 Conclusions

In this Chapter we presented KSEMF and KSEMF_SD, two novel methods for student Progress Modeling based on online updating MF performance prediction and Kalman Filters. Progress Modeling is proposed, as amelioration of domain independent performance prediction, it allows showing the evolution of the students over time in a plausible way. We showed that it is possible to give a specific interpretation to latent features which represents the state of the student and the characteristics of a task. In future work, the relationship between the computed kn could be mapped with the real knowledge evolution with the final goal to deliver an effortless analysis tool to teachers and developers. In order to do so, an idea could be to apply the same approach to the contents' latent features associating the normed sum of the latent features with the estimated difficulty level of a task. With a laboratory experiment it would be easier to map the from the algorithms retrieved curve to the available tasks' domain information, rather than trying to map predicted and real students' knowledge.
KSEMF and KSEMF_SD also showed appealing properties in comparison to other potential domain independent progress modeler. First, the algorithm requires less resources as the entire student's history is not necessary to compute the updated latent features. Then, the algorithm is still domain independent because the tagged skills of the tasks are not used to deliver a score prediction. Finally, KSEMF_SD reduced the prediction error and is less sensitive to the lack of data. In future work we believe to further be able to reduce the error by developing a better initialization of the students' latent features. In conclusion, in this Chapter we showed that Recommender Systems and Kalman Filters can be successfully

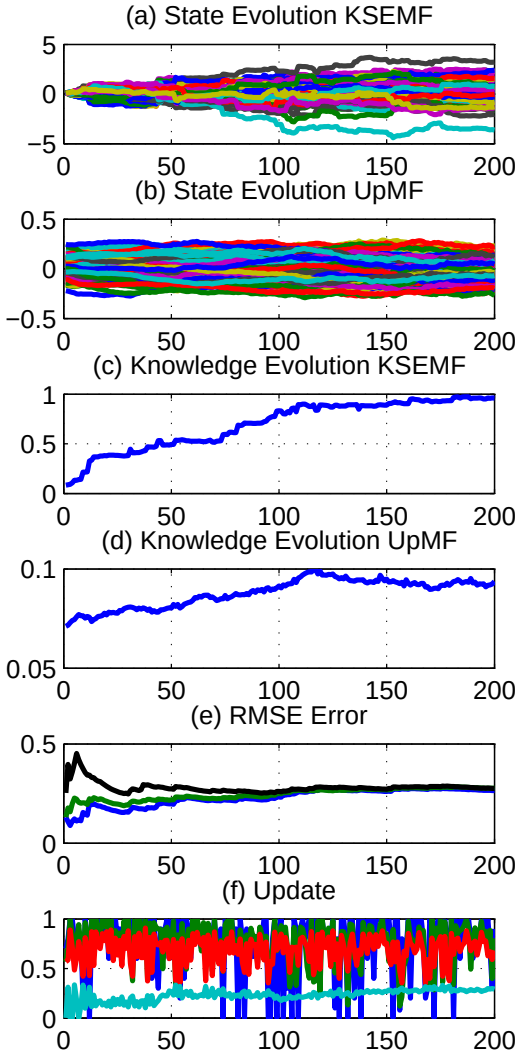

Figure 5.11: **x-Axis:** Number of tasks seen by the student or interactions.
y-Axis: *(a)* how the state evolves according to KSEMF_SD with K=62.
(b) shows how the state evolves according to UpMF algorithm with K=102.
(c) and *(d)* show the knowledge evolution, computed as in Eq. (5.9).
(e) RMSE of KSEMF_SD (blue), RMSE of MF (green) and UpMF (black).
(f) Actual Performance of the student (blue), predicted performance of the student by the KSEMF_SD (green), predicted performance by MF (red) and $\tilde{y}$ (turquoise).

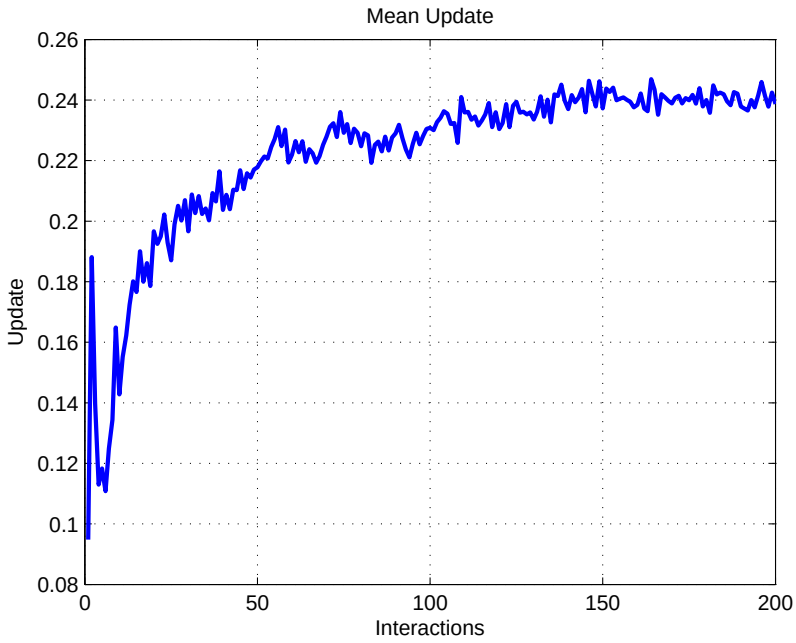

Figure 5.12: **Mean Update Over Time** Update behavior at each interaction on average for all students.

combined. We are then looking forward in testing such approach with other user modeling applications.

Chapter 6

The Vygotsky Policy Sequencer

Contents

6.1 Content Sequencing Structure 73

6.1.1 The Sequencer Structure 74

6.1.2 Simulated Learning Process 76

6.2 Experiment Session 78

6.2.1 Experiments on the Simulated Learning Process 80

6.2.2 Sensitivity Analysis on the Vygotsky Policy 80

6.2.3 VPS Evaluation 81

6.2.4 Advanced Experiments 82

6.3 VPS Feasibility and Utility 86

6.3.1 Sequencing VPS Feasibility 88

6.3.2 Sequencing VPS Utility 89

6.4 Conclusions 90

Sequencing contents, like tasks, hints, and feedbacks, is an open issue for Intelligent Tutoring Systems. The common approach is based on domain analysis by experts, who characterize each content with skills involved and a difficulty level. In addition, Machine Learning based sequencers require a specific dataset collection to create users' models and a sequencing policy, which needs to be tested online with strong ethical requirements and a high number of users.

In this Chapter we propose a novel method of sequencing based on Matrix Factorization Performance Prediction and Vygotsky's concept of Zone of Proximal Development. This approach represents a valid alternative to Reinforcement Learning and Bayesian Knowledge Tracing. Reinforcement Learning, can be reconnected to robotics, which has an availability of accurate simulators and tireless test subjects. The same cannot be said for ITS where, generally, apart from adults, also children of any age are involved.
To perform an preliminary offline evaluation of the developed sequencer, we designed a simulated learning environment with customizable scenarios. The sequencer's tests allow showing that a performance prediction method can be used to create offline fully personalized students' models that combined with a score-based sequencing policy propose contents without domain engineering/authoring effort. Its results in the designed simulated environment are promising in comparison to curriculum based policies.
Therefore, the main contributions of the developed sequencer are:

1. A content sequencer based on a Performance Prediction system that (1) can be set up and preliminary evaluated in a laboratory, (2) models multiple skills and individualization without engineering/authoring effort, (3) adapts to each combination of contents, levels and skills available.

2. Simulated environment with multiple skill contents and students' knowledge representation, where knowledge and performance are modeled in a continuous way.

3. Experiments on different scenarios with direct comparison with informed baseline.

After a successful offline evaluation, several other considerations before integration in an ITS need to be done. The final evaluation with an online experiment can be found in Chapter 7, as well as a designed method for lightweight integration.
These contributions are also published in: [42, 45].
This Chapter is structured as follows. We will first discuss the issues related to the evaluation of systems interacting with humans and the reason why the design of a simulated learning process should be a mandatory first step in every sequencer's evaluation. After having discussed the plausibility of the simulated learning process, we present our sequencer the so-called Vygotsky Policy Sequencer (VPS). We analyze the results under different perspectives, by exploiting the simulated learning process we consider different sequencing policies and analyses the effect they have on learning. We then discuss how flexibly the sequencers adapts to the different situations. We continue discussing next steps towards integration of

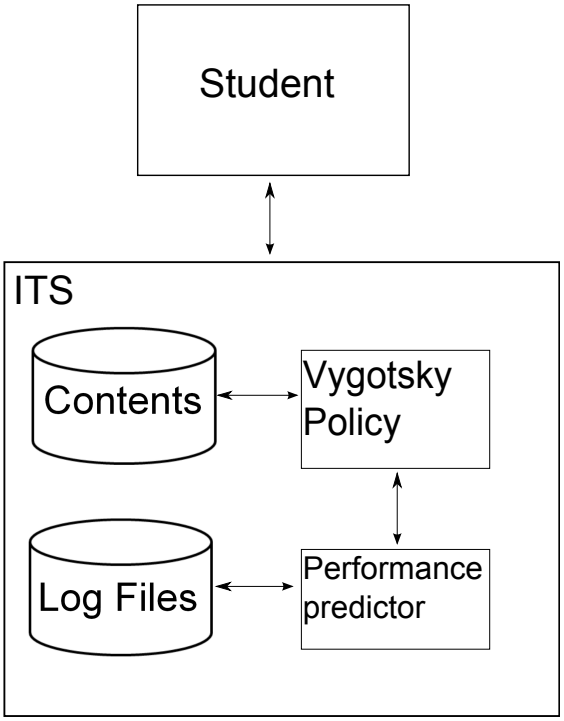

Figure 6.1: System structure in a block diagram.

the VPS into a real system. In particular, we point to utility and feasibility of the VPS. Finally, we introduce potential ameliorations that could lead to a more personalized sequencer.

6.1 Content Sequencing Structure

The designed system is an ITS (see Fig. 6.1) that consists of different modules: the available contents, the previous interactions of the students with the system (log files), the students' model computed by the Performance Predictor and the Sequencer Policy. The Performance Predictor needs the log files of students interacting with the contents that are used to predict their potential performance or score in the next contents. The policy is applied in an adaptive way thanks to the information on the predicted scores shared between Performance Predictor and Sequencer.
First step toward a working prototype requires testing in a laboratory. Since sequencing problems can only be evaluated online, i.e. the sequence optimality can be measured only after a student worked with it, we designed a simulated learning process that is described in Sec. 6.1.2. We excluded the possibility of collecting an

exploratory corpus because making students practice with very easy and very difficult exercises in random order could be frustrating, especially if they are children. After a first validation with real students, only a common dataset collection will be necessary to set up the system with new contents, giving also the possibility to calibrate the environment and later use it for new sequencing methods.
In this Chapter we use the word content to refer to the activities a student interacts with, although our main focus here is task sequencing. Taking advantage of the simulated learning process characteristics described, we can later interpret contents as different ITS elements. As explained in [44], a content could be a hint, a feedback, a topic or a task, whose sequencing could in each case take advantage of the designed system as we will discuss later.

6.1.1 The Sequencer Structure

Let $C \subseteq \mathbb{C}$ and $S \subseteq \mathbb{S}$ be respectively a set of contents and students, d_{c_j} be the difficulty of a content defined as $d_{c_j} = \sum_{k=0}^{K} \psi_{j,k}$. The content sequencing problem consists in finding the optimal policy π^*:

$$\pi^* : (\mathbb{C} \times \hat{Y}_i) \rightarrow \mathbb{C}. \tag{6.1}$$

that selects the next content given the available contents and the predicted score on the contents $\hat{Y}_i$., i.e. without knowing the difficulties of the contents and the required skills to solve them.
A common problem in designing a policy for ITS is retrieving the knowledge of the student from the given information, e.g. score, time needed, previous exercises, etc. The previous mentioned data types are just an indirect representation of the knowledge, which cannot be automatically measured, but needs to be modeled inside the system. Hence, integrating the curriculum and skills structure is the cause of high costs.
In this work we try to keep the contents in the Vygotskys Zone of Proximal Development (ZPD) [57], i.e. the area where the contents neither bore or overwhelm the learner. We mathematically formalized the concept with the following policy, that we called Vygotsky Policy (VP):

$$c^{t*} = \text{argmin}_c \left| y_{th} - \hat{y}^t (c) \right| \tag{6.2}$$

where y_{th} is the threshold score, i.e. the score that keeps the contents in the ZPD. The policy will select at each time step the content with the predicted score $\hat{y}^t$ at time t most similar to y_{th}.
Our approach is related but different from [25], where it is hypothesized that the

ZPD could be used to select the next task, by considering the predicted probability of answering a task correctly. The authors of [25] propose to select those tasks where the probability of answering correctly is at 50 %, since the new data recorded would also ameliorate the prediction ability of the model used for performance prediction and 50 % would theoretically be in the middle of a ZPD defined between 0 and 1. Nevertheless the feasibility of this idea was not evaluated. We disagree with this decision, as we want to focus on the sequencing performances rather than on the prediction performances. A human teacher would select tasks that students are more likely to solve and leave the ones that (s)he is not sure about for later. In Sec. 6.2.2 and 7.2 we will suggest ways to select the correct y_{th} value. Especially in Sec. 7.2 we discuss how this value must be carefully decided.
The peculiarity of the VP is the absence of the difficulty and "required skills" concept. Defining the difficulty for a content in a simulated environment as ours is easy, because we mathematically define the skills required. In the real case it is not trivial and quite subjective. Also the required skills are considered as given in the other state of the art methods like PFM and BKT, where a table represents the connection between contents and skills required. Without skills information not only BKT and PFM performance prediction cannot be used in our formalization, also sequencing methods [22] have no information to work with. Therefore, to compute $\hat{y}^t$ please refer to Sec. 4.1 for the description of Biased Matrix Factorization (BMF), which is the performance predictor used in this work. At the same time please also consider that specific Performance Predictor and policy were chosen, however, nothing is against using other ones in the future following the same approach.
Matrix Factorization was selected for sequencing as it has several advantages in comparison to BKT state of the art presented in Sec 3.1.1:

1. Domain independence. Ability to model each skill, i.e. no engineering/authoring effort in individuating the skills involved in the contents.

2. Having better performances in Root Mean Square Error on Benchmark datasets.

3. Possibility to build the system without an exploratory corpus, which is considered as unethical by pedagogues.

Despite the fact that MF is generally used for large datasets, the algorithms also outperforms other state of the art algorithms for small datasets, as reported by [38].

6.1.2 Simulated Learning Process

Given the necessity of preliminary evaluation in a laboratory, it is of crucial importance to have a simulated environment able to model reality with a certain degree of fidelity. For our system we required a score and skill representation between 0 and 1, to be able to test following aspects:

1. Possibility to use score as single success indicator for sequencing
2. Ability to model a multiple skill domain and students' knowledge by the performance predictor
3. Possibility to change number of skills involved to test flexibility
4. Possibility to test also noisy processes

We designed a simulated student based on the following assumptions:
(1) A content is either of the correct difficulty for a student, or too easy, or too difficult. (2) A student cannot learn from too easy contents and learns from difficult ones proportionally to his knowledge level. (3) It is impossible to learn from a content more than the required skills to solve it. (4) The total knowledge at the beginning is different than zero. (5) The general ability on connected skills helps solving and learning from a content. The last assumption is more plausible because we assume to sequence activities of the same domain. For instance, in order to solve a fraction addition, a student needs more related skills: multiplication, fraction expansion etc. It is unlikely for a student to do a fraction expansion without knowing how multiplication works. At the same time the knowledge of multiplication will help him solving the steps on fraction expansion.
A student simulator is a tuple (S, C, y, τ) where, given a set $S \subseteq [0,1]^K$ of students, s_i is a specific student described as a vector $\boldsymbol{\varphi}^t$. The latter is of dimension K, where K is the number of skills involved. $C \subseteq [0,1]^K$ is a set of contents, where c_j is the j–th content, defined with a vector $\boldsymbol{\psi}_j$ of K elements representing the skills required. $\varphi_{i,k} = 0$ means student's i skill level k is zero, whereas $\varphi_{i,k} = 1$ means having full ability. In addition In $\varphi_{ik} > \varphi_{fk}$ means student i is more knowledgeable than student f on skill k and $\psi_{jk} > \psi_{gk}$ means that more knowledge on skill k is required to solve content j than to solve content g. By comparing ψ_{jk} and φ_{ik} we define the ZPD as described in the formulas below. $\tau : \mathbb{S} \times \mathbb{C} \to \mathbb{S}$ is a function defining the amount of skills acquired by a student and therefore the follow-up state $\boldsymbol{\varphi}^{t+1} = \boldsymbol{\varphi}^t + \tau$ of a student $s_i \in \mathbb{S}$ after working on contents c_j^t at time t. In particular $\mathbb{S}$ and $\mathbb{C}$ are the spaces of the students and contents respectively.

Finally, a function y^t defines the performance $y^t(\varphi_i, \psi_j)$ at time t.
y and τ can be formalized as follows:

$$\begin{aligned} y(\varphi_i, \psi_j) &:= \max(1 - \frac{||\boldsymbol{\alpha}||}{||\boldsymbol{\varphi_i}||}, 0) \\ \tau(\varphi_i, \psi_j)_k &:= y(\varphi_{ik}, \psi_{jk})\alpha_k \\ \tilde{y} &:= y\epsilon \end{aligned} \tag{6.3}$$

where

$$\alpha_k^{i,j} = \max(\psi_{jk} - \varphi_{ik}, 0) \tag{6.4}$$

and ϵ is proportional to the beta distribution $\mathcal{B}(p, q)$. We selected p and q in order to have $\tilde{y} \sim \mathcal{B}(y, \sigma^2)$, where σ^2 is the variance, i.e. the amount of noise. We chose the beta distribution because it is defined between zero and one as the score. Consequently it will not change the codomain of the y function. The characteristic of the formulas are the following.

1. The performance of a student on a content decreases proportionally to his skill deficiencies w.r.t. the required skills.

2. The student will improve all the required skills of a content proportionally to his performance and his skill-specific deficiency up to the skill level a content requires.

3. As a consequence it is not possible to learn from a content more than the difference from the required and possessed skills. When $\psi_{jk} < \varphi_{ik}$ means that content j is too easy for student i and (s)he cannot learn from it.

4. A further property of this model is that contents requiring twice the skills level that a student has, i.e. $\|\boldsymbol{\psi_j}\| \geq 2\|\boldsymbol{\varphi}_i\|$, are beyond the reach of a student. For this reason his performance will be zero ($y = 0$).
   This is easily demonstrated with the following reasoning. Assuming there is no noise, if a student i is not able to solve a content the score y must be equal to zero. The score is zero when $\frac{||\boldsymbol{\alpha}||}{||\boldsymbol{\varphi_i}||} \geq 1$, but this holds only when $||\boldsymbol{\alpha}|| \geq ||\boldsymbol{\varphi_i}||$. The only case in which this occur is when for each skill k

$$\|\boldsymbol{\psi_j} - \boldsymbol{\varphi_i}\| \geq \|\boldsymbol{\psi_j}\| - \|\boldsymbol{\varphi_i}\| \geq \|\varphi_{ik}\|$$

$$\|\boldsymbol{\psi_j}\| \geq \|\boldsymbol{\varphi_i}\| + \|\boldsymbol{\varphi_i}\| \geq 2\|\boldsymbol{\varphi_i}\|$$

c_j	d_c	y	τ_k
$[0.1, 0.1]$	0.2	1	$[0, 0]$
$[0.5, 0.6]$	1.1	0.617	$[0.12, 0.0617]$
$[0.5, 0.7]$	1.2	0.515	$[0.1, 0.1]$
$[0.9, 0.9]$	1.8	0	$[0, 0]$

Table 6.1: Simulated learning process with two skills. A simulated student with $\boldsymbol{\varphi} = \{0.3, 0.5\}$ scores y and learning τ after interacting with different contents c_j.

With a simple experiment without noise, we can show the plausibility of the designed simulator. We inserted values in Eqs. 5.7 as follows. Let us consider a system with two skills and represent the student knowledge as $\boldsymbol{\varphi} = \{0.3, 0.5\}$. As it is possible to see in Tab. 6.1 with the increase of the content difficulty the learning increases and the score decreases until $\|\boldsymbol{\psi_i}\| \geq 2 \left\|\boldsymbol{\varphi}_j\right\|$. The maximal difficulty level is equal to the number of skills since a single skill value cannot be greater than one. The simulated environment in Tab. 6.1 can be represented in a 2D diagram as in Fig. 6.2. The red circle represents the simulated student's proficiency level on the generic skills one and two. The circle radius, instead, represents the total knowledge of the simulated students. The light blue circles are contents whose difficulty level is represented by the radius of the circle, whereas the knowledge required to solve them is shown by their position in the diagram. The area between the red lines describes the Zone of Proximal Development of the student. Only contents in that area can increase the students'knowledge. If the simulated student interacts with the contents located under the first red line, in the bottom left corner, the score is one but no new knowledge is acquired. Contents over the second red line, without intersection, are too difficult for the student and consequently do not increase the knowledge of the student who will then perform with a score equal to zero.

6.2 Experiment Session

In this Section we show the experiments performed on the learning process simulator and on the sequencer interacting with it in different scenarios and introducing noise.
A scenario is represented by a number of contents n_c, a number of difficulty levels n_d, a number of skills n_k, and a number of students for each group n_t.
All the first experiments will have no noise, i.e. $\tilde{y} = y$.

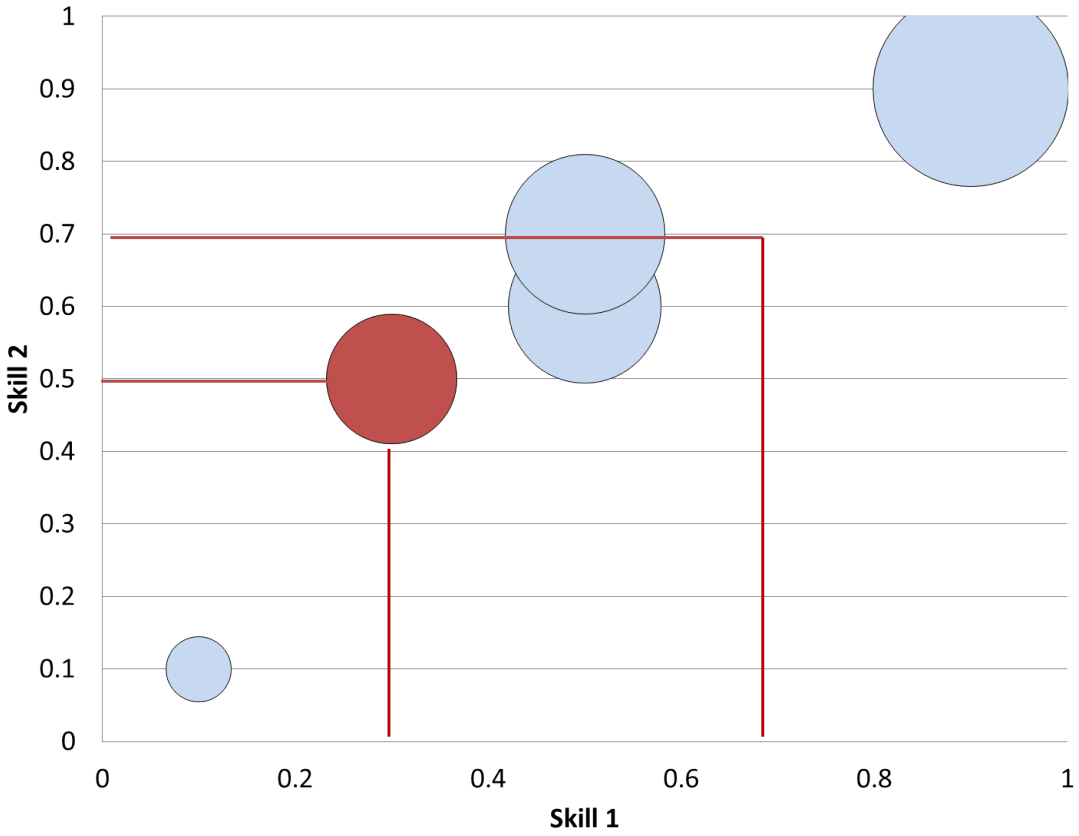

Figure 6.2: Student with two skills, $s_i = [0.3, 0.5]$ represented with a red circle, could interact with the contents described in Tab. 6.1 represented as the light blue circles. The radius of the circles indicates difficulty for contents and ability level for the student. The red lines shows the ZPD according to the simulated learning environment. Contents whose center lies in the ZPD can be solved by the student.

6.2.1 Experiments on the Simulated Learning Process

To prove the operating principle of the simulator we tested basic sequencing methods in a particular scenario. The one we chose is described in Fig. 6.3, with $n_d = 7$ and $n_c = 150$. For representation purposes we created the contents with increasing difficulty, so that IDs implicitly indicates the difficulty. For example, a content with ID 2 is easier than a content with ID 100, see Fig. 6.3. The proposed scenario mimics an interesting situation for sequencing, i.e. when more apparently equivalent exercises are available. The two policies we used are (1) Random (RND), where contents are selected randomly, and (2) the in range policy (RANGE), where each second content is selected in difficulty order. This strategy is informed on the domain because it knows the difficulty of the contents. We initialized the students and contents skills with an uniform random distribution between 0 and 1. Again for representation purposes we show the average total knowledge of the students that is represented by average of the students skills sum at each time step. We chose to perform the tests on 10 skills, i.e. the maximal total knowledge possible is equal to 10. We considered the scenario mastered when the total knowledge of the student group is greater than or equal to the 95% of the maximal total knowledge. Fig. 6.4 shows the total knowledge of two groups of $n_t = 200$ students, one group was trained with random policy the other one with the in range policy. RANGE is characterized by a low variance in the learning process. RND, instead, has a high variance because the knowledge level of the students at each time step is given by chance. It is shown that the order in which the student practices on the contents is important for the total final learning. Fig. 6.4 also shows how the practice on too many contents of the same difficulty level, after a while, saturates the knowledge acquisition. All these aspects demonstrate that the learning progress is plausibly simulated.

6.2.2 Sensitivity Analysis on the Vygotsky Policy

In order to evaluate the VP we created two more sequencing methods that exploit information not available in reality. The best sequencing knows exactly which is the content maximizing the learning for a student, for this reason we called it Ground Truth (GT). Vygotsky Policy Sequencer Ground Truth (VPSGT), instead, uses the Vygotsky Policy and the true score y of a student to select the following content. GT and VPSGT can be considered the upper bound of the sequencer potential in a scenario, but are impossible in reality because they use information that is not available. In order to select the correct value of y_{th} we plot the average knowledge level at time $t = 11$ for the policy with different y_{th}. From Fig. 6.5 one can see that the policy is working for $y_{th} \in [0.4, 0.7]$, this because of the

relationship between Eqs. 5.7 of the student simulator. In a real environment the interpretation of these results is twofold. First we assume y_{th} will be approximately the score keeping the students in the ZDP. Second, this value would allow finding the trade–off between exploring new concepts and exploiting the already possessed knowledge. Moreover, as one can see in Fig. 6.6, the policy obtains good results if compared with GT for some y_{th}, but for others the policy is outside the ZPD and the students do not reach the total knowledge of the scenario. In some experiments we noticed that the width of the curve in Fig. 6.5 decreased so that the outer limits of the y_{th} interval create a sequence outside the ZPD. As consequence we selected the value $y_{th} = 0.5$, that means a student would solve half of the task correctly and was successful in most of the scenarios.
How this value could be selected for a real ITS is discussed in Sec. 7.2.3.

6.2.3 VPS Evaluation

Similarly to the approach explained in Sec. 2.1.2, MF was previously trained with n_s students that were used to learn the characteristic of the contents. Consequently, the dimensions of the MF during the simulated learning process are: $\Psi \in \mathbb{R}^{n_c \times P}$ and $\Phi \in \mathbb{R}^{(n_s+n_t)\times P}$, so that $Y \approx \hat{Y} = \Psi\Phi$.
The scenario we selected for the tests with the VPS has $n_c = 150$, $n_d = 6$, $n_k = 10$ and $n_t = 400$. In order to train the MF–model a training and test dataset need to be created. We used $n_s = 300$ students who learned with all the contents in order of difficulty. We used 66% of the data to train the MF–model and the remaining 34% to evaluate the Root Mean Squared Error (RMSE) for selecting the regularization factor λ and the learning rate of the gradient descent algorithm. We performed a full Grid Search and selected the parameters shown in Tab. 6.2. The sequencing experiments are done on a separate group of n_t students. In order to avoid the cold start problem 5 contents are shown to them and their scores added to the training set of the MF. For $T = 40$ the best content c_j^{*t} is selected with the policy VP for the n_t students, using the predicted performance $\hat{y}_{ij}^t$. In order to avoid the deterioration of the model, after each time step the model is trained again once all students saw an exercise. A detailed description of the algorithm of the sequencer can be found in Alg. 7, where Y_0 is the initial dataset.
The retraining of the Matrix Factorization is performed here with a full retraining as in Sec. 4.1, nevertheless all of the previously mentioned update could be used, e.g. Abernathy [1, 55], UpMF [40], KSEMF, and KSEMF_SD (See Chapter 5). As one can see in Fig. 6.7 the VPS selects the first content similarly to RANGE. Then the prediction allows to skip unnecessary contents speeding up the learning. Once the total knowledge arrives around 95%, the selection policy cannot find contents

Parameters	Choice
Learning Rate	0.01
Latent Features	60
Regularization	0.02
Number of Iteration	10

Table 6.2: Parameters MF

that fit to the requirements. Consequently the students learn as slow as the RND group, as one can see from the saturating curve. In Fig. 6.8 GT selects the contents in difficulty order skipping the unnecesary ones. The average sequence of the VPS, instead, is also with approximately increasing difficulty but in an irregular way. This is due to the error in the prediction performance. In conclusion the proposed sequencer gains 63% over RANGE and 150% over RND. The presented

Algorithm 7 Vygotsky Policy based Sequencer

Input: $\mathbb{C}$, Y_0 π, s_i, T
Train the MF using Y_0;
for t = *1 to* T **do**
  for *All* $c \in \mathbb{C}$ **do**
    Predict $\hat{y}(c_j, s_i)$ Eq. 4.4;
  end
  Find c^{t*} according to Eq. 6.2;
  Show c^{t*} to s_i with Eq. 6.3;
  Add $y(s_i, c^{t*})$ to Y_t;
  Retrain the MF; // Corrects over- or underestimation by the MF
end

experiments show how the MF is able, without domain information, to model the different skills of students and contents and partially mimics the best sequence, which is the one selected by GT in Fig. 6.8.

6.2.4 Advanced Experiments

In this Section we want to show the correct working of the sequencer changing the parameters of the scenario n_k and n_c and later adding noise. In order to do so we consider the percentage of gain of VPS with respect to RANGE considering a specific time step $t = 30$ with $n_k = 10$ and $n_d = 6$. As one can see in Fig.

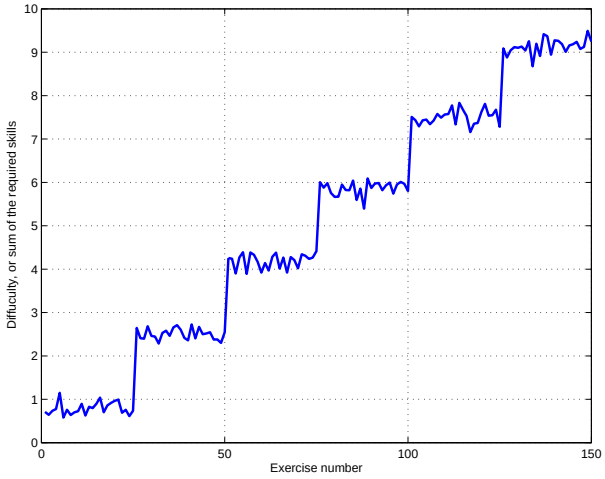

Figure 6.3: Scenario: content number and difficulty level.

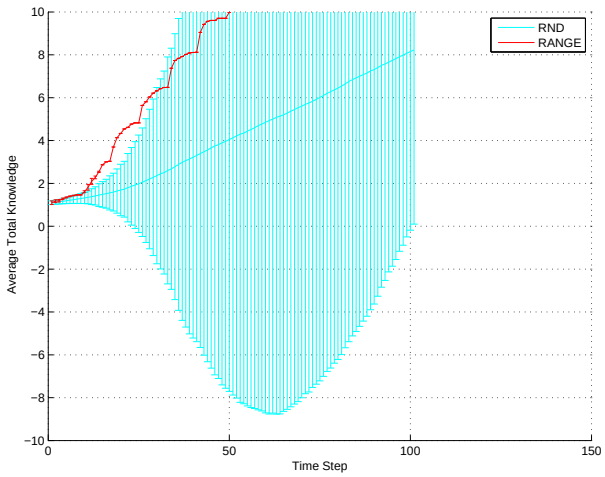

Figure 6.4: Comparison between RANGE and RND. Average skills sum, i.e. knowledge, over all the students with variance

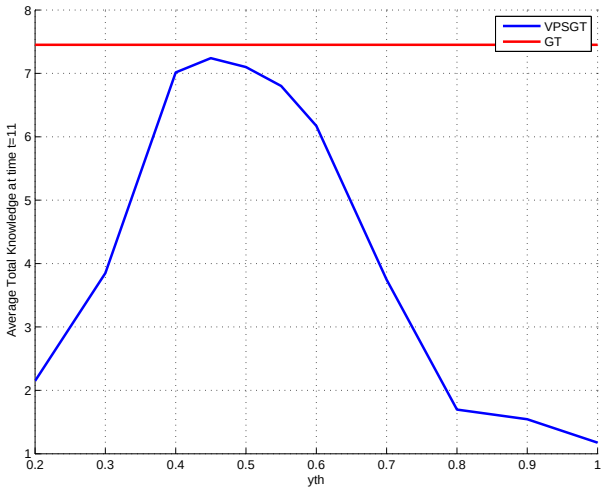

Figure 6.5: Policy selection, i.e. the performance of the Vygotsky policy with different y_{th} at the same time step. Different groups of students learned with the Vygotsky policy with y_{th} values going from 0.1 to 0.9. As shown in the figure the knowledge levels change according to the y_{th} selected.

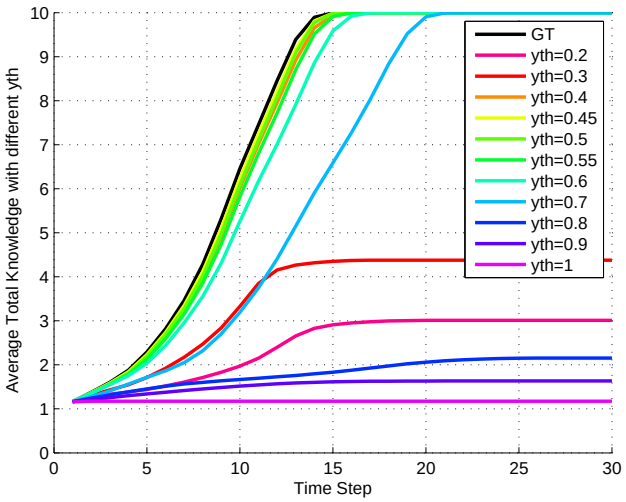

Figure 6.6: Effects of the different y_{th} on the final knowledge of the students. The learning curves of the student groups that learned with the different Vygotsky policies.

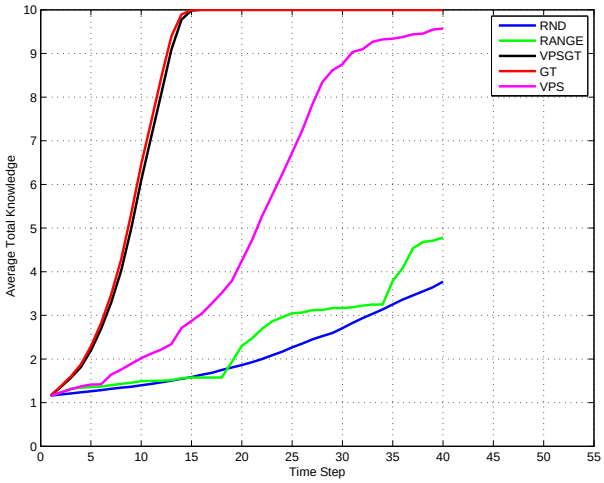

Figure 6.7: Average Total Knowledge. How the average learning curve of the students changes over time.

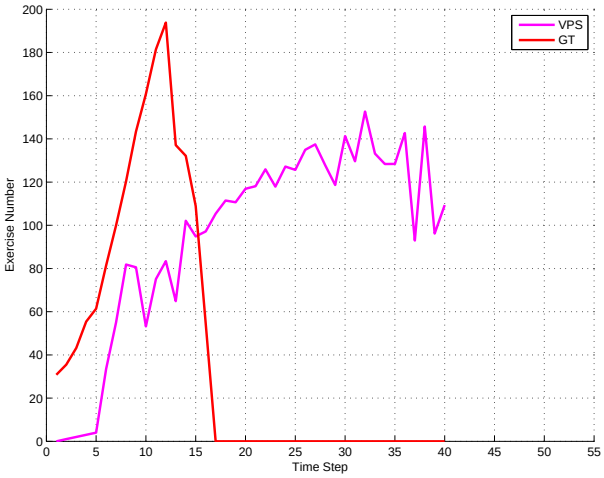

Figure 6.8: Average sequence selected by the GT and the VPS. The VPS approximate the optimal sequence that GT computes thanks to the real skills of the students.

Policy	Description
Random (RND)	Contents are selected randomly
In Range (RANGE)	Each second content is selected in difficulty order.
Ground Truth (GT)	Selects the contents according to which is the one maximizing the learning.
Vygotsky Policy based Sequencer Ground Truth (VPSGT)	Chooses the next content using the policy and the real score of a student.
Vygotski Policy based Sequencer (VPS)	Chooses the next content using the policy and the predicted score of a student.

Table 6.3: Sequencers Description

6.10 the gain obtained by the sequencer depends on the available number of contents. Since in RANGE each second content is selected, with $n_c < 60$ there are not enough contents for all time steps. Our sequencer can adapt without problems to the situation. The optimal point for the in range policy is when $n_c = 60$ because there is exactly the necessary number of contents for the student to learn. When $n_c > 60$ the students see many unnecessary contents and consequently learn slower. Fig. 6.9 with $n_c = 60$, $t = 30$ and $n_d = 6$ shows the dependencies between skills and gain. The experiments demonstrated a high adaptability of the sequencer to the different scenarios.
Last we experimented the results robustness adding noise, i.e. $\tilde{y} = y\epsilon$. We experimented with $\sigma^2 \in [0, 0.5]$. As one can see in Fig. 6.11 with $\sigma^2 = 0.1$ the Vygotsky sequencers are still able to produce a correct learning sequence but more time is required. The VPSGT is the one that suffered the most from the introduction of noise, probably related to the selection of y_{th}.

6.3 VPS Feasibility and Utility

There are several considerations that needs to be taken into account before the presented Sequencer can be tested integrated in an ITS. Therefore, we discuss hereafter Feasibility and Utility of the VPS as preliminary evaluation of the effort required to perform an online experiment.

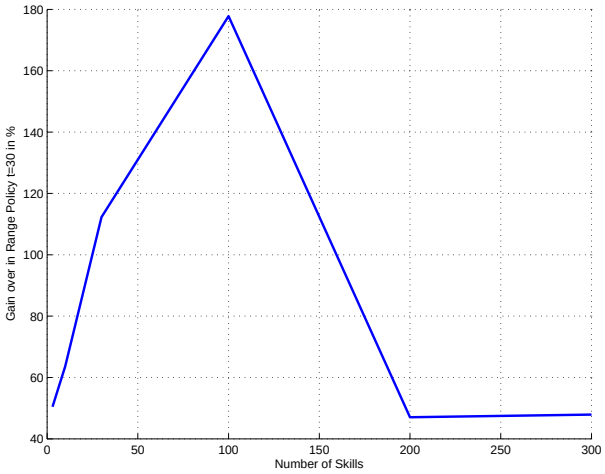

Figure 6.9: Gain over RANGE policy varying n_k. The gain is measured at a specific time step in percentage, considering the average knowledge level of the two groups of students, one practicing with the RANGE sequencer and one with the VPS.

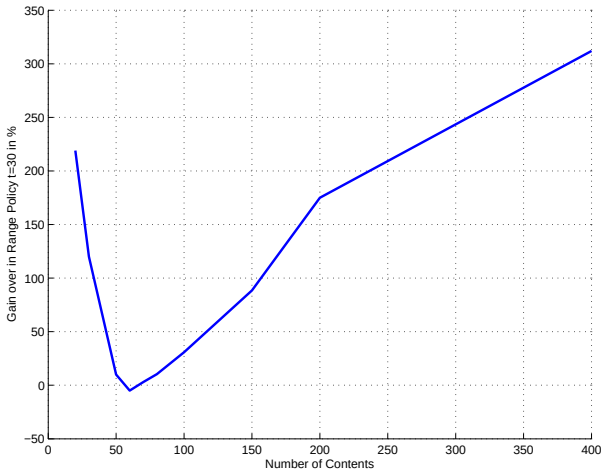

Figure 6.10: Gain over RANGE policy varying n_c. The gain is measured at a specific time step in percentage, considering the average knowledge of the two groups of students, one practicing with the RANGE sequencer and one with the VPS.

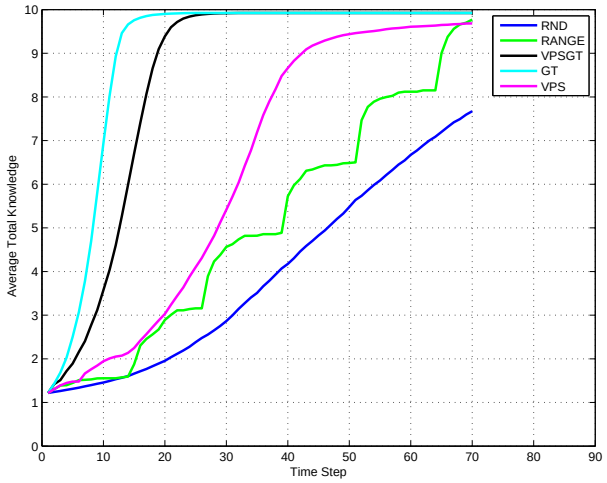

Figure 6.11: Effect of noise in the simulated learning process. Beta distribution noise with $\sigma^2 = 0.1$.

6.3.1 Sequencing VPS Feasibility

As aforementioned the presented experiments represent a proof of concept, i.e. they give a hint of what could happen if the VPS were integrated in an ITS. Since the VPS could deliver the same or better results than an informed state of the art with less authoring effort, we can safely suggest that it is worth to try to confirm our hypotheses. Before this can happen, the following questions still needs to be answered.

Theoretically, the selection of y_{th} implies the repetition of an evaluation experiment several times, as we did in the related section with the simulated learning environment. Therefore y_{th} can be considered as a sequencer hyperparameter. Nevertheless, simple reasoning can help us defining a reasonable range that could allow a first experiment. First of all, in our use case we should consider the score range that indicates whether the student sufficiently learned from the task and y_{th} should be set in this interval. As we will see in Chapter 7, a possible simple solution involves an analysis of the log files. In the same Chapter, we also present a more complex solution, which was developed in strong collaboration with the authors of the cited papers.

In addition, as pointed out by the experiments with the simulated environment, the scenario may influence the sequencers' performances. Here, we want to stress the fact that a large system with a large number of interactions and alternative actions is the most interesting scenario to test a domain independent sequencer. The added contribution of the sequencer is its adaptivity. If just a few tasks are avail-

able a fixed sequence will work as a VPS. The time to design the fixed sequence will be reduced and will not require the infrastructure that the VPS and each Machine Learning method need for their implementation. Moreover, the domain may be composed of different topics whose skills acquisitions are independent from one another or with a minimal overlapping. Also this aspect should be investigated in an online experiment.
Another important point to be taken under consideration is related to the cold start problem. The latter is strongly connected with the personalized prediction of Matrix Factorization and affects the utility of the sequencer as well. Therefore, more information can be found in the next Section, Sec. 6.3.2.

6.3.2 Sequencing VPS Utility

In [56] the cold start problem's effect on the MF performances applied in ITS was analyzed. From this analysis insights about the utility of the VPS can be obtained.
Utility defines how many interactions of a student are required by the system before the VPS can get a reasonable task sequenced. Since the VPS error is strongly dependent from the MF error, the VPS utility is strongly connected to the MF and its sensitivity to the cold start problem. As discussed by [39], at least 10 interactions are required to neglect this problem. This condition often occurs in ITS where systems are shown to novel students in different classes for trials.
How small datasets affects the performance of MF was already evaluated in [38] and, despite the lack of data, MF was still the better performing of the different algorithms tested. The dataset considered by [56], described in Sec. 2.4.2, is small and also sparse. Therefore, it can be considered more challenging than the one in [38].
The performance evaluation, presented by [56], considers the error of the MF that predicts scores for students with a short history length, i.e. for students of which just a few interactions are available. For this reason the impact of the cold start problem is expected to be particularly strong . [56] tested the performance of MF derived algorithms on different students' sets that changed according to how much data was available for one student. The authors considered the History Lengths (HL), i.e. the number of interactions available for each student. The error was evaluated with LOO considering the students that had at least interacted with is 3 for training. As typical for LOO approach, for each student the last interaction was left out from the training set. They repeated the experiment then several times, removing from the test set last interaction from the students with $HL \geq 3$, then $HL \geq 4$, etc. until $HL \geq 8$. By comparing the RMSE results of MF

so computed with those of other state of the art algorithms, we can foresee the expected behavior of MF in an online experiment.
Under the interpretation of the author, MF and derived algorithms can be used for prediction as soon as they beat a simple Global Average Predictor (GAP). BMF and MF are in general influenced by data of students with short history negatively at the beginning, although, for students with a longer history, these data can be used to ameliorate performances. As shown by [56], from $HL \geq 5$ MF, BMF and prediction performed using just the BMF biases have not sufficient information about the students to predict their performance.
In conclusion, this should hold theoretically also for the use of the VPS, although an online experiment is required for a full evaluation.

6.4 Conclusions

In this Chapter we presented the VPS, a sequencer based on performance prediction and Vygotsky's concept of ZPD for multiple skills contents with continuous knowledge and performance representation. We showed that MF is able dealing with the most actual problems of Intelligent Tutoring Systems, like time and personalization, retrieving automatically skills required and difficulty. We proposed VP, a performance based policy that does not require direct input of domain information, and a student simulator that helps in preliminary off-line evaluation. The designed system achieved time gain over random and in range policy in almost each scenario and is robust to noise. This demonstrates how the sequencer could solve many engineering/authoring efforts. Nevertheless, an experiment with real students is required to better confirm the validity of the assumptions of the simulated learning process. Preliminarily to this online experiment we presented a discussion on Feasibility and Utility of the algorithm. Such aspects need to be taken into account for the online experiment presented in the next Chapter. In conclusion, the VPS was designed to work without domain analysis: the MF will reconstruct it thanks to a continuous score representation. This will allow the integration of intelligent sequencers in ITS whose content analysis is not affordable.

Chapter 7

Large Scale Experiment

Contents

7.1 Lightweight Integration of Machine Learning algorithms 93

7.1.1 Machine Learning Requirements 94

7.1.2 A novel Protocol for Machine Learning Integration . . . 95

7.2 Sequencers' Integration in Commercial ITS 98

7.2.1 Commercial ITS Dataset Preprocessing 100

7.2.2 Online Update Integration 101

7.2.3 Vygotsky Policy Integration 102

7.2.4 Technical Integration 103

7.3 Experiment Session 103

7.3.1 Experiment Design 104

7.3.2 Results from Dataset Analysis 104

7.3.3 Post Test 105

7.3.4 Questionnaire 107

7.3.5 Integration 108

7.4 Computational Requirements 108

7.5 VPS with adaptive Threshold 109

7.6 Conclusions 111

Many Machine Learning solutions for Learning Analytics are born and die in laboratories due to the high integration requirements and constraints. As reported in [30] only one half of the developed task recommenders interacts, in a final evaluation stage, with students, whereas the other half stays at design or prototype stage of development. Adaptations of models to be integrated into already existing systems cannot be evaluated as a whole and need to be observed from different perspectives [5]. This is also important if we consider that in an integration experiment partners, especially if coming form different research areas, have different interests and qualitative measures. Integration within already existing ITS is even more challenging since those systems were not intended to be combined with Machine Learning methods, so the tendency is to develop ad-hoc learning environments. Such an integration is becoming unfeasible for Machine Learning sequencing experiments also because large dataset are needed, time is required to show students' learning gains and it is necessary to have enough tasks to sequence. This impose large scale experiment in sufficiently large systems, that cannot be created ad-hoc.
In this Chapter we show how we adapted the Machine Learning based domain independent sequencer of Chapter 6, composed of a Performance Predictor and a score based task sequencing policy, in order to be integrated in the large commercial online maths ITS described in Sec. 2.4.1.
Thanks to the work done, we could trial the sequencer with 100 students for a month and discuss the obtained online experiment's results from different perspectives. The developed sequencer showed the following promising characteristics:

1. Lightweight integrability of Machine Learning based sequencer in not ad-hoc constructed systems.

2. Having comparable response time as the actual rule based system.

3. Achieving the same post-test results with almost no curriculum authoring effort.

4. Possessing a better user modeling, better adapting to the knowledge acquisition rate of the students.

5. Outperforming the current sequencer in the perceived experience questionnaire.

These contributions are also published in: [43, 48].
In order to show how we achieved the aforementioned contributions we present our work as follows. In Sec. 7.1 we present a new API that allows to integrate Machine Learning algorithms with a single method, in Sec. 7.2 we present how the Vygotsky Sequencer was adapted to a real time working web service. Finally, in Sec. 7.3 we present the designed experiments and discuss the results.

7.1 Lightweight Integration of Machine Learning algorithms

The impossibility to have tangible results on the specific use case without an online test, especially for sequencing, is not sufficient to motivate the effort of a complete ML integration. An easier ML integration would subsidize the possibility to: (1) evaluate different Machine Learning (ML) methods, (2) increase data collection by increasing the ITS users, (3) maintain the algorithms, e.g. because new data are available or the model is outdated.
Previous discussion about the topic can be found in [33], who considers the ITS as a pedagogical problem and underlines the necessity to separate the student modeling from the other components. The ITS server has the task to provide anonymous interaction between intelligent components on the basis of a specific communication protocol. Whereas the student/teacher interface interacts with the ITS server forwarding the contents to the students and resuming statistics to the teachers. This framework oversimplifies Artificial Intelligence problems neglecting the fact that Artificial Intelligence comprises both very simple heuristics, like uninformed search algorithms, or more complicated ML methods. [17], instead, proposes a plug-in framework for separating the contents from the sequencing, so that the sequencing technology could be interchanged easily. Nevertheless, the applicability of that framework to ML-powered sequencers is not discussed.
In this Chapter we fill the gap between data collectors and ML developers by:

1. exploring the requirements that ML systems have to be applied to a specific learning problem (sequencing), and

2. proposing a minimally invasive protocol (based on web services) to easily integrate Learning Analytics Services into e–learning systems. The latter point allows also to minimize integration requirements reducing the risks and necessary corrective actions in case of experiment failure.

7.1.1 Machine Learning Requirements

ML is a part of Artificial Intelligence. It has the task to abstract from past data to a model that can be used to predict the outcome of future cases. Considering ITS, the two most common ML applications are Performance Prediction and sequencing. Such ML applications consist of 3 steps that need to be integrated:

1. **Training**, where the model is built,

2. **Validating**, where the model error is evaluated first off-line, i.e. in a laboratory, and then online, i.e. applied within a bigger system as decision method

3. **Model Maintenance**, where ML expertise is required to determine whether the model is still data representative or a retraining is required.

The early integration for online validation is problematic. In case of Learning Analytics humans are interacting with the system imposing ethically correct experiments and higher degree of confidence. Other technical requirements exist: due to the strong statistical approach, data collection or data exchange and analysis could be considered as mandatory requirements for a ML experiment. Similarly to medical applications, also in educational data mining there is the problem of having access to data in reasonable amount[1]. The computational time necessary to train the model is often not addressed in this application type and is strongly algorithm dependent. Also if a small amount of time is required, several models with different hyperparameters need to be built for the evaluation phase[2]. Slightly different is the approach required for validating sequencing algorithms. The full testing can be done only online [10], since a brute force approach, where all possible sequences are tested on a single person, is not feasible. When the validation phase is over, the model can be used as decision function for Performance Prediction or sequencing. Generally, this operation does not require particular resources, if we assume that the model does not need maintenance.
With these premises it is clearly necessary, especially for sequencing, to have an early integration with low system coupling.

[1] With reasonable amount is meant a quantity that allows the model to generalize from the single examples to an unique model.

[2] Hyperparameter tuning is done by comparing the prediction error of the model on a separate part of the dataset.

7.1.2 A novel Protocol for Machine Learning Integration

In this Section, we propose a web service oriented integration protocol. In particular, we discuss the specific context of collaboration between a ML provider, who has the ML expertise, and a ML client, who has the potentially exploitable data. According to the requirements mentioned, all steps in Alg. 8 necessitate to be integrated. In order to reduce the integration effort, several implementation options are possible.
For the training it can be either assumed that the model is built by the client or by the provider. In the first case the client must be sure to have enough computational resources to perform a model training and validation. In the second case, the data are transferred to the provider that later returns the model to the client. This is the simplest option since it reduces the required ML expertise of the client. In case of maintenance or updates, it allows the provider to eventually change the algorithm and just transfer the model. The validation phase of a Performance Prediction method, instead, is done only in case of testing, whereas for sequencing the *Get_Next_Exercise* method needs to be integrated with Database (DB) access already in the validation phase. The architecture in [17] could be extended to ML based sequencers by granting reading and writing rights to the DB. Nevertheless, this approach still requires the API definition for model transfer and DB integration. This is not straightforward if the developed method needs an online formative evaluation and consequent refinements. Moreover, it neglects the lack of ML expertise, the possible degradation of the model, and improper implementations of the sequencer.
Consequently, we designed the following system for increasing the chance to apply ML methods to big amount of unexploited data.
Considering a Performance Prediction experiment the ML evaluation can be done in a laboratory. Whereas for a sequencing problem, the integration of a ML sequencer in the ITS system must take place at the beginning of validation step without previous exhaustive evaluation. The small integration requirements reduces the risks and necessary corrective actions in case of experiment failure.
The provider could be represented in our case from a ML expert group that developed a set of ML methods for Learning Analytics. The client, instead, could be another research group or a company, that would like to use those method in its system.
Web services are programming language independent services deployed over the web with two main parties: the service provider and the service client.
To avoid also these problems a Web Service deploying the ML-powered sequencer could be use. In order to do so a further separation of the sequencer is required

Algorithm 8 Methods for consuming ML-powered sequencers

```
Build_the_Model(Train Dataset, Test Dataset)
Load Data from DB;
foreach hyperparameters do
    Create Model;
    Validate Model;
    Save best actual Model;
end

Get_Next_Exercise()
Load the Model from the DB;
Use the Model for sequencing;
Record the real and predicted outcome;


Maintain_the_Model(Recorded data)
Evaluate Model degradation;
if Training done inhouse then
    Build_the_Model();
else
    Import the Model from a third party;
end
```

by eliminating the client database access. A part of the database will be duplicated at the beginning of the cooperation and used by the provider for the training phase. As a consequence, the provider will be able to create the model, tune it and update it whenever it is necessary without any other requirement. The API for the designed sequencer service is:

$Get\_Next\_Exercise(StudentID, PreviousExerciseID, PreviousScore, NextExID, Timestamp)$.

The parameters passed by the method *Get_Next_Exercise* can be found in Tab. 7.1. The parameters passed by the method *Get_Next_Exercise* are the student

Parameter	**Description**
StudentID	Identifcation number of the student. Used to update the DB.
PreviousExerciseID	Identification number of the task solved by the student. Used to update the DB.
PreviousScore	Score obtained by the student in the task he just solved. Used to update the DB.
NextExID	Identification number of the task selected by the VPS to be shown to the student. Used to manage A/B tests.
Timestamp	Timestamp, used for synchronizations between DB.

Table 7.1: Parameter of the ML API

ID, his last score and in which task he obtained it. The last parameter is a Timestamp used mainly for synchronization. This value can also be used to avoid that for any transmission error, two times the same DB entry is recorded. The last data recorded is necessary in order to maintain the copy of the DB up to date and give the possibility to both sides to evaluate the results. It is important to notice that the interests of provider and client are different and consequently different validations could take place. The method *Get_Next_Exercise* should be called after the use of the old sequencer, if there is one, and before the sequencing decision is applied by the ITS as shown in Alg. 9. By doing so, the client can decide with a boolean flag whether the previous or the new sequencer has to be used. In order for the system to be robust to connection problems a timeout variable is used. The server side will take care of all parts that require a ML expertise and create the wrapper in Fig.7.1 for sequencing the steps listed in Alg. 8.
The simple integration in Alg. 9 allows the client to decide with no additional costs to change to another ML method offered by the Learning Analytics platform

or continue to exploit the method as it is for further tests.
A comparison study can be run easily by maintaining the current sequencer version as shown in Fig. 7.1. In case of local exploitation the web service structure can be integrated within the client system importing from the provider the relevant parts as done in [17]. Fig. 7.1 shows also how the DB of the Learning Analytics platform is divided in three components. The log files DB, is partly composed from the previous data of the company and the data collected by the service. The model DB stores the developed ML models and the Domain DB contains curriculum information (e.g. skills involved in the tasks, difficulty level, task domain) that has to be taken into consideration while sequencing. The further separation of the sequencer from the ITS envisages the possibility not to use curricula-based sequencers but a novel kind, as proposed in Chapter 6, that schedules tasks exploiting the statistical information of the students (e.g. scores, time needed, previous exercise solved). For these reasons in Fig. 7.1 the domain DB is in brackets. The increased complexity for the server side in Fig. 7.1 allows therefore also the use of the same sequencer for different ITS. The same could be done with other Learning Analytics ML applications by defining one API for each of them.

Algorithm 9 Implementing the Web Service, client side

Input: StudentID, PreviousExerciseID, PreviousScore, Timestamp
NextEx = Get_Next_Exercise_Curriculum_Based();
if *!timeout* $\wedge$ *Flag* **then**
    Get_Next_Exercise(StudentID, PreviousExerciseID, PreviousScore, NextExID, Timestamp);
end

7.2 Sequencers' Integration in Commercial ITS

In this Section we want to integrate the domain independent sequencer called Vygotsky Policy Sequencer (VPS) presented in the Chapter 6 with the large Intelligent Tutoring System presented in Sec. 2.4.1 by means of the lightweight API presented in Sec. 7.1. This will involve further modifing two components: the Performance Prediction method and the score based policy as described in Sec. 7.2.2 and in Sec. 7.2.3 respectively. The experiments performed in Sec. 7.3 represents the online and final evaluation of the VPS in comparison to the state of the art sequencer designed over the years by the experts for the commercial ITS.
In Chapter 6, the evaluation on groups of simulated students in comparison to several sequences showed the advantages of such a system. There is no authoring

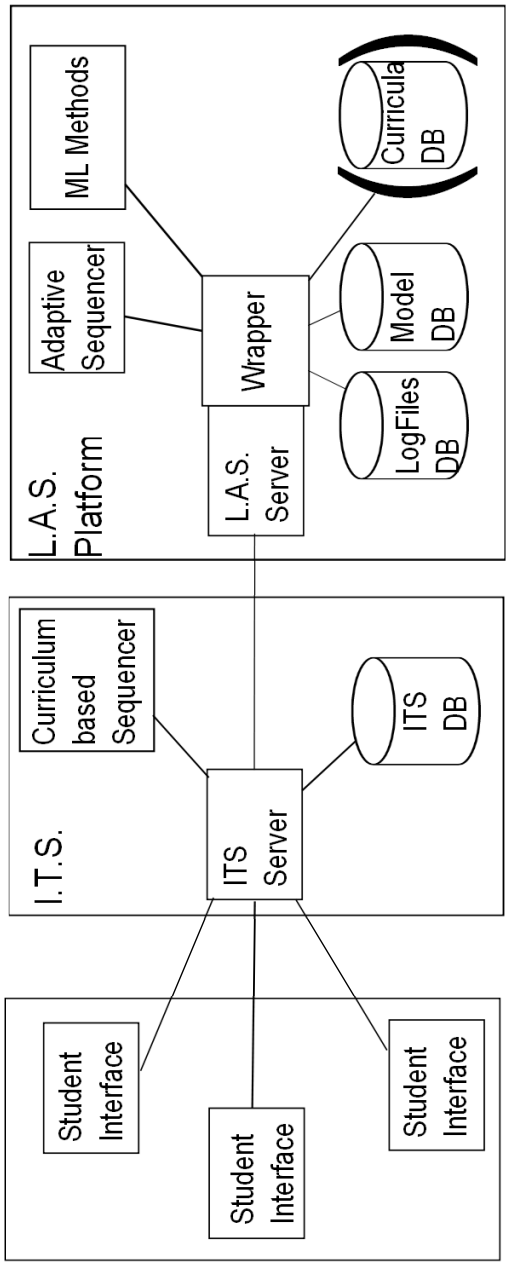

Figure 7.1: New Framework for lightweight ML integration. Structure and interaction between the ITS platform and the Learning Analytics Services platform

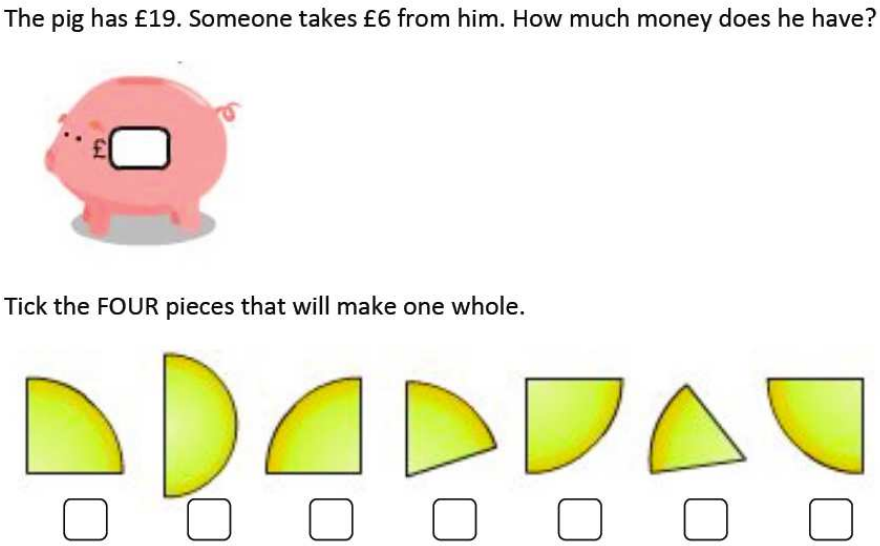

Figure 7.2: Two questions of the commercial ITS

effort for sequencing required, since the system is flexible enough to adapt to ITS with different number of tasks to practice per difficulty level. Therefore, most advantages are gained with large or small task availability per difficulty level. Given a curriculum that possesses the correct number of tasks to practice with for each difficulty level, the performances between a sequencer that selects tasks in order of difficulty and the VPS are comparable. On the contrary, if there are more tasks of the same difficulty level, VPS is able to skip the unnecessary ones. With just few tasks pedagogical experts would be able to set up a unique sequence in a short amount of time, since not many sequences could be implemented.
In conclusion, the most interesting scenario for an evaluation would need an ITS with many tasks, where the sequencer could reduce the burden of pedagogical experts. Given the interdisciplinary knowledge required for creating such an ITS from scratch the experiment, without integration with an already existing system, would not have been affordable.

7.2.1 Commercial ITS Dataset Preprocessing

As aforementioned in Sec. 2.4.1, the ITS has 20 topics about maths for children aged from 6 to 14, who can practice on over 2000 tasks at school or at home. An example of questions posed to the students in exercise and test tasks can be found in Fig. 7.2. The score, as in [45], is represented in a continuous interval which goes from 0 to 10. The topics and new skills to be acquired are introduced following

the curriculum of the country.
This also defines the Math Age which represents both the student's skills level on a topic and task's difficulty level. Tasks are assigned to topics and students have a Math Age per topic. Considering a specific topic, a task of Math Age 10 means that the difficulty should be solved by students of that age. It means also that a student with Math Age of 10 in a topic knows more than a student of Math Age 9.
The commercial system possesses a rule–based adaptive sequencer that was designed and refined by pedagogical experts over the year that we call state of the art (SotA) sequencer. The Math Age is the indicator the SotA sequencer uses to monitor the progress within the curriculum and select the next tasks. A student can see tasks of the next difficulty level only if the ones of the previous level are completed.
We analyzed the dataset collecting the information summarized in Tab. 4.2. In order to avoid sparseness, which strongly affects MF, each line is generally abstracted to Knowledge Component (KC) level, i.e. the algorithm predicts if the student is going to answer correctly modeling his knowledge on a KC. Since we did not have this information, we preprocessed the dataset at task level predicting the future score for each task. In order to select the best performing model we followed the standard approach in the field to divide the dataset temporally in two thirds for training and one third for testing, evaluating the performances with the Root Mean Square Error (RMSE) as done in Sec. 2.1.1. We considered the last 3 years of data, excluding the oldest ones, since the tasks were slightly modified over time. We also removed the skipped tasks, where the score is automatically assigned to 7.5 out of 10 (7.5/10) by the ITS. Those data were considered noisy since there was no evidence that 7.5/10 could represent the knowledge of the student at that time. With a full Grid Search we selected the best hyperparameters' combination ($P = 50$,$\lambda = 0.01$, learn rate = 0.02, 100 iterations per training).

7.2.2 Online Update Integration

It is well known to those working with recommender technology that MF deals with slow evolving states, i.e. models are kept constant over a long period of time, whereas here we have a fast evolving state. As a consequence if the student is repeating a task his mark is computed as the weighted sum of the previous performances. Two past data on the same task are considered equally. Since the students'features change after each interaction in Chapter 6 the model was retrained each time. Given the data amount this was not feasible while students were interacting with the system. In order to keep the model up to date, we implemented

the online update UpMF proposed in [40] and in Sec. 4.2.1 that was previously tested for cold start problems in recommender systems. As already described, the method is an approximation of a full MF retrain. Moreover, the paper reports how the performances of the update deteriorates over time in comparison to the full retrain. Considering Alg. 7, we solved again the minimization problem of Eq. 4.3 optimizing the student's latent feature vector $\boldsymbol{\varphi}$ with gradient descend algorithm. We noticed that after approximately 20 interactions the model's update was not updating features as expected. This is coherent with the errors behavior reported in [40]. As a consequence, each night we retrained the model, assuming students would see approximately 10 tasks per day. Given the large data availability we had no cold–start problem, which is experienced in MF when not enough data on tasks or on students are available. As aforementioned, the task cold–start problem is not common to the movie rating applications since there the data availability is higher, but could be experienced in ITS. For this reason it was crucial to have a partner with large data availability both on tasks and on students. We were able to select 100 students coming from the same school and that had already experience with the system, so that also the student cold–start problem could be avoided.
Despite the fact that KSEMF or KSEMF_SD would have represented a better choice to be implemented as Progress Modeling algorithm, they were not available by the time the online evaluation took place. Since it was not possible to run the experiment again with the novel algorithm, in Sec. 7.4 we evaluate theoretically the advantages of having KSEMF approach instead of UpMF as predictor analyzing also the computational requirements of the sequencer integrated as Web Service.

7.2.3 Vygotsky Policy Integration

The policy integration consisted first of all in selecting the threshold score y_{th} (see Eq. 6.2). In Sec. 6.2.2 a sensitiveness analysis of the VP to y_{th} was done with simulated students. The same approach in a real scenario was considered detrimental for children, consequently the threshold score was selected according to the authors experience with the system and according to following considerations. In Sec. 6.3.1 it was discussed that, in order to keep the student in the ZPD, the selected score should be good, in order to be able to assume that the student was learning something from the task, but also not excellent, to avoid unnecessary repetitions on already known concepts. [44] suggested to select the threshold in the middle of the passing range, i.e. given a passing score of 5.5/10 one should select as threshold score 8/10. Our decision was made also on further considerations.

Given the characteristics of the MF we assumed that the model was going to underestimate the performances of the student, so 8 as threshold score would have been too high. Moreover, experts, that were analyzing the tasks, suggested that the path through the curriculum could be ameliorated by removing unnecessary repetitions rather than increasing practice. Consequently, we decided to choose a threshold score $y_{th} = 6.5$, i.e. the lowest possible in the passing range keeping a safe guard in case of overestimation by the model.
A further policy adjustment was required since in exercise modality bottom out hints are available. We decided to consider only tasks in test model (see Sec. 2.4.1) to evaluate students'ability. As a consequence only tests are selected with the VP. Whether or not the correspondent exercise should be shown is decided considering the Performance Prediction, if the score predicted is higher than 9.5/10 the exercise is skipped.

7.2.4 Technical Integration

As proposed in [48] and in Sec. 7.1, we integrated the VPS within the ITS with a single method API. The minimal integration effort, visible in Alg. 9, was crucial to convince the commercial partner to invest time and effort integrating a sequencer still not fully evaluated. The parameters passed by the method Get_Next_Task have the following function. The student ID, the task ID and the relative score obtained are required in order to maintain the VPS DB up to date. The method Get_Next_Task was called after the use of the old sequencer by the ITS but before the sequencing decision is applied as shown in Alg. 9. $NextTaskID$ contained the task suggestion of the state of the art sequencer, so that the VPS was able to manage the A/B test, deciding which students were practicing with which sequencer. Moreover, to be robust to connection problems we used a Timestamp indicating when the data was recorded in order to avoid inserting duplicates in the DB and a timeout variable was used at ITS side for connection problems.
This single method was exposed as Web Service.

7.3 Experiment Session

The purposes of the trial with the students were several. Firstly, we wanted to show that it is possible to sequence tasks by just considering students' score without frustrating them. Secondly, we wanted to evaluate sequencer performances in comparison with the current sequencer used by the ITS, the so called SotA Sequencer, which was adapted over the years to the tasks and countries curricula. In order to answer these questions we analyzed the following success criteria: learning

gains evaluated with trial data analysis, comparison of questionnaire and post test scores with the SotA sequencer, and finally we evaluate integration performances.

7.3.1 Experiment Design

To demonstrate learning gains by just changing sequences is known to be problematic since the most used indicator is the learning gain of students. This is generally obtained with a large number of students and over extensive time. Schools agreed with us to let 98 children interact with the ITS 45 minutes a week for 4 weeks. Another appointment in the fifth week was granted for a 30 minutes post test and a five question questionnaire for the perceived experience. Students were able to practice also at home and use all other related features of the ITS, e.g. spend coins gained for passing tasks for decorating their virtual room. Of the 98 students that were assigned to the study we randomly assigned them to two groups one was practicing with the SotA sequencer and one with the VPS. Students did not know to which system they were assigned. Given the reduced amount of time we could not let students practice with all 20 topics in order to be able to monitor any learning gain. We selected 3 topics and one recall topic, i.e. Fractions, Properties of Numbers, Solving Problems as well as Rapid Recall on additions and subtractions. Moreover, we limited the VPS degree of freedom by defining an active range for each topic, i.e. we selected a subset of tasks between which the VPS could choose in order to limit difficulty jumps. The active range of each topic is initialized with the Math Age of the most difficult task of a topic a student could solve, this represents the center of the range. We then allowed only tasks around +/- one year Math Age from the center. Each time the student is able to solve a more difficult task in test mode the center of the active range is updated. Although from simulated experiments in Chapter 6 an active range seemed not to be required, we preferred to adopt this risk minimization procedure in order to avoid frustrating excessively the students in case of experiment failure. Experts considered the range adequate for an ethically correct experiment and large enough for being able to evaluate the ability of the VPS to construct a reasonable path. For introducing new topics we adopted the simple policy of showing them the easiest available task, further tasks on the topic are selected in the active range with the Vygotsky policy.

7.3.2 Results from Dataset Analysis

From the 98 students we filtered those that practiced on less than 10 tasks and/or did not participate to all tests having 80 students left. From the trial data analysis we could notice that there was no big usage difference. Both groups approximately

	VPS	SotA
Avg Math Age Improvement	0.06±0.038	0.03±0.0354
Avg Start Math Age	8.41±1.42	8.26±1.94
Avg End Math Age	9.72±1.92	8.84±2.02
Avg Tasks Score	6.82±0.901	7.9±0.992
Inter Topic StD	0.64 ±0.30	0.32±0.44
MF Error	0.317±0.10	-

Table 7.2: Trial Data Analysis. Values are indicated with ± standard deviation

saw 2000 tasks in a month. In order to have a learning gain comparison we computed the average Math Age per student and per topic at the beginning (Avg Start Math Age) and at the end (Avg End Math Age) of the experiment. We then normalized the difference, i.e. the students' learning gain, with the number of tasks seen, in order to exclude also amount of practice differences (Avg Math Age Improvement). As one can see from in Tab. 7.2 the average improvement of VPS students per task is double as much as the SotA ones. This proved that the VPS is able to propose tasks in a way that students can proceed in the curriculum also if this is composed by different topics.
However, by observing Fig. 7.3 it is possible to see how the average standard deviation (Inter Topic StD) between topics' Math Age (Inter Topic Standard Deviation) is higher for those who interacted more with the system. This means that the inter knowledge standard deviation between topics' knowledge will increase until the student finishes the tasks of some topics, i.e. when the tasks of the mastered topics will be too easy to be in the ZPD and the VPS will select those of the uncompleted topics.
The Performance Prediction was working correctly. As one can see in Tab. 7.2 the average score for VPS students is 6.82, i.e. the threshold score 6.5 plus a slight overestimation, as expected. The average score of the SotA students is coherent with those of the data used to train the model as one could see comparing Tab. 7.2, 4.2.

7.3.3 Post Test

The trial post-test comprised 15 questions; 5 corresponding to each of the three topics, excluding the recall. The questions were sourced by experts from the ITS library of tasks ensuring that the Math Age assigned to the task was consistent with their perceived difficulty of each chosen question, and that the questions range of difficulty level was broad and considering each difficulty level. Given the

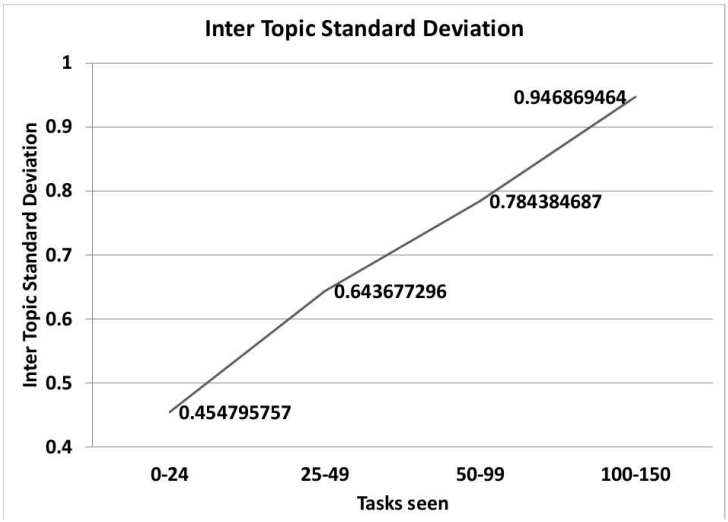

Figure 7.3: Inter topic standard deviation, i.e. average of the standard deviation between Topic Math Age for students that interacted with 10-24, 25-49, 50-99, or 100-150 tasks.

average age of student this range was from Math Age 7.5 to 11.0. All 15 questions were taken from the test tasks. These originally existed in paper form and were adapted to the online ITS tutor. Thus, converting the digital questions back to their original form required no work, and the questions lost no key form factors (such as digital interactivity) in the conversion process. For each question one point was assigned so that the sum of points obtained, normalized in a 0 to 10 range, represents the score of the post test. Moreover, taking the tests from the ITS curriculum, allowed the experts to reassess the student knowledge in order to compute the actual Math Age. As expected due to the short time period of the trial, no difference in average score and Math Age can be seen between the two groups (Tab. 7.3). This means that the sequencer does not damage learning. This result could appear at first glance in contrast with the average Math Age improvement, however the final average Math age reported in Tab. 7.2 of VPS students, i.e. the most difficult task students could solve, is more similar to those evaluated by experts in the post test. Finally, the VPS system was able to better model the current knowledge of the student and adapt to it.

	Post Test Score	Average Math Age
VPS	6.68±1.53	9.94±0.60
SotA	6.70±1.79	9.96±0.71

Table 7.3: Post Test Comparison. Values are indicated with ± standard deviation

7.3.4 Questionnaire

In order to evaluate the perceived experience of the students following questions were posed to them.

- Q1: Was the ITS fun?
- Q2: Were the exercises repetitive?
- Q3: Were the exercises easy?
- Q4: Was the ITS helpful?
- Q5: Was the ITS easy to understand?

The students could give a vote between 1 and 5 where 5 meant strong agreement and 1 strong disagreement.
As one can see in Tab.7.4 in almost all questions the VPS is slightly better than the SotA sequencer except from Q4 where the outcome is equal. In general the experience was positive, as one can see from Q1 and Q4. There where no usability issues related to introduction of the sequencer as reported from Q5. In Q3 students stated that tasks were between the adequate difficulty and too easy. This is coherent with the outcome of the post tests since the tasks proposed were too easy at the beginning. The VPS was better at adapting to the students' learn rate, so the sequence proposed by the VPS was perceived by the students to be of a more correct difficulty level. This agrees also with the data analysis where the average score of the VPS students is lower than the those of the SotA group. The only negative comment reported was the repetitiveness that could have been perceived by both groups for several reasons. Firstly because the set of questions in the tasks cannot be interrupted, secondly because the recall tasks are similar with one another but in the commercial version are presented interleaved with more topics.

	Q1	Q2	Q3	Q4	Q5
VPS	3.76 ±1.03	3.69 ±1.22	3.26 ±1.13	3.56 ±0.98	4 ±0.95
SotA	3.59 ±1.23	3.9 ±1.12	3.49 ±1.13	3.51 ±1.29	3.49 ±1.11

Table 7.4: Questionnaire comparison. 1: strong disagreement, 5: strong agreement. Values are indicated with ± standard deviation

7.3.5 Integration

We rented a server with 8 virtual CPUs, 30Gb RAM and two SSD of 80Gb each. The adapted VPS required $6s$ worst case to: identify the student, his appertaining group, update the model, and select the next task from the subset with the VP. The last action was not always necessary, if, for instance, the student had to practice with the correspondent test of an exercise or if he was of the SotA group and then the suggestion of the ITS needed just to be forwarded. The sequencing time could be further reduce by indexing the DB, but we did not require to do so, since response times were already comparable to those of the current SotA sequencer. The implementation was tested with 30 students practicing at the same time, but we do not exclude it could work with more.

7.4 Computational Requirements

In this Section we show by discussing the computational requirements of KSEMF and UpMF, why KSEMF would have been a better option as Performance Predictor than UpMF for a large experiment such as the one described in this Chapter. As reported in in the previous Sections, for 30 students in parallel UpMF required 6 seconds to update a student's state. However, these 6 seconds included also the time required for the selection policy. Therefore, we present hereafter a more detailed analysis of the time required for the updates of KSEMF and UpMF. As already said, the approach of [40] has the disadvantage that the more samples are available, the more interactions are needed to UpMF to converge, and the more computational time is required in an online experiment. As it is possible to see in Fig. KSEMF is slower for short students' histories but constant over time. This allows a better evaluation of the characteristics of the machine required to run the algorithm. This information is particularly important for systems where the algorithm interacts with other processes that require additional computational time like systems that combine different classifiers to take their decision [18, 19].

Our additional purpose is to develop a real time predictor, so that it could be use to sequence also hints and feedbacks that occurs at event level. According to Human Computer Interaction literature, 0.1s is the maximal reaction time for a perceived real time system [34]. Given Fig. 7.4, we could think that both KSEMF and UpMF would allow real time interactions. However, UpMF algorithm exceeds the 0.1s when access to a DB is required. In Fig. 7.5 we can see a small test that should give a taste of the time performances required if UpMF is running on an online experiment with read and write DB operations. In such cases a DB is a requirement since not all model parameters can be stored in memory and consequently more time needs to be taken into account. Particularly demanding is the extraction with a query of the entire history of one student requested as input parameter for Alg. 3. The DB generally possesses a caching system, i.e. the last requested queries' results are kept in the cache until the available space is required for something else. Since we simulated the DB interaction with only one student we deleted the cache after 20 interactions. As a result, each time a query is made which is not in the cache a peak is registered. Such peaks appears more often when more students interact with the system. KSEMF does not require these demanding DB accesses to extract the entire student's history since it uses only information of the current time step to predict the next one.
In conclusion, in addition to reduced RMSE, KSEMF would have delivered reduced computational requirements if implemented in the large experiment described in this Chapter.

7.5 VPS with adaptive Threshold

The VPS has an advantage in comparison to other state of the art methods because it does not require a detailed analysis of the skills involved. Nevertheless, some steps were required for the VPS to be integrated within a learning platform, as described by Sec. 7.2. In Sec. 7.2.3 we present an example about how y_{th} could be selected for a large commercial ITS. Nevertheless, this approach is far from being optimal.
An automatic way to adapt y_{th} was suggested in [18, 19, 20]. As y_{th} determines the score we want ideally a student to obtain, it is easy to understand that this value could effect, not only if the student is indeed in the ZPD, but also his perceived experience. As a matter of fact, students feel differently over- or under-challenged and reaction to failures is generally different for every person. Therefore, the selection of a unique y_{th} is a pragmatic solution, but not the best one. As shown in Fig. 7.6, the idea presented by Janning et al. hypothesize that from audio perceived task difficulty recognition could be retrieved and used to adapt y_{th}.

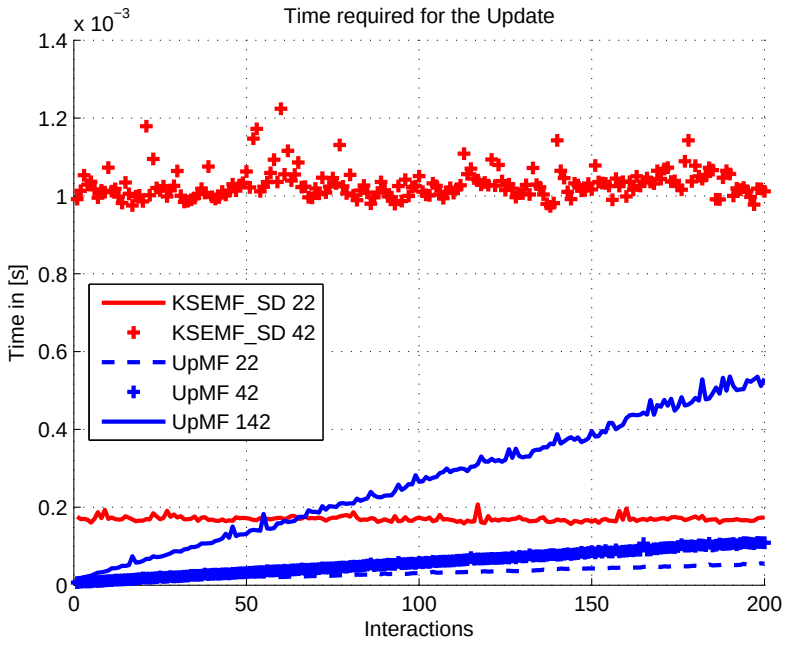

Figure 7.4: Time required on a laptop with an Intel Core i5 CPU (2.6GHz) and 8GB RAM without DB accesses for updating one student's model by UpMF (blue) and KSEMF_SD (red).

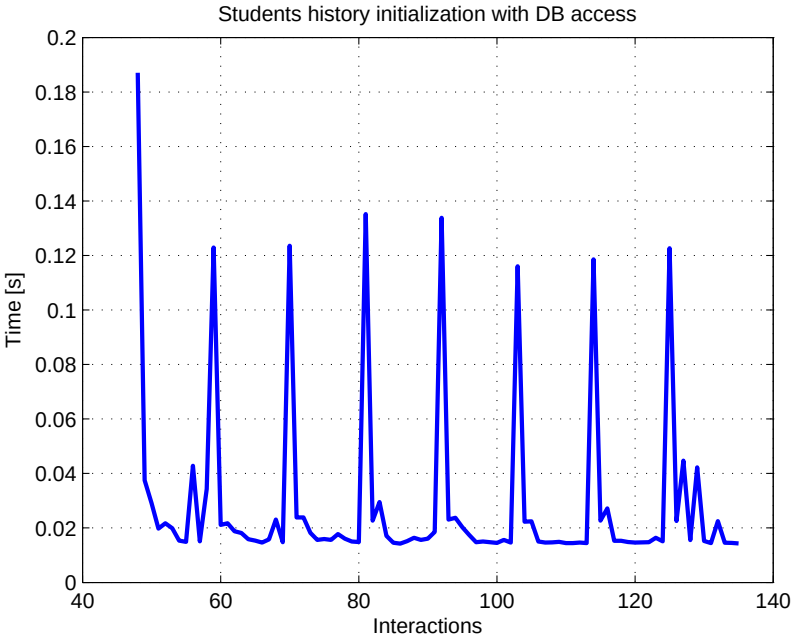

Figure 7.5: Time required to extract one student's history form a DB with 926000 lines on a laptop with an Intel Core i5 CPU with 2.6GHz and 8GB RAM.

Once it is determined that the student was over challenged by the task, one could, for instance, increase the y_{th} so that easier tasks are proposed. Similarly, if the tasks are perceived as too easy, the y_{th} values should be decreased.

7.6 Conclusions

In this Chapter we presented first a possible architecture for a minimal invasive ML integration in ITS and then how we applied it to integrate the VPS into a commercial ITS.
By presenting the steps of ML experiments and requirements we showed how ML sequencers could be adapted to the current available frameworks. Moreover, we reduced the integration effort for potential ML consumers proposing sequencers as Learning Analytics Services. The latter allows an easy management of the ML methods by the server side and a not binding trial opportunity for the client. In addition, the novel proposed method allows parallel exploitation of ML-powered sequencers and other ML-based modules by different ITS. With this lightweight integration method we want to increase the possibility of integration in large systems, especially those with a high number of contents and users.

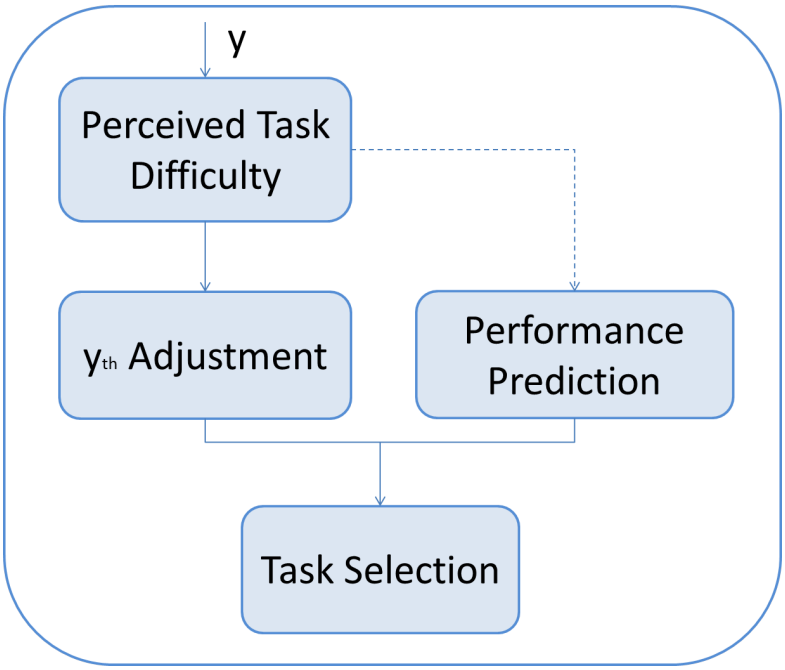

Figure 7.6: [18, 19, 20], VPS with adaptive threshold

In order to prove that our lightweight integration method and the VPS work, we integrated the VPS in a commercial ITS analyzing its potential from different perspectives. Although there was a high standard deviation due to the small sample, results are promising, showing how the sequencer was able to better model the students'knowledge. This result is coherent with the results of the questionnaire and the post test. The latter tests also show that the VPS is not damaging learning and that children had an experience comparable with the SotA sequencer, a sequencer that was modeled by experts over the years.

Chapter 8

Conclusions and Future Work

Contents

8.1 Achieved Results 114
8.2 Future Work 116

In this thesis we focused on algorithms for task sequencing in ITS, with the final goal to allow students proceeding through tasks without the help of human teachers. We worked on algorithms used for Educational Data Mining extending them to Learning Analytics.
In particular we proposed for each student a sequence of tasks adapted to his/her needs and that maximizes his/her learning amount and we then extend Performance Prediction developing Progress Modeling, that allows modeling the state of the student in a meaningful way over time.
Many of the results presented in this work were achieved thanks to the FP7 EU project called "Talk, Tutor, Explore, Learn: Intelligent Tutoring and Exploration for Robust Learning"–iTalk2Learn (grant no. 318051), where the University of Hildesheim was the coordinator. As such, we contributed to the main goal of the project, which aimed at building an intelligent platform able to collect, analyze and adapt to student's data and therefore ameliorate current state of the art of ITS.
The goal of developing "new methods for automatic intervention selection" is fulfilled here by considering sequencing as intervention method with Progress Modeling as a way to deliver such intervention without detrimental data collections or too extensive domain information. In achievement of statistical significance, the

sequencer was evaluated with a large amount of students and was able to sequence the rich structured contents selected.
Moreover, thanks to the collaboration with other authors, we contributed to the integration of the VPS with other machine learning algorithms empowered with additional information, such as speech, that allows ameliorating the algorithm performances.

8.1 Achieved Results

Hereafter we list the achieved results.

Chapter 4 First, in Chapter 4, we discussed the feasibility evaluation of domain independent Matrix Factorization applied in Intelligent Tutoring Systems for Performance Prediction. As a result, we show that, when using Matrix Factorization, task IDs, student IDs, and scores can be used to obtain a prediction for dataset not possessing the level of detail that benchmark datasets of the area have. There we underlined the necessity of having an time evolving algorithm to model students' knowledge acquisition. Therefore we adapted and evaluated domain independent update methods for online learning Matrix Factorization. As a consequence, the algorithms designed for Data Mining purposes were extended to work with the online learning problem in a Learning Analytics context.

Chapter 5 In Chapter 5, a novel Matrix Factorization update method based on Kalman Filters approach is presented in two variations, KSEMF and KSEMF_SD, which have two different updating functions: (1) a simple one considering the task just seen, and (2) another one, which is aware of the skills' deficiency of the students.
KSEMF and KSEMF_SD implement Progress Modeling as amelioration of domain independent Performance Prediction and allow showing the evolution of the students over time in a plausible way. We demonstrated that it is possible to give a specific interpretation to latent features which represents the state of the student and the characteristics of a task.
KSEMF and KSEMF_SD also showed appealing properties in comparison to other potential domain independent progress modeler. First, the algorithm requires less resources as the entire student's history is not necessary to compute the updated latent features. Then, the algorithm is still domain independent because the tagged skills of the tasks are not used to deliver a score prediction. Finally, KSEMF_SD reduced the prediction error and is less sensitive to the lack of data. In conclusion,

in this work we showed that Recommender Systems and Kalman Filters can be successfully combined.

Chapter 6 In Chapter 6 a new environment for offline testing of machine learning controlled sequencers is proposed. We designed a simulated environment composed by simulated students and tasks with continuous knowledge and score representation and different difficulty levels. Moreover, we presented an alternative to Reinforcement Learning for tasks sequencing. The VPS approach does not require detrimental data collection for users and extensive authoring effort. We showed that MF is able dealing with the most actual problems of ITS, like time and personalization, retrieving automatically skills required and difficulty. Then, we proposed VP, a performance based policy that does not require direct input of domain information, and a student simulator that helps in preliminary off-line evaluation. The designed system achieved time gain over random and in range policy in almost each scenario and is robust to noise. This demonstrates how the sequencer could solve many engineering/authoring efforts. Preliminarily to an online experiment we presented a discussion on feasibility and utility of the algorithm. In conclusion, to use the VPS, theoretically no content analysis is required, since the MF will reconstruct the domain information, thanks to continuous score representation.

Chapter 7 In Chapter 7 we presented first a possible architecture for a minimal invasive ML integration in ITS and then how we applied it to integrate the VPS into a commercial ITS. The design of a minimal invasive API for the lightweight integration of ML components in larger systems aims at minimizing the risk of integration and the cost of expertise transfer. The API allowed integration of the developed sequencer in a large commercial ITS, that could not allow the effort of a invasive integration. By presenting the steps of ML experiments and requirements we showed how ML sequencers could be adapted to the current available frameworks. Moreover, we reduced the integration effort for potential ML consumers proposing sequencers as Learning Analytics Services. The latter allows an easy management of the ML methods by the server side and a not binding trial opportunity for the client. In addition, the novel proposed method allows parallel exploitation of ML-powered sequencers and other ML-based modules by different ITS. With this lightweight integration method we want to increase the possibility of integration in large systems, especially those with a high number of contents and users. In addition, the proposed service approach could be used for ML applications different than Learning Analytics.

In the same Chapter a large scale evaluation of the designed sequencer is proposed. The large scale evaluation ran in a commercial system with 100 users and one month. The sequencer proved to have comparable learning gains and perceived experience results with those of the ITS sequencer, which was designed over the years by experts. In addition, the VPS proved to have better modeling abilities, so that the students could proceed faster through the curriculum. This is an appealing property for companies that develop ITS, since their goal is to prove not only the best learning gains but also that they can be obtained in a reduced amount of time in comparison to the other systems.

8.2 Future Work

The presented work is suitable for several extensions.

Student Modeling Chapter 5 presented kn, a way to show that the norm of the student latent features increases over time. The latter could be mapped with the real knowledge evolution with the final goal to deliver an effortless analysis tool to teachers and developers. In order to do so, an idea could be to apply the same approach to the contents' latent features associating the normed sum of the latent features with the estimated difficulty level of a task. It would be easier to map the from the algorithms retrieved curve to the available tasks' domain information, rather than trying to map predicted and real students' knowledge. Once this is achieved, the same approach could be followed to determine the student's knowledge.
With respect to KSEMF approach one could extend the method by learning the functions computing the next state from the previous state and the control values, as well as the matrix computing the next observed measure form the current state estimate. Another possible extension consists in ameliorating the model initialization, which seems to be one of the reasons why the algorithms have not so good performances in the first iterations. This amelioration would also reduce the cold start problem, which occurs by using personalized models. Other methods for the solution of the cold start problem could be used such as Transfer Learning, Active Learning, etc.

Sequencing As shown by the simulated environment, it is plausible to believe that an ameliorate performance predictor could deliver better performances. Therefore, one could extend the VPS by integrating it with KSEMF Progress Modeling. As suggested by [18, 19] the VPS could be further ameliorated by integrating a Performance Predictor analyzing other aspects of the student's state

such as the perceived task difficulty.
In the large scale experiments another potential extension is shown by considering the performances of UpMF that requires the entire student's history to update the latent features. Therefore, KSEMF should not only reduce the error but also the computational requirements of the algorithm. Finally, another extension is required to sequence different topics. As shown in Chapter 7, after the experiment the students that practiced with the VPS sequencer proceeded faster through the curriculum, but had in the end an unbalanced profile, i.e. they had a different amount of knowledge in the different topics.

References

[1] Abernethy, J., Canini, K., Langford, J., and Simma, A. Online collaborative filtering. 32, 41, 43, 53, 81

[2] Baker, D., Ryan, S., Corbett, A. T., and Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*, pages 406–415. Springer. 26, 28, 39

[3] Baker, R., Pardos, Z., Gowda, S., Nooraei, B., and Heffernan, N. (2011). Ensembling predictions of student knowledge within intelligent tutoring systems. *User Modeling, Adaption and Personalization*, pages 13–24. 29

[4] Beck, J., Woolf, B. P., and Beal, C. R. (2000). Advisor: A machine learning architecture for intelligent tutor construction. *Association for the Advancement of Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 552–557. 35

[5] Brusilovsky, P. (2001). Adaptive hypermedia. *User modeling and user-adapted interaction*, pages 87–110. 92

[6] Cen, H., Koedinger, K., and Junker, B. (2006). Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer. 28, 29

[7] Chang, K.-m., Beck, J., Mostow, J., and Corbett, A. (2006). A bayes net toolkit for student modeling in intelligent tutoring systems. In *International Conference on Intelligent Tutoring Systems*, pages 104–113. Springer. 39

[8] Chen, Z. (2003). Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, pages 1–69. 33

[9] Chi, M., Koedinger, K. R., Gordon, G. J., Jordon, P., and VanLahn, K. (2011a). Instructional factors analysis: A cognitive model for multiple instructional interventions. 29

[10] Chi, M., VanLehn, K., Litman, D., and Jordan, P. (2011b). Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *UMAI*. 17, 21, 35, 94

[11] Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. (2009). *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. ohn Wiley & Sons. 30

[12] Corbett, A. and Anderson, J. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *UMAI*. 26, 40, 46

[13] Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L., and Nejdl, W. (2012). Real-time top-n recommendation in social streams. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 59–66. ACM. 32

[14] Feng, M., Hansen, E. G., and Zapata-Rivera, D. (2009). Using evidence centered design for learning (ecdl) to examine the assistments system. In *annual meeting of the American Educational Research Association (AERA), San Diego, California*. 18

[15] Gong, Y., Beck, J. E., and Heffernan, N. T. (2010). Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *International Conference on Intelligent Tutoring Systems*, pages 35–44. Springer. 28, 29

[16] Guàrdia-Sebaoun, E., Guigue, V., and Gallinari, P. (2015). Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 281–284. ACM. 31

[17] Gutiérrez, S., Pardo, A., and Kloos, C. D. (2006). A modular architecture for intelligent web resource based tutoring systems. In *International Conference on Intelligent Tutoring Systems*. 93, 95, 98

[18] Janning, R., Schatten, C., and Schmidt-Thieme, L. (2014a). Feature analysis for affect recognition supporting task sequencing. In *European Conference for Technology-Enhanced Learning*. xi, 108, 109, 112, 116

[19] Janning, R., Schatten, C., and Schmidt-Thieme, L. (2014b). Multimodal affect recognition for adaptive intelligent tutoring systems. In *Workshop on Feedback from Multimodal Interactions in Learning Management Systems at the International Conference of Educational Data Mining*. xi, 108, 109, 112, 116

[20] Janning, R., Schatten, C., and Schmidt-Thieme, L. (2016). Perceived task-difficulty recognition from log-file information for the use in adaptive intelligent tutoring systems. *International Journal of Artificial Intelligence in Education*, pages 1–22. xi, 29, 109, 112

[21] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45. 32, 46

[22] Koedinger, K., Pavlik, P., Stamper, J., Nixon, T., and Ritter, S. (2011). Avoiding problem selection thrashing with conjunctive knowledge tracing. In *International Conference of Educational Data Mining*. 26, 28, 34, 75

[23] Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. *Advances in neural information processing systems*, 12:1008–1014. 36

[24] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37. 30

[25] Krohn-Grimberghe, A., Busche, A., Nanopoulos, A., and Schmidt-Thieme, L. (2011). Active learning for technology enhanced learning. In *Towards Ubiquitous Learning*, pages 512–518. Springer. 74, 75

[26] Lee, J. I. and Brunskill, E. (2012). The impact on individualizing student models on necessary practice opportunities. *International Conference on Educational Data Mining*. 28

[27] Leszczenski, J. M. and Beck, J. E. (2007). Whats in a word? extending learning factors analysis to model reading transfer. In *13th International Conference on Artificial Intelligence in Education, Educational Data Mining Workshop*. 29

[28] Li, B., Zhu, X., Li, R., Zhang, C., Xue, X., and Wu, X. (2011). Cross-domain collaborative filtering over time. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2293–2298. Association for the Advancement of Artificial Intelligence Press. 31

[29] Malpani, A., Ravindran, B., and Murthy, H. (2011). Personalized intelligent tutoring system using reinforcement learning. In *FLAIRS*. 35, 36

[30] Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., and Koper, R. (2011). Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer. 46, 92

[31] Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A. M., and Gama, J. (2015). Forgetting methods for incremental matrix factorization in recommender systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 947–953. ACM. 29, 32, 42, 43

[32] Mazziotti, C., Holmes, W., Wiedmann, M., Loibl, K., Rummel, N., Mavrikis, M., Hansen, A., and Grawemeyer, B. (2015). Robust student knowledge: Adapting to individual student needs as they explore the concepts and practice the procedures of fractions. *Intelligent Support in Exploratory and Open-ended Learning Environments Learning Analytics for Project Based and Experiential Learning Scenarios*, page 32. ix, 3, 23, 34, 35

[33] Mühlenbrock, M., Tewissen, F., Hoppe, U., et al. (1998). A framework system for intelligent support in open distributed learning environments. *International Journal of Artificial Intelligence in Education (IJAIED)*, 9:256–274. 93

[34] Nielsen, J. (1994). *Usability engineering*. Elsevier. 12, 32, 109

[35] Pardos, Z. A. and Heffernan, N. T. (2010). Modeling individualization in a bayesian networks implementation of knowledge tracing. In *User Modeling, Adaptation, and Personalization*. Springer. 28

[36] Pardos, Z. A. and Heffernan, N. T. (2011). Kt-idem: introducing item difficulty to the knowledge tracing model. In *User Modeling, Adaptation, and Personalization*, pages 243–254. Springer. 28

[37] Pavlik, P., Cen, H., and Koedinger, K. (2009). Performance factors analysis-a new alternative to knowledge tracing. In *Artificial Intelligence in Education*. 29, 46

[38] Pero, Š. and Horváth, T. (2015). Comparison of collaborative-filtering techniques for small-scale student performance prediction task. In *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*, pages 111–116. Springer. 75, 89

[39] Pilászy, I. and Tikk, D. (2009). Recommending new movies: Even a few ratings are more valuable than metadata. In *RecSys*. 20, 53, 62, 89

[40] Rendle, S. and Schmidt-Thieme, L. (2008). Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 251–258. ACM. 32, 41, 42, 43, 53, 81, 102, 108

[41] Sarma, B. S. and Ravindran, B. (2007). Intelligent tutoring systems using reinforcement learning to teach autistic students. In *Home Informatics and Telematics: ICT for The Next Billion*, pages 65–78. Springer. 36

[42] Schatten, C., Janning, R., and Schmidt-Thieme, L. (2014a). Vygotsky based sequencing without domain information: A matrix factorization approach. In *Computer Supported Education*, pages 35–51. Springer. 9, 72

[43] Schatten, C., Janning, R., and Schmidt-Thieme, L. (2015). Integration and evaluation of a machine learning sequencer in large commercial its. In *Association for the Advancement of Artificial Intelligence 2015*. Springer. 8, 9, 12, 22, 32, 34, 37, 41, 42, 65, 93

[44] Schatten, C., Mavrikis, M., Janning, R., and Schmidt-Thieme, L. (2014b). Matrix factorization feasibility for sequencing and adaptive support in its. In *International Conference of Educational Data Mining*. 8, 21, 37, 74, 102

[45] Schatten, C. and Schmidt-Thieme, L. (2014). Adaptive content sequencing without domain information. In *International Conference on Computer Supported Education*. 9, 19, 38, 40, 51, 68, 72, 100

[46] Schatten, C. and Schmidt-Thieme, L. (2016a). Hybrid matrix factorization update for progress modeling in intelligent tutoring systems. *Communications in Computer and Information Science*. 8, 47

[47] Schatten, C. and Schmidt-Thieme, L. (2016b). Student progress modeling with skills deficiency aware kalman filters. In *International Conference on Computer Supported Education*. 8, 47

[48] Schatten, C., Wistuba, M., Schmidt-Thieme, L., and Gutirrez-Santos, S. (2014c). Minimal invasive integration of learning analytics services in its. In *International Conference on Advanced Learning Technologies*. 9, 93, 103

[49] Schilling, N., Wistuba, M., Drumond, L., and Schmidt-Thieme, L. (2015). Joint model choice and hyperparameter optimization with factorized multilayer perceptrons. In *IEEE 27th International Conference on Tools with Artificial Intelligence*, pages 72–79. IEEE. 54

[50] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. Cambridge Univ. Press. 36

[51] Thai-Nghe, N., Drumond, L., Horvath, T., Krohn-Grimberghe, A., Nanopoulos, A., and Schmidt-Thieme, L. (2011). Factorization techniques for predicting student performance. *Educational Recommender Systems and Technologies: Practices and Challenges. IGI Global*. 30

[52] Thai-Nghe, N., Drumond, L., Horvath, T., and Schmidt-Thieme, L. (2012). Using factorization machines for student modeling. In *User Modeling, Adaptation, and Personalization Workshops*. 30

[53] Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., and Schmidt-Thieme, L. (2010). Recommender system for predicting student performance. *Procedia Computer Science*, 1(2):2811–2819. xiii, 12, 20, 30, 38, 39, 40, 46, 53

[54] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press. 33, 49

[55] Vinagre, J., Jorge, A. M., and Gama, J. (2014). Fast incremental matrix factorization for recommendation with positive-only feedback. In *User Modeling, Adaptation, and Personalization*, pages 459–470. Springer. 13, 15, 16, 32, 41, 43, 53, 81

[56] Voss, L., Schatten, C., and Schmidt-Thieme, L. (2015). A transfer learning approach for applying matrix factorization to small its datasets. In *International Conference on Educational Data Mining*. 3, 15, 20, 21, 23, 29, 53, 54, 62, 89, 90

[57] Vygotsky, L. L. S. (1978). *Mind in society: The development of higher psychological processes*. HUP. 51, 74

[58] Wang, Y. and Heffernan, N. (2011). Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes. 19, 29

[59] Wang, Y., Ostrow, K., and Heffernan, N. (2016). Partial credit revisited: Enhancing the efficiency and reliability of group differentiation at scale. *Student Modeling from Different Aspects*, page 22. 28, 29

[60] Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015). Sequential model-free hyperparameter tuning. In *2015 IEEE International Conference on Data Mining*, pages 1033–1038. IEEE. 54

[61] Witt, M. (2014). *Primary Mathematics for Trainee Teachers*. Learning Matters. 21

[62] Xiong, L., Chen, X., Huang, T.-K., Schneider, J. G., and Carbonell, J. G. (2010). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *International Conference on Data Mining*, pages 211–222. SIAM. 31

[63] Xu, Y. and Mostow, J. (2013). Using item response theory to refine knowledge tracing. In *International Conference of Educational Data Mining*, pages 356–357. 28