DISS. ETH NO. 24505

Sensor-free learner models for trait discovery and identification in intelligent tutoring systems

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
Severin Achill Klingler

Master of Science ETH in Computer Science, ETH Zurich
born on 05.06.1987
citizen of Gossau (SG), Switzerland

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. Andreas Krause, co-examiner
Dr. Tanja Käser, co-examiner

2017

Abstract

Intelligent tutoring systems (ITS) are gaining importance in education, as they can individualize the training for each learner. The individualization relies on the student model that infers cognitive, metacognitive or affective states of the learner based on the history of interactions with the system. Research on student models has mainly focused on how to better model student knowledge and student learning, but recently, models of metacognitive and affective states, as well as automatic assessments of students, has gained interest. In this thesis, we present data-driven models for the identification and discovery of student traits. We focus on sensor-free models relying entirely on interaction logs.

First, we present a completely supervised pipeline for integrating automatic assessment seamlessly into an ITS and apply the method to the case of developmental dyscalculia (DD). We demonstrate that interaction logs provide enough information to identify children at risk of DD with high accuracy. In addition, we demonstrate test validity and reliability comparable to traditional assessments. Our model is able to adapt the duration of the assessment to the individual child and can classify a child at risk of DD with an accuracy of 91% after 11 minutes on average.

Second, we present a semi-supervised classification pipeline that makes effective use of unlabeled data to significantly improve model quality for detecting student characteristics. We employ deep variational auto-encoders to learn efficient feature embeddings that improve the performance of standard classifiers by up to 28%. We demonstrate that our method outperforms previous methods for finding efficient feature embeddings and generalizes better to imbalanced data sets compared to expert features. The method provides the ability to improve performance on a variety of classification tasks in educational data mining.

Third, we propose an unsupervised evolutionary clustering pipeline to discover student characteristics. Our pipeline can be used as a black box for any ITS and is optimized to improve cluster stability over multiple training sessions in the presence of noise. Our model selection is designed such that relevant cluster evolution effects can be captured. Our method outperforms previous work regarding clustering performance and stability on synthetic data. We demonstrate that the

clustering pipeline is able to detect interesting student behavior and properties of learning environments based on data from real-world ITS.

Zusammenfassung

Intelligente Tutorensysteme (ITS) werden für die Bildung immer wichtiger, da sie Lernenden individualisierte Trainingsmöglichkeiten bieten. Die Individualisierung basiert auf dem Studentenmodell, welches mit Hilfe von Interaktionsdaten Rückschlüsse auf die kognitiven, meta-kognitiven und emotionalen Zustände des Lernenden bzw. der Lernenden zieht. Forschung im Bereich der Studentenmodelle konzentrierte sich bisher vor allem auf der Verbesserung der Repräsentation vom Wissensstand und von Lerneffekten. In letzter Zeit wurden meta-kognitive und emotionale Modelle sowie die automatische Beurteilung von Benutzereigenschaften wichtiger. Diese Dissertation beschreibt datenorientierte Modelle für das Erkennen und Entdecken von Charakteristiken von Lernenden basierend auf Log-Dateien.

Als erstes stellen wir eine überwachte Pipeline zum automatischen und integrierten Erkennen von Benutzereigenschaften vor und wenden sie für den Fall von Dyskalkulie an. Wir zeigen, dass Interaktionsdaten genügend Informationen enthalten um Kinder mit Dyskalkulie mit grosser Genauigkeit und mit einer zu Standardtests vergleichbaren Validität und Reliabilität zu erkennen. Unser Modell passt die Dauer des Tests individuell dem Kind an und klassifiziert Kinder mit Dyskalkulie mit einer Genauigkeit von 91% nach durchschnittlich 11 Minuten.

Zweitens stellen wir eine teilüberwachte Klassifikationspipeline vor welche Daten ohne Kennsatz verwendet um die Qualität der Detektion von Benutzereigenschaften signifikant zu verbessern. Wir benutzen Deep Variational Autoencoder um effiziente Datenrepräsentation zu finden, welche die Leistung von Standardklassifikatoren um bis zu 28% erhöht. Unsere Methode übertrifft vorherige Methoden für das Finden von Datenrepräsentation und generalisiert besser auf unausgeglichene Datensätze im Vergleich zu anderen Methoden.

Drittens schlagen wir eine Pipeline zur evolutionären Clusteranalyse zum Entdecken von Benutzereigenschaften vor. Unsere Pipeline kann als Blackbox mit jedem ITS verwendet werden und wurde für eine verbesserte Stabilität der Cluster optimiert zudem erkennt unser Modelauswahlprozess Clusterentwicklungseffekte automatisch. Auf künstlich erzeugten Daten zeigt unsere Methode eine bessere Leistung und stabilere Resultate als vorherige Modelle. Weiter zeigen wir, dass

unsere Methode mit Daten von realen ITS zum Erkennen von interessanten Benutzereigenschaften verwendet werden kann.

Acknowledgments

First of all, I would like to thank my advisor Prof. Markus Gross who convinced me to pursue a Ph.D. at the Computer Graphics Laboratory in the first place. His vision to create intelligent software that truly adapts to children with special needs sparked my interest for the exciting field of educational data mining. I am very grateful for the opportunity to continue the research on learning environments at the lab. His unconditional support, trust, and scientific guidance were an invaluable help throughout my Ph.D.

I am very grateful for all the support of Dr. Tanja Käser, who introduced me to the scientific world already during my master thesis. Her extensive knowledge about the area of tutoring systems allowed me to navigate these waters much more efficiently. I am happily looking back to many exciting discussions with her that challenged my ideas and allowed me to put my work into the greater context of computer-based education. I am sincerely thankful to Dr. Barbara Solenthaler for her enduring support, advice and scientific intuition during my Ph.D. Her ability to ask the right questions at the right time helped me to focus my work time and again and allowed me to stay on track.

Furthermore, I would like to thank all my collaborators for their support. Dr. Alberto Giovanni Busetto helped me with many technical aspects of the thesis. I am very thankful to Prof. Dr. von Aster and Dr. Juliane Kohn for the offered expertise in study design and psychological aspects of this work. I would like to thank Christian Vögeli, Felix Fontein and all the other coworkers at Dybuster AG, who helped me with many technical questions and provided invaluable support to run the study in summer 2014. I wish them all the best for their future, and I hope their adaptive learning systems will take the world by storm.

A special thank goes to all current and former members of CGL, IGL, and DRZ. You all made the lab a great place to work. I never worked at a place where discussions were more open-minded and genuinely exciting and fun.

I am deeply thankful to my family and friends. You gave me the perspective and support to go through ups and downs of this exciting time. Lastly, I am most thankful to you, Salomé, for your unconditional love, your understanding, and your ability to make my world a better place.

This work was supported by ETH Research Grant ETH-23 13-2.

Contents

Abstract iii

Zusammenfassung v

Acknowledgements vii

Contents ix

List of Figures xiii

List of Tables xv

Introduction 1
- 1.1 Overview . . . 4
- 1.2 Principal Contributions . . . 7
- 1.3 Thesis outline . . . 10
- 1.4 Publications . . . 10

Student model landscape (related work) 13
- 2.1 Cognitive student models . . . 15
  - 2.1.1 Hybrid models . . . 16
  - 2.1.2 Student model properties . . . 16
- 2.2 Models of student characteristics and traits . . . 17
  - 2.2.1 Student trait identification . . . 18
  - 2.2.2 Student trait discovery . . . 20

Data 23
- 3.1 Orthograph . . . 23
- 3.2 Calcularis . . . 25

Performance characteristics of student knowledge models 29
- 4.1 Investigated Models . . . 30
  - 4.1.1 Bayesian Knowledge Tracing . . . 30
  - 4.1.2 Item Response Theory . . . 32
  - 4.1.3 Latent Factor Knowledge Tracing . . . 32

4.1.4 Feature Aware Knowledge Tracing . . . 33
4.2 Synthetic data generation . . . 34
4.3 Experimental Setup . . . 36
4.4 Results . . . 37
4.4.1 Error Metrics . . . 37
4.4.2 Model Comparison . . . 38
4.4.3 Parameter Influence . . . 42
4.5 Discussion . . . 44

Supervised student trait identification **49**
5.1 Adaptive Classification Algorithm . . . 50
5.1.1 Feature extraction . . . 51
5.1.2 Feature selection . . . 53
5.1.3 Probabilistic classifier . . . 54
5.1.4 Feature ordering . . . 55
5.1.5 Stopping criterion . . . 55
5.2 Experimental Evaluation . . . 56
5.2.1 Method . . . 56
5.2.2 Content validity . . . 58
5.2.3 Criterion-related validity . . . 59
5.2.4 Construct validity . . . 60
5.2.5 Reliability . . . 61
5.2.6 Test duration . . . 61
5.3 Generalization Capabilities . . . 62
5.3.1 Method . . . 63
5.3.2 Feature generalizability . . . 65
5.3.3 Performance on classroom data . . . 65
5.4 Discussion . . . 66

Semi-supervised student trait identification **69**
6.1 Background . . . 70
6.1.1 Auto-encoder . . . 71
6.1.2 Variational auto-encoder . . . 72
6.2 Method . . . 72
6.2.1 Student snapshots . . . 74
6.2.2 Simple student auto-encoder (S-SAE) . . . 74
6.2.3 CNN student auto-encoder (CNN-SAE) . . . 75
6.2.4 Feature selection . . . 76
6.2.5 Semi-supervised classification pipeline . . . 76
6.3 Results . . . 77
6.3.1 Experimental Setup . . . 78
6.3.2 Implementation . . . 79

6.3.3 Network comparison . . . 80
6.3.4 Classification performance . . . 81
6.3.5 Comparison to our specialized models . . . 85
6.3.6 Robustness on sample size . . . 85
6.4 Discussion . . . 86

Unsupervised student trait discovery **89**
7.1 Method . . . 90
7.1.1 Action Sequences . . . 91
7.1.2 Action Processing . . . 91
7.1.3 Similarity Computation . . . 92
7.1.4 Clustering . . . 92
7.1.5 Model Selection . . . 93
7.2 Synthetic experiments . . . 95
7.2.1 Experimental setup . . . 95
7.2.2 Clustering Quality & Robustness . . . 97
7.2.3 Stability . . . 99
7.2.4 Interpretability . . . 99
7.3 Exploratory data analysis . . . 102
7.3.1 Experimental Setup . . . 102
7.3.2 Navigation Behavior . . . 102
7.3.3 Input & Help-Seeking Behavior . . . 105
7.4 Discussion . . . 108

Conclusion **111**
8.1 Limitations & Future work . . . 112

Instruments **117**

References **121**

List of Figures

1.1 Thesis overview 6

2.1 Principle components of an ITS 14

3.1 The intelligent tutoring system Orthograph 25
3.2 The intelligent tutoring system Calcularis 28

4.1 Bayesian knowledge tracing and Rasch model 31
4.2 Latent factor knowledge tracing and feature-aware knowledge tracing 33
4.3 Combined model for data sampling 35
4.4 Performance for predicting task outcomes of different student models 40
4.5 Performance for predicting knowledge states of different student models 41
4.6 Relative performance of student models 43
4.7 Influence of sampling parameters on model performance 45

5.1 Pipeline for supervised student trait identification 51
5.2 Features for detecting developmental dyscalculia 58
5.3 Performance for detecting developmental dyscalculia 59
5.4 Test duration for detecting developmental dyscalculia 62
5.5 Feature generalizability 64
5.6 Test scores of misclassified children 66

6.1 Network layout of our variational auto-encoders 73
6.2 Semi-supervised classification pipeline 77
6.3 Comparions of variational auto-encoders 81
6.4 Classification performance for different feature embeddings 83
6.5 Robustness of feature embeddings 86

7.1 Overview of our clustering pipeline 91
7.2 Overview of synthetic data generation 96
7.3 Comparison of clustering methods 98
7.4 Extracting temporally stable clusters 100
7.5 Simulated examples of four types of cluster events 101

7.6 Navigation behavior in Orthograph 103
7.7 Analyzing the navigation behavior 104
7.8 Markov Chains for the *Input Behavior* and the *Help-Seeking Behavior* in *Orthograph* and *Calcularis*. 105
7.9 Clusters of Input and Help-Seeking Behavior 106

List of Tables

4.1 Best and worst case performance of student models. 39

5.1 List of extracted features 52
5.2 Analysis of convergent and divergent validity 60

6.1 Performance comparison of different feature embeddings 84

CHAPTER 1

Introduction

As our world is becoming increasingly complex, the amount of knowledge we need for navigating everyday life has increased over the past decades. For this reason, efficient methods to acquire new knowledge are necessary. Efficient teaching, however, not only requires a deep understanding of the subject taught but requires a thorough understanding of the diverse learning needs of students as well. Teaching and learning are greatly impacted by the way students process information, their overall approach to learning (surface, deep, strategic) and their attitude towards knowledge in general [Felder and Brent, 2005]. Educators, teachers, and tutors are specialized in identifying these needs and adjusting the training of students accordingly. Teaching decisions are as much based on the cognitive state of the learner (e.g. student's knowledge and skills) as they are based on the affective state (e.g. a student's motivation, attention, emotion) [Lepper and Chabay, 1988]. Student achievement was found to be heavily influenced by teacher-related factors such as provided feedback, the instructional quality, class environment and challenge of goals [Hattie, 2003]. Even non-professional peer tutoring was found to improve student achievement consistently and was shown to improve the attitude of students towards the subject taught [Cohen et al., 1982]. Tutoring was found to improve mean achievement scores by up to two standard deviations [Bloom, 1984] compared to learners that did not receive additional tutoring.

Computer-based tutoring provides an inexpensive alternative or extension to human tutoring. These systems provide a fear-free, engaging and playful environment for students to learn. The use of technology in classroom applications has a significant positive effect on student's achievement and

attitude [Schmid et al., 2014]. In particular, intelligent tutoring systems (ITS) have gained importance in education over the past years. ITS are computer-based tutors that model the learner's psychological state to provide individualized tutoring of students. ITS have proven to be more effective than instructions in large groups led by a teacher or computer-based instruction that was not based on a ITS [Ma et al., 2014]. Student learning can already be significantly increased by employing ITS for homework support only [Vanlehn et al., 2005]. Further, ITS have shown learning gains at least similar to individualized human tutoring or instructions within small groups [Ma et al., 2014; Woolf et al., 2009]. ITS prove to be robust and efficient instructional tools improving the performance of students significantly over conventional classes in multiple and diverse domains [Kulik and Fletcher, 2016].

Over the past decades a lot of research has been conducted in the area of ITS to represent and model student knowledge accurately, to design effective curricula or to develop optimal instructional policies. A major component of any ITS is the student model that infers cognitive, metacognitive or affective states of the learner based on the interactions with the system. The student model is used as the basis for teaching decisions within an ITS. However, modeling cognitive, metacognitive and affective states from user interactions events is challenging, since the way a learner interacts with an ITS provides only partial and ambiguous information on a student's true state [Conati and Kardan, 2013]. In particular, Woolf et al. [2013] formulated five grand challenges for AI in educational systems. Here we give a summary of the challenges:

1. **Mentors for every learner.** Learning science research shows that people have different needs when learning. Students differ in how they learn efficiently and how they keep engaged with a topic even when learning challenging concepts. To support every learner, ITS need to model the changes that occur in learners with respect to cognitive, metacognitive and affective states. Student models need to go beyond identifying the knowledge and skills mastered, but should represent students holistically, e.g. their preferences, goals, communicative competencies, behavior changes, affective self-regulation capabilities, etc. Further, to allow for the wide adoption of ITS methods need to be more generic as current student models often need to be adapted for every domain or application.

2. **Learning 21^{st} century skills.** Essential skills in the 21^{st} century include cognitive skills (such as systems thinking or critical thinking) as well as interpersonal skills (e.g. active listening, conflict resolution) and intrapersonal skills (e.g. adaptability, self-management).

Today, knowledge changes at high frequencies and teachers and learners need to constantly adapt. To prepare students for lifelong learning, technological advances are needed to develop alternative ways of teaching compared to the still predominant teacher-centered instructions. Technologies that can mentor students when solving complex and ill-structured problems are needed. Such methods should support students exploitative behavior and creativity.

3. **Interaction data to support learning.** ITS provide unique types of data sets collected from learners interacting with these systems. Over the past decades, large repositories of interaction data have accumulated. Data analysis should be generic and include multiple tutoring systems, classes, and games to evaluate the general competencies of a student. Mining of these data repositories should reveal the overall capabilities and needs of learners such as student motivation, learning difficulties, problem-solving strategies and capabilities for critical thinking, self-regulation and active listening. However, the effective use of these data sets to not only identify students that need special support but to help teachers understand the difficulties of students and to improve the curriculum are challenging goals. Efficient techniques need to draw from the fields of machine learning and data mining and these methods need to be adapted for the specific requirements of the educational domain.

4. **Universal access to global classrooms.** Global classrooms have the potential to significantly improve the way we traditionally learn by providing universal and inclusive access to learning resources at any time for all students. Global access to the latest and most efficient methods for learning a particular subject as well as providing an almost unlimited supply of other students to converse with has the potential to change how we learn. Today, massively open online courses (MOOCs) exhibit little content individualization and struggle from huge dropout rates. MOOCs have shown to be successful only for learners with high levels of initial knowledge and motivation. New techniques are needed to help students engage in these global classrooms and to overcome language and cultural issues. Methods for the automatic and reliable grading of student assessments and the improvement of learning content based on previous user behavior are needed to allow global classrooms to reach their full potential.

5. **Lifelong and lifewide learning.** AI technologies should allow

> learning over the entire life (lifelong) and across all aspects of life (lifewide). Today learning and education are not equivalent. Education takes place within strictly defined levels (school, college, professional development), in specific places (institutions, work), according to certain learning types (e.g. typical and special students). However, mobile technology has the potential to provide ubiquitous and seamless learning across these established boundaries. Indeed, as advances in learning technology enable learners to gain knowledge equally well outside of classrooms, boundaries between informal and formal learning can disappear. For this, we need to better understand how learning fundamentally occurs within humans. We need to support learners at a personal level and build upon personal strengths and interests. Virtual, AI-based characters can serve as lifelong learning companions and teachers. These characters can provide natural interfaces to tools that support lifelong and lifewide learning and can store detailed profiles of learners.

In this thesis, we contribute to the first and third grand challenge. Namely, we develop techniques to leverage interaction data from ITS to develop methods to support teachers and student models to adapt to personality traits and characteristics that are not ITS-specific. In this way, our data-driven models provide insights into student learning that go beyond more traditional students models that are concerned with modeling cognitive states only. More specifically, we develop frameworks for the identification of well-known student characteristics based on student interaction data. We present two different frameworks for supervised and semi-supervised prediction of student traits and evaluate these frameworks for the specific case of developmental dyscalculia. Further, we develop a clustering pipeline to help practitioners explore and discover new student characteristics, again based on the interaction data of students with an ITS. This work draws from different fields such as data mining, machine learning, learning sciences and research on intelligent tutoring systems to expand the student models to incorporate student traits.

1.1 Overview

This thesis describes the data-driven development of models for student traits and characteristics. From an initial analysis of performance characteristics of recent student models, over supervised and semi-supervised methods for student trait identification to unsupervised methods for exploring student characteristics.

Student model characteristics. We review the state-of-the-art in student modeling with a special focus on the most common student models: models of student knowledge and learning. We provide an in-depth analysis based on simulated data of two recent student models that unify and expand on previous models. We conclude that even state-of-the-art knowledge models provide only marginal performance gains for inferring student knowledge. This serves as the motivation to expand student models to include student characteristics and traits that are typically not directly observable within an intelligent tutoring system.

Models of student traits. There are three fundamental ways to incorporate student characteristics into student models of intelligent tutoring systems (ITS).

First, we can assess student characteristics offline in a separate test (e.g. a pre-test before the training). The results of such a traditional assessment can then provide parameters to configure the learning environment or the student models. While this approach allows for very detailed and high quality student profiles, traditional assessments are often time consuming and commonly have to be supervised by an expert, rendering them expensive in practice. Hence, this approach does not scale and is therefore not suitable in many cases, such as massive open online courses (MOOC), large university courses, or widespread screenings in elementary schools to enable early detection of learning disabilities.

Second, students can potentially be assessed based on the actions within a tutoring system. In this scenario we use machine learning techniques to classify student characteristics based on the observed behavior. However, it is unclear what types of student characteristics and traits can be predicted based on interaction data only. Compared to the first approach this method requires only a small set of manual assessments to allow the training of the classification method.

Third, instead of predicting the student characteristics explicitly we can implicitly model them by grouping students based on a similarity measure between students. This approach has the advantage that no labeled data is needed (which means that we do not need manual assessments), but this approach has the disadvantage that the captured student characteristics directly depend on the similarity metric used.

Sensor-free models. In this thesis we explore the second and third approach: data-driven machine learning methods for the detection and identification of student characteristics and traits. Building predictive models of student characteristics such as knowledge level, learning disabilities, per-

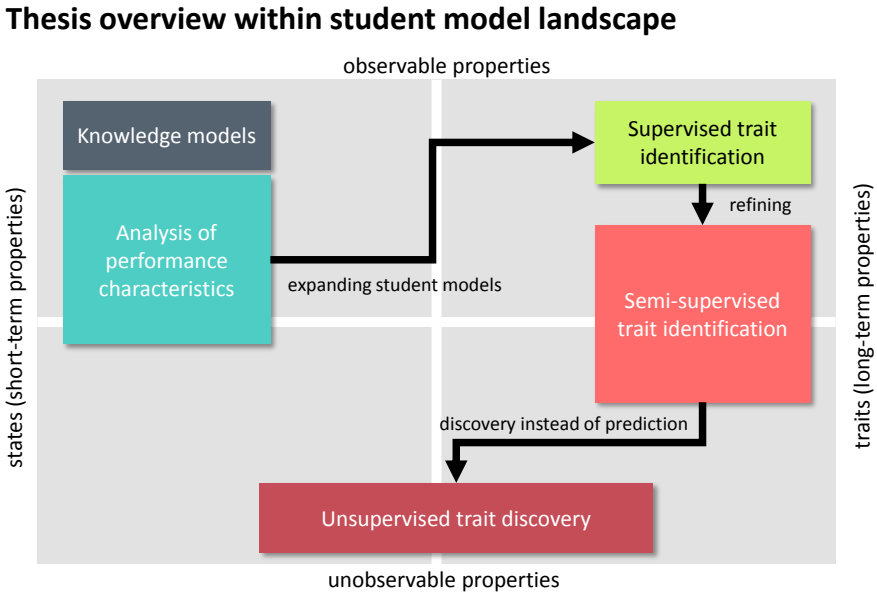


Figure 1.1: *Conceptual overview of the thesis. We categorize student models according to the duration of the property they model (state vs. trait) and to what degree the modeled property can be observed (observable vs. unobservable). Based on the analysis of performance characteristics of current cognitive student state models we develop supervised and semi-supervised methods for detecting student traits. In addition we present a unsupervised framework for the discovery of temporally coherent student characteristic.*

sonality traits or engagement is one of the big challenges in educational data mining (see the discussion of the grand challenges for AI in educational systems at the beginning of this chapter). Such detailed student profiles allow for a better adaptation of the curriculum to the individual needs and are crucial for fostering optimal learning progress. We focus on sensor-free models relying entirely on student interaction events gathered from log data. While sensor data (cameras, microphones, bio-sensors) allow for more detailed information about the student, recording such personal data often requires a specialized setup and can lead to privacy issues outside of controlled user studies. Today, most intelligent tutoring systems are already recording detailed information about all the interactions of students within the system such as keystrokes, mouse inputs, navigational actions, performance on tasks, etc. Therefore, sensor-free methods can be included into existing intelligent tutoring systems at relatively low cost as the relevant data is already available, making these methods widely applicable to different intelligent tutoring systems.

Figure 1.1 provides a conceptual overview of the work presented in this the-

sis. For this overview, we categorize student models according to the duration of the student property they model (state vs. trait) and to what extent the modeled property can be observed (observable vs. unobservable). With student states, we denote properties of a student that change or evolve relatively quickly (typically within a single training session). The knowledge of a student, for example, evolves during training with every solved task, or the current engagement level might change depending on the question the student is currently trying to answer. Student traits, on the other hand, denote longer-term properties of a student such as his or her general attitude towards learning, potential learning disabilities, or learning styles. While these properties can change over time as well, they typically do so only on a much larger timescale and not within a single training session. Behavior patterns of students (e.g. how proficient a student is with the learning environment) typically lie somewhere in the middle of this state-trait scale.

Based on an in-depth performance analysis of recent student models that analyses the predictive quality for observable and unobservable variables, we first present a classification pipeline for predicting student characteristics tailored to small sample sizes, which typically arise from controlled study experiments. We evaluate our method for the specific case of developmental dyscalculia. Next, we identify the need to develop semi-supervised techniques as for large portions of the available data in educational data sets, traits of students are unknown. We present a semi-supervised classification pipeline that makes use of recent discoveries in research on neural network. Finally, we present a clustering pipeline that allows for temporally coherent clustering of student behavior that provides a tool to discover and observe student behavior in ITS.

1.2 Principal Contributions

In the following we list the main contributions of the work presented in this thesis:

- **Performance characteristics of recent student models.** In extension to previous work on the analysis of student model properties [Beck and Chang, 2007; Khajah et al., 2014b; Fancsali et al., 2013], we empirically evaluate the performance characteristics of two recent student models latent factor knowledge tracing (LFKT) [Khajah et al., 2014a] and feature aware student knowledge tracing (FAST) [González-Brenes et al., 2014] on synthetic data and compare them to two standard student modeling approaches: Bayesian knowledge trac-

ing (BKT) [Corbett and Anderson, 1994] and item response theory (IRT) [Wilson and De Boeck, 2004]. By evaluating the models on 66'000 different parameter configurations, we demonstrate the relative performance gains between the models for various regions of the parameter space.

- **Generalized student sampling model.** Many processes of interest in educational data mining are not directly observable such as for example student learning. Ground truth about latent variables enables the experimental evaluation of various properties of a model. We therefore propose a synthetic generative student model that encompasses four well-known student models. We carefully design the set of parameter configurations to match real-world conditions. This allows us to study model performance under different parameter settings, and to test model robustness against violations of specific model assumptions.

- **Classification pipeline for student traits.** We propose a pipeline for integrating automatic detection of student traits directly into a tutoring system. Since our pipeline, leverages standard interaction data available from intelligent tutoring systems the costs for model building are low and the accuracy of the classifier can be continuously improved as more student data is added over time. Our classifier can be seamlessly embedded into an intelligent tutoring system (in our experiments *Calcularis* [Käser et al., 2013c]), where the assessment runs continuously and non-intrusively in the background. In addition, our method can be used as a screening tool in which case test time can be adapted to each child individually, which reduces the average test duration substantially.

- **Extensive comparison to expert tests.** One of the main challenges of computer-based tests is to show equivalence to conventional assessments in terms of accuracy, practicability, and validity [Jenkins et al., 2007]. We therefore validate our classification pipeline for the specific case of developmental dyscalculia (DD) (a specific learning disability affecting the acquisition of arithmetic skills [von Aster and Shalev, 2007a]) and compare our predictions in terms of content-validity, criterion-related validity, construct validity and reliability to standardized tests for detecting DD. Our results demonstrate that we can identify children at risk of DD with a high accuracy (91% sensitivity, 91% specificity) within a short time (11 minutes on average).

- **Variational auto-encoders for learning data.** Recently, variational auto-encoders have outperformed other semi-supervised classifica-

tion approaches in computer vision [Kingma and Welling, 2014; Rezende et al., 2014]. Inspired by these advances, we explore variational auto-encoders for student classification in the educational context. We employ deep variational auto-encoders to learn efficient feature embeddings from unlabeled student interaction data. We optimize the architecture of two different neural networks for educational data - a simple variational auto-encoder and a convolutional variational auto-encoder.

- **Semi-supervised classification pipeline.** Based on the developed variational auto-encoders we present a complete semi-supervised classification pipeline for student traits. We validate our framework on a large and unlabeled data set with interaction data from more than 7K students and measure the classification performance on two independent small and labeled data sets with 83 and 155 students. Our pipeline improves the classification performance (in terms of area under the ROC curve) for standard classifiers by up to 28% compared to completely supervised training. Additionally, our approach shows improved robustness to class imbalance compared to other feature embedding methods. This improved robustness is especially advantageous when predicting relatively rare student conditions such as developmental dyscalculia.

- **Temporally coherent clustering for student characteristics.** We present a complete processing pipeline for evolutionary clustering that can be used as a black box for any intelligent tutoring system that records student interactions. Our pipeline uses an evolutionary clustering method [Xu et al., 2014] for finding temporally coherent clusters. We demonstrate that temporal smoothing of clusters has beneficial properties for extracting student behavior and groups from educational data. Our pipeline can capture cluster evolution events, such as merging, splitting, dissolving and forming of clusters, which is crucial to understand the evolution of learning data. To capture these events automatically, we compute the optimal cluster count for each time step.

- **Robust similarity measure for students.** We present a robust similarity measure between students based on student interaction data. We propose to summarize student actions as Markov Chains and demonstrate that this approach is superior to direct sequence mining techniques [Bergner et al., 2014; Köck and Paramythis, 2011] with respect to noise cancellation and the ability to identify groups of students with similar behavior. We compute student similar-

ity based on individual Markov Chains and demonstrate that the Hellinger distance outperforms other metrics that are frequently used in the educational data mining literature [Bergner et al., 2014; Köck and Paramythis, 2011].

1.3 Thesis outline

This thesis is organized as follows.

- **Chapter 2** gives an overview of intelligent tutoring systems and a overview of related work on student modeling, including cognitive state models, models for predicting student traits and clustering methods.
- **Chapter 3** describes the two intelligent tutoring systems that have been used to evaluate most of our methods in this thesis.
- **Chapter 4** compares performance characteristics of recent student models using synthetic data sets and provides a discussion for potential directions to improve student models. This chapter motivates our work on expanding student models for the identification and discovery of student traits.
- **Chapter 5** describes a data-driven supervised classification pipeline for the detection of student traits and evaluates the pipeline for the specific case of developmental dyscalculia.
- **Chapter 6** expands on the work in Chapter 5 by making use of large unlabeled data sets to improve classification accuracy. We present a semi-supervised classification pipeline using recent advances in the field of deep neural networks.
- **Chapter 7** describes our clustering pipeline for the discovery of student characteristics and student behavior within intelligent tutoring systems.
- **Chapter 8** concludes this thesis by a discussion of the contributions and an outlook to potential future work.

1.4 Publications

In the context of this thesis, the following peer-reviewed publications have been accepted:

S. KLINGLER, R. WAMPFLER, T. KÄSER, B. SOLENTHALER and M. GROSS (2017). Efficient Feature Embeddings for Student Classification with Variational Auto-encoders. *Proceedings of the International Conference on Educational Data Mining (Wuhan, China, June 25-28, 2017)*, pp. 72-79. **[Best Paper Award]**

S. KLINGLER, T. KÄSER, B. SOLENTHALER and M. GROSS (2016). Temporally Coherent Clustering of Student Data. *Proceedings of the International Conference on Educational Data Mining (Raleigh, US, June 29- July 2, 2016)*, pp. 102-109.

S. KLINGLER, T. KÄSER, A.G. BUSETTO, B. SOLENTHALER, J. KOHN, M. VON ASTER and M. GROSS (2016). Stealth Assessment in ITS - A Study for Developmental Dyscalculia. *Proceedings of the International Conference on Intelligent Tutoring Systems (Zagreb, Croatia, June 6-10, 2016)*, pp. 79-89.

S. KLINGLER, T. KÄSER, B. SOLENTHALER and M. GROSS (2015). On the Performance Characteristics of Latent-Factor and Knowledge Tracing Models. *Proceedings of the International Conference on Educational Data Mining (Madrid, Spain, June 26-29, 2015)*, pp. 37-44.

This thesis includes the contents of all above papers, but includes additional implementation and evaluation details not present in the papers.

During the course of this thesis the following peer-reviewed publications have been accepted that are not directly linked to the work presented in this thesis.

T. KÄSER, **S. KLINGLER**, A.G. SCHWING and M. GROSS (2017). Dynamic Bayesian Networks for Student Modeling. *IEEE Transactions on Learning Technologies*.

R. SIEGFRIED, **S. KLINGLER**, M. GROSS, R.W. SUMNER, F. MONDADA and S. MAGNENAT(2017). Improved mobile robot programming performance through real-time program assessment. *Proceedings of Innovation and Technology in Computer Science Education (Bologna, Italy, July 3-5, 2017)*, pp. 341-346.

T. KÄSER, **S. KLINGLER** and M. GROSS (2016). When to stop? - Towards Universal Instructional Policies. *Proceedings of Learning Analytics and Knowledge Conference (Edinburgh, United Kingdom, April 26-29, 2016)*, pp. 289-298. **[Best Paper Award]**

T. KÄSER, **S. KLINGLER**, A.G. SCHWING and M. GROSS (2014). Beyond Knowledge Tracing: Modeling Skill Topologies with Bayesian

Networks. *Proceedings of the International Conference on Intelligent Tutoring Systems (Honolulu, Hawaii, June 5-9, 2014)*, pp. 188-198. [**Best Paper Award**]

S. MÜLLER, M. KAPADIA, S. FREY, **S. KLINGLER**, R.P. MANN, B. SOLENTHALER, R.W. SUMNER and M. GROSS (2015). Statistical Analysis of Player Behavior in Minecraft. *Proceedings of Foundation of Digital Games (Pacific Grove, CA , USA, June 21-25, 2015)*.

S. MÜLLER, M. KAPADIA, S. FREY, **S. KLINGLER**, R.P. MANN, B. SOLENTHALER, R.W. SUMNER and M. GROSS (2015). HeapCraft Social Tools: Understanding and Improving Player Collaboration in Minecraft. *Poster Proceedings of Foundation of Digital Games (Pacific Grove, CA , USA, June 21-25, 2015)*.

S. MÜLLER, M. KAPADIA, S. FREY, **S. KLINGLER**, R.P. MANN, B. SOLENTHALER, R.W. SUMNER and M. GROSS (2015). HEAPCRAFT: Quantifying and Predicting Collaboration in Minecraft. *Artificial Intelligence and Interactive Digital Entertainment (Santa Cruz, USA,, November 14-18, 2015)*, pp. 156-162 .

S. MÜLLER, B. SOLENTHALER, M. KAPADIA, S. FREY, **S. KLINGLER**, R.P. MANN, R.W. SUMNER and M. GROSS (2015). HeapCraft: Interactive Data Exploration and Visualization Tools for Understanding and Influencing Player Behavior in Minecraft. *Proceedings of Motion in Games (Paris, France, November 16-18, 2015)*, pp. 237-241.

S. MAGNENAT, M. BEN-ARI, **S. KLINGLER**, R.W. SUMNER (2015). Enhancing Robot Programming with Visual Feedback and Augmented Reality. *Proceedings of Innovation and Technology in Computer Science Education (Vilnius, Lithuania, July 6-8, 2015)*, pp. 153-158.

CHAPTER 2

Student model landscape (related work)

Intelligent tutoring systems (ITS) have been used successfully in many different applications such as physics [Conati et al., 2002], algebra [Koedinger et al., 1997] and basic arithmetic [Käser et al., 2013c; Käser et al., 2013a]. ITS consist of four main components [Corbett et al., 1997], as displayed in Figure 2.1. Learners are interacting with a computer-based environment solving tasks provided by the system in the *problem-solving environment*. Student actions are evaluated based on *domain knowledge* (typically provided by experts) to assess the correctness of the provided solutions. This information along with various features from the problem-solving environment (such as response times, time to notice an error or problem-solving strategies) are used to update the *student model* (a mathematical representation of the student). Based on the student model the *pedagogical module* performs teaching decisions to optimize the learning outcome for each student. The student model is a fundamental part of an ITS, since task selection and evaluation of the student's learning progress are based on this model. The quality of the student model greatly influences the learning experience and the learning outcome of a student. In order for the pedagogical module to adapt the learning experience to the individual needs of a student detailed information about a student's cognitive, metacognitive and affective state is desirable (as discussed in Chapter 1). In addition, identifying and discovering student traits such as learning disabilities or learner styles allows for further adaption. Further, multi-modal student models can provide various insights into the learning process and allow experts to improve the curriculum as well as identify students that need additional support. Expanding student models

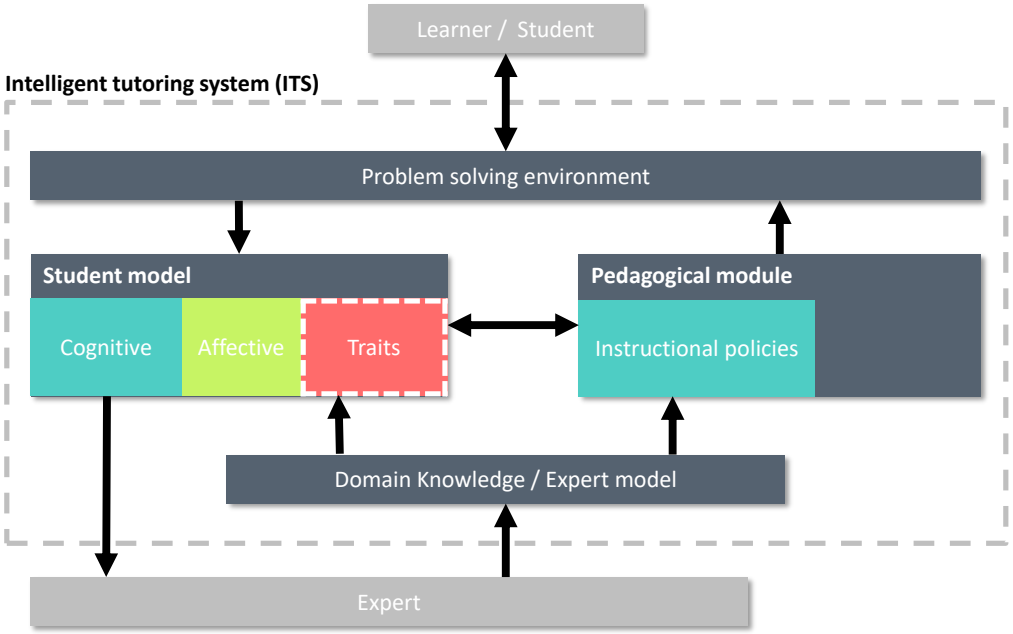


Figure 2.1: *Main components of a generic intelligent tutoring system (ITS) (figure adapted from [Solenthaler et al., under review]). Learners are interacting with a learning environment. These interactions are used to build a representation of the cognitive and affective states of the student (called student model) and can be used to identify student traits. Based on such a student profile, the pedagogical module performs teaching decisions to provide an optimal learning experience. Experts provide domain knowledge to assess student solutions and build domain knowledge. Developing models of student traits and characteristics is the focus of this thesis (dashed box).*

to represent student traits will be the focus of this thesis (dashed square in Figure 2.1).

Cognitive models of student knowledge have been the major focus in research on student models. In Section 2.1 we will therefore first review the work on modeling student knowledge as well as other cognitive states such as models of learning progress or knowledge gaps.

As recent ITS provide more detailed user interaction data, student models have expanded beyond modeling cognitive and affective student states to discover and predict student behavior and student traits. We will review related work on clustering and predicting student behavior and student traits in Section 2.2.

This chapter is intended to give a broad overview of related work in the

field of student modeling for intelligent tutoring systems. More specific related work to methods presented in this thesis is discussed in the respective chapters.

2.1 Cognitive student models

Modeling student knowledge and student learning has been the major focus in research on cognitive student models. Many popular knowledge modeling approaches are based on two common basic models of student learning. The first model called Bayesian Knowledge Tracing (BKT) is a special case of a Hidden Markov model and was introduced by Corbett and Anderson [1994] to model student knowledge as a binary variable inferred from binary observations. For BKT the observations consist simply of correct or incorrect solution attempts to a particular task. The performance of the original BKT model has been improved by using individualization techniques such as modeling the parameters by student and skill [Pardos and Heffernan, 2010a; Wang and Heffernan, 2012; Yudelson et al., 2013] or per school class [Wang and Beck, 2013]. It has been shown that individualizing student models to each student has a significant effect on the number of practice opportunities: Lee and Brunskill [2012] found that a significant amount of students were over or under practicing skills when using a population model (same model parameters for the entire population). In addition, student-specific parameters provide better performance for predicting student task outcomes, of which student-specific learning rates was found to be most important [Yudelson et al., 2013]. Clustering approaches have also proven successful in improving the prediction accuracy of BKT [Pardos et al., 2012]. Dynamic Bayesian networks provide a unified framework for modeling temporal stochastic processes and were used to model student performance in various domains ranging from mathematics learning [Käser et al., 2013c; Käser et al., 2013a] to serious games [Lester et al., 2013].

The second modeling family is Item Response Theory (IRT) [Wilson and De Boeck, 2004]. The concept of IRT assumes that the probability of a correct response to an item is a mathematical function of student and item parameters. The Additive Factors Model (AFM) [Cen et al., 2007; Cen et al., 2008] fits a learning curve to the data by applying a logistic regression. Another technique called Performance Factors Analysis (PFA) [Pavlik et al., 2009] is based on the Rasch item response model [Fischer and Molenaar, 1995].

2.1.1 Hybrid models

Recently several hybrid models have been proposed that combine the modeling techniques from BKT and IRT. In [Johns and Woolf, 2006] a dynamic mixture model has been presented to trace performance and affect simultaneously. The KT-IDEM model extends BKT by introducing item difficulty parameters [Pardos and Heffernan, 2011]. Other work focused on individualizing the initial mastery probability of BKT by using IRT [Xu and Mostow, 2013]. Logistic regression has also been used to integrate subskills into BKT [Xu and Mostow, 2011]. Recently, two models have been introduced which synthesize IRT and BKT. Latent Factor Knowledge Tracing (LFKT) [Khajah et al., 2014a] individualizes the guess and slip probabilities of BKT based on student ability and item difficulty. Feature Aware Student Knowledge Tracing (FAST) [González-Brenes et al., 2014] generalizes the individualized guess and slip probabilities to arbitrary features.

2.1.2 Student model properties

The analysis of properties of student models such as Bayesian Knowledge Tracing (BKT) has been an important part of research over the past years. Practitioners and researchers employing student models in their intelligent tutoring systems are rarely only interested in the predictive power of these models (will a student be able to solve a particular task?), but need to interpret model parameters related to student learning (e.g. how fast is a student learning?). Beck and Chang [2007] showed that learning BKT models exhibits fundamental identifiability problems, i.e., different model parameter estimates may lead to identical predictions about student performance but would require very different interventions based on the parameters. This problem was addressed by using an approach that biases the model search by Dirichlet priors to get statistically reliable improvements in predictive performance. This work has been further extended by performing a fixed point analysis of the solutions of the BKT learning task and by deriving constraints on the range of parameters that lead to unique solutions [Van de Sande, 2013]. Furthermore, it has been shown that the parameter space of BKT models can be reduced using clustering [Ritter et al., 2009].

Other research focused on analyzing convergence properties [Pardos and Heffernan, 2010b] of the expectation maximization algorithm (EM) for learning BKT models and exploring parameter estimates produced by EM [Gu et al., 2014]. It has been shown that convergence in the log likelihood space does not necessarily mean convergence in the parameter space. Fancsali

et al. [2013] have studied how good BKT is at predicting the moment of mastery. Different thresholds to assess mastery and their corresponding lag, i.e., the number of tasks that BKT needs to assess mastery (after mastery has already been achieved), have been investigated. Using multiple model fitting procedures, BKT has been compared to performance factors analysis (PFA) [Gong et al., 2010]. While no differences in predictive accuracy between the models have been reported, it has been shown that for knowledge tracing EM achieves significantly higher predictive accuracy than Brute Force. Findings from other studies, however, suggest the opposite [Baker et al., 2008; Baker et al., 2010a]. In [Beck and Xiong, 2013], upper bounds on the predictive performance have been investigated by employing various cheating models. It has been concluded that BKT and PFA perform close to these limits, suggesting that other factors such as robust learning or optimal waiting intervals should be considered to improve tutorial decision-making. Recently, the predictive performance of the hybrid models LFKT and FAST has been compared to BKT and IRT models using data from different intelligent tutoring systems [Khajah et al., 2014b].

2.2 Models of student characteristics and traits

Building predictive models of student characteristics and traits such as knowledge level, intelligence, learning disabilities or personality traits is one of the big challenges in educational data mining (see discussion of the grand challenges in Chapter 1). To overcome these challenges a large body of work has focused on mining the data logs collected from intelligent tutoring systems (ITS). Important topics in the area of student behavior and student traits modeling are automatic stealth assessments (classification) and the extraction of student properties (clustering).

First, stealth assessments provide an unobtrusive way for the evaluation of student learning or the detection of student traits and characteristics by predicting student properties in the background of an ITS [Shute, 2011]. Traditional assessments are often time-consuming and have to be supervised by an expert, rendering them expensive in practice. Hence, this approach does not scale and is therefore not suitable in many cases, such as massive open online courses (MOOCs), large university courses, or widespread screenings in elementary schools to enable early detection of learning disabilities. Related work on predictive models for the detection of student traits is discussed in Section 2.2.1.

Second, clustering approaches provide tools for the automatic extraction and discovery of student properties. On the one hand, the identification of stu-

dent abilities and behavior patterns allows us to draw conclusions about human learning. On the other hand, the extracted properties can be used to improve the adaptation of the underlying intelligent tutoring system (ITS). We discuss related work on clustering data logs in ITS in Section 2.2.2.

2.2.1 Student trait identification

Previous work has investigated stand-alone automatic digital assessments, including research on automatic scoring [Attali, 2015], item generation [Graf and Fife, 2013] and game-based assessment [Hao et al., 2015]. Furthermore, digital screening programs replacing traditional neuropsychological tests, for example for dyscalculia [Butterworth, 2003] or dyslexia [Cisero et al., 1997] have been developed. Ideally, such computer-based screening programs are seamlessly integrated into an intelligent tutoring system (ITS). This enables not only automatic and non-intrusive assessment of students, but also analysis and detection of student traits that allow for a better adaptation of the curriculum to the individual needs of learners. Despite these advantages, only a few works have addressed such ITS with a fully integrated assessment. One step in this direction are integrated behavior detectors identifying students gaming the system [Baker et al., 2004], finding wheel-spinning students [Beck and Gong, 2013] or modeling engagement, e.g. [Beck, 2005; Arroyo and Woolf, 2005; Cooper et al., 2010]. User types have been modeled based on observed action sequences using expert domain knowledge and psychological behavior theory [Cowley and Charles, 2016]. Other work combined clustering and classification approaches to detect students' mathematical characteristics [Käser et al., 2013b]. One of the main challenges of automatic computer-based assessments is to show equivalence to conventional assessment in terms of accuracy, practicability, and validity [Jenkins et al., 2007].

In order to build such predictive models of student traits, smaller-scale and controlled user studies are typically conducted where detailed information about student characteristics are at hand (labeled data). However, gathering labeled data in educational data mining is a time and cost-intensive task. Nevertheless, the amount of available training data directly influences the quality of predictive models. Unlabeled data, on the other hand, is readily available in high volumes from ITS and massive open online courses. Semi-supervised learning bridges this gap by making use of patterns in bigger unlabeled data sets to improve predictions on smaller labeled data sets (this is also the focus of Chapter 6). These techniques are well explored in a variety of domains and it has been shown that classifier performance can be im-

proved for, e.g., image classification [Kingma et al., 2014], natural language processing [Turian et al., 2010] or acoustic modeling [Liao et al., 2013].

In the education community, semi-supervised classification has been used employing self-training, multi-view training, and problem-specific algorithms. Self-training has e.g. been applied for problem-solving performance [Min et al., 2014] and the generation of concept maps [Jiang et al., 2016]. In self-training, a classifier is first trained on labeled data and is then iteratively retrained using its most confident predictions on unlabeled data. Self-training has the disadvantage that incorrect predictions decrease the quality of the classifier. Multi-view training uses different data views and has been explored with co-training [Tam et al., 2015] and tri-training [Kostopoulos et al., 2015] for predicting prerequisite rules and student performance, respectively. Co-training trains two classifiers on two independent feature sets (or data views) and uses the predictions of one classifier to enlarge the training set of the other classifier and vice versa [Blum and Mitchell, 1998]. In tri-training, the predictions for which two classifiers agree are used to improve the training set for the third classifier [Zhou and Li, 2005]. The performance of these methods, however, largely depends on the properties of the different data views, which are not yet fully understood [Xu et al., 2013a]. Problem-specific semi-supervised algorithms have been used to organize learning resources in the web [Labutov and Lipson, 2016], to automatically score short text answers [Jing, 2015] or for the analysis of learning objectives [Miller and Soh, 2013]. But of course, problem-specific algorithms have the disadvantage that they cannot be directly applied to other classification tasks.

On the other hand models for representation learning automatically compute efficient data representations without the need for prior expert knowledge [Bengio et al., 2013]. In the past years, deep neural networks have provided breakthroughs in understanding data in many areas, including but not limited to computer vision, natural language processing or game AI (most notably Atari and Go games) [LeCun et al., 2015; Silver et al., 2016; Mnih et al., 2015]. Recently, it has been shown (outside of the education context) that variational auto-encoders have the potential to outperform the commonly used semi-supervised classification techniques. A variational auto-encoder is a neural network that includes an encoder that transforms a given input into a typically lower-dimensional or constraint representation, and a decoder that reconstructs the input based on the latent representation [Kingma et al., 2014]. Hence, variational auto-encoders learn an efficient feature embedding (feature representation) using unlabeled data that can be used to improve the performance of any standard supervised learning algorithm [Kingma et al., 2014]. This property greatly reduces the

need for problem-specific algorithms. Moreover, variational auto-encoders feature the advantage that the trained deep generative models are able to produce realistic samples that allow for accurate data imputation and simulations [Rezende et al., 2014], which makes them an appealing choice for educational data mining.

2.2.2 Student trait discovery

Clustering of sequential data is a common approach to detect similar behavior patterns and student characteristics. Sequential data clustering has been successfully applied to a variety of applications such as reading comprehension [Peckham and McCalla, 2012], online collaboration tools [Perera et al., 2009], table-top environments [Martinez-Maldonado et al., 2013], web browsing [Wang and Heffernan, 2012], physics simulations [Bergner et al., 2014] or homework assignments [Herold et al., 2013].

Beal and Cohen [2008] demonstrated that data from intelligent tutoring systems (ITS) exhibit interesting structure at different scales: from short action sequences to long-term patterns over multiple training sessions. A variety of different student behavior at these different scales has been investigated over the past years. [Miller and Soh, 2013] identified students that impose challenges for the student models and allowed to retrain the models for these challenging students to improve model performance. Clustering approaches allowed to detect homogenous groups of students with similar behavior [Trivedi et al., 2011; Trivedi et al., 2012; Käser et al., 2013b]. Extracted group information has been used to improve post-test score prediction [Trivedi et al., 2011; Trivedi et al., 2012; Käser et al., 2013b] and to increase the precision of student models [Pardos et al., 2012]. Further, two different variants of an ITS have been compared to identify differences in student interactions using directed graphs [Rebolledo-Mendez et al., 2013]. Other work studied the relation between interaction patterns and the performance of students to identify patterns that positively or negatively influence the performance [Andres et al., 2015; Kinnebrew and Biswas, 2012]. The relation between student action sequences and the affective states boredom and confusion among the students has been studied [Andres et al., 2015]. Groups of students that show interesting temporal behavior have been mined from interaction data [Kinnebrew et al., 2013]. Using state transition graphs and graph measures (such as density and centrality) self-regulated learning has been explored [Hadwin et al., 2007]. King et al. [2007] identified different subtypes of children with dyslexia (a learning disability).

Common techniques for the analysis of sequential data include sequence mining [Agrawal and Srikant, 1995; Martinez-Maldonado et al., 2013; Nesbit et al., 2007], differential pattern mining [Herold et al., 2013] or Hidden Markov models (HMM) [Biswas et al., 2010; Boyer et al., 2000; Soller and Lesgold, 2007]. Sequential pattern mining techniques have been contextualized using piecewise linear segmentation [Kinnebrew and Biswas, 2012]. Further, pattern frequency analysis of action sequences [Andres et al., 2015] and bootstrapped aggregated clustering [King et al., 2007] has been used. Others have employed semi-supervised graph clustering using the predictions from a student model as additional constraints [Miller and Soh, 2013] or used the dependent Dirichlet process mixture model for clustering with an a priori unknown number of clusters [Campbell et al., 2013]. Static clustering approaches have been used to cluster sequential data by employing similarity measures on state sequences [Bergner et al., 2014; Desmarais and Lemieux, 2013]. These state sequences can be aggregated into Markov Chains modeling the state transitions [Köck and Paramythis, 2011; Benevenuto et al., 2009]. Hidden Markov models have been employed to extract stable groups from temporal data by joint optimization of the model parameters and the cluster count [Li and Biswas, 2000].

While the previous work discussed above analyze student clusters at a given point in time, a temporal analysis would allow identifying how interaction patterns change over time and how groups of similar students evolve. Temporal effects of cluster evolution have been analyzed in [Kinnebrew et al., 2013], based on static clustering at each time step. Static approaches are sensitive to noise in the data and may result in temporally inconsistent clusters. Evolutionary clustering methods [Chakrabarti et al., 2006] address this problem as they consider multiple subsequent time steps. The temporal smoothing increases the resulting cluster stability notably and allows for a better analysis of the clusters, i.e., the student properties and interaction patterns in our case. Recently, an evolutionary clustering approach called AFFECT [Xu et al., 2014] has been introduced that smooths proximities of objects over time followed by static clustering. AFFECT was shown to outperform static clustering algorithms. Similar approaches for time series clustering have been successfully applied in other domains such as for example clustering of motion capture data [Li and Prakash, 2011].

CHAPTER 3

Data

In this chapter we introduce the two intelligent tutoring systems `Orthograph` and `Calcularis` that have provided the data for most experimental results in this thesis. Both systems have been developed at ETH Zurich and are commercially available through Dybuster AG, Switzerland[1].

3.1 Orthograph

Reading and writing skills are crucial in today's society as written communication is ubiquitous in everyday life. However, 5-17% of the population in English- and German-speaking countries [Shaywitz, 1998; Rüsseler et al., 2006] suffer from developmental dyslexia (DL), a specific learning disability affecting the acquisition of reading and writing skills [World Health Organization, 1993]. Research indicates that DL is a neurological disorder with genetic origin [Galaburda et al., 1985; Galaburda et al., 2006; Schulte-Körne et al., 2004; Démonet et al., 2004; Ziegler et al., 2005]. Individuals suffer from auditory difficulties in the discrimination of sounds and the phoneme-to-grapheme mapping [Baldeweg et al., 1999], from visual impairments causing visual confusion and transposition of letters [Lovegrove et al., 1990], and motor impairments involving time estimation and coordinative abilities [Nicolson and Fawcett, 1990].

The computer-based multi-modal training environment `Orthograph` [Kast et al., 2007; Kast et al., 2011] was developed to improve spelling learn-

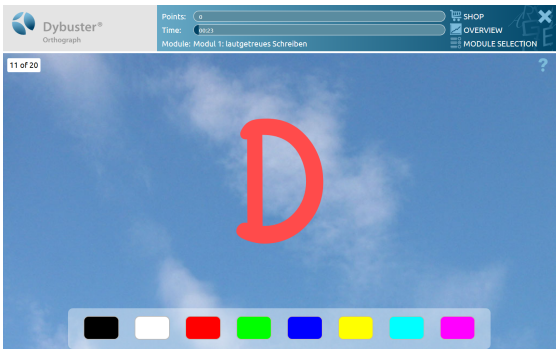
[1] `http://www.dybuster.ch`

ing focusing on students with developmental dyslexia. The training system uses multi-modal recoding of information from letters and words into multiple visual and auditory cues. Letters are mapped to specific colors using a multi-objective optimization, which assigns similar letters such as e.g. 'n' and 'm' to very distinct colors. Additionally, different letter types are mapped to different shapes. Small letters are represented as spheres, capital letters as cylinders and umlauts as pyramids (see Figure 3.1 (c)). Finally, a graph structure visualizes the decomposition of the word into syllables and letters. Auditory cues are generated by synthesizing a melody based on the type and number of letters. Multi-sensory learning stimulates different senses and has been shown to enhance perception and facilitate memory retrieval [Lehmann and Murray, 2005; Shams and Seitz, 2008].

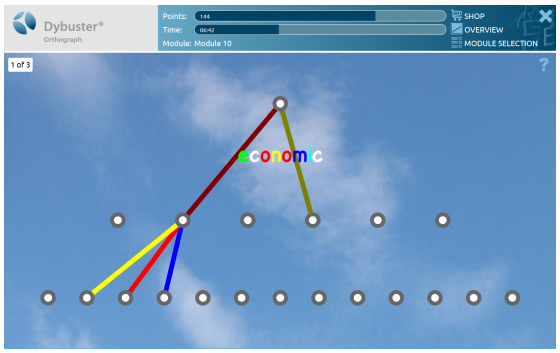
`Orthograph` consists of three different games: Two supporting games that help the learner train the mappings from the word to the visual cues (`Color` and `Graph` game) and the main game, where students learn how to spell words (`Word Learning` game). In the `Color` game (see Figure 3.1 (a)) students learn to memorize the color of each letter. The colors of presented letters are fading to white over time and students have to pick the correct color from the list of colors. In the `Graph` game (see Figure 3.1 (b)) students learn how the graph structure is representing the word decomposition into syllables and graphemes. For this learners have to reproduce the graph structure for different words by connecting the white circles to form the correct tree structure.

The `Word Learning` game (see Figure 3.1 (c)) represents the actual learning game, where students spend most of their time during training. The word is dictated along with a word melody (computed based on the letters, syllables and the word length) and learners can see the color of all letters and the graph structure of the word to spell. Learners then have to enter the correct spelling of the word by keyboard. Erroneous input is immediately detected and the system provides auditory and visual feedback to the student (see highlighted sphere for letter 'ü' in Figure 3.1 (c)). This prevents the display of wrong spellings of entire words. Word selection is based on the student ability and the error profile of the learner.

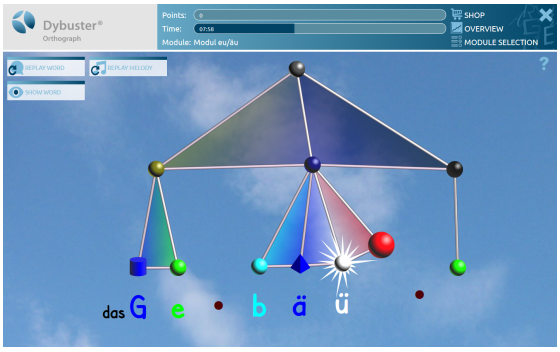
In `Orthograph` words are grouped into different modules based on the frequency in the language and an estimate of the word difficulty [Baschera, 2011]. The word difficulty is estimated based on the word length, the number of silent letter pairs and the number of potential dyslexic confusions. Students are linearly progressing through the different modules. The word selection controller provides and adaptive learning experience by choosing words within a module that maximize the learning gain. Learning gain is



(a) The `Color` game.



(b) The `Graph` game.



(c) The `Word Learning` game.

Figure 3.1: *`Orthograph` consists of three different games. The support games (Figure (a), (b)) train the visual encoding of words. In the `Word Learning` game, students learn how to spell words (Figure (c)). Screenshots were taken with permission of Dybuster AG, Switzerland.*

measured as the reduction in uncertainty in the spelling of words in the current module. Uncertainty, on the other hand, is computed as the error entropy given a global letter confusion matrix and a local error history for each word. A student progresses to the next module once the error entropy falls below a certain threshold for the words in the current module.

3.2 Calcularis

In today's society, the possession of mathematical skills is crucial as numerical cognition and calculations are ubiquitous in everyday life. However, 3-6% of the population in English- and German-speaking countries [Lewis et al., 1994; Shalev and von Aster, 2008] suffer from developmental dyscalculia (DD), a specific learning disability affecting the acquisition of arithmetic

skills [von Aster and Shalev, 2007a]. Recent research indicates that DD is a brain-based disorder, although environmental effects and poor teaching might be relevant as well [Shalev, 2004]. Children with DD show deficits in basic numerical skills such as number comparison [Landerl et al., 2004; Butterworth, 2005], number processing [Kucian et al., 2006] and arithmetic procedures [Ostad, 1997; Ostad, 1999]. Early identification of DD and targeted interventions in the first years of education have been shown to be essential for later mathematical performance [Landerl et al., 2004; Mazzocco and Thompson, 2005].

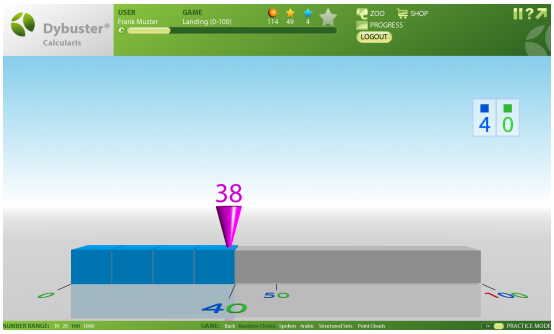
The computer-based training environment *Calcularis* [Käser et al., 2013c; Käser et al., 2013a] was specifically designed to improve mathematical abilities of children with DD or difficulties in learning mathematics. The program turns current neurocognitive theory into the design of different instructional games that train number representations and number understanding as well as arithmetic operations. All games use a special design for numerical stimuli: auditory and visual cues such as color, form, and topology encode different number properties. The learning environment is fully adaptive: A student model and a controlling algorithm optimize the ordering and selection of tasks presented to a child to provide an ideal level of cognitive stimulation.

The games of the training environment are hierarchically structured according to different number ranges (0-10, 0-100, 0-1000) and can be further classified into two distinct areas. Games belonging to the first area train the different number representations (Arabic digits, spoken word, position on a number line) and their interrelationships, as well as the concepts of number understanding (cardinality, ordinality, and relativity). The games in this area are ordered according to a neuropsychological theory on the development of number understanding [Von Aster and Shalev, 2007b; Dehaene and Cohen, 1995]. In the following, we describe four example games of this area. The `Landing` game (Figure 3.2 (a)) asks children to indicate the position of a given number on a number line. In the `Ordering` game (Figure 3.2 (b)), children need to decide if a given number sequence is sorted in ascending order. Another game is the `Secret Number` game (Figure 3.2 (c)). In this game, children have to guess a number in a given interval in as few steps as possible. After each guess, they are told if the searched number is larger or smaller than the guessed number. In the `Estimation` game (Figure 3.2 (d)), children need to assign a point set to a given number. The second area focuses on training concepts and automation of arithmetic operations. Children solve addition and subtraction tasks at different difficulty levels. The difficulty of a task depends on the magnitude of the numbers involved, the complexity of the task and the means allowed to solve the task.

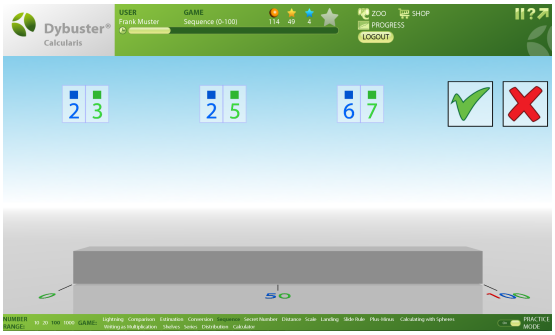
An example game of this area is the `Plus and Minus` game (Figure 3.2 (e)). In this game, children are asked to model arithmetic tasks using colored blocks representing hundreds (red), tens (blue) and units (green). Another game of this area, the `Calculator` game (Figure 3.2 (f)) asks children to perform mental calculation.

`Calcularis` provides optimal learning conditions by adapting the task selection to the needs of the individual child. The program models the knowledge of the children using a dynamic Bayesian network representing different mathematical skills and their relationships. These skills are connected based on their hierarchy, i.e., two skills A and B have a directed connection if skill A is a prerequisite for knowing skill B. For example, to be able to solve an addition with a carry (such as '9+5=?') the child needs to have mastered simple addition tasks (such as '2+3=?'). The resulting skill net contains 100 mathematical skills that are associated with the different games of the training program. To be able to adapt to the type of errors children committed, the program has access to a bug library storing typical error patterns. If a child commits a specific error pattern several times, the program selects remediation tasks for this error type. An example of an error pattern is the switching of digits ('8+5=31'). Remediation games for this error type train the Arabic notation.

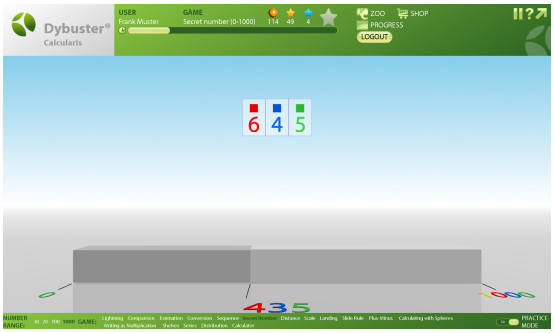
All children start the training with the same game. After each solved task, the knowledge state of the child is updated. Due to the structure of the skill net, each child persecutes a different trajectory over the course of the training. This variety is increased by randomly repeating less sophisticated skills. A detailed description of the mathematical concepts and the control algorithm can be found in [Käser et al., 2013c; Käser et al., 2013a].



(a) In the `Landing` game, the position of a displayed number (40) has to be marked on the number line using the red cone.



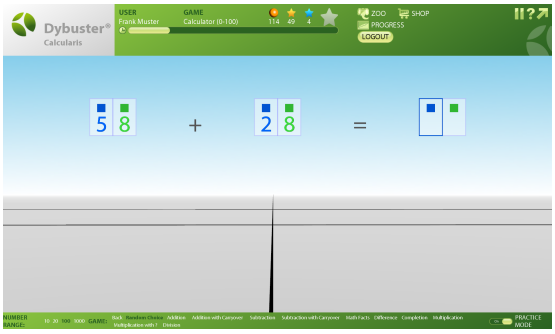
(b) In the `Ordering` game, the correctness of number ordering (ascending) has to be checked.



(c) In the `Secret Number` game, a hidden number has to be found by repeatedly guessing numbers and reducing the search interval.

(d) Given an Arabic number, the corresponding point cloud has to be indicated in the `Estimation` game.

(e) The `Plus and Minus` game asks users to model simple arithmetic operations using blocks representing tens (blue) and ones (green).



(f) In the `Calculator` game, children have to perform mental calculations.

Figure 3.2: *Examples of games in `Calcularis`. Screenshots were taken with permission of Dybuster AG, Switzerland.*

CHAPTER 4

Performance characteristics of student knowledge models

Student models are an integral part of any intelligent tutoring system (ITS) because task selection and evaluation of the student's learning progress are based on this model. Over the last decades, much research has focused on building more accurate models of student knowledge and learning. Most of this work has focused on improving one of the two major student model families: latent factor or knowledge tracing models. Recent studies investigated the limits to the predictive performance of these models. Results show that knowledge tracing and latent factor models perform close to these limits (see Chapter 2). These findings suggest that student models should include other factors such as robust learning or optimal waiting intervals to improve tutorial decision making [Beck and Xiong, 2013]. To create more powerful student models, recently, two models have been introduced that synthesize latent factor models and knowledge tracing models. Latent Factor Knowledge Tracing (LFKT) individualizes model parameters of knowledge tracing based on student ability and item difficulty (the two most common factors in latent factor models) [Khajah et al., 2014a]. Feature Aware Student Knowledge Tracing (FAST) on the other hand generalizes LFKT by allowing individualized model parameters based on arbitrary features [González-Brenes et al., 2014].

In this chapter, we are interested in the properties of these recent hybrid student modeling approaches combining latent factor and knowledge tracing models. In extension to previous work and especially to [Khajah et al., 2014b], we empirically evaluate the performance characteristics of the two

recent hybrid models LFKT and FAST on synthetic data and compare them to the underlying approaches of Bayesian knowledge tracing (BKT) and item response theory (IRT). We sample from a combined student model that encompasses all four models. By using synthetic data generated from the combined model, we show the robustness of the models under breaking model assumptions. By evaluating the models on 66'000 different parameter configurations, we rigorously explore the parameter space to demonstrate the relative performance gain between models for various regions of the parameter space.

Our findings show that for the generated data sets FAST significantly outperforms all other methods for predicting the task outcome for the majority of the investigated parameter configurations. BKT, on the other hand, is significantly better than FAST and LFKT at predicting the latent knowledge state. Furthermore, we identify the influence of different properties of a data set on model performance using regression and show best and worst case performances of the models. Finally, we relate our findings to other studies on student model performance and highlight potential directions to improve student models. These future directions serve as the main motivation for the presented methods in the remainder of this work that will explore techniques for the discovery and identification of student traits within ITS (Chapter 5 to Chapter 7).

4.1 Investigated Models

In an intelligent tutoring system (ITS) a student is typically presented with a set of tasks to learn a specific skill. For each student n, the system chooses at time t an item i from a set of items corresponding to a particular skill. The system then observes the answer $y_{n,t}$ of the student, which is assumed to be binary for all models that we compare in this chapter. In the following, we first present the two standard techniques to model student knowledge in the context of ITS: Bayesian knowledge tracing and Item Response Theory. We then briefly present the two different hybrid techniques (LFKT and FAST) to model various latent states of the student and the tutoring environment.

4.1.1 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) [Corbett and Anderson, 1994] models the knowledge acquisition of a single skill and is a special case of a Hidden

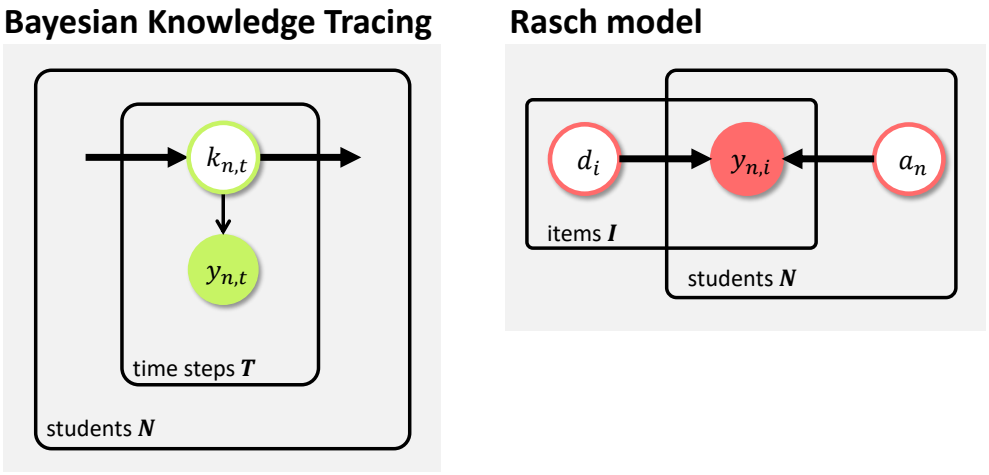


Figure 4.1: *Plate notation of Bayesian knowledge tracing BKT (left) and the Rasch model (right). In BKT task outcomes $y_{n,t}$ depend only on the latent knowledge state $k_{n,t}$ of the student. In the Rasch model task outcomes depend on the difficulty of the presented item d_i as well as the ability of the student a_n.*

Markov Model (HMM) [Reye, 2004]. Figure 4.1, left gives the plate notation of the BKT model. BKT uses two latent states (*known* and *unknown*) to model if a student n has mastered a particular skill $k_{n,t}$ at time t, and two observable states (*correct* and *incorrect*) to represent the outcome of a particular task. Therefore, the probabilistic model can be fully described by a set of five probabilities. The initial probability of knowing a skill a priori $p(k_{n,0})$ is denoted by p_I. The transition from one knowledge state $k_{n,t-1}$ to the next state $k_{n,t}$ is described by the probability p_L of transitioning from the *unknown* latent state to the *known* state and the probability p_F of transitioning from the *known* to the *unknown* state:

$$p(k_{n,t}) = k_{n,t-1}(1 - p_F) + (1 - k_{n,t-1})p_L. \tag{4.1}$$

In the case of BKT, p_F is fixed at 0. Finally, the task outcomes $y_{n,t}$ are modeled as

$$p(y_{n,t}) = k_{n,t}(1 - p_S) + (1 - k_{n,t})p_G, \tag{4.2}$$

where p_S denotes the *slip probability*, which is the probability of solving a task incorrectly despite knowing the skill, and p_G is the *guess probability*, which is the probability of correctly answering a task without having mastered the skill. Learning the parameters for a BKT model amounts to maximum likelihood estimation (MLE). The likelihood for the BKT model for N users and

T observations per user is given as

$$p(\mathbf{Y} \mid \theta) = \prod_{n=1}^{N} \left[\sum_{K} \prod_{t=1}^{T} p(y_{n,t} \mid \theta, K) \right],$$

where K denotes the set of latent knowledge states, $\theta = (p_I, p_L, p_S, p_G)$ the set of model parameters and $\mathbf{Y}$ the observed task outcomes. As the BKT model is a special case of a HMM, we can evaluate the likelihood very efficiently using the forward algorithm [Yudelson et al., 2013]. The maximization of the likelihood function is typically performed using brute-force grid-search [Baker et al., 2010a], expectation maximization [Chang et al., 2006], or gradient descent [Yudelson et al., 2013].

Prediction of the outcomes y is performed in the following way: We predict the first observation $y_{n,1}$ of student n based on the initial probability p_I. We then predict the t^{th} observation $y_{n,t}$, for $t \in \{2, \ldots, T\}$, based on the previous $t-1$ observations.

4.1.2 Item Response Theory

Item Response Theory (IRT) [Wilson and De Boeck, 2004] models the response of a student to an item as a function of latent student abilities a_n and latent item difficulties d_i . The simplest form of an IRT model is the Rasch model as depicted in Figure 4.1, right, where each student n and each item i are treated independently. The outcome $y_{n,t}$ at time t is modeled using the logistic function

$$p(y_{n,t}) = \left(1 + e^{-(a_n - d_i)}\right)^{-1}. \tag{4.3}$$

Note that d_i refers to the difficulty of the item presented at time t. As an example, a student with an ability of $a_n = d_i$ has a 50% chance of getting the item i correct. In contrast to BKT, IRT does not model knowledge acquisition, because IRT was developed for the design and analysis of testings or questionnaires. The model parameters for the Rasch model are learned using the expectation maximization algorithm (EM).

4.1.3 Latent Factor Knowledge Tracing

The Latent Factor Knowledge Tracing (LFKT) [Khajah et al., 2014a] model combines BKT and IRT using a hierarchical Bayesian model as shown in

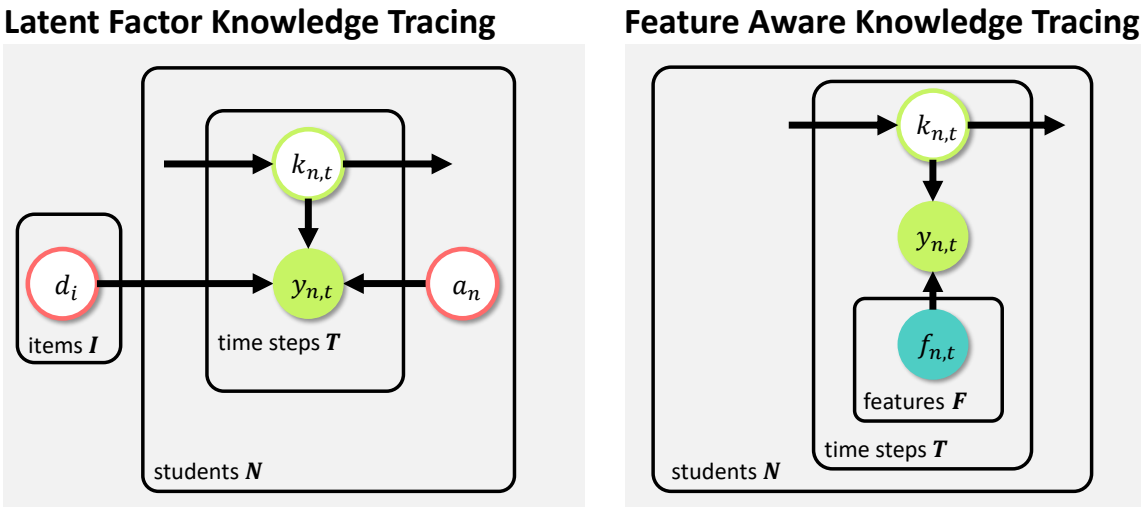


Figure 4.2: *Plate notation of latent factor knowledge tracing (LFKT, left) and feature-aware student knowledge tracing (FAST, right). For LFKT task outcomes $y_{n,t}$ depend on student knowledge $k_{n,t}$, item difficulty d_i and student ability a_n. For FAST task outcomes $y_{n,t}$ depend on arbitrary features $f_{n,t}$*

Figure 4.2, left. Based on the BKT model, slip and guess probabilities are individualized based on student ability and item difficulty as

$$p_{G_{n,t}} = \left(1 + e^{-(d_i - a_n + \gamma_G)}\right)^{-1} \tag{4.4}$$

$$p_{S_{n,t}} = \left(1 + e^{-(a_n - d_i + \gamma_S)}\right)^{-1}, \tag{4.5}$$

where γ_G and γ_S correspond to offset parameters for the guess and slip probabilities. The individualized guess and slip probabilities are then used to predict the outcome $y_{n,t}$ based on equation (4.2) as

$$p(y_{n,t}) = k_{n,t}(1 - p_{S_{n,t}}) + (1 - k_{n,t})p_{G_{n,t}}. \tag{4.6}$$

LFKT therefore assumes that the task outcome depends on item difficulty d_i, student ability a_n and the latent knowledge state $k_{n,t}$ (as shown in Figure 4.2, left). The model is fit by calculating Bayesian parameter posteriors using Markov Chain Monte Carlo.

4.1.4 Feature Aware Knowledge Tracing

Feature Aware Student Knowledge Tracing (FAST) [González-Brenes et al., 2014] allows for unification of BKT and IRT as well, but generalizes the individualized slip and guess probabilities to arbitrary features as shown in

Figure 4.2, right. Given a vector of features $\mathbf{f}_{n,t}$ for a student n at time t the adapted emission probability reads as

$$p(y_{n,t}) = \left(1 + e^{-(\boldsymbol{\omega}^T \mathbf{f}_{n,t})}\right)^{-1}, \tag{4.7}$$

where $\boldsymbol{\omega}$ is a vector of learned feature weights. To model the dependency on the latent knowledge state $k_{n,t}$, FAST uses two types of indicator features. Features $\mathbf{f}^+$ are only active when a student mastered a skill and features $\mathbf{f}^-$ that are only active when the student has not mastered the skill. Additionally, FAST can represent the item difficulties d_i and student abilities a_n from the IRT model if a set of binary indicator functions for the items and the students are used. If features are designed in this way, the emission probability (given in equation (4.7)) reduces to a form that explicitly depends on knowledge state, item difficulty, and student ability (see equation (4.8)). The parameters are fit using a variant of EM [Berg-Kirkpatrick et al., 2010].

4.2 Synthetic data generation

Synthetic data is needed to have ground truth about the underlying data generating model, which enables the experimental evaluation of various properties of a model. In particular, we are not only interested in the predictive powers of the student models regarding the task outcomes (whether a student will correctly answer a task). Instead, we are additionally interested in the quality of mastery prediction for a skill (How precise is the inferred knowledge state?), which is not possible to assess with real-world data, since the actual knowledge state of a student cannot be observed.

The sampling procedure starts by generating N student abilities a_n from a normal distribution $\mathcal{N}(0, \sigma)$. Then, it generates I item difficulties d_i from a uniform distribution $\mathcal{U}(-\delta, \delta)$. Based on the initial probability p_I and the learn probability p_L a sequence of knowledge states $k_{n,0}, k_{n,1}, \ldots, k_{n,T}$ is sampled based on (4.1) and we, therefore, simulate data from only one skill. The time t^* at which $k_{n,t^*} = 1$ for the first time is considered as the moment of mastery. The number of sampled knowledge states is then given as $T = t^* + L$, where L denotes the lag of the simulated mastery learning system. For each student, we generate a random sequence of items, i.e., item indices i. Arbitrary features from the training environment, such as answer times, help calls, problem-solving strategy, the engagement state of the student and gaming attempts, can have an influence on the performance of a student. To simulate those influences in a principled way, we add a single feature f to the data-generating model with a varying feature weight ω (and thus varying correlation to the task outcomes $y_{n,t}$).

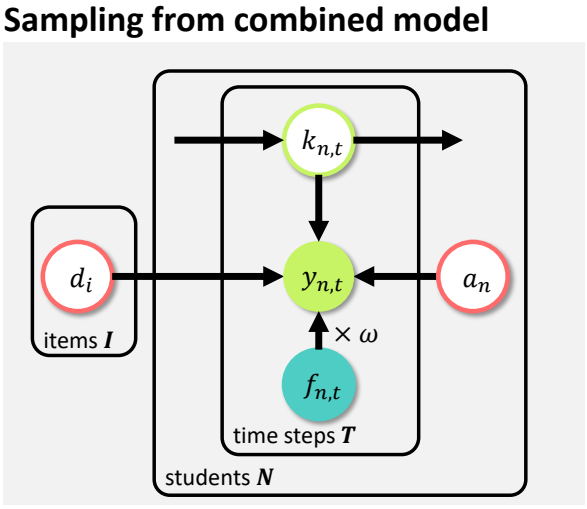


Figure 4.3: *Combined student model used for synthetic data generation. The model corresponds to LFKT with the addition of a single feature. The relative dependencies of the observable nodes (solid circles) and the latent nodes (circles with border) are shown. $k_{n,s}$ denotes the latent knowledge state, d_i the item difficulty, a_n the student ability, $y_{n,t}$ the observation, and $f_{n,t}$ the feature value.*

Based on these quantities, we sample the observations $y_{n,t}$ from a Bernoulli distribution with probability

$$p(y_{n,t}) = \left(1 + e^{-(a_n - d_i - \log \gamma_{n,t} + \omega f_{n,t})}\right)^{-1}, \tag{4.8}$$

where

$$\gamma_{n,t} = \left(k_{n,t}(1 - p_S) + (1 - k_{n,t})p_G\right)^{-1} - 1.$$

Figure 4.3 gives a graphical overview of the described sampling procedure. Our sampling model has the following nine parameters: $p_I, p_L, p_S, p_G, \delta, \sigma, \omega, I, N$. The sampling procedure allows sampling of data that exactly matches the model assumptions of all four models. To sample BKT data we set $\delta = \sigma = \omega = 0$ and (4.8) simplifies to the standard BKT formulation. By setting $p_S = p_G = 0.5$ and $\omega = 0$ we can sample from an IRT model. To sample from an LFKT model we set $\omega = 0$ and for FAST none of the parameters are restricted.

4.3 Experimental Setup

Parameter space. We generated a vast number of parameter configurations to analyze the four models. The set of parameter configurations has been carefully designed to match real-world conditions. The BKT parameters (p_I, p_G, p_S, p_L) are based on parameter clusters found on real-world data [Ritter et al., 2009]. Using a normal distribution with a standard deviation of 0.02, we sampled up to 30 points (depending on the cluster size) around each cluster mean. According to common practice [Harris, 1989], we scaled the student abilities a_n to have a mean of 0 and a variance of 1 and therefore $\sigma = 1$. We sampled the parameter δ (determining the range of the item difficulties) uniformly from $[0, 3]$ according to [Harris, 1989]. Despite simulating only one skill, we varied the item difficulties to account for the fact that skill models tend to be imperfect in practice [Cen et al., 2007; Stamper and Koedinger, 2011; Koedinger et al., 2013]. In accordance to the item difficulties, the feature weight ω was varied uniformly across $[0, 1.5]$. Feature values $f_{n,t}$ were sampled from the uniform distribution $\mathcal{U}(-1, 1)$.

For every parameter configuration, we generated five folds with $N = 300$ simulated students. Each fold was randomly split up into two parts of an equal number of students. The first part was used as training data and the second part for testing. Therefore, the test data did contain unseen students only. As we simulated data from a mastery learning environment the number of simulated tasks for each student was determined by the moment of mastery. Based on the results on typical lag in mastery learning systems presented by Fancsali et al. [2013], we set the lag of the simulated system to $L = 4$ tasks from the moment of mastery. We simulated $I = 15$ different items with random item order.

In total, we generated 66′000 parameter configurations for $p_I, p_G, p_S, p_L, \delta, \omega$, this amounts to a total evaluation time (training and test) of 1′280 hours and 1′351 hours for LFKT and FAST respectively. The evaluation time for the BKT was 99 minutes, and all configurations were evaluated in 58 minutes for the IRT model.

Implementation. To train BKT models, we used our custom code that trains BKT using the Nelder-Mead simplex algorithm minimizing the log-likelihood [Nelder and Mead, 1965]. We thoroughly tested our implementation against the BKT implementation of Yudelson et al. [2013]. The IRT models were fit by joint maximum likelihood estimation [Meyer and Hailey,

2012] implemented in the psychometrics library[1]. FAST using IRT features was shown to be equivalent to LFKT except for the parameter estimation procedure [Khajah et al., 2014b]. The experiments comparing the performance of the different estimation procedures have not been conclusive to whether one procedure is in general superior to the other [Khajah et al., 2014b]. As this work did not investigate different parameter estimation techniques, both models were trained and evaluated using the publicly available FAST student modeling toolkit[2].

4.4 Results

Using the generated data, we investigated the performance characteristics of the four models and evaluated their predictive power and robustness under varying parameter configurations. For our results, we generated 66′000 parameter configurations, and for each of them, we created synthetic data for 1′500 students. Note that there are many ways to characterize performance differences among student models and we only cover a subset of these possibilities.

4.4.1 Error Metrics

The right choice of error metrics when evaluating student models has recently gained increased interest in the educational data mining community. In [Pelánek, 2014] some of the standard options for error metrics are discussed, highlighting possible issues with the accuracy and the area under the ROC curve (AUC). Correlations between various performance metrics and the accuracy of predicting the moment of mastering a skill have been investigated in [Pardos and Yudelson, 2013], showing that the F-measure (equaling to the harmonic mean of precision and recall) and the recall are two metrics with a high correlation to the accuracy of knowledge estimation. The root mean squared error (RMSE) and log-likelihood, on the other hand, are well suited if one wants to recover the true learning parameters. Similarly, Dhanani et al. [2014] concluded from results of 26 synthetic data sets that RMSE is better at fitting parameters than the log-likelihood.

In line with this previous work, we investigated correlations between accuracy, RMSE, and F-measure across all four models. For this, all models were

[1] An open source Java library for measurement, available at https://github.com/meyerjp3/psychometrics.
[2] http://ml-smores.github.io/fast/

trained and evaluated on data using 66′000 different parameter configurations. All metrics are strongly correlated $|\rho| > 0.75, p \ll 0.001$. Our inspections of the metric correlations revealed no significant differences in the metric correlations among the different models. Thus, to a large extent the measures capture the same characteristics for the models we considered in this chapter. In the following, we therefore focus our analysis on the RMSE.

4.4.2 Model Comparison

Overall Performance. In a first step, we investigated the overall performance of the models. For every parameter configuration, we calculated the average RMSE over the five generated folds. Table 4.1 summarizes the parameters for the best and worst data set for every model when model assumptions are met (see Section 4.2). Results show that all models that include a knowledge state (all except IRT) perform best if the slip probability is low and the guess probability is high. This setting leads to a data set that exhibits a high ratio of correct observations. IRT performs best on data that has very distinguished item difficulties (δ is large). Notably, the best performance of FAST is achieved on a data set without features ($\omega = 0$). We assume that this is due to the decreased complexity of the data set, compared to one that exhibits high ω. Consistently, worst case data sets show high symmetric values for guess and slip probabilities. In the case of LFTK and FAST worst case data sets additionally do not distinguish between items (difficulty range $\delta = 0$) and for FAST the feature weights are low.

We then performed the non-parametric Friedman test over all parameter configurations to assess performance differences between the models. We found that there is a statistically significant difference in the performance of the models ($\chi^2(3) = 13'065, p < 0.0001$). Performing a post-hoc analysis using Scheffe's S procedure [Scheffe, 1999] shows all model differences to be significant at $p < 0.0001$ with average ranks of 1.7156, 2.3017, 2.6898 and 3.2929 for FAST, LFKT, BKT, and IRT, respectively. FAST therefore significantly outperforms the other methods on our synthetic data sets.

In [Khajah et al., 2014b] IRT performed not significantly worse than LFKT and FAST on four different data sets. The good performance of IRT was attributed to the deterministic item ordering that allows IRT to infer knowledge acquisition confounded with item difficulty. Our results support this hypothesis as in our synthetic data set the items are in random order, and IRT exhibits the worst overall performance.

Parameter Space Investigation. To gain a better understanding of the performance characteristics of the different models, we analyzed their perfor-

Table 4.1: *Parameters of best and worst case data sets for each model. We only considered data sets that meet the model assumptions. Parameters denoted with * are fixed according to the model assumptions (see Section 4.2).*

Model	δ	pI	pL	pS	pG	ω	RMSE
BKT							
Best	0.00*	0.71	0.41	0.01	0.47	0.00*	0.25
Worst	0.00*	0.10	0.12	0.50	0.49	0.00*	0.48
IRT							
Best	3.00	0.10	0.08	0.50*	0.50*	0.00*	0.42
Worst	0.00	0.10	0.10	0.50*	0.50*	0.00*	0.50
LFKT							
Best	0.75	0.69	0.40	0.01	0.46	0.00*	0.25
Worst	0.00	0.53	0.16	0.28	0.29	0.00*	0.51
FAST							
Best	0.75	0.67	0.40	0.01	0.46	0.00	0.25
Worst	0.00	0.56	0.16	0.28	0.28	0.00	0.51

mances across the parameter space. For every pair of parameters θ_i and θ_j, we divided the parameter configurations into bins with similar values for θ_i and θ_j. We used five bins for each parameter (θ_i and θ_j) resulting in a total of 25 bins. The performance of each model was assessed by calculating the mean RMSE for each bin. The significance of the observed performance differences was computed using the Friedman test and $p < 0.05$.

Figure 4.4 shows the relative performance of the best model for each parameter pair. The models are color-coded: BKT is shown in red, IRT in green, LFKT in yellow, and FAST in blue. The color gradient indicates the relative improvement of the winning model over the second best model, where darker colors indicate higher values. White-colored areas indicate that there is no significant difference between the models. The plot shows that FAST is robust to parameter variations and outperforms the other models in vast areas of the parameter space. In parts with small feature weights, i.e., where the feature f shows only a low correlation with task outcomes, LFKT outperforms FAST. When the variance δ of item difficulties d_i is low, BKT is the best model. A small variance in d_i implies a good skill model, with all tasks having approximately the same difficulty.

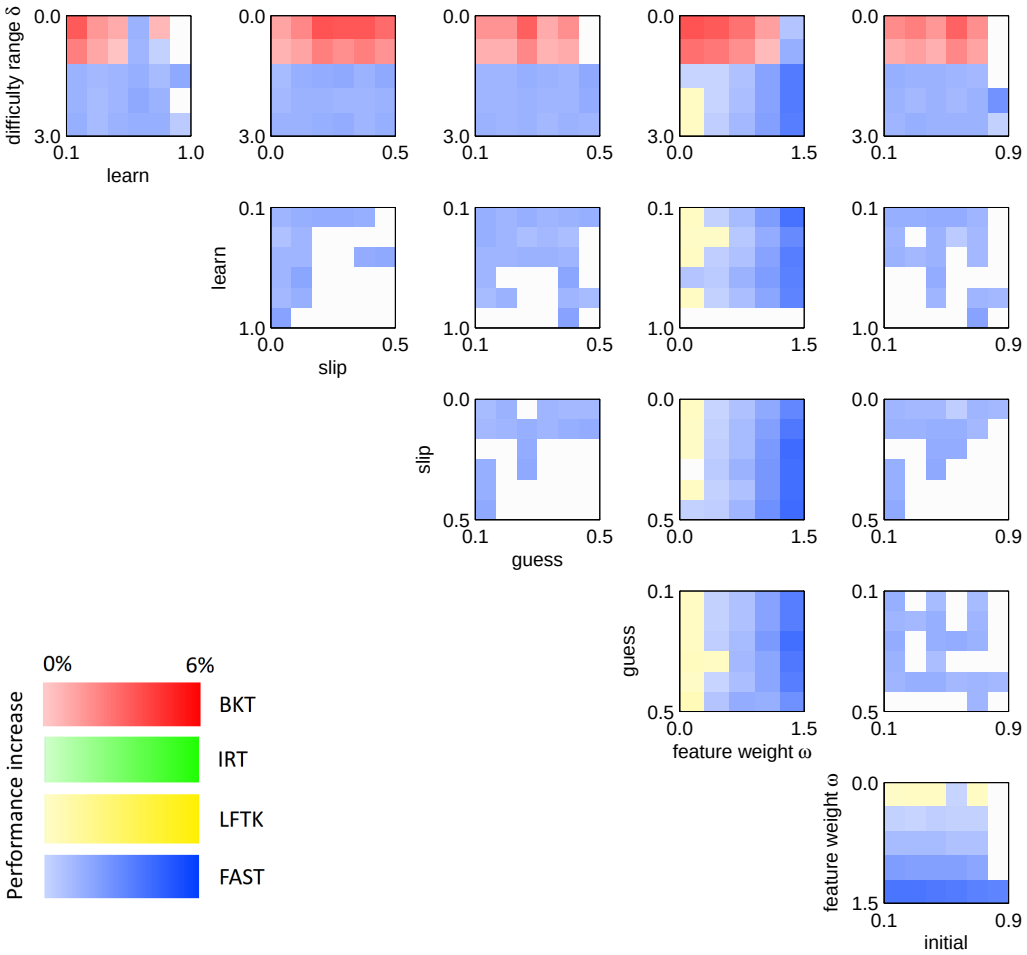


Figure 4.4: *Best performing models (RMSE) regarding prediction of task outcomes. The color for each bin indicates the best performing model, averaged over all other parameters. We investigated BKT (red), IRT (green), LFKT(yellow), and FAST(blue). White-colored bins exhibit no significant difference in model performance. The color brightness indicates the relative improvement of the best performing model over competing models, with dark colors referring to higher values. FAST is robust to parameter variations and outperforms the other models in large parts of the parameter space when predicting task outcomes. BKT is the best model if the variance of the item difficulty is low.*

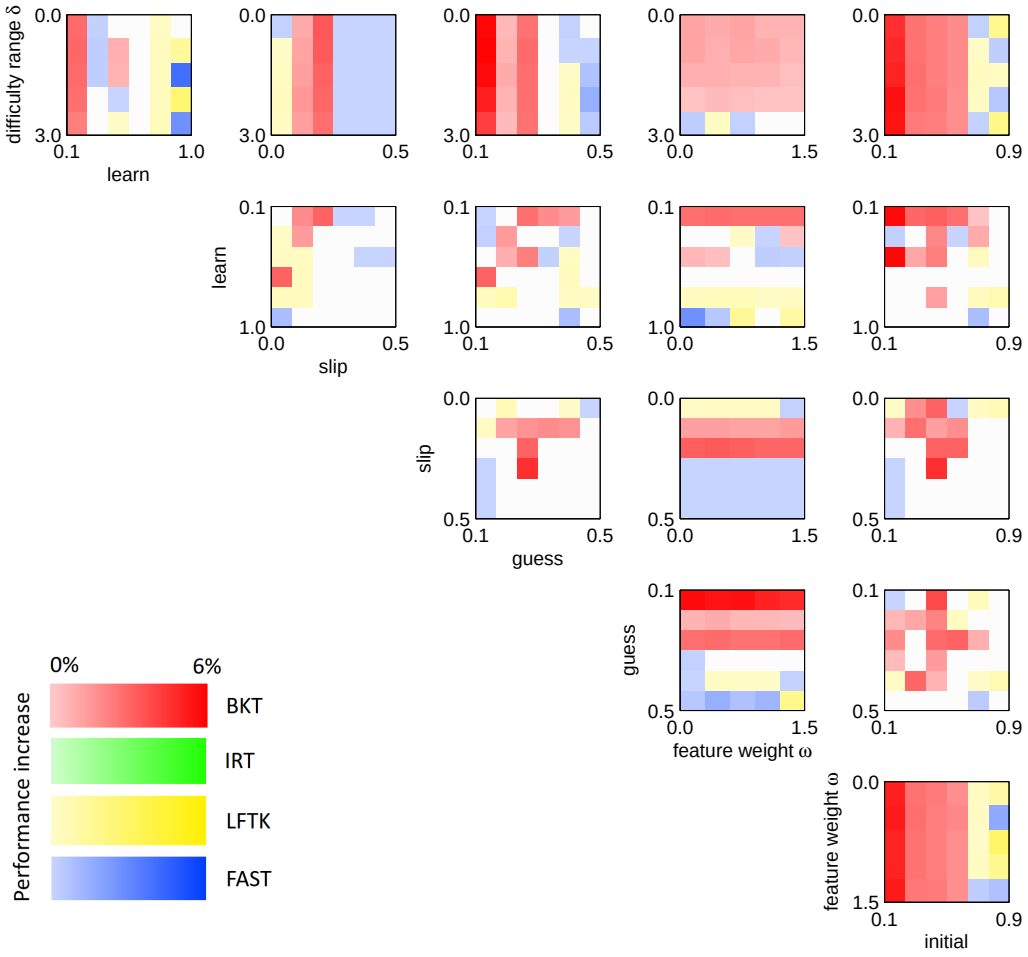


Figure 4.5: *Best performing models (RMSE) regarding prediction of knowledge states. The color for each bin indicates the best performing model, averaged over all other parameters. We investigated BKT (red), IRT (green), LFKT(yellow), and FAST(blue). White-colored bins exhibit no significant difference in model performance. The color brightness indicates the relative improvement of the best performing model over competing models, with dark colors referring to higher values. BKT is superior to the other models in large parts of the parameter space when predicting knowledge states.*

In contrast to Figure 4.4, where we assessed the prediction of task outcomes, we analyzed the quality of the prediction of knowledge states $k_{n,t}$ using the RMSE in Figure 4.5. Ultimately, we want to predict whether a student has mastered a skill or not [Pardos and Yudelson, 2013; Baker et al., 2010b]. The plot uses the same parameter pairs and color codings as Figure 4.4. Interestingly, LFKT and FAST are not superior to BKT when it comes to prediction of the latent state. The additional parameters that LFKT and FAST use, have a direct influence on the predicted task outcomes and therefore improve performance when predicting task outcomes. They have, however, no direct influence on the latent state $k_{n,t}$ of the model.

Robustness. Next, we tested the robustness of the different models against each other. We generated ideal data (meeting the model assumptions) for all the models and then interpolated the parameter values between these ideal cases. The classes of data sets that meet the model assumptions for the four models are described in Section 4.2. From every class of data sets, we selected the extreme case with the least amount of noise. In the following, we describe these cases.

For BKT, data is generated using $\delta = \omega = 0$, assuming a perfect skill model (all tasks have the same difficulty) and setting the influence of additional (not captured) features to 0. Furthermore, we removed the randomness by setting $p_G = p_S = 0$. For IRT, the extreme case data was generated using $p_G, p_S = 0.5$, $\omega = 0$ and by additionally setting $\delta = 3$. As LFKT is a combination of IRT and BKT, we set the parameters to $p_G, p_S = 0.25$ and $\delta = 1.5$. Furthermore, we set $\omega = 0$, again assuming no influence of not captured features. For FAST we used the same parameters as for LFKT, but additionally introduced a feature influence by setting $\omega = 1.5$. We linearly interpolated the parameter space in-between these extreme cases to assess model robustness when model assumptions are violated. Figure 4.6 displays the model with best RMSE in this subspace that contains the extreme (ideal) cases, where p_L and p_I are averaged over the BKT parameter clusters presented in [Ritter et al., 2009]. From these results, we can see that BKT tends to be robust to increased feature influence as long as $p_G, p_S \leq 0.15$. If the feature weight $\omega > 0.75$, FAST outperforms all the other classifiers. For large differences in item difficulties and large guess and slip probabilities, LFKT has a slight advantage over IRT.

4.4.3 Parameter Influence

To analyze the influence of the model parameters on the performance of the student models, we used linear regression to predict the RMSE based on the

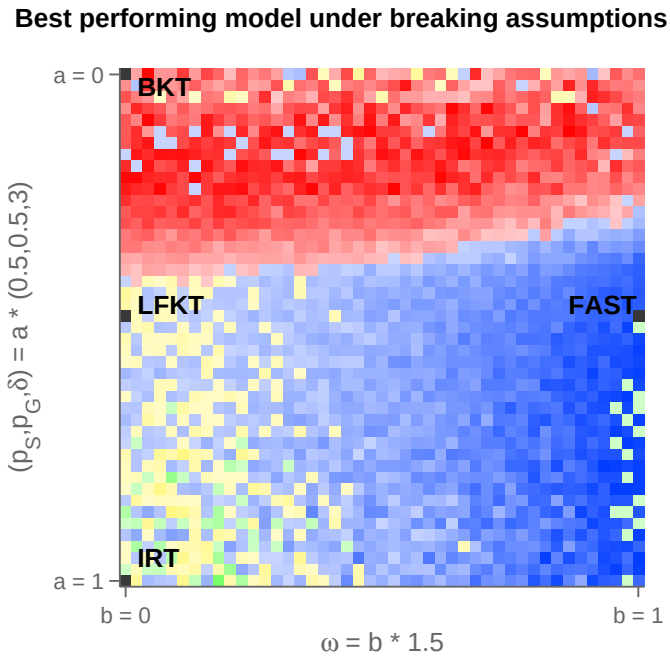


Figure 4.6: *Relative model performance on ideal data sets generated by linearly interpolating between parameters. The colors refer to the models BKT (red), IRT (green), LFKT (yellow) and FAST (blue). The color gradient indicates the relative performance as in Figure 4.4. BKT and FAST are more robust to the invalid assumptions of our experiment than IRT and LFKT.*

parameters of the sampling model. This procedure allowed us to identify statistically significant correlations between the sampling parameters and the performance of the models despite the high dimensionality of the parameter space.

The sampling parameters have a direct influence on the ratio of correct observations in the data, e.g., a high learning probability with low guess and slip parameters leads to a high ratio of correct observations. Further, if the parameters model fast learners, then the average number of tasks tends to be low since we are simulating a mastery learning environment. The three models IRT, LFKT and FAST which explicitly model items are sensitive to this kind of lacking data, as by having fewer observed items per student the estimation of item difficulty becomes more challenging. To investigate the effect of both factors, we added the two variables *correct ratio* and *average number of tasks* as predictors to the regression model. In order to make correlation coefficients comparable, all sampling parameters have been normalized to have mean 0 and standard deviation 1.

Figure 4.7 shows the regression coefficients for all four models, with red and green denoting statistically significant and not significant coefficients,

respectively. The variables *correct ratio* and *average number of tasks* have the largest influence on the RMSE. Both effects are significant and positive (reducing the RMSE). A wider range of item difficulties δ has a positive influence on the performance of all models except for the BKT model. This is expected as BKT does not account for variations in item difficulty and thus larger variations in item difficulties are treated as noise by BKT, which makes prediction harder. IRT, LFKT, and FAST, on the other hand, benefit from larger variations. We assume that this is due to the better identifiability of the effects of the different items. Interestingly, increasing the feature range ω has no significant negative effect for the models that do not take features into account (BKT, IRT, LFKT), but has a positive effect for FAST. The initial probability and the learning probability have a small negative and small positive effect on performance, respectively. While these coefficients are partially significant, they have very small magnitude. The positive effect of the slip probability p_S for all models except BKT (the effect is not significant) is rather surprising. However, the effect of a high slip probability in our sampling model is that it weakens the influence of the latent knowledge state on the task outcomes. This effect could explain the positive influence for models that estimate item difficulty since the difficulty estimates are less convoluted with effects from the knowledge state. Further work is needed to prove this effect.

4.5 Discussion

In this chapter, we investigated the performance characteristics of latent factor and knowledge tracing models by exploring their parameter space. To do so, we generated a vast amount of 66'000 synthetic data sets for different parameter configurations containing data for 1'500 students each. Synthetic data allowed us to study the model performances under different parameter settings, and to test the robustness of the models against violations of specific model assumptions.

We showed best and worst case performances for all the models and investigated the relative performance gain in various regions of the parameter space. Our results revealed that the two recently developed models LFKT and FAST, which synthesize item response theory and knowledge tracing, perform better than BKT and IRT when predicting future task outcomes. FAST even significantly outperformed LFKT if reasonable features can be extracted from the learning environment. As the quality of features is highly dependent on the learning environment, the choice of using FAST instead of LFKT should be based on the specific learning environment at hand. An-

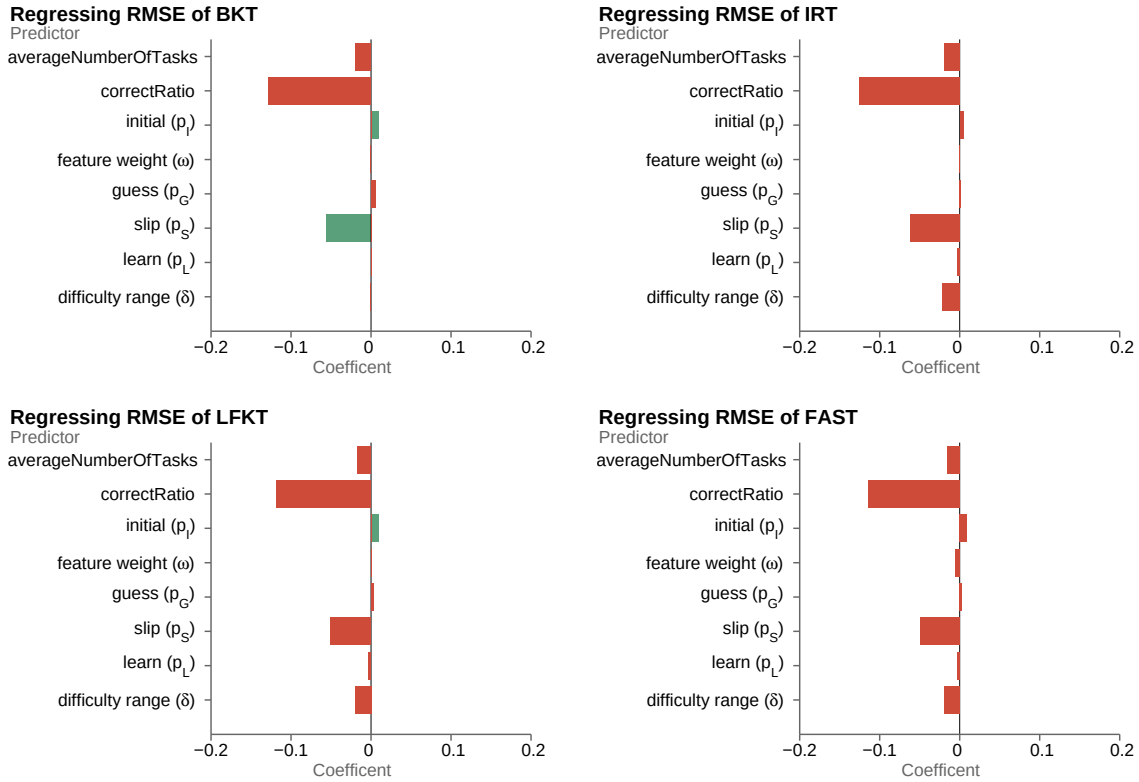


Figure 4.7: *Regression coefficients to predict RMSE based on the sampling parameter values for the models BKT, IRT, LFKT and FAST. Parameters with positive coefficients have a negative effect on the performance and vice versa. Red denotes significant coefficients with $p < 0.001$, green coefficients are not significant.*

other interesting finding is that, IRT exhibited the worst performance of all the models, which supports the hypothesis by Khajah et al. [2014b] that random item ordering has a negative influence on the performance of IRT models. However, more analyses are needed to investigate this effect thoroughly. Further, we investigated the models' abilities to predict the latent knowledge state and demonstrated that LFKT and FAST are outperformed by BKT. We hypothesize that this might be due to parameter identifiability issues for LFKT and FAST in that multiple sequences of latent states are equally good at predicting the observed task outcomes. The findings raise the question of how to adjust the two recent methods LFKT and FAST if the aim is to predict knowledge states; we leave this exploration for future work.

Limitations. While all sampling parameters have been carefully chosen to match real-world conditions, we expect real-world data to exhibit more noise and additional effects not covered by our synthetic data. Thus, the

achieved performance can be considered an upper bound on the performance achievable in real-world settings. The performance of BKT depends on the quality of the underlying skill model. We have simulated imperfect skill models by introducing item effects, but we did not take other sources for imperfect skill models into account. Furthermore, the simulated data consisted of a fixed set of items. For tutoring systems offering many variations of tasks, reliable estimation of item effects is challenging, which in turn influences the performance of IRT, LFKT, and FAST. Moreover, the performance of FAST is driven by feature quality, which may vary between different tutoring systems.

Effect on learning. Despite the significant differences between the recent models and BKT and IRT, the improvements are small in magnitude across the parameter space. We show relative improvements in RMSE between models of up to 6%. The magnitude of the improvements is in line with recent work on student models. Even recent results from Piech et al. [2015] employing deep recurrent neural networks (named deep knowledge tracing) for the prediction of task outcomes demonstrated relative performance gains over standard BKT of only 9.6% on average (over different data sets). Moreover, in a recent study comparing these deep neural networks to variants of knowledge tracing model the maximal improvement when using deep knowledge tracing dropped to 3.6% [Piech et al., 2015; Khajah et al., 2016]. The effect of small-scale improvements in the accuracy of student models on the learning outcome has been discussed controversially [Beck and Gong, 2013; Yudelson et al., 2013]. According to [Beck and Gong, 2013], it is unclear how an improvement in accuracy on predicting the outcome of the next task translates to better teaching decisions within an ITS. They argue that the predictions from a student model have to be actionable withing the ITS. In another work, Beck and Xiong [2013] investigated the limits to the accuracy of student models on different real-world data sets. They conclude that most of the recent student models demonstrate comparable performance and are relatively close to the estimated upper bounds on the accuracy. The primary goal of student modeling is inferring student knowledge but as our results demonstrate improved prediction quality (of task outcomes) is not a guarantee for more accurate detection of knowledge states and recent models have mostly focused on task outcome prediction.

Expanding cognitive student models. For these reasons, expanding cognitive student models beyond modeling current knowledge states has gained interest recently. Efficient instructional policies based on student models have been developed that optimize teaching sequences [Muldner and Conati, 2007; Murray et al., 2004; Clement et al., 2015; Rafferty et al., 2011; Mandel et al., 2014]. Different policies for when-to-stop teaching a skill have

been explored [Käser et al., 2016; Rollinson and Brunskill, 2015]. Other work examined the optimal time interval between task repetitions [Pashler et al., 2009] to account for the spacing effect (learning is greater if task opportunities are spread over time) [Pavlik and Anderson, 2005]. Further, new methods for evaluating student models have been explored that assess the quality of student models in terms of student effort and the learning outcome using instructional policies based on these student models [Gonzalez-Brenes and Huang, 2015]. Other work has focused on building predictors for knowledge retention that allow predicting future student performance after a delay of up to 10 days [Wang and Beck, 2012]. Further, regression methods for the detection of shallow learning (knowledge cannot be transferred to similar tasks) have been developed that allow for personalizing interventions that have shown to be effective for fostering deep learning [Baker et al., 2012]. Other work focused on modeling answer times based on student abilities that has the potential to improve task selection by planning for the expected amount of time required to solve a task [Jarušek and Pelánek, 2012].

Motivated by our findings and the various research directions on expanding cognitive student models, we develop methods for the identification and discovery of student characteristics and traits in the remainder of this work. Knowledge about traits and student characteristics can serve as valuable and actionable input to practitioners (e.g. a child with a specific learning disability can receive special tutoring). Also, the prediction of student traits can serve as an input to student models such as FAST (presented in this chapter) to build student models that allow for better tailoring of the learning experience to individual students.

CHAPTER 5

Supervised student trait identification

Parts of this chapter are based on the work presented in [Käser, 2014] and [Klingler, 2013]. The theoretical derivations of the models are detailed in these works.

Many interesting student characteristics and traits are often not directly observable within a learning system. However, student characteristics such as learning disabilities, personality traits or engagement are known to influence the learning outcome of students. Pure knowledge models that do not account for such characteristics have shown only marginal performance improvements over the past years and the effect on the learning outcome of such marginal improvements has been questioned (see Chapter 4 for a discussion). Automatic assessment of student characteristics such as learning disabilities can not only help in improving these student models and in better tailoring the learning experience to the individual student but can also help educators and teachers to better adapt to the specific learning needs of students. In this chapter, we present a method for the automatic, supervised detection of student traits. We propose a pipeline for integrating automatic assessment, i.e. detectors of student traits, directly into an intelligent tutoring system (ITS). We validate our approach for the case of developmental dyscalculia (DD) and the game-based training environment `Calcularis`.

Our pipeline leverages the potential of machine learning algorithms and relies on log data from student interactions only. Its data-driven nature features several advantages. First, since it builds upon a large set of student training data, the costs for model building are low, and the accuracy of the classifier can be continuously improved as more student data is added over time. The test duration can be adapted to each child individually, which

reduces the average test length substantially. Second, our classifier can be seamlessly embedded into an ITS (in our case `Calcularis` [Käser et al., 2013c]), where the assessment runs continuously and non-intrusively in the background. This integration reduces testing expenses and emotional stress imposed on children is kept to a minimum. The embedding allows the ITS to leverage the information from the stealth assessment during the training. Third, our pipeline has the potential to be applied to a different ITS and be used for the assessment of various student traits. Fourth, our method only relies on interaction logs (no additional sensor data is needed), which allows our pipeline to be readily integrated into a variety of ITS.

One of the main challenges of automatic computer-based assessments is to show equivalence to conventional assessment regarding accuracy, practicability, and validity [Jenkins et al., 2007]. Therefore, we extensively evaluate the accuracy, practicability, and validity of our approach on data logs from 68 children. Our results demonstrate that we can identify children at risk of DD with a high accuracy (91% sensitivity, 91% specificity) within a short time (11 minutes on average). We conclude from our results that recorded user inputs alone could potentially allow for a detailed reconstruction of student traits and that the integration of stealth assessments may refine the adaptation of the curriculum that ITS are currently providing.

5.1 Adaptive Classification Algorithm

Our adaptive classification is based on the training environment `Calcularis` [Käser et al., 2013c] that we introduced in Section 3.2. As a reminder, `Calcularis` is a computer-based system for learning mathematics designed for children with developmental dyscalculia (DD). The program is structured into different instructional games, which are designed based on current neurocognitive theory. `Calcularis` consists of ten different games representing 100 different skills that are essential for learning mathematics.

Our model building process consists of four steps. We first extract a large set of candidate features and then perform feature selection based on common similarity measures. Next, we build our adaptive classifier by first sorting the selected features and then defining a Naive Bayes model. An overview of the processing pipeline is shown in Figure 5.1.

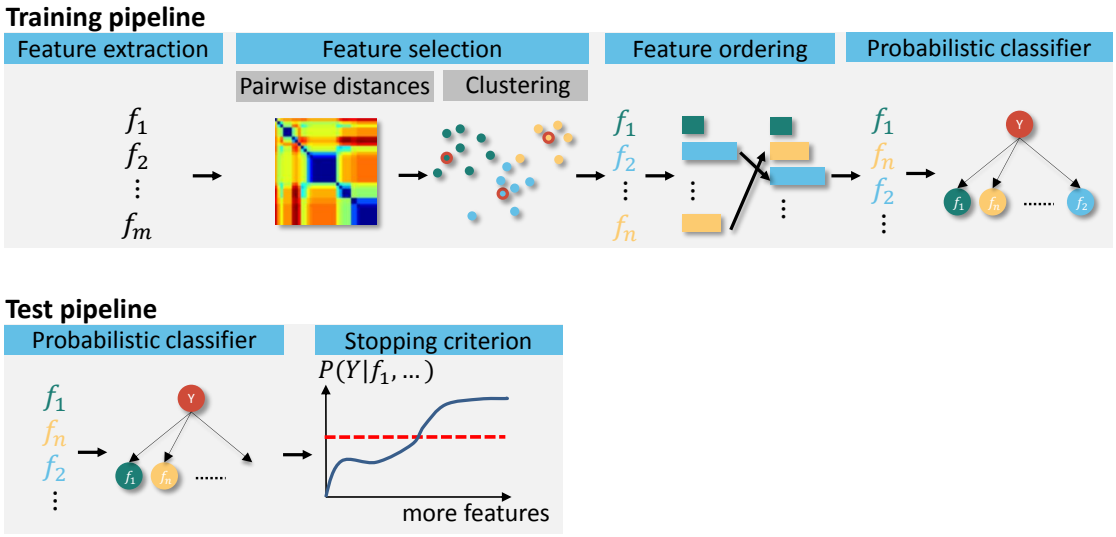


Figure 5.1: *Processing pipeline: Pairwise distances of features f serve as input for the clustering. We select the representative feature per cluster and determine an optimal feature ordering. A Naive Bayes model is trained on the selected features. The probabilistic output of the classifier is used to adapt the test duration to each child.*

5.1.1 Feature extraction

We identified a set of recorded features that describe different mathematical properties of the user. These features can be classified into *skill-* and *game dependent* features, and are summarized in Table 5.1. *Skill dependent* features provide information about tasks associated with a specific skill and *game dependent* features provide information unique to a particular game.

Performance. The performance **P** for a skill measures the ratio of correctly solved tasks for a given number of tasks. We expect children without DD to outperform children with DD on these tasks since mathematical abilities of children with DD are at a level comparable to the level of children without DD of lower age [von Aster, 2000].

Answer time. Answer time **AT** is measured for all skills as children with DD tend to have longer answer times compared to children without DD [Geary et al., 1991]. They often show deficits in fact retrieval and tend to have difficulties in acquiring arithmetic procedures [Ostad, 1997] which increases answer times for simple arithmetic tasks.

Typical mistakes. We count typical mistakes **TM** for a subset of games where such a measure is meaningful. **TM** are extracted by matching the erroneous result to a set of error patterns. As an example switching the dig-

Table 5.1: *Extracted features and abbreviations (bold) used for the detection of developmental dyscalculia.*

Feature	Description
Skill dependent features (extracted for specific skills)	
Performance	Ratio of correctly solved tasks.
Answer **T**ime	Average answer time.
Typical **M**istakes	Number of typical mistakes committed.
Game dependent features	
Estimation	Estimating the number of displayed points.
	E is the ratio between number of overestimates and task count.
Secret **N**umber	Guessing a number in as few steps as possible.
	S is the ratio by which the remaining search interval is reduced.
Ordering	Is a number sequence ordered ascending?
	O is the ratio of false positive and incorrectly solved tasks.
Landing	Positioning a number on a number line.
	L is the distance to the correct position of the given number).

its of the result in an arithmetic task is considered a typical mistake (e.g. 15 + 9 = 42). The complete set of error patterns is described in [Käser et al., 2013c].

Game dependent features. Additional *game dependent* features were chosen related to specific games. The estimation game feature **E** measures the relative number of overestimates when estimating the number of points in a point cloud. Whether children with DD are less sensitive to differences in this number representation is not consistently supported by recent work [Noël and Rousselle, 2011]. The feature **SN** for the secret number game measures the reduction of the search interval while repeatedly guessing the same number. This feature quantifies common problem-solving strategies such as bisection of the search interval or linear search. The ordering game feature **O** measures the ratio of false positives when assessing whether numbers are in ascending order. Children with DD are shown to be less efficient when processing numbers [Landerl et al., 2004], therefore we hypothesize that they will perform worse when comparing numbers. The landing game feature **L** measures the error of the number estimate. Deficits in spatial number representation as often shown by children with DD [Kucian et al., 2011] are obstructive to this task; thus we expect children with DD to perform significantly worse compared to peers without DD.

5.1.2 Feature selection

Our feature extraction yields a few hundred features, each corresponding to a set of tasks the user has to solve. Therefore, the number of features directly influences the test duration. To limit the test duration and to remove possible correlations between features, we only use a subset of features for classification. We cluster the features into groups based on their similarity and select one representative feature per cluster. As the different feature types have different domains (e.g., $\mathbf{P} \in [0,1]$, **AT** seconds > 0) a direct comparison between the features is not meaningful. We therefore process the features to make them comparable.

In a first step, we compute a similarity matrix $\mathbf{K_i} \in [0,1]^{S \times S}$ for each feature f_i, where S denotes the number of children. Therefore, $\mathbf{K_i}$ contains the pairwise similarities between each pair of children regarding feature f_i. We design the matrices based on the nature of each feature and in particular exploiting invariances of the feature types. As the basis for all kernels, we employ a Gaussian kernel given by

$$\mathbf{K_i}(s,u) = \exp\left(-\frac{\left\|f_i^s - f_i^u\right\|^2}{2\sigma^2}\right) \tag{5.1}$$

where f_i^s and f_i^u denote the respective feature values for children s and u and $\sigma > 0$. The Gaussian kernel is shift-invariant, and the projection of the feature values for any user u have unit length $\mathbf{K_i}(u,u) = \exp(0) = 1$. Furthermore, the parameter σ controls how sensitive the kernel is. For small values of σ, the kernel matrix is close to the identity matrix, whereas for large values the matrix tends to a uniform matrix.

For the answer time features **AT** we combine a Gaussian kernel with a log transform to obtain

$$\mathbf{K_i^{AT}}(s,u) = \exp\left(-\frac{\left\|\log(f_i^s) - \log(f_i^u)\right\|^2}{2\sigma^2}\right). \tag{5.2}$$

The log transform has proven useful in similar applications [Baschera et al., 2011] and the shift-invariance is a natural requirement. For a single observed answer time, we cannot decide whether the task was solved quickly or not. Only when comparing to answer times of other users, such a conclusion can be drawn.

We incorporate a cumulative beta distribution within the standard Gaussian

kernel to design the similarity matrix for the performance features **P**

$$\mathbf{K_i^P}(s,u) = \exp\left(\frac{\left\|\text{betacdf}(f_i^s) - \text{betacdf}(f_i^u))\right\|^2}{2\sigma^2}\right). \tag{5.3}$$

The beta cumulative distribution function allows to emphasize differences at the boundary of possible performance values (the performance on a task is bounded by 0 and 1).

For the secret number **SN** feature, we designed an exponential kernel

$$\mathbf{K_i^{SN}}(s,u) = \exp\left(\frac{\left\|\exp(-f_i^s) - \exp(-f_i^u) - 2\right\|^2}{2\sigma^2}\right). \tag{5.4}$$

The kernel is designed to be more sensitive to small differences for valid guesses (inside the search interval) than for invalid guesses (outside of the search interval).

For all other features (**TM**, **L**, **O**, **E**) we apply a standard Gaussian kernel. Further details regarding the design of the different kernels can be found in [Käser, 2014].

In a second step, we cluster the features using pairwise-clustering [Hofmann and Buhmann, 1997] based on the pairwise distances $d_{ij} = \left\|\mathbf{K_i} - \mathbf{K_j}\right\|_F$ between all feature pairs using the Frobenius norm. We then compute an optimal matrix **T**, which contains the pairwise Hamming distances between child labels, i.e., $\mathbf{T}(s,u) = 0$ if s and u belong to the same group $\in \{DD, control\}$, with *control* referring to control group, and $\mathbf{T}(s,u) = 1$ otherwise. For each cluster, we select one representative feature, which is the one with the smallest distance $dt_i = ||\mathbf{K_i} - \mathbf{T}||_F$ to matrix **T**.

5.1.3 Probabilistic classifier

Based on the selected features, we develop a probabilistic model that adapts the test duration to the individual child. The classification task is solved using an adapted Naive Bayes model, which assumes conditional independence of all the features f_i given the group label Y ($Y = 0$ child with DD, $Y = 1$ control), but was shown to perform optimally even if the independence assumption is violated [Zhang, 2004]. Correlations between features are low in our case (average $\rho = 0.07$, $< 1\%$ significant correlations at $\alpha = 0.001$) because of our feature selection step. The posterior probability of the group label Y for a child given N observed features is proportional to

$$p(Y|f_1,...,f_N) \propto \prod_{i=1}^{N} p(f_i|Y) \cdot p(Y), \tag{5.5}$$

where for every feature we choose the density $p(f_i|Y)$ from a set of standard distributions that best models the data according to the Bayesian information criterion (BIC) score. We assume a normal distribution for the features **E**, **SN**, **O**, and **L**, and a Beta, Gamma, and Poisson distribution for **P**, **AT**, and **TM**, respectively. The prior probability $p(Y)$ is set to the estimated prevalence of DD [Shalev and von Aster, 2008]. Due to the independence assumption, we can deal with cases where we only observe a subset of all features. After observing the first feature f_1, we can compute $p(Y = 1|f_1)$. Having observed f_2, we infer $p(Y = 1|f_1, f_2)$, etc. For any threshold $\tau \in [0, 1]$, the predicted group label $\hat{Y}$ can then be computed as

$$\hat{Y} = \begin{cases} 1 & p(Y = 1|f_1, ..., f_n) > \tau \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

5.1.4 Feature ordering

To determine the optimal ordering of the tasks in the test, we compute the amount of group information contained in each feature. We prefer features where the feature values differ substantially across the groups (DD and control) and are similar within the group. To assess the quality of each feature f_i, we use an unpaired t-test for a difference in means of the two independent groups. We then order the features by sorting the calculated p-values in ascending order, *i.e.*, the feature with the smallest p-value is asked first.

5.1.5 Stopping criterion

The optimal point in time to stop the test is heuristically determined. After observing the first t features, the classifier has a current belief about the group label of a child and predicts the label based on $p(Y|f_1, \ldots, f_t) > \tau$ (see Equation (5.5)). Intuitively, we stop the test if observing the next feature would not contradict our current belief about the group label. As the next feature value f_{t+1} is unknown, the feature value in the training data $\hat{f}_{t+1}$ that contradicts the model's current belief the most is taken instead. We stop if observing $\hat{f}_{t+1}$ is not changing the current belief, i.e., if

$$p(Y = 1|f_1, ..., f_t) > \tau \quad \text{and} \quad p(Y = 1|f_1, ..., f_t, \hat{f}_{t+1}) > \frac{\tau}{2}.$$

5.2 Experimental Evaluation

We experimentally evaluated our proposed method on a real-world data set recorded during a user study with *Calcularis* for the specific case of developmental dyscalculia (DD). We aimed at investigating the following questions:

(i) *How do computer-extracted features from a game-based learning environment relate to neuropsychological findings on DD?*

(ii) *Can machine learning algorithms provide the same quality assessments of a mathematics disability as standard neuropsychological tests?*

To answer these questions, we assessed the validity of the proposed model, namely the content validity to investigate question (i), and the criterion-related and construct validities as well as the test duration for answering question (ii).

5.2.1 Method

The experimental evaluation of our method is based on log files from 68 participants (32 DD, 36 control) of a multi-center user study conducted in Germany and Switzerland [Von Aster et al., 2015]. The study was conducted with a total of 83 participants, but since the group of children in 4th and 5th grade (15 children) consisted entirely of children with DD, this group was excluded from all of the presented analysis in this chapter. During the study, children trained with *Calcularis* at home for five times per week during six weeks and solved on average 1551 tasks. There were 28 participants in the 2^{nd} grade (9 DD, 19 control) and 40 children in the 3^{rd} grade (23 DD, 17 control). The diagnosis of DD was based on standardized neuropsychological tests [von Aster et al., 2006; Haffner et al., 2005; Esser et al., 2008a]. In the following, we describe the participants of the entire study as well as the criteria applied to diagnose DD in more detail.

Participants. During the user study, 83 children (56 females, 27 males) trained with *Calcularis*. There were 47 participants with DD (34 females, 13 males) and 36 control children (22 females, 14 males) without DD. The average age of the children in the control group was 8.08 years (SD 0.48) and 9.06 years (SD 0.80) in the group of children with DD. There were 28 children (age M=7.97 years, SD=0.42; 21 females, 7 males, 9 children with DD, 19 control children) in 2nd grade, 40 children (age M=8.67, SD=0.66; 24 females, 16 males, 23 children with DD, 17 control children) in 3rd grade and 15 children (age M=9.68, SD=0.73; 11 females, 4 males, all 15 children with DD) in 4th and 5th grade. All participating children visited regular public schools and

were German-speaking. The children were required to train with *Calcularis* during six weeks with a frequency of 5 sessions per week. The duration of a single training session was 20 minutes. During the training, the input behavior and performance of the children was automatically recorded in log files. The present work includes only log files from children that completed at least 26 training sessions. The average number of tasks the participants solved during the training period is M=1541 (SD=231), where the group of control children solved M=1583 (SD=190) tasks on average and the children with DD M=1506 (SD=257) tasks on average.

Diagnosis. The diagnosis of DD was based on an estimation of the IQ of the participants as well as an assessment of their mathematical abilities. The *general intelligence* was assessed using the average T-score of four sub-tests, namely, the verbal intelligence and non-verbal intelligence subtests from BUEGA [Esser et al., 2008a]; the 'finding commonalities' and the mosaic subtest from the HAWIK-IV [Esser et al., 2008b; Petermann and Petermann, 2008]. The assessment of *mathematical performance*, on the other hand, was based on the average T-score of the four following sub-tests: Sub-tests for addition and subtraction from HRT, BUEGA calculation sub-test and the number line II sub-test of ZAREKI-R [Haffner et al., 2005; Esser et al., 2008a; von Aster et al., 2006]. All children in the user study have been preselected based on their IQ that ranges from 85 to 115. A DD diagnosis was made if a participant achieved a general intelligence score of $T \geq 42$ (resulting in an estimated IQ score of ≥ 85) and a mathematical performance score of $T \leq 40$. All the instruments used for the diagnosis are described in detail in Appendix A.

We calculated the accuracy, the specificity and the sensitivity of our model based on the predicted and the true label of the students (either DD or control). All results were computed on unseen students in the test set. Training and test sets were created using .632 bootstrap with resampling ($B = 300$). All parameter estimates are based on maximum likelihood estimation using Nelder-Mead simplex direct search. The optimization stops when the improvement in the likelihood is $< 10^{-4}$ or after 400 iterations. Hyperparameters (parameters for kernels and features) and features (including feature ordering) were selected using nested cross-validation, employing .632 bootstrap with resampling ($B = 300$) on top of 10-fold cross validation. The optimal number k^* of clusters in the feature selection step was heuristically determined by limiting the maximal test duration to <35 minutes. Since we required five recorded tasks per feature (average recorded task time: 0.39 minutes), this test duration results in $k^* = 17$ clusters (which leads to 85 tasks in the test).

Type/Nr.	Order	Skill
AT/1	2	Addition 2,1 TC* ('13+8=21')
AT/2	14	Point set estimation
AT/3	4	Subtraction 3,1 TC* ('122-7=115')
AT/4	8	Addition 3,1 TC* ('128+4=132')
AT/5	15	Are numbers sorted ascending
AT/6	12	Spoken to written number
P/1	5	Spoken to written number
P/2	10	Subtraction 2,2 TC* ('56-38=18')
P/3	1	Find neighbor numbers ±10
P/4	11	Spoken to written number
SN/1	7	Guess a number
SN/2	13	Guess a number
TM/1	3	Subtraction 2,1 TC* ('74-9=65')
TM/2	9	Assign spoken number to number line
TM/3	16	Addition 2,2 TC* ('23+18=41')
TM/4	6	Subtraction 2,2 ('48-36=12')
TM/5	17	Assign written number to number line

* TC : with carrying / borrowing

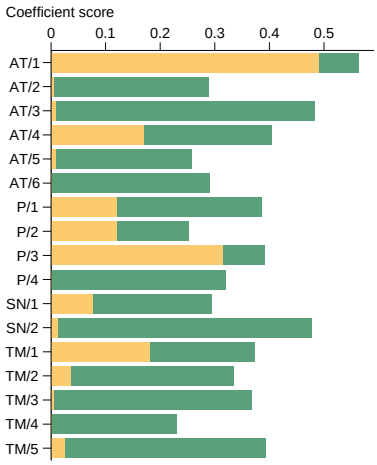


Figure 5.2: *Selected features and their corresponding skills and ordering in the test. The relationship between a feature and the test score is shown on the right, using Pearson's correlation coefficient (yellow) and the maximal information coefficient MIC (green).*

5.2.2 Content validity

The 17 features listed in Figure 5.2 were automatically selected based on the recorded data alone. For all features, we calculated Pearson's correlation coefficient ρ^2 and the maximal information coefficient (MIC) [Reshef et al., 2011] between the feature and the test score to measure the linear and non-linear relationships, respectively. For most features, the relationship is highly non-linear, which prohibits the use of simple prediction methods such as linear regression. The feature ordering yields the optimal task sequence in the test as listed in Figure 5.2.

The automatically selected features agree well with findings in previous work on DD. Deficits in number comparison that are shown by children with DD [Landerl et al., 2004] are captured by considering temporal and performance values (**AT**/5, **P**/3). Children with DD exhibit deficits in number processing [Cohen Kadosh et al., 2007]. Number processing skills are captured in various features and include again temporal and performance information (**AT**/2, **AT**/6, **P**/1, **P**/4). The features extracted from the number line game (**TM**/2, **TM**/5) capture typical mistakes in spatial number representation [Butterworth et al., 2011]. Furthermore, different problem-solving strategies are analyzed based on the Secret Number game (**SN**/1, **SN**/2). Finally, difficulties acquiring simple arithmetic procedures and deficits in fact retrieval that are frequently shown by children with DD [Ostad, 1997]

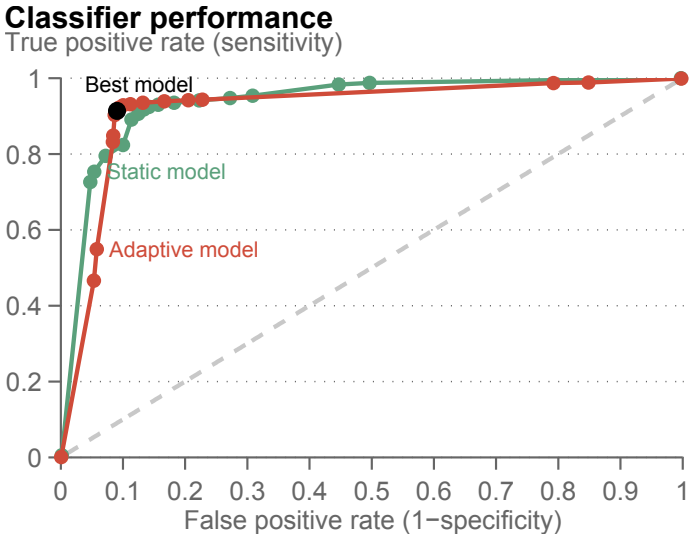


Figure 5.3: *Performance comparison of the classifiers using ROC curves. The adaptive approach with reduced test duration (red) shows comparable performance to the classifier using all features (green). Points on the curves correspond to different probability thresholds τ at which the model decides if a child has DD.*

are captured measuring answer times for various arithmetic procedures in **AT**/1, **AT**/3. Interestingly, no features from tasks associated with subitizing are selected, although subitizing is considered one of the basic functions often impaired for children with DD [Landerl et al., 2004].

Most of the selected features correspond well with the type of tasks used in standardized tests for DD such as counting, number comparison, number representation and simple arithmetical tasks [von Aster et al., 2006]. Note that our method makes use of some features such as typical mistakes and problem-solving strategies that are not captured by paper tests. Further, the type of the selected features agrees well with other computer-based screening tools that measure answer time, performance and typical mistakes on tasks such as dot enumeration, number comparison, single digit arithmetic (Dyscalculia Screener Digital [Butterworth, 2003]) or recognizing reading and writing of natural numbers (DyscaliUM [Beacham and Trott, 2005]).

5.2.3 Criterion-related validity

In Figure 5.3, we compare the performance of the static and adaptive Bayesian network model with ROC curves. In the static case (green line), we used all features, *i.e.*, all tasks, while in the adaptive case (red line) we

Table 5.2: *Spearman correlations ρ between the probabilistic output of our screener and various related and unrelated abilities of the study participants. Our method shows moderate to high correlations for related concepts and weak correlations to unrelated concepts. A description of all tests can be found in Appendix A.*

Test	ρ	p-value
Convergent validity		
Non verbal intelligence [Esser et al., 2008a]	0.44	$<10^{-3}$
Cognitive competence [Asendorpf and Van Aken, 1993]	0.63	$<10^{-7}$
Math anxiety test [Krinzinger et al., 2007]	0.42	$<10^{-2}$
Discriminant validity		
Working memory [Hasselhorn, 2012]	0.19	0.13
Verbal intelligence [Esser et al., 2008a]	0.23	0.06
Attentional performance [Zimmermann and Fimm, 2009]	0.25	0.10
Sport competence [Asendorpf and Van Aken, 1993]	-0.17	0.18
Peer acceptance [Asendorpf and Van Aken, 1993]	0.08	0.51

used early test abortion based on our stopping criterion. Every point on the curves corresponds to a different threshold τ for the probabilistic classifier. Our best classifier (selected by cross-validation) exhibits a high sensitivity and specificity of 0.91 for a threshold $\tau = 0.3$ (black dot). There is no significant decrease in performance when we stop the test early with our adaptive model, i.e., on average, children are not misclassified more frequently. In fact, the adaptive classifier that is based on partial data is outperforming the static approach for a specificity in the range $[0.05, 0.15]$. As the features are ordered based on how much information they carry about the group label, it can be advantageous to neglect those with little information since they tend to have more noisy information. Our classifier achieves a higher sensitivity compared to the stand-alone digital screening test DyscalculiUM; no comparison can be made with the Dyscalculia Screener Digital as it was standardized independent of traditional tests for DD.

5.2.4 Construct validity

Construct validity of our method was assessed by correlating the probabilistic output of our screener with a series of tests measuring different cognitive aspects of all participants. The tests were chosen to measure concepts related to DD such as *non-verbal intelligence* and *cognitive competence* as well as concepts that are not related to DD but can have an in-

fluence on the performance of students such as *verbal intelligence* or *attentional performance* (more details on the employed tests can be found in Appendix A). We performed standardized tests to assess the convergent validity and the discriminant validity as listed in Table 5.2. We observed moderate to high correlation coefficients for all measures capturing related cognitive concepts and weak correlations to the set of tests measuring unrelated concepts. These results are comparable to construct validity analysis of standardized neuropsychological tests that assess mathematical abilities. Correlations for these tests range from 0.22 to 0.73 [Woolger, 2001; Desoete and Grégoire, 2006].

5.2.5 Reliability

A lower bound on the test reliability is commonly measured with Cronbach's alpha. It measures the interrelatedness of the items in a test. However, in our setting two assumptions underlying Cronbach's alpha are violated. First, our test output is a non-linear function of the measurements and second the measurements are not tau-equivalent. We therefore investigated the split-half reliability of our proposed model. The average of all possible split-half reliabilities is equivalent to Cronbach's alpha under some assumptions [Cortina, 1993]. We repeatedly calculated the correlation between model outputs based on random half splits until we reached convergence for the estimate of the reliability (after 300 iterations). Splits were computed by randomly partitioning the five tasks for every feature into two sets. All correlations were corrected for the test length by using the Spearman-Brown formula. For our test, the Spearman-Brown correction is only an approximation as the assumptions of parallel tests are not met.

We observe a reliability of 0.87. This result is comparable to other mathematical tests where a reliability in the range of 0.7 to 0.92 is reported [Desoete and Grégoire, 2006; Esser et al., 2008a].

5.2.6 Test duration

Due to our stopping criterion, the test duration is adapted to the individual child. Figure 5.4, shows the test duration for all children (gray) and children with developmental dyscalculia (DD, red). On average, our adaptive screener classifies a child as DD or control after only 11 minutes (at which point the test is stopped). This is notably shorter than screener durations reported in previous work. In comparison, the test duration of the Dyscalculia Screener Digital is reported to be between 15 and 30 minutes [Butterworth,

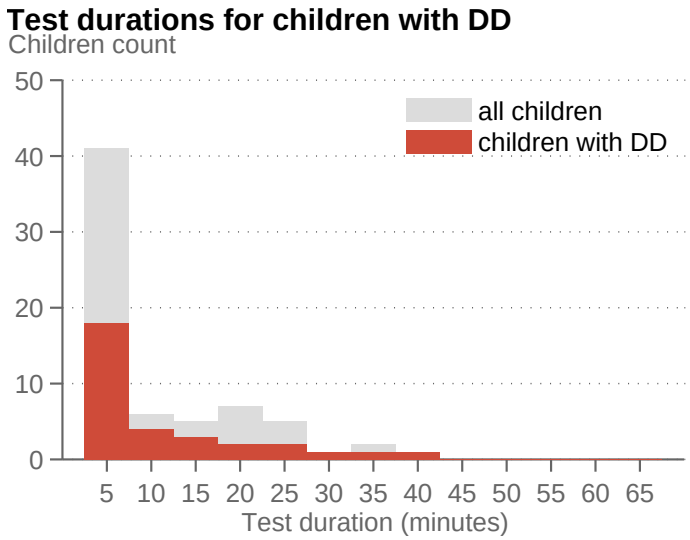


Figure 5.4: *Test durations for all children (grey) and DD (red). Our adaptive screener requires on average 11 test minutes to classify a child. Around 40% can be classified already after 5 minutes.*

2003]. For Higher Education, a test duration of 48 minutes was reported using the computer-based screener for DD DyscalculiUM. With our adaptive screener, roughly 40% of children are already classified after five test minutes. Our static screener test takes 26.6 minutes on average, which emphasizes the importance of the adaptivity. The adaptive stopping criterion is necessary to retain classification accuracy as for 43% of the children the initial classification changed until the stopping criterion was met.

5.3 Generalization Capabilities

ITS are used in many different settings, e.g. within a classroom, as a homework exercise, or self-training at home. Depending on the setting, the degree of support from parents or teachers, the amount of distraction by peers, and the motivation level of students is very different. Besides, there are multiple different neuropsychological tests to identify students at risk of developmental dyscalculia (DD). We would like our classification pipeline and features to be robust to different scenarios in that we would want to be able to detect children at risk of DD in all of those scenarios.

Specifically, to allow for early identification and intervention, a widespread, simultaneous screening of school classes would be beneficial. However, our classification pipeline has been trained on data from children using the

learning software at home. To assess the generalization capabilities of our developed model, we conducted an initial pilot study with ten Swiss school classes. For this study, we created a screening tool based on the training environment `Calcularis` that consisted of all tasks that were found relevant for the detection of DD in Section 5.2.

In the following, we first describe the participants of the study as well as the criteria to diagnose DD. Second, we investigate the generalizability of our feature set, and third, we examine the overall performance of our classifier for this changed setting.

5.3.1 Method

Participants. In the pilot study, the performance and interactions of 156 children (79 females, 77 males) with our screening tool were recorded. Eight participants were diagnosed with DD (6 females, 2 males). The average age of children without DD was 9.01 (SD 0.66) and 9.19 years (SD 0.48) in the group of children with DD. 98 participants attended the 2nd grade(age: M=8.69, SD=0.54; 52 females, 46 males; 3 children with DD) and 58 children attended the 3rd grade (age: M=9.58, SD=0.41; 27 females, 31 males; 5 children with DD). All participants visited regular public schools. The computer-based screener was executed directly in the classroom using laptop computers or in dedicated computer labs (depending on the availability at the school).

Diagnosis. The DD diagnosis was based on an estimation of the *general intelligence* as well as an assessment of the mathematical performance of the children. The *general intelligence* was estimated using subtests of the CFT1 [Cattell et al., 1997] or the CFT20-R [Weiss, 2006] (depending on the age). The assessment of the *mathematical performance* was based on the entire ZAREKI-R test [von Aster et al., 2006] and the addition and subtraction subtests from the HRT [Haffner et al., 2005]. A participant was diagnosed with DD if his or her estimated IQ score was in the range $85 - 115$ and one of the following conditions was true.

1. The total score achieved in ZAREKI-R was below the 10^{th} percentile
2. The scores of specified subtests of ZAREKI-R (see [von Aster et al., 2006] for an itemization) were below the 10^{th} percentile
3. The total score achieved in ZAREKI-R was at or below the 25^{th} percentile and the average performance of the addition and subtraction sub-tests from HRT was at or below the 10^{th} percentile.

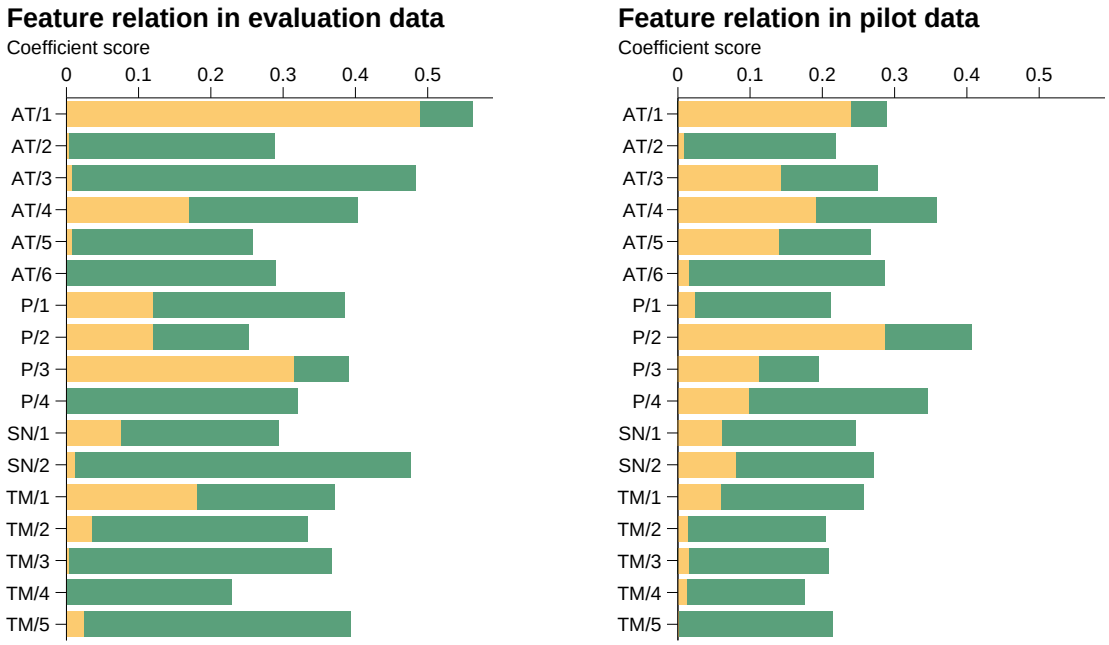


Figure 5.5: *The relationships between our selected features and the test score are shown for the evaluation data set (left) as well as the data set from the classroom study (right), using Pearson's correlation coefficient (yellow) and the maximal information coefficient MIC (green).*

All children solved our computer-based screener directly in the classroom. For this group setting, we trained our method on the data of our initial experimental evaluation (see Section 5.2). Children, therefore, had to solve 85 tasks associated with 17 selected features (detailed in Figure 5.2). The average test time of this non-adaptive screener version was 26 minutes. Due to the group setting, children were exposed to more distraction while solving the computer test; thus real-world testing conditions are well reflected.

We investigated the generalizability of the selected features to the changed test settings as well as the performance of our model under these settings. All results were obtained using the holdout method: the data set from the initial experimental evaluation was used for training and the data set from the pilot study for testing. All correlations were assessed using the phi coefficient and Spearman's rank correlation coefficient.

5.3.2 Feature generalizability

First, we investigated the applicability of the selected features in the classroom setting. We compared the MIC scores obtained on the 17 selected features to the MIC scores achieved in our initial experimental evaluation (see Figure 5.5). In general, features extracted in the classroom setting show lower MIC scores. We believe that this is due to the different tests that were used for assessing DD. Investigating the relative importance of the different feature types, we observe that in the classroom setting **AT** features are less relevant which might be caused by more distraction that increases the noise level for time measurements. These findings are in line with the lower scores for the **TM** features. Investigating the number of errors users made during the classroom study, we found that more users were committing typical mistakes for the features **TM**/3 (0.21 vs. 0.30), **TM**/4 (0.12 vs. 0.19), and **TM**/5 (0.15 vs. 0.39) compared to the evaluation data set. Again, this increased number of typical mistakes might be attributed to the group setting leading to more distraction.

5.3.3 Performance on classroom data

We analyzed the correlation between the screener output and the test score obtained from the standardized math tests (see Section 5.3.1) and found highly significant moderate correlations $r = 0.43, p < 10^{-7}$ between the two. For comparison, the correlation between ZAREKI-R and Arithmetic of WISC-III (Wechsler Intelligence Scale for Children) [Woolger, 2001] was found to be $r = 0.64$. Investigating the predictive performance of the model we found a promising but lower sensitivity and specificity compared to the results obtained for the evaluation data set. With changed study settings, we achieve a sensitivity of 0.63 while keeping the specificity at 0.85 ($\tau = 0.01$). Thus, the extracted features and the trained classifier capture important aspects of DD that generalize to the changed test settings. However, the low threshold $\tau = 0.01$ shows that our classifier is not well calibrated. In addition, the estimated sensitivity and specificity of our classification pipeline are not sufficient to be of practical use as a screening tool.

Investigating the predictions of misclassified students further, Figure 5.6 demonstrates that most classification errors were observed at the decision boundary for the DD diagnosis (see Section 5.3.1 for details on the diagnosis). More than 90% of the misclassified children show below average math performance according to our testing. As expected, the false positive rate de-

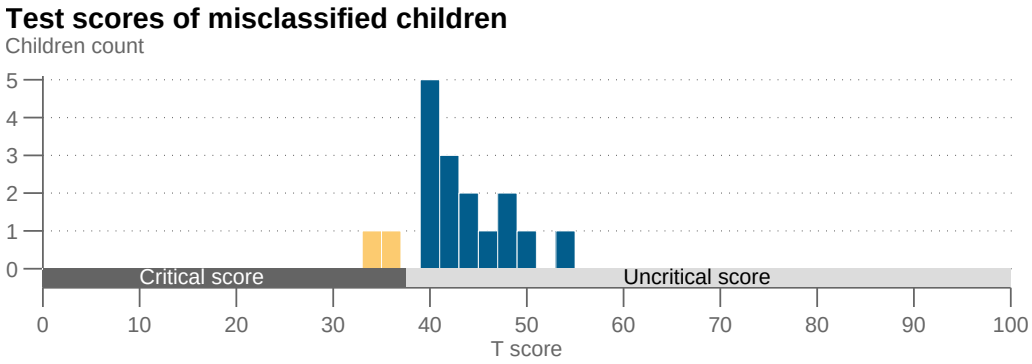


Figure 5.6: *Test scores of false negatives (yellow) and false positives (blue) children in the pilot study. Users with a critical score (T score $\leq$ 37.5) are diagnosed with DD.*

creases with increasing study scores. This decrease is a desirable property, as the children at the decision boundary are particularly challenging.

5.4 Discussion

In this chapter, we developed a fully data-driven pipeline for the automatic detection of student traits that can be seamlessly embedded into an ITS. We validated the method for the case of developmental dyscalculia (DD), allowing for non-intrusive and unsupervised screening of children while they are training with the ITS. The automatically selected features are covering a broad range of different characteristics of the children and are in accordance with the literature on DD. The classifier exhibits high sensitivity (0.91) and specificity (0.91) and adapts the test duration to each child individually, resulting in an average duration of as little as 11 minutes. Further, our method exhibits good construct validity (high correlations to tests measuring mathematical abilities, low correlations to tests assessing dissimilar abilities).

These findings demonstrate that student traits can be effectively learned from user inputs alone. This knowledge about student traits allows an ITS to further adapt the curriculum to the specific needs of the students. While we evaluated the proposed model only for the screening of children at risk of DD, there is nothing inherently DD specific in the method. As such, our framework can be applied for the unobtrusive detection of other student traits and using different learning environments.

Limitations. Based on a pilot study with 156 children we investigated the generalizability of the proposed model to a new test setting, where all chil-

dren are assessed simultaneously in the classroom. This new setting differs to the initial evaluation regarding two main aspects. First, this setting naturally leads to more disturbances since the test environment is noisier than in the evaluation study (where children used the system mostly at home) and children are interacting with each other during the test. Second, children in the pilot study solved all tasks consecutively while students in the evaluation study solved the tasks over the course of several weeks as part of their normal training with the software.

In this setting, our method exhibits significantly lower sensitivity and specificity. We found that our method can detect only about half of the children diagnosed with DD. The decrease in performance of our model is most likely due to the different changes of the data collection setting as well as timing effects due to different spacing of the tasks. These differences render our feature set less relevant for detecting DD.

To adapt our classifier to this new setting, we could re-train the model using the new data set from the pilot study. Since DD is a relatively rare condition, only 8 participants were diagnosed with DD. This small number of students provides only insufficient information about the characteristics of students with DD. Indeed, experiments for which we re-trained our Naive Bayes classifier on the classroom dataset, revealed that the learned models did not generalize to unseen students. In the next chapter, we, therefore, explore semi-supervised methods to learn more robust features based on a large unlabeled data set.

CHAPTER 6

Semi-supervised student trait identification

In order to build predictive models of student characteristics and traits, controlled user studies are typically conducted as we have done in Chapter 5. Such controlled studies provide detailed information about student characteristics (labeled data). However, gathering labeled data in educational data mining is a time and cost-intensive task, which most often limits the number of study participants due to time and budget constraints. The quality of the predictive models, however, inherently depends on the number of study participants. When building models for relatively rare conditions this issue becomes especially pronounced as data sets from small-scale studies can provide only a few examples of the rare condition as we demonstrated in Chapter 5. In contrast to such controlled user studies, digital learning environments such as intelligent tutoring systems (ITS), educational games, learning simulations, and massive open online courses (MOOCs) produce high volumes of data. These data sets provide rich information about student interactions with the system but come with no or only little additional information about the user (unlabeled data).

Semi-supervised learning bridges this gap by making use of patterns in bigger unlabeled data sets to improve predictions on smaller labeled data sets. This is also the focus of this chapter. Recently, it has been shown (outside of the education context) that variational auto-encoders (VAE) have the potential to outperform the commonly used semi-supervised classification techniques. A VAE is a neural network that includes an encoder that transforms a given input into a typically lower-dimensional representation,

and a decoder that reconstructs the input based on the latent representation. Hence, VAEs learn an efficient feature embedding (feature representation) using unlabeled data that can be used to improve the performance of any standard supervised learning algorithm [Kingma et al., 2014]. This property greatly reduces the need for problem-specific algorithms. Moreover, VAEs feature the advantage that the trained deep generative models are able to produce realistic samples that allow for accurate data imputation and simulations [Rezende et al., 2014], which makes them an appealing choice for educational data mining.

Inspired by these advantages, and the demonstrated superior classifier performance in other domains such as computer vision [Kingma and Welling, 2014; Rezende et al., 2014], this chapter explores VAE for student classification in the educational context. We present a complete semi-supervised classification pipeline that employs deep VAEs to extract efficient feature embeddings from unlabeled student data. We have optimized the architecture of two different networks for educational data - a simple variational auto-encoder and a convolutional variational auto-encoder. While our method is generic and hence widely applicable, we apply the pipeline to the problem of detecting students suffering from developmental dyscalculia (DD), as in Chapter 5. The large and unlabeled data set at hand consists of student data of more than 7K students, and we evaluate the performance of our pipeline on the two independent small and labeled data sets that we presented in Chapter 5 with 83 and 155 students, respectively.

Our evaluation first compares the performance of the two networks, where our results indicate the superiority of the convolutional VAE. We then apply different classifiers to both labeled data sets, and demonstrate not only improvements in classification performance of up to 28% compared to other feature extraction algorithms, but also improved robustness to class imbalance when using our pipeline compared to other feature embeddings. The improved robustness of our VAE is of particular importance for predicting relatively rare student conditions - a challenge that is often met in educational data mining applications.

6.1 Background

In the semi-supervised classification setting, we have access to a large data set $\mathcal{X}_B$ without labels and a much smaller labeled data set $\mathcal{X}_S$ with labels $\mathcal{Y}_S$. The idea behind semi-supervised classification is to make use of patterns in the unlabeled data set to improve the quality of the classifier beyond what

would be possible with the small data set $\mathcal{X}_S$ alone. There are many different approaches to semi-supervised classification including transductive SVMs, graph-based methods, self-training or representation learning [Zhu, 2006].

In this chapter, we focus on learning an efficient encoding $\mathbf{z} = E(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X}_B$ of the data domain using the unlabeled data $\mathcal{X}_B$ only. This learned data transformation $E(\cdot)$ - the encoding - is then applied to the labeled data set $\mathcal{X}_S$. Well-known encoders include principle component analysis (PCA) or Kernel PCA (KPCA). PCA is a dimensionality reduction method that finds the optimal linear transformation from an N-dimensional to a K-dimensional space (given a mean-squared error loss). Kernel PCA [Schölkopf et al., 1997] extends PCA allowing non-linear transformations into a K-dimensional space and has, among others, been successfully used for novelty detection in non-linear domains [Hoffmann, 2007].

Recently, variational auto-encoders (VAE) have outperformed other semi-supervised classification techniques on several data sets [Kingma et al., 2014]. VAE combine variational inference networks with generative models parametrized by deep neural networks that exploit information in the data density to find efficient lower dimensional representations (feature embeddings) of the data.

6.1.1 Auto-encoder

An auto-encoder or autoassociator [Bengio and others, 2009] is a neural network that encodes a given input into a (typically lower dimensional) representation such that the original input can be reconstructed approximately. The auto-encoder consists of two parts. The encoder part of the network takes the N-dimensional input $\mathbf{x} \in \mathbb{R}^N$ and computes an encoding $\mathbf{z} = E(\mathbf{x})$ while the decoder D reconstructs the input based on the latent representation $\hat{\mathbf{x}} = D(\mathbf{z})$. If we train a network using the mean squared error loss and the network consists of a single linear hidden layer of size K, e.g.

$$E(\mathbf{x}) = \mathbf{W_1}\mathbf{x} + \mathbf{b_1} \quad \text{and} \quad D(\mathbf{z}) = \mathbf{W_2}\mathbf{z} + \mathbf{b_2} \tag{6.1}$$

for weights $\mathbf{W}_1 \in \mathbb{R}^{K \times N}$ and $\mathbf{W}_2 \in \mathbb{R}^{N \times K}$ and offsets $\mathbf{b}_1 \in \mathbb{R}^K$ and $\mathbf{b}_2 \in \mathbb{R}^N$, the autoencoder behaves similar to PCA in that the network learns to project the input into the span of the K first principle components [Bengio and others, 2009].

For more complex networks with non-linear layers, multi-modal aspects of the data can be learned. Auto-encoders can be used in semi-supervised classification tasks because the encoder can compute a feature representation $\mathbf{z}$

of the original data $\mathbf{x}$. These features can then be used to train a classifier. The learned feature embedding facilitates classification by clustering related observations in the computed latent space.

6.1.2 Variational auto-encoder

Variational auto-encoders [Kingma et al., 2014] are generative models that combine Bayesian inference with deep neural networks. They model the input data $\mathbf{x}$ as

$$p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x};\mathbf{z},\theta) \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0,I) \tag{6.2}$$

where f is a likelihood function that performs a non-linear transformation with parameters θ of $\mathbf{z}$ by employing a deep neural network. In this model, the exact computation of the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is not computationally tractable. Instead, the true posterior is approximated by a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ [Kingma and Welling, 2014]. This inference network $q_\phi(\mathbf{z}|\mathbf{x})$ is parametrized as a multivariate normal distribution as

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \mathrm{diag}(\sigma^2_\phi(\mathbf{x}))), \tag{6.3}$$

where $\mu_\phi(\mathbf{x})$ and $\sigma^2_\phi(\mathbf{x})$ denote vectors of means and variance respectively. Both functions $\mu_\phi(\cdot)$ and $\sigma^2_\phi(\cdot)$ are represented as deep neural networks. Hence, variational autoencoders essentially replace the deterministic encoder $E(\mathbf{x})$ and decoder $D(\mathbf{z})$ by a probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder $p_\theta(\mathbf{x}|\mathbf{z})$.

Direct maximization of the likelihood is computationally not tractable. Therefore, a lower bound on the likelihood has been derived [Kingma and Welling, 2014]. The learning task then amounts to maximizing this variational lower bound

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right], \tag{6.4}$$

where KL denotes the Kullback-Leibler divergence. The lower bound consists of two intuitive terms. The first term is the reconstruction quality while the second one regularizes the latent space towards the prior $p(\mathbf{z})$. We perform optimization of this lower bound by applying a stochastic optimization method using gradient back-propagation [Kingma and Ba, 2015].

6.2 Method

In the following, we introduce two networks. First, a simple variational auto-encoder consisting of fully connected layers to learn feature embed-

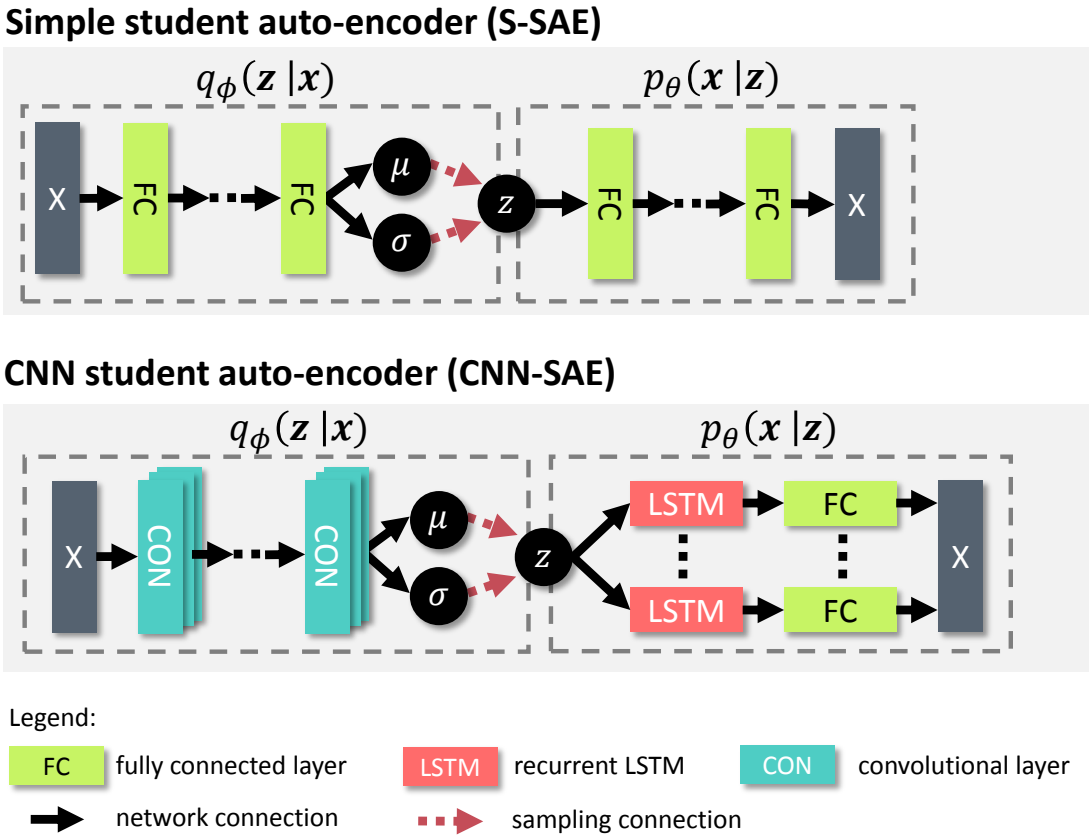


Figure 6.1: *Network layouts for our simple student auto-encoder (top) using only fully connected layers and our improved CNN student auto-encoder (bottom) using convolutions for the encoder and recurrent LSTM layers for the decoder. In contrast to standard auto-encoders, the connections to the latent space z are sampled (red dashed arrows) from a Gaussian distribution.*

dings of student data. These encoders have shown to be powerful for semi-supervised classification [Kingma et al., 2014], and are often applied due to their simplicity. Second, an advanced auto-encoder that combines the advantages of VAE with the superiority of asymmetric encoders. This second network is motivated by the fact that asymmetric auto-encoders have shown superior performance and more meaningful feature representations compared to simple VAE in other domains such as image synthesis [van den Oord et al., 2016].

6.2.1 Student snapshots

There are many applications where we want to predict a label y_n for each student n within an ITS based on behavioral data X_n. These labels typically relate to external variables or properties of a student, such as age, learning disabilities, personality traits, learner types, or the learning outcome. Similar to Bayesian Knowledge Tracing (see Section 4.1.1) we propose to model the data $X_n = \{\mathbf{x}_{n1}, \ldots, \mathbf{x}_{nT}\}$ as a sequence of T observations. In contrast to Bayesian Knowledge Tracing, we store F different feature values $\mathbf{x}_{nt} \in \mathbb{R}^F$ for each element in the sequence, where t denotes the t^{th} opportunity within a task. This approach allows us to simultaneously store data from multiple tasks in $\mathbf{x}_{nt}$, e.g. $\mathbf{x}_{n1}$ stores all features for student n that were observed during the first task opportunities. For every task in an ITS, we can extract various features that characterize how a student n was approaching the task. These features include performance, answer times, problem-solving strategies, etc. We combine this information into a student snapshot $\mathbf{X}_n \in \mathbb{R}^{T \times F}$, where T is the number of task opportunities, and F is the number of extracted features.

6.2.2 Simple student auto-encoder (S-SAE)

Our simple variational autoencoder is following the general design outlined in Section 6.1 and is based on the student snapshot representation. For ease of notation, we use $\mathbf{x} := \text{vec}(\mathbf{X_n})$, where $\text{vec}(\cdot)$ is the matrix vectorization function to represent the student snapshot of student n.

The complete network layout is depicted in Figure 6.1, top. The encoder and decoder networks consist of L fully connected layers that are implemented as an affine transformation of the input followed by a non-linear activation function $\beta(\cdot)$ as

$$\mathbf{x}_l = \beta(\mathbf{W}_l \mathbf{x}_{l-1} + \mathbf{b}_l), \tag{6.5}$$

where l is the layer index and $\mathbf{W}_l$ and $\mathbf{b}_l$ are a weight matrix and offset vector of suitable dimensions. Typical choices for $\beta(\cdot)$ include tanh, rectified linear units or sigmoid functions [Goodfellow et al., 2016]. To produce latent samples $\mathbf{z}$ we sample from the normal distribution (see Equation (6.3)) using re-parametrization [Kingma and Welling, 2014]

$$\mathbf{z} = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\epsilon, \tag{6.6}$$

where $\epsilon \sim \mathcal{N}(0, 1)$, to allow for back-propagation of gradients. For $p_\theta(\mathbf{x}|\mathbf{z})$ (see (6.2)) any suitable likelihood function can be used. We used a Gaussian distribution for all examples in this chapter. Note that the likelihood function is parametrized by the entire (non-linear) decoder network.

The training of variational auto-encoders can be challenging as stochastic optimization was found to set $q_\phi(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ in all but vanishingly rare cases [Bowman et al., 2016], which corresponds to a local maximum that does not use any information from $\mathbf{x}$. We, therefore, add a warm-up phase that gradually gives the regularization term in the target function more weight:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \alpha \,\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right], \tag{6.7}$$

where $\alpha \in [0, 1]$ is linearly increased with the number of epochs. The warm-up phase has been successfully used for training deep variational auto-encoders [Sønderby et al., 2016]. Furthermore, we initialize the weights of the dense layer computing $\log(\sigma^2_\phi(\mathbf{x}))$ to 0 (yielding a variance of 1 at the beginning of the training). This was motivated by our observations that if we employ standard random weight initialization techniques (glorot-norm, he-norm [He et al., 2015]) we can get relatively high initial estimates for the variance $\sigma^2_\phi(\mathbf{x})$, which due to the sampling leads to very unreliable samples $\mathbf{z}$ in the latent space. The large variance in sampled points in the latent space leads to bad convergence properties of the network.

6.2.3 CNN student auto-encoder (CNN-SAE)

Following the recent findings in computer vision, we present a second, more advanced network that typically outperforms simpler variational auto-encoders. In [van den Oord et al., 2016], for example, these asymmetric auto-encoders resulted in a superior reconstruction of images as well as more meaningful feature embeddings. A specific kind of convolutional neural network was combined with an auto-encoder, being able to capture low-level pixel statistics directly and hence to extract more high-level feature embeddings.

Inspired by this previous work, we combine an asymmetric auto-encoder (and a decoder that is able to capture low-level statistics) with the advantages of variational auto-encoders. Figure 6.1, bottom, shows our combined network. We employ multiple layers of one-dimensional convolutions to parametrize the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ (again we assume a Gaussian distribution, see (6.3)). The distribution is parameterized as follows:

$$\begin{aligned} \mu_\phi(\mathbf{x}) &= \mathbf{W}_\mu \mathbf{h} + \mathbf{b}_\mu \\ \log(\sigma^2_\phi(\mathbf{x})) &= \mathbf{W}_\sigma \mathbf{h} + \mathbf{b}_\sigma \\ \mathbf{h} &= \mathrm{conv}_l(\mathbf{x}) = \beta(\mathbf{W}_l * \mathrm{conv}_{l-1}(\mathbf{x})), \end{aligned}$$

where $*$ is the convolution operator, $\mathbf{W}_l, \mathbf{W}_\mu, \mathbf{W}_\sigma, \mathbf{b}_\mu, \mathbf{b}_\sigma$ are weights of suitable dimensions, $\beta(\cdot)$ is a non-linear activation function, and l denotes the

layer depth. Further, $\text{conv}_0(\mathbf{x}) = \mathbf{x}$. We keep the standard variational layer (see (6.6)) while changing the output layer to a recurrent layer using long term short term units (LSTM). Recurrent layers have successfully been used in auto-encoders before, e.g. in [Fabius and van Amersfoort, 2015]. LSTM were very successful for modeling temporal sequences because they can model long and short term dependencies between time steps. Every LSTM unit receives a copy of the sampled points in latent-space, which allows the LSTM network to combine context information (point in the latent space) with the sequence information (memory unit in the LSTM cell). Using LSTM cells the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ assumes a Gaussian distribution and is parametrized as follows:

$$\begin{aligned}\mu_{\theta t}(\mathbf{z}) &= \mathbf{W}_{\mu z} \cdot \text{lstm}_t(\mathbf{z}) + \mathbf{b}_{\mu z} \\ \log(\sigma^2_{\theta t}(\mathbf{z})) &= \mathbf{W}_{\sigma z} \cdot \text{lstm}_t(\mathbf{z}) + \mathbf{b}_{\sigma z},\end{aligned}$$

where $\mu_{\theta t}(\mathbf{z})$ and $\sigma^2_{\theta t}(\mathbf{z})$ are the t^{th} components of $\mu_\theta(\mathbf{z})$ and $\sigma^2_\theta(\mathbf{z})$, respectively, $\text{lstm}_t(\cdot)$ denotes the t^{th} LSTM cell and $\mathbf{W}_*$ and $\mathbf{b}_*$ denote suitable weight and offset parameters.

6.2.4 Feature selection

Variational auto-encoders provide a natural way for performing feature selection. The inference network $q_\phi(\mathbf{z}|\mathbf{x})$ infers the mean and variance for every dimension z_i. Therefore, the most informative dimension z_i has the highest KL divergence from the prior distribution $p(z_i) = \mathcal{N}(0, 1)$ while uninformative dimensions will have a KL divergence close to 0 [Higgins et al., 2016]. The KL divergence of z_i to $p(z_i)$ is given by

$$KL\left[q_\phi(z_i|\mathbf{x})||p(z_i)\right] = -\log(\sigma_i) + \frac{\sigma_i^2 \mu_i^2}{2} - \frac{1}{2}, \tag{6.8}$$

where μ_i and σ_i are the inferred parameter for the Gaussian distribution $q_\phi(z_i|\mathbf{x})$. Feature selection proceeds by keeping the K dimensions z_i with the largest KL divergence.

6.2.5 Semi-supervised classification pipeline

The encoder and the decoder of the variational auto-encoder can be used independently of each other. This independence allows us to take the trained encoder and map new data to the learned feature embedding. Figure 6.2 provides an overview of the entire pipeline for semi-supervised classification.

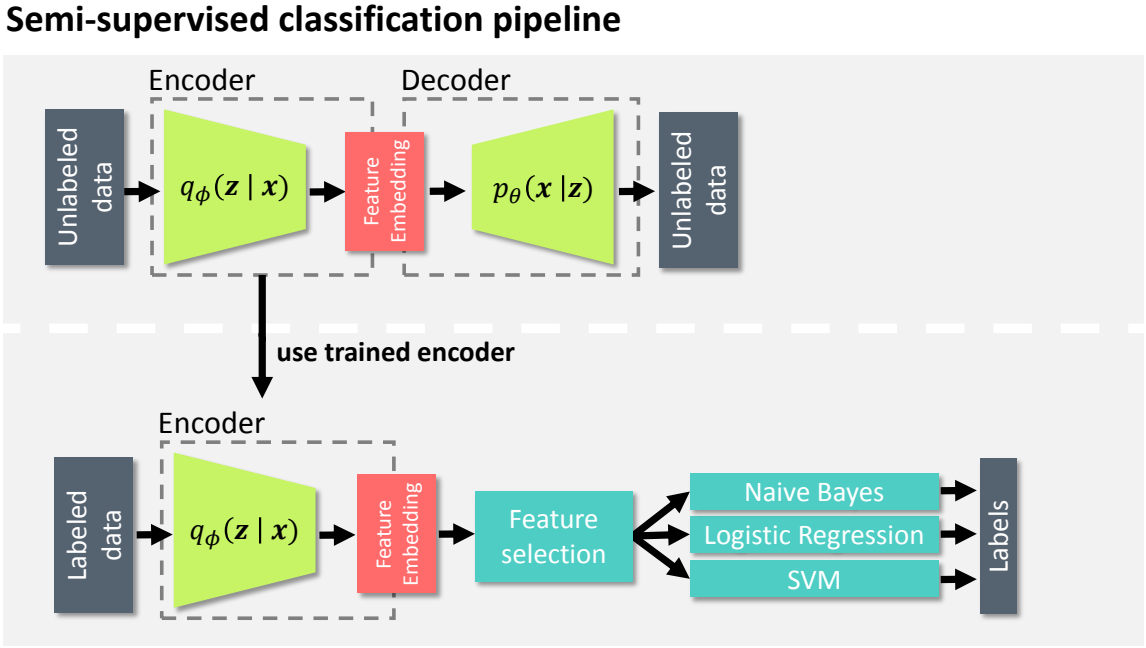


Figure 6.2: *Overview of our semi-supervised classification pipeline. We train the variational auto-encoder on a large unlabeled data set. The trained encoder of the auto-encoder can be used to transform other data sets into an expressive feature embedding. Based on this feature embedding we train different classifiers to predict the student labels.*

In a first unsupervised step, we train a VAE on unlabeled data. The learned encoder $q_\phi(\mathbf{z}|\mathbf{x})$ is then used to transform labeled data sets to the feature embedding. We finally apply our feature selection step that considers the relative importance of the latent dimensions as previously described. We then train standard classifiers: Logistic Regression, Naive Bayes and Support Vector Machine (SVM) on the feature embeddings.

6.3 Results

We evaluated our approach for the specific example of detecting developmental dyscalculia (DD), which is a learning disability affecting the acquisition of arithmetic skills [Von Aster and Shalev, 2007b]. Based on the learned feature embedding on a large unlabeled data set the classifier performance was measured on two independent, small and labeled data sets from controlled user studies (see Chapter 5 for a detailed description of the studies).

In this chapter, however, we refer to them as *balanced* and *imbalanced* data sets since their distribution of DD and non-DD children differs: the first study (the evaluation study described in Section 5.2.1) has approximately 50% DD, while the second study (the classroom study in Section 5.3.1) includes 5% DD which corresponds to the typical prevalence of DD.

6.3.1 Experimental Setup

All three data sets were collected from the intelligent tutoring system (ITS) *Calcularis* (see Section 3.2 for details). As a reminder, *Calcularis* consists of different games for training number representations and calculation. In Chapter 5 we identified a set of games that are predictive of DD within *Calcularis*. Since timing features were found to be one of the most relevant indicators for detecting DD [Butterworth, 2003] and to facilitate comparison to other feature embedding techniques we limited our analysis to log-normalized timing features, for which we can assume normal distribution [van der Linden, 2006]. Therefore, we evaluated our pipeline on the subset of games from Chapter 5 for which meaningful timing features could be extracted and sufficient samples were available in all data sets (we used > 7000 samples for training the VAEs). Since our pipeline currently does not handle missing data only students with complete data were included.

Timing features were extracted for the first 5 tasks in 5 different games. The selected games involve addition tasks (adding a 2-digit number to a 1-digit number with ten-crossing; adding two 2-digit numbers with ten-crossing), number conversion (spoken to written numbers in the ranges 0-10 and 0-100) and subtraction tasks (subtracting a 1-digit number from a 2-digit number with ten-crossing). For every task, we extracted the total answer time (time between the task prompt until the answer was entered) and the response time (time between the task prompt and the first input by the student). Hence, each student is represented by a 50-dimensional snapshot $\mathbf{x}$ (see Section 6.2).

Unlabeled data set. The unlabeled data set was extracted using live interaction logs from the ITS *Calcularis*. In total, we collected data from 7229 children. Note that we have no additional information about the children such as DD or grade. We excluded all teacher accounts as well as log files that were $<$ 20KB. A single user was excluded for technical reasons. Since every new game in *Calcularis* is introduced by a short video during the very first task, we excluded this particular task for all games.

Balanced data set. The first labeled data set was introduced in Section 5.2.1. In the following, we give a summary of the study: The data set is based on

log files from 83 participants of a multi-center user study conducted in Germany and Switzerland, where approximately half of the participants were diagnosed with DD (47 DD, 36 control) [Von Aster et al., 2015]. During the study, children trained with *Calcularis* at home for five times per week during six weeks and solved on average 1551 tasks. There were 28 participants in 2^{nd} grade (9 DD, 19 control), 40 children in 3^{rd} grade (23 DD, 17 control), 12 children in 4^{th} grade (12 DD) and 3 children in 5^{th} grade (3 DD). The diagnosis of DD was based on standardized neuropsychological tests [Von Aster et al., 2015].

Imbalanced data set. The second labeled data set was already introduced in Section 5.3.1. In the following, we give a summary of the study: The data set is based on a user study conducted in the classroom of ten Swiss elementary school classes. In total, 155 children participated, and a prevalence of DD of 5% could be detected (8 DD, 147 control). There were 97 children in 2^{nd} grade (3 DD, 94 control) and 58 children in 3^{rd} grade (5 DD, 53 control). The DD diagnosis was computed based on standardized tests assessing the mathematical abilities of the children [von Aster et al., 2006; Haffner et al., 2005]. During the study, children solved 85 tasks directly in the classroom. On average, children needed 26 minutes to complete the tasks.

6.3.2 Implementation

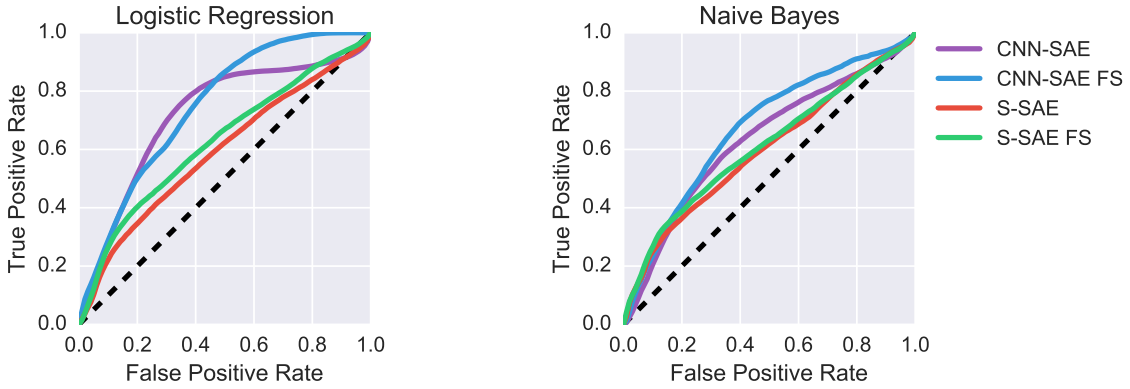
The unlabeled data set was used to train the unsupervised VAE for extracting compact feature embeddings of the data. Based on the learned data transformations we evaluated two standard classifiers: Logistic Regression (LR) and Naive Bayes (NB). We restricted our evaluation to simple classification models because we wanted to assess the quality of the feature embedding and not the quality of the classifier. More advanced classifiers typically perform a (sometimes implicit) feature transformation as part of their data fitting procedure. To represent at least one model that performs such an embedding we included Support Vector Machine (SVM) in all our results. All classifier parameters were chosen according to the default values in *scikit-learn* [Pedregosa and others, 2011]. Note that we have additionally performed randomized cross-validated hyperparameter search for all classifiers, which, however, resulted in marginal improvements only. Because of that, and to keep the model simple and especially easily reproducible, we use the default parameter set in this work. For Logistic Regression, we used L2 regularization with $C = 1$, for Naive Bayes, we used Gaussian distributions and for the SVM, RBF kernels and data point weights have been set inversely proportional to label frequencies. All results are cross-validated

using 30 randomized training-test splits on the unlabeled data (test size 5%). The classification part of the pipeline is additionally cross-validated using 300 label-stratified random training-test splits (test size 20%) to ensure highly reproducible classification results.

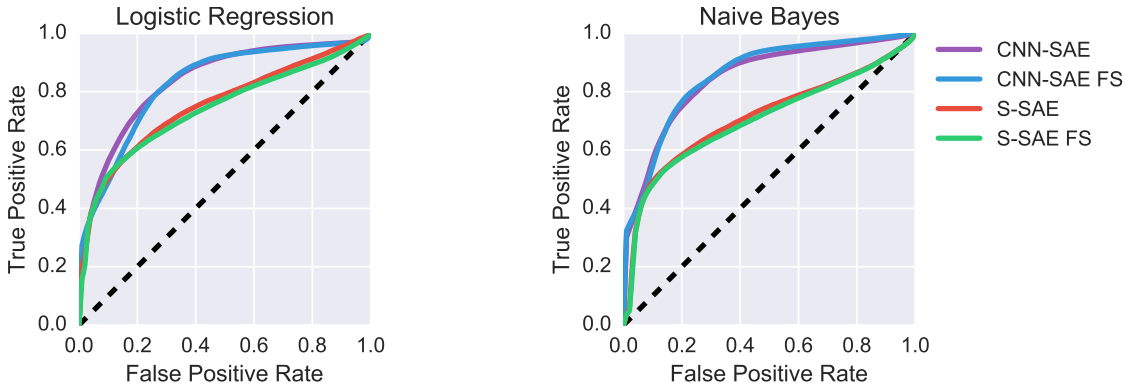
Network hyperparameters were defined using the approach described in [Bengio, 2012]. We increased the number of nodes per layer, the number of layers and the number of epochs until a good fit of the data was achieved. We then regularized the network using dropout [Srivastava et al., 2014] with increasing dropout rate until the network was no longer overfitting the data. Activation and weight initialization have been chosen according to common standards: We employ the most common activation function, namely rectified linear activation units (RELU) [LeCun et al., 2015], for all activations. Weight initialization was performed using the method by He et al. [He et al., 2015]. Following this procedure, the following parameters were used for the S-SAE model: encoder and decoders used 3 layers of size 320. The CNN-SAE model was parametrized as follows: 3 convolution layers with 64 convolution kernels and a filter length of 3. We used a single layer of LSTM cells with 80 nodes. We used a batch size of 500 samples and batch normalization and dropout ($r = 0.25$) at every layer. The warm-up phase (see Section 6.2) was set to 300 epochs. Training was stopped after 1000 (S-SAE) and 500 (CNN-SAE) epochs. The number of latent units was set to 8 in accordance with our findings on detecting students with DD from Chapter 5. There, we used 17 features for the prediction but found that about half of the features were sufficient to detect DD with high accuracy. When feature selection was applied, we set the number of features to $K = 4$ and thus we kept exactly half of the latent space features. All networks were implemented using the Keras framework [Chollet, 2015] with TensorFlow™ [Abadi et al., 2015] and optimized using Adam stochastic optimization with standard parameters according to [Kingma and Ba, 2015].

6.3.3 Network comparison

In a first experiment, we compared the feature embeddings generated by our simple S-SAE and our asymmetric CNN-SAE with and without feature selection. Figure 6.3 illustrates the average ROC curves of our complete semi-supervised classification pipeline. Our feature embeddings based on asymmetric CNN-SAE clearly outperform the ones from the simple S-SAE on both the imbalanced and the balanced data set for Naive Bayes (NB) and Logistic Regression (LR). For both models, feature selection improves the area under the ROC curve (AUC) for the imbalanced data set (CNN-SAE:



(a) Imbalanced data set



(b) Balanced data set

Figure 6.3: *ROC curves for the two proposed models with and without feature selection (FS). Our asymmetric CNN-SAE outperforms the simple S-SAE consistently with (blue) and without (purple) feature selection. Feature selection improves performance only on the imbalanced data set.*

LR 4.2%, NB 6.3%; S-SAE: LR 6.8%, NB: 1.6%), but has no effect for the balanced data set. We believe that this is due to the ability of the classifiers to distinguish useful features from noisy ones given enough samples. Since the performance of the classifiers with feature selection (FS) is better or equal to no feature selection in each experiment, we used the CNN-SAE FS model for all further evaluations.

6.3.4 Classification performance

Our VAE models are trained to extract efficient feature embeddings of the data. To assess the quality of these computed feature representations, we compare the classification performance of our method to previous techniques for finding efficient feature embeddings. In Figure 6.4 we compare

classification performance of our method based on VAE to two well-known methods for finding optimal feature embeddings, namely principle component analysis (PCA, green) and Kernel PCA (KPCA, red) [Schölkopf et al., 1997]. For comparison and as a baseline for the performance of the different methods, we include direct classification results (gray), for which no feature embedding was computed. We used $K = 8$ (dimensionality of feature embedding) for all methods. The features extracted by our pipeline compare favorably to PCA and Kernel PCA showing improvements in terms of AUC of 28% for Logistic Regression and 23% for Naive Bayes on the imbalanced data set and an improvement of 3.75% for Logistic Regression and 7.5% for Naive Bayes on the balanced data set. By using simple classifiers, we demonstrated that our encoder learns an effective feature embedding. More sophisticated classifiers (such as SVM with non-linear kernels) typically proceed by first embedding the input into a specific feature space that is different from the original space.

For the imbalanced data set the overall performance for SVM is significantly lower for all embeddings. This result is in line with previous work [Imam et al., 2006] showing that for imbalanced data sets, the decision boundaries of SVMs are heavily skewed towards the minority class resulting in a preference for the majority class and thus a high miss-classification rate for the minority class. Indeed, we found that SVM predicted only majority labels on the imbalanced data set. For the balanced data set our feature embedding shows improvements of 2.5% over alternative embeddings when using SVM.

Further, Table 6.1 shows the performance of all feature embeddings using three additional common classification metrics: root mean squared error (RMSE), classification accuracy (Acc.) and area under the precision recall curve (AUPR). We statistically compared the classification metrics of our feature embedding to the best alternative feature embedding using an independent t-test and Bonferroni correction for multiple tests ($\alpha = 0.05$). Our feature embedding significantly outperformed alternative embeddings for all classifiers on both the balanced and imbalanced data sets on most metrics. The main exception was the performance of SVM on the imbalanced data set, which exhibited large variance for all feature embeddings and the worst overall classification performance (compared to the other classifiers).

When comparing classification performance on the imbalanced and the balanced data sets we observed that our pipeline using VAEs showed significant performance improvements compared to other methods for finding feature embeddings. While the unlabeled and the balanced data sets stem from an adaptive version of *Calcularis*, the imbalanced data was collected using

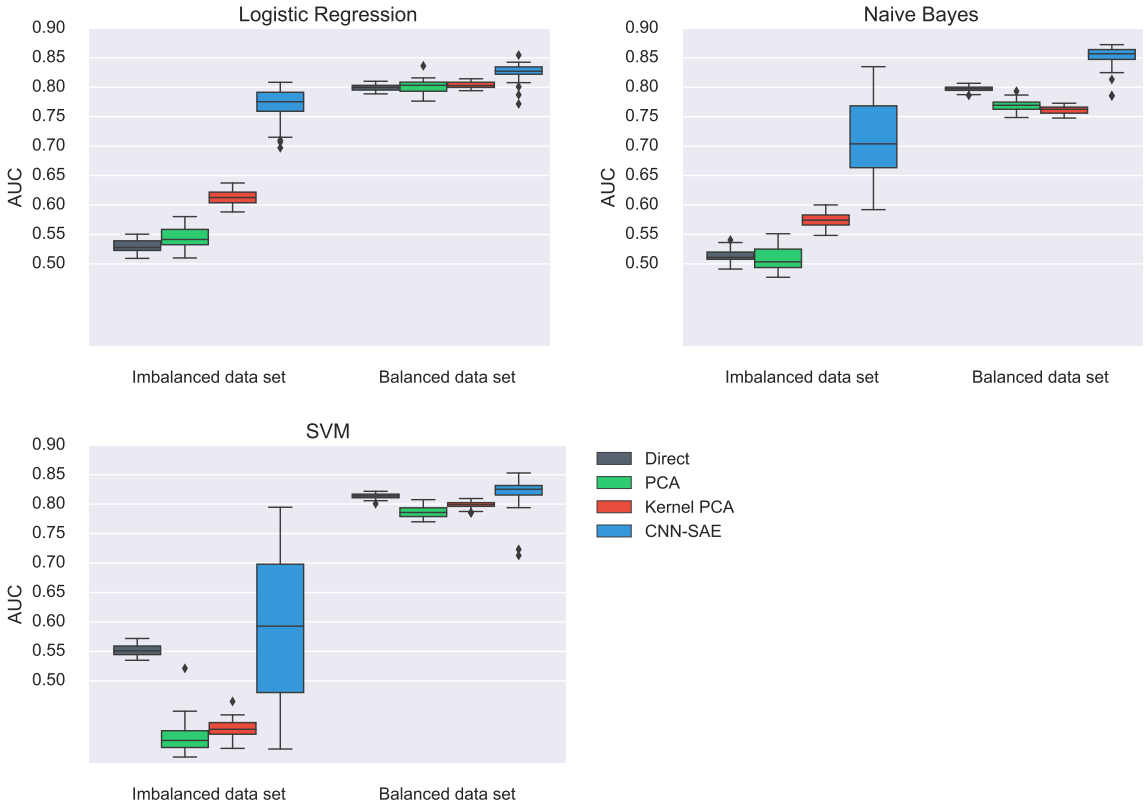


Figure 6.4: *Classification performance for different feature embeddings. Our variational auto-encoder (blue) outperforms other embeddings by up to 28% (imbalanced data set) and by up to 7.5% (balanced data set).*

a fixed task sequence. As our method shows larger improvements on the imbalanced data, we believe CNN-SAE learned an embedding that is robust beyond adaptive ITS. The relative improvements of our feature embeddings are smallest for SVM on the balanced data set. We believe that this is due to the ability of the SVM to learn complex decision boundaries given sufficient data. However, the ability for complex decision boundaries renders SVMs more vulnerable to class imbalance, yielding performance at random level on the imbalanced data set.

Table 6.1: *Comparison of our method to alternative embeddings. Our approach using a variational auto-encoder (CNN-SAE) significantly outperforms other approaches for most cases. The best score for each metric and classifier is shown in bold. *= statistically significant difference (t-test with Bonferroni correction, $\alpha = 0.05$).*

	Direct				**PCA**				**Kernel PCA**				**CNN-SAE**			
	AUC	RMSE	AUPR	Acc.	AUC	RMSE	AUPR	Acc.	AUC	RMSE	AUPR	Acc.	AUC	RMSE	AUPR	Acc.
Imbalanced data set																
Logistic Regression	0.53	0.27	0.18	0.91	0.54	0.25	0.17	0.93	0.61	0.25	0.16	0.93	**0.78***	**0.24***	**0.28***	**0.94***
Naive Bayes	0.51	0.29	0.23	0.91	0.50	0.29	0.10	0.90	0.57	0.28	0.20	0.91	**0.70***	**0.25***	**0.24**	**0.93***
SVM	0.55	0.25	**0.22***	0.94	0.40	0.25	0.08	0.94	0.42	0.25	0.09	0.93	**0.59**	0.25	0.16	0.94
Balanced data set																
Logistic Regression	0.80	0.44	0.82	0.73	0.80	0.42	0.84	0.73	0.80	0.42	0.83	0.75	**0.83***	**0.40***	0.84	**0.77**
Naive Bayes	0.80	0.49	0.80	0.73	0.77	0.46	0.77	0.71	0.76	0.46	0.76	0.70	**0.86***	**0.38***	**0.86***	**0.80***
SVM	0.81	0.42	**0.84***	0.75	0.79	0.43	0.81	0.73	0.80	0.43	0.83	0.73	**0.83**	**0.40***	0.81	**0.79***

6.3.5 Comparison to our specialized models

Next, we compare our feature embeddings to the specialized Naive Bayes classifier (S-NB) from Chapter 5. Along with the classifier, we presented a set of features optimized for the detection of developmental dyscalculia. The development of S-NB including the set of features was based on the balanced data set used in this chapter. The feature selection step for S-NB depended on the classification task in two ways: 1) the kernels used for feature comparison were designed with the classification task in mind and 2) representative features are chosen based on information about the student labels (see Section 5.1.2 for details). In comparison to our S-NB method, the approach presented in this chapter relies on timing data only and the extracted features are independent of the classification task.

We compared the performance of S-NB to our CNN-SAE model with the best performing classifier (Logistic Regression or Naive Bayes) on both data sets. For the balanced data set, we found an AUC of 0.94 for the specialized model (S-NB) compared to an AUC of 0.86 for Naive Bayes using our feature embedding. On the imbalanced data set, we found an AUC of 0.67 for S-NB compared to an AUC of 0.77 using Logistic Regression with our feature embedding. These findings demonstrate that while the feature embedding extracted by the variational auto-encoder performs slightly worse on the balanced data set (for which the S-NB was developed), classifiers using our automatically learned feature embedding significantly outperform S-NB by 15% on the imbalanced data set, which suggests that our VAE model automatically extracts feature embeddings that are more robust than the feature set we extracted based on the balanced data alone (see Chapter 5 for details).

6.3.6 Robustness on sample size

Ideally, a classifier's performance should gracefully decrease as less data is provided. A useful feature embedding allows a classifier to generalize well based on few labeled examples because similar samples are clustered together in the feature embedding. We, therefore, investigated the robustness of the different feature representations with respect to the training set size. For this, we used the balanced data set where we varied the training set size between 7 (10% of the data) and 62 (90% of the data) by random label-stratified sub-sampling. Figure 6.5 compares the AUC of the different feature embeddings over different sizes of the training set. In the case of Naive Bayes and Logistic Regression, our embedding provides superior

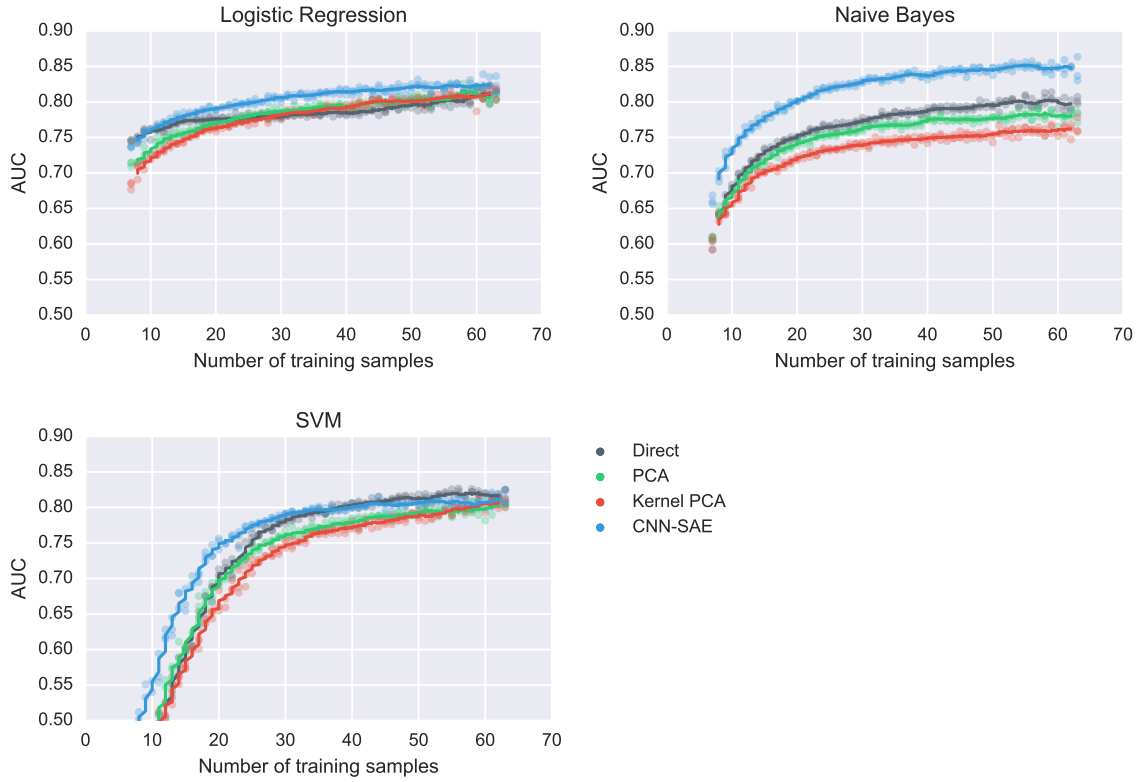


Figure 6.5: *Comparison of classifier performance on the balanced data for different training set sizes (moving average fitted to data points). The features automatically extracted by our variational auto-encoder (blue) maintain a performance advantage even if the training size shrinks to 7 samples (10% of the original size).*

performance for all training set sizes. For large enough data sets SVM using the raw feature data (Direct, gray) is performing as well as using our embedding (CNN-SAE, blue). However, for smaller data sets starting at 30 samples the performance of SVM based on the raw features declines more rapidly compared to the SVM based on our feature embedding.

6.4 Discussion

In this chapter, we adapted the recently developed variational auto-encoders to educational data for the task of semi-supervised classification of student characteristics. We presented a complete pipeline for semi-supervised classification that can be used with any standard classifier. We demonstrated that extracted structures from large-scale unlabeled data sets could significantly improve classification performance for different labeled data sets. Our find-

ings show that the improvements are especially pronounced for small or imbalanced data sets. Imbalanced data sets typically arise in educational data mining when detecting relatively rare conditions such as learning disabilities. We demonstrated that the feature embedding learned by our variational auto-encoder is more robust to new settings compared to our specialized feature set developed in Chapter 5. While we applied our method to the particular case of detecting developmental dyscalculia, the presented pipeline is generic and thus can be applied to any educational data set and used for the detection of any student characteristic.

CHAPTER 7

Unsupervised student trait discovery

So far we have explored supervised and semi-supervised methods for the detection of known student characteristics and student traits. Unsupervised methods would allow us to gain new insights into student behavior and potentially reveal new student characteristics. The unsupervised extraction of student properties and traits is a central element in educational data mining. On the one hand, the identification of student abilities and behavior patterns allows us to draw conclusions about human learning. On the other hand, the extracted properties and traits can be used to improve the adaptation of the underlying intelligent tutoring system (ITS).

In this chapter, we present a complete processing pipeline for evolutionary clustering that can be used as a black box for any ITS. We incorporate a variation of the adaptive evolutionary clustering method (AFFECT) [Xu et al., 2014] into our pipeline and demonstrate that temporal smoothing has beneficial properties for extracting student behavior and groups from educational data. We propose several extensions of the original method tailored towards learning data.

Our approach is articulated in four steps. In a first step, we extract action sequences from ITS log data and aggregate them using Markov Chains. We show that the Markov Chain representation of the actions is superior to direct sequence mining techniques [Bergner et al., 2014; Köck and Paramythis, 2011] with respect to noise cancellation and the ability to identify groups of students with similar behavior. The second step consists of computing pairwise similarities between the Markov Chains. While the proposed pipeline provides flexibility in the choice of similar-

ity measure, the Hellinger distance outperforms other metrics that are frequently used in the educational data mining literature [Bergner et al., 2014; Köck and Paramythis, 2011]. Based on the obtained similarities, evolutionary clustering [Xu et al., 2014] is performed in the third step. The temporal aspect of the student data leads to changing behavior patterns, i.e., we expect the number of clusters and cluster sizes to change over time. Therefore, capturing cluster evolution events, such as merging, splitting, dissolving and forming of clusters, is crucial in order to analyze sequential data. To capture these events automatically, we compute the optimal cluster count for each time step using the AICc criterion [Hurvich and Tsai, 1989].

Using synthetic data, we demonstrate that our method exhibits a higher performance and is more robust to noise than previous work [Bergner et al., 2014; Köck and Paramythis, 2011]. We further show that our pipeline can extract stable clusters over time and reliably detects all cluster events. In an exploratory analysis on real-world data, we apply our pipeline to log data from our two tutoring systems: `Orthograph` and `Calcularis` for spelling and mathematics learning, respectively (see Chapter 3 for a description of both systems). Finally, we present a set of visual tools that are powerful to analyze temporal data and student clusters.

7.1 Method

Our method for student clustering is designed to address two challenges when clustering temporal data. First, the method provides temporally consistent clusters. Second, our pipeline is able to capture changes in cluster sizes as well as in the number of clusters. Four *cluster events* are of particular interest in the context of educational data mining: merging, splitting, dissolving and forming of clusters. If the behavior of students from two different clusters becomes more similar over time, we expect the clusters to *merge* (this could mark a training effect). If on the other hand the behavior of students in a cluster sufficiently diverges clusters might *split* (this could mark the development of different learning strategies). If a distinct behavior disappears within a group of students, we assume the cluster will *dissolve*, meaning students will uniformly change to other clusters. In contrast, *forming* clusters have the potential to mark the development of distinct strategies within students.

The resulting clustering pipeline addressing these challenges is illustrated in Figure 7.1. The only input required are action sequences extracted from student log data. These action sequences are transformed into Markov

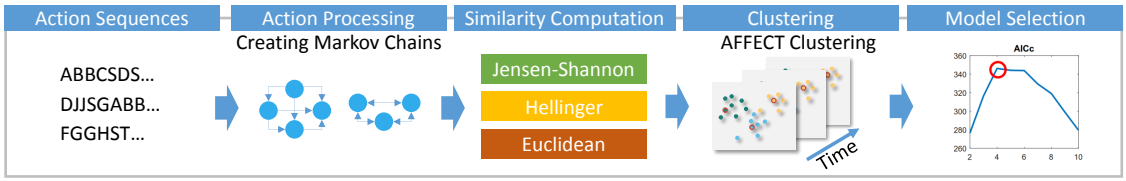


Figure 7.1: *Overview of our clustering pipeline. Action sequences are extracted from log data and transformed into Markov Chains per session. Pairwise similarities between students are computed for every session. Clustering is performed using evolutionary clustering. Finally, the AICc criterion selects the best model.*

Chains for every session, and pairwise similarities between these chains are computed. Students are clustered based on these similarities while enforcing temporal consistency over consequent training sessions. Finally, we compute the optimal number of clusters for each training session.

7.1.1 Action Sequences

In a first step, we extract action sequences $A_u^t = (a_0, a_1, \ldots, a_n)$ for every session t of a user u. To do so, we map events in the log files of an intelligent tutoring system (ITS) to the actions a_i. Typical actions within an ITS include for example correct or wrong keyboard inputs, help calls, off-task behavior or prompts for new tasks. As the particular actions depend on the ITS, the extraction of action sequences has to be changed depending on the ITS. However in many ITS, user interaction events are already logged as distinct events which renders the extraction of action sequences less labor intensive.

7.1.2 Action Processing

While action sequences provide rich temporal information about the exact ordering of actions, we expect that they exhibit a considerable amount of noise. We therefore transform the action sequences into an aggregated representation using Markov Chain models, similar to [Köck and Paramythis, 2011]. Markov Chains provide an aggregated view of the pairwise transition probabilities of actions and can be fully described by these transition probabilities $t_{i,j} := p_{a_j|a_i}$ from any state a_i (in our case an action) to any other state a_j. Markov Chains can be extracted using maximum likelihood estimates of the transition probabilities $t_{i,j}$.

7.1.3 Similarity Computation

To cluster student behavior, a suitable similarity (or distance) measure between students has to be defined; either on the action sequences or, as in our method, on Markov Chains. In educational data mining, popular choices for measuring distances between action sequences are the longest common subsequence (LCS) and the Levenshtein distance (see e.g. [Bergner et al., 2014]). LCS measures the length of the largest set of characters that appear in left-to-right order within the string, not necessarily at consecutive places. The Levenshtein distance computes the number of insertions, deletions, and replacements needed to transform one string into the other.

Instead of computing distances directly on action sequences, we can apply the computation to the aggregated values of Markov Chains. Previous work [Köck and Paramythis, 2011] has been using the Euclidean distance between the transition probabilities of two Markov Chains. A potential disadvantage of the Euclidean distance is that it is not designed for the comparison of probabilities. Therefore, we propose to use metrics that are specifically designed for comparing probability distributions. Since the conditional probabilities describing a Markov Chain do not form a proper probability distribution (the entries of the transition probability matrix do not sum up to one), we compute the expected transition probabilities using the stationary distribution over the actions and compare these expected transition frequencies $\bar{t}_{i,j}$ instead of the conditional probabilities $t_{i,j}$. We use two common metrics: the Jensen-Shannon Divergence and the Hellinger distance [Pardo, 2005] to compute the distances between the expected transition frequencies $\bar{t}_{i,j}$ of the Markov Chains. For two Markov Chains T and S, the Jensen-Shannon Divergence is given as

$$d_S(T,S) = \frac{1}{2}\sum_{i,j} \bar{t}_{i,j} \log(\frac{\bar{t}_{i,j}}{\bar{s}_{i,j}}) + \frac{1}{2}\sum_{i,j} \bar{s}_{i,j} \log(\frac{\bar{s}_{i,j}}{\bar{t}_{i,j}}), \tag{7.1}$$

where, $\bar{t}_{i,j}$ and $\bar{s}_{i,j}$ denote the expected transition frequencies for Markov Chains T and S respectively. Using the same notation the Hellinger distance is defined as

$$d_H(S,T) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i,j}(\sqrt{\bar{t}_{i,j}} - \sqrt{\bar{s}_{i,j}})^2}. \tag{7.2}$$

7.1.4 Clustering

Using the measures defined above we compute a pairwise similarity matrix W^t for every session t of the training (entries of the matrix measure how

similar two students are during that particular training session). These similarity matrices can then be clustered by any standard clustering method. However, clustering students for each session individually does not make use of the temporal information available. Recently, a method for clustering evolutionary data has been proposed that accurately tracks the time-varying similarities of objects over discrete time steps [Xu et al., 2014]. The method assumes that the observed similarities W^t are a linear combination of the true similarity between students Ψ^t and random noise N^t:

$$W^t = \Psi^t + N^t. \tag{7.3}$$

Instead of performing clustering directly on W^t, a smoothed similarity matrix $\hat{\Psi}^t$ is proposed, given as

$$\hat{\Psi}^t = \alpha^t \hat{\Psi}^{t-1} + (1 - \alpha^t) W^t, \tag{7.4}$$

where α^t controls the amount of smoothing applied to the observed similarity matrix W^t. Under some assumptions (detailed in [Xu et al., 2014]) an optimal choice for α^t is

$$\alpha^t = \frac{\sum_i \sum_j \mathrm{var}(n_{ij}^t)}{\sum_i \sum_j (\hat{\psi}_{ij}^{t-1} - \psi_{ij}^t)^2 + \mathrm{var}(n_{ij}^t)}. \tag{7.5}$$

Hence, the optimal α^t is based on a trade-off between the estimated noise in W^t and the amount of new information that W^t contains compared to previous similarity matrices. If W^t exhibits a lot of noise, we more heavily rely on previous observations (high α^t) but if we observe large discrepancies between the previous similarity estimates and the current ones (e.g. some students show a novel behavior) we emphasize the similarities from the current session (low α^t). Finally, we use the standard clustering algorithm K-Means to cluster the smoothed similarity matrices $\hat{\Psi}^t$.

7.1.5 Model Selection

The assumption of temporal consistency in the pairwise similarities between students does not prohibit evolution of clusters if students change their behavior over the course of the training. Such long-term drifts lead to growing and shrinking of clusters eventually, and even to dissolving and forming of clusters over time. In contrast to the original AFFECT method [Xu et al., 2014], we, therefore, compute the optimal number of clusters in every time step. Deciding on the number of clusters is a variant of the model selection problem, for which various criteria exists. The Akaike information criterion

(AIC) and the Bayesian information criterion (BIC) are among the most common criteria for model selection. The main difference between BIC and AIC is that the BIC penalizes the number of clusters more strongly than AIC. AICc corrects the AIC criteria for finite sample sizes. For our experiments, we used AICc as it potentially reveals more clusters, which is important for our exploratory analysis of learning data. To compute the AICc the log likelihood (LL) of the model is needed. According to [Pelleg and Moore, 2000], the log likelihood for K-Means can be formulated as

$$LL = \sum_i \log(\frac{N_{c(i)}}{N}\phi(x_i|\mu_{c(i)}, \sigma)), \tag{7.6}$$

where N denotes the number of samples, $c(i)$ the cluster index of sample x_i and $N_{c(i)}$ the number of samples in cluster $c(i)$. The likelihood of a sample x_i that was assigned to cluster $c(i)$ can be computed using the probability distribution $\phi(x_i|\mu_{c(i)}, \sigma)$, where $\mu_{c(i)}$ denotes the centroid of the cluster and σ the empirical variance of the data. In our case (as suggested by [Pelleg and Moore, 2000]), the probability distributions ϕ are identical spherical Gaussians. To compute the log likelihood, we embed our data points in a Euclidean space in which the distances between the points match the similarities extracted from the action sequences. To perform this embedding, we use the method presented in [Hofmann and Buhmann, 1997] that transforms N objects with pairwise similarities to a $D = N - 1$ dimensional Euclidean space.

An issue with this approach is that the dimensionality of the space $(N - 1)$ is likely to be much higher than the effective dimensionality of the data. Hence, the likelihood computation and estimation of the number of parameters P would be inaccurate. Therefore, we estimate the effective dimensionality $\hat{D}$ of our data set as the sum of eigenvalues λ_i of the covariance matrix divided by the largest eigenvalue λ_1 (see [Kirkpatrick, 2009]):

$$\hat{D} = \frac{\sum_i \lambda_i}{\lambda_1} \tag{7.7}$$

Based on this estimate, the effective number of parameters P for the K-Means clustering is

$$P = (\hat{D} + 1)k, \tag{7.8}$$

where k is equal to the number of clusters (see e.g. [Pelleg and Moore, 2000] for a derivation). Based on the log likelihood LL and the estimated effective dimensionality of our data $\hat{D}$, we calculate the AICc as

$$AICc = -2LL + 2P + \frac{2P(P+1)}{n - P - 1}. \tag{7.9}$$

7.2 Synthetic experiments

We analyzed the properties of our clustering algorithm using synthetic data, and we compared the performance and stability of our method to previous algorithms for clustering sequential educational data. Finally, we also validated our model selection step.

7.2.1 Experimental setup

We simulated student actions for 80 students over 50 sessions in a simulated learning environment. Students needed to solve 20 tasks per session. Student abilities θ and task difficulties d were simulated as part of a Rasch model [Wilson and De Boeck, 2004]. Student abilities for all students were sampled from a normal distribution with mean μ and variance σ. Task difficulties were sampled uniformly from the range $[-3, 3]$ in agreement with the common range of task difficulties [Harris, 1989]. Each task y consisted of eight steps s_j that a student had to complete to finish the task (this could e.g. be letters of a word to spell, performing steps of a calculation or solving a physics problem). The probability of a student correctly solving a task was then given by the Rasch model as

$$p(y) = (1 + e^{-(\theta - d)})^{-1} \tag{7.10}$$

In our simulation (in accordance with many ITS) a task was correctly solved if all the substeps are correctly solved, which defines the probability of correctly solving a step of a task s_j to be $p(s_j) = (p(y))^{\frac{1}{8}}$. Finally, a student could request help at any point in time during the training. Whether the student asked for help was sampled from a Bernoulli distribution with p_H. An overview of the described sampling procedure is given in Figure 7.2. Based on the sampling procedure we emitted the following actions for a student: *new task*, *help*, *correct*, *incorrect*, *correction*, *task completed*. The number of sampled actions per student and session depended on the performance of the student (e.g. a student who gets every step of a task correct completes a task after eight *correct* actions, whereas another student who requests help and commits an error requires more actions to complete the task).

For our experiments, we simulated student groups with different behavior. For the chosen range of task difficulties, student abilities are found to be normally distributed with mean $\mu = 0$ and variance $\sigma = 1$ (see [Harris, 1989] for details). We simulated good-performing students by setting $\theta = 1$ and bad-performing students by setting $\theta = -1$. According to [Aleven et

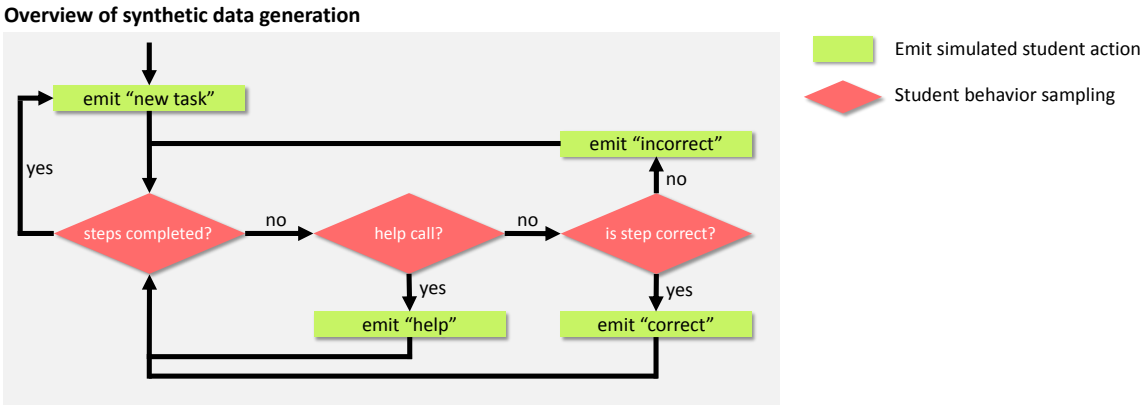


Figure 7.2: *Student actions (green squares) are simulated based on sampled student behavior (red diamonds) from a simple mastery learning system. Sampling parameters for student behavior (red diamonds) are adjusted to simulate different student traits.*

al., 2006], the most frequent form of help abuse are multiple consecutive help requests. We simulated this behavior by a large probability $p_H = 0.2$ to ask for help instead of working on the task, while normal help seeking behavior has a smaller probability for requesting help $p_H = 0.05$. Based on these different properties we simulated four groups of 20 students as follows. Group A contains bad-performing students ($\theta = -1$) that rarely ask for help ($p_H = 0.05$). Group B consists of bad-performing students ($\theta = -1$) that frequently use the help system ($p_H = 0.2$). Group C and D consist of good-performing students ($\theta = 1$) with rare ($p_H = 0.05$) and frequent ($p_H = 0.2$) help requests, respectively.

Our proposed pipeline offers flexibility in the choice of the similarity measure (see Section 7.1). We used the Jensen-Shannon divergence [Pardo, 2005], the Hellinger distance [Pardo, 2005] and the Euclidean distance for our experiments, and refer to these approaches as *Ours_SD*, *Ours_HD*, and *Ours_EUC*. To measure the influence of the different elements of the pipeline on the overall performance, we compared the proposed method to previous work on clustering of action sequences. The first approach [Bergner et al., 2014] works directly on the action sequences and uses the longest common subsequences (LCS) as similarity measure. Clustering is performed using an agglomerative clustering. However, to be able to compare clustering results better, we used the proposed similarity measure together with K-Means. We refer to this pipeline as *LCS_KM*. Similar to our method, the second approach used for comparison [Köck and Paramythis, 2011] computes the similarities between students using Markov Chains. Similarities are measured using the

Euclidean distance and clustering is performed using K-Means. The pipeline for this approach is denoted by *MC_EUC_KM*.

7.2.2 Clustering Quality & Robustness

In a first experiment, we computed the clustering quality of the different approaches with increasing noise levels. The performance P was measured using the cluster agreement in comparison to the ground truth labels. The different noise levels were simulated by increasing the variance in student abilities σ for the sampling of the data. Figure 7.3 (top) illustrates the performance of the different approaches with increasing noise. Note that the performance was computed using the correct cluster count of $k = 4$. Our pipeline (colored in green, red, and brown) exhibits the highest performance over all noise levels. The average agreement of our best performing pipeline (P_{Ours_HD}) is substantially higher than the average agreement of the best previous approach ($P_{MC\_EUC\_KM}$), both for a low variance ($P_{Ours\_HD,\sigma=1} = 0.82$, $P_{MC\_EUC\_KM,\sigma=1} = 0.53$) and for noisy data ($P_{Ours\_HD,\sigma=10} = 0.45$, $P_{MC\_EUC\_KM,\sigma=10} = 0.34$).

To investigate these differences between the approaches, we measured their performance over different numbers of clusters at preset noise levels. Figure 7.3 (bottom, left) illustrates the results for data with a relatively low noise level ($\sigma = 2$), while Figure 7.3 (bottom, right) shows the clustering quality of the different pipelines on noisy data ($\sigma = 8$). In the case of small noise in the data, all methods exhibit the best performance for the correct number of clusters ($k = 4$), which is a desirable property. The results demonstrate that using Markov Chains ($P_{MC\_EUC\_KM,k=4} = 0.44$) instead of working directly on action sequences ($P_{LCS\_KM,k=4} = 0.40$) leads to a higher clustering quality. A further increase in performance is achieved by our proposed algorithm: The variations of our pipeline exhibit a substantially higher clustering quality ($P_{Ours\_EUC,k=4} = 0.66$, $P_{Ours\_HD,k=4} = 0.70$, $P_{Ours\_SD,k=4} = 0.70$) than the previous work. This substantial increase in performance ($\Delta P_{k=4} = 0.26$ compared to *MC_EUC_KM*) is due to two changes in the pipeline. First, the proposed pipeline uses the AFFECT method for clustering leading to an increase in performance of $\Delta P_{k=4} = 0.20$. Second, while *MC_EUC_KM* computes the similarity measure directly on the transition probabilities, we use the expected transition probabilities as a basis for the similarity computations (see Section 7.1) accounting for an improvement in performance of $\Delta P_{k=4} = 0.06$. Within our approach, the choice of similarity measure has only a small impact on the clustering quality. Figure 7.3 (bottom, right) demonstrates that our proposed method

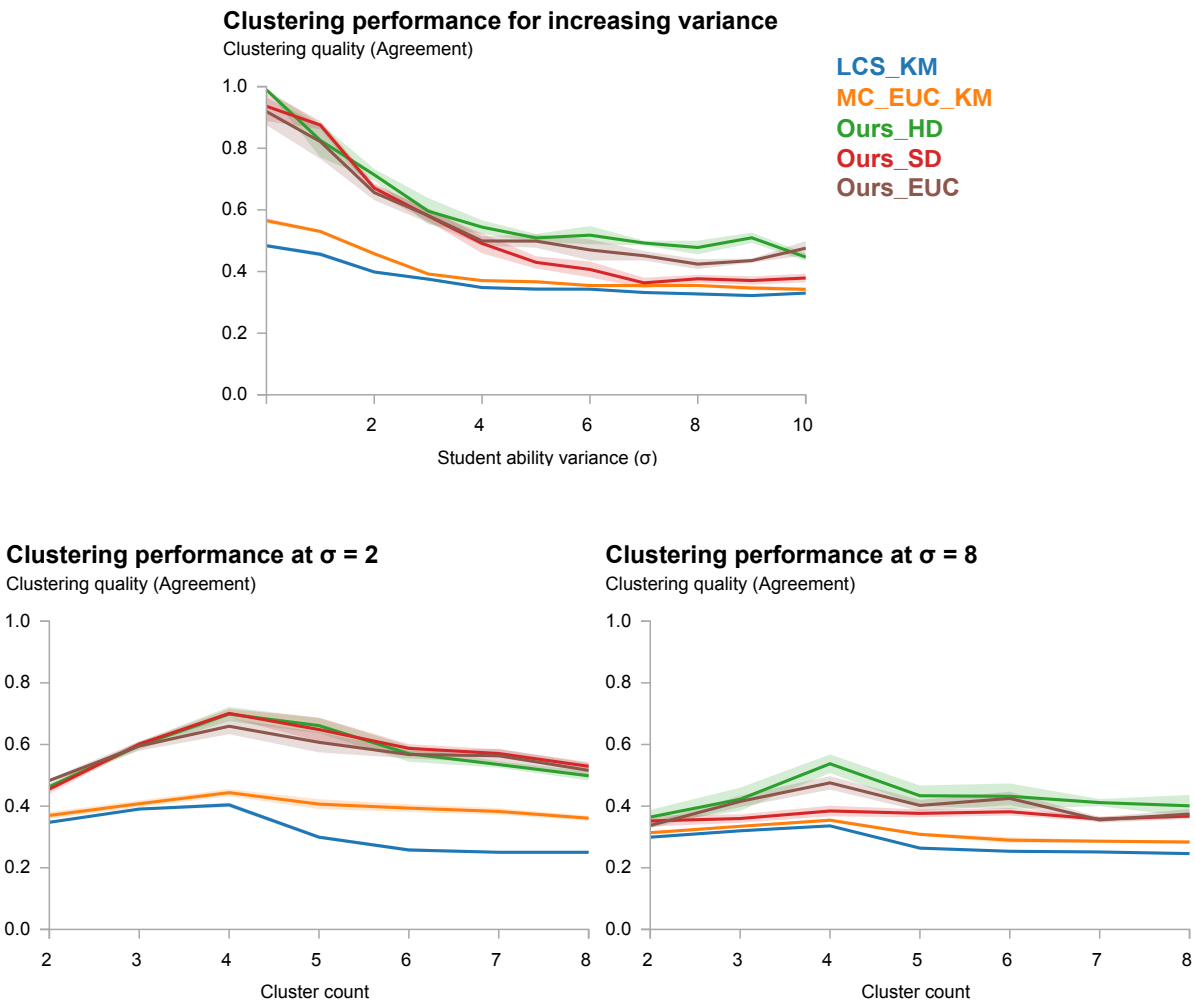


Figure 7.3: *Comparison of clustering methods over increasing noise levels (top) and over different numbers of clusters for fixed noise levels $\sigma = 2$ (bottom, left) and $\sigma = 8$ (bottom, right). Our method (Ours_HD, Ours_SD, Ours_EUC) shows less degradation of clustering quality (agreement with ground truth) for increasing noise levels.*

is more robust to noise than previous work [Köck and Paramythis, 2011; Bergner et al., 2014]. The best variation of our pipeline (colored in green) still achieves a reasonable performance ($P_{Ours\_HD,\sigma=8} = 0.54$). At these noise levels, the choice of action processing (Markov Chains vs. direct processing of action sequences) does not significantly influence performance ($P_{LCS\_KM,k=4} = 0.34$, $P_{MC\_EUC\_KM,k=4} = 0.35$). The choice of the clustering algorithm on the other hand is important. The increased performance of our method can be attributed to the use of AFFECT for clustering: AFFECT takes into account data from previous time steps to perform the clustering. Interestingly, the pipeline using the Jensen-Shannon divergence (*Ours_SD*) seems less robust to noise than the other pipelines (*Ours_HD* and *Ours_EUC*). We currently do not have an explanation for this effect, and further investigations into this are needed.

7.2.3 Stability

When clustering student actions over time, temporal consistency of clusters is essential. We measured the temporal stability of our method by computing the cluster size and cluster stability over the 50 simulated sessions (see Figure 7.4 for the cluster sizes). We compared the best performing pipeline from the first experiment (*Ours_HD*) to the previous approaches (*LCS_KM*, *MC_EUC_KM*) using again $k = 4$ clusters. The cluster stability ρ was computed using the bounded n-invariant variation of the split-merge measure [Meila, 2005]. Cluster stability of our best pipeline *Ours_HD* ($\rho = 0.002$) is significantly better than the stability of previous methods *LCS_KM* ($\rho = 0.468$) and *MC_EUC_KM* ($\rho = 0.425$). As can be seen from Figure 7.4 (top), our method provides a smooth temporal clustering with stable cluster sizes over time. The clusters found by *MC_EUC_KM* (Figure 7.4 (bottom, left)) and *LCS_KM* (Figure 7.4 (bottom, right)), on the other hand, are unstable: cluster sizes vary significantly over time. These results are as expected, as static clustering approaches identifying groups of students at each point in time are very sensitive to noise. The proposed method solves this problem by applying an evolutionary clustering algorithm and therefore takes into account multiple time steps.

7.2.4 Interpretability

Since we are clustering student behavior over multiple sessions, we expect the number of clusters and the cluster sizes to change over time. We expect clusters to merge, split, dissolve and form (see Section 7.1 for details). We

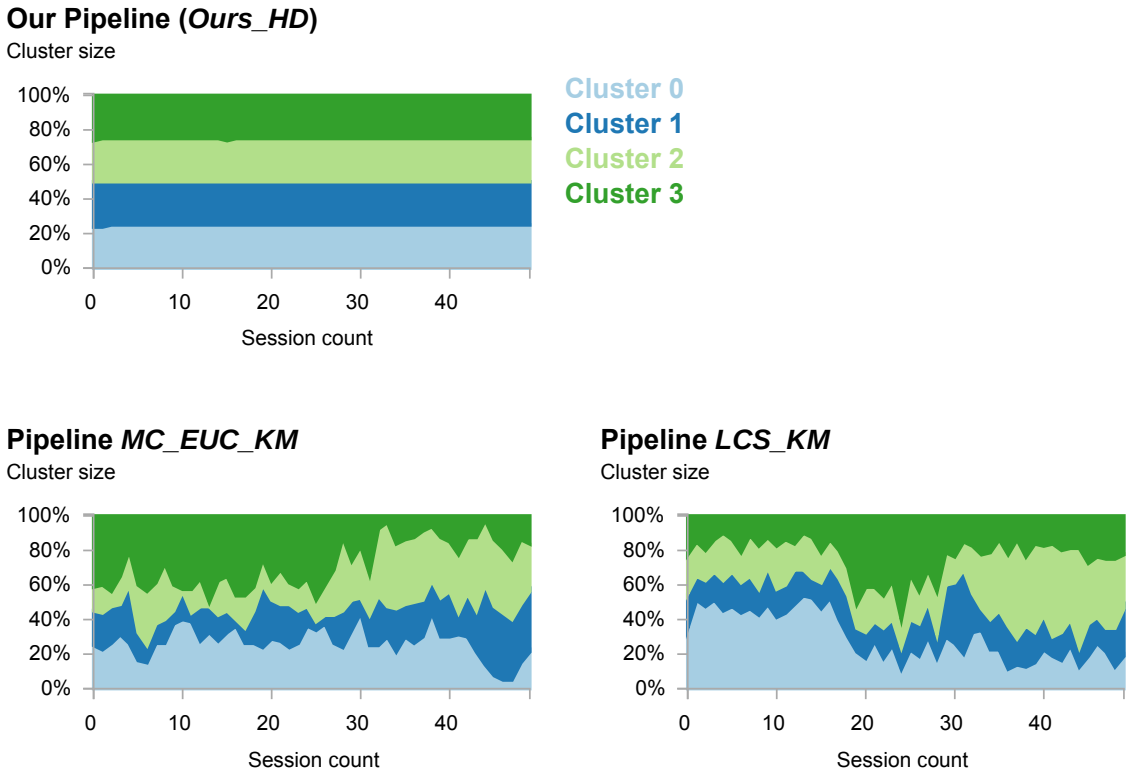


Figure 7.4: *Relative cluster sizes (for $k = 4$ clusters) over 50 simulated sessions. Our method performs best in extracting temporally stable clusters (Ours_HD) compared to previous methods.*

evaluated the *Ours_HD* pipeline on four scenarios using synthetic data. Note that these scenarios are artificial and are used only to demonstrate that the pipeline can capture the described events; we will show real-world examples of these events in Section 7.3.

In the first scenario (Figure 7.5, top left), group A consisting of bad-performing students with rare help calls (colored in dark green) merges into group B (colored in dark blue), i.e. the students of group A also start abusing the help. In our simulation, we initiate the cluster merge after $t = 20$ sessions and let group A completely vanish after $t = 50$ sessions, a behavior that is nicely captured by our method.

The second scenario (Figure 7.5, top right) starts with only three groups (B, C, and D), assuming that all bad-performing students frequently use the help. Over time, the bad-performing students split into a group abusing the help (group B, colored in dark blue) and a cluster consisting of students with rare help calls (group A, colored in dark green), i.e. in the simulation, some of the bad-performing students stop abusing the help over time.

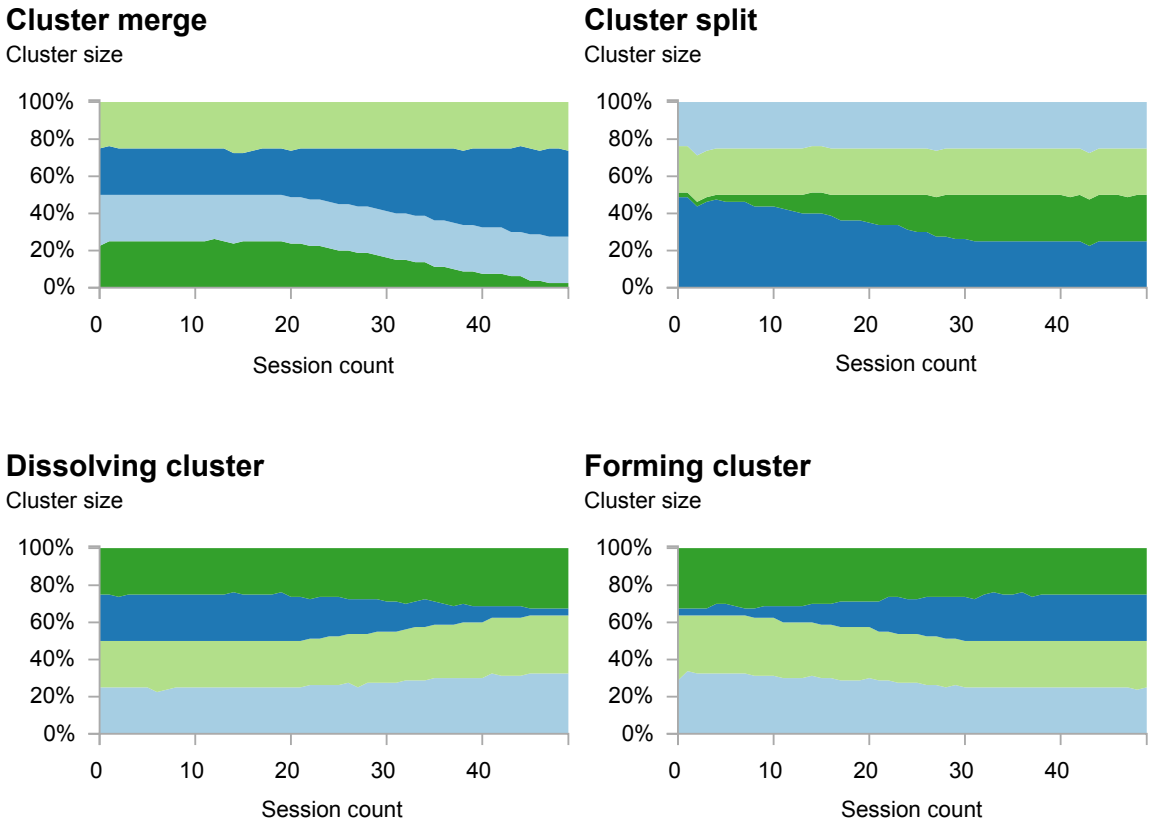


Figure 7.5: *Simulated examples of four types of cluster events. Our pipeline correctly identifies cluster merges/splits as well as dissolving/forming clusters.*

In the third scenario (Figure 7.5, bottom left) a dissolving cluster is simulated: Over time, group B (colored in dark blue) completely dissolves and the students are distributed to the other three clusters.

The fourth scenario (Figure 7.5, bottom right), finally, simulates a forming cluster event. The simulation starts with only three clusters (groups A, C, and D). With an increasing number of sessions, a fourth cluster forms (group B, colored in dark blue) and students from the other three clusters slowly switch to the new cluster until all the groups have equal size (after $t = 50$ sessions). This event is again correctly captured by our method.

The presented experiments demonstrate that the proposed pipeline is able to reliably identify changing cluster numbers and sizes. The results also demonstrate the validity of the model selection step of the pipeline: The AICc correctly identifies the number of clusters for all scenarios.

7.3 Exploratory data analysis

We applied our method to clustering of student interactions to data from `Orthograph` and `Calcularis`, focusing on the identification and interpretation of *cluster events*.

7.3.1 Experimental Setup

The first data set contained log data from 106 students and was collected using *Orthograph*, the computer-based training program for elementary school children with dyslexia [Gross and Vögeli, 2007]. To recap, *Orthograph* consists of one main learning game, where children have to type a dictated word (see Section 3.1 for details). The second data set contained data from 134 students and was collected from *Calcularis*, an ITS for elementary school children with difficulties in learning mathematics [Käser et al., 2013c]. As a reminder, *Calcularis* consists of different games for training number representations and calculation (see Section 3.2 for details). For all students, we extracted the first 15 training sessions with a minimal duration of $t = 5$ minutes from each student.
All results have been computed using our pipeline *Ours_HD* (see Section 7.1), applying the Hellinger Distance to measure similarities between Markov Chains of different students.

7.3.2 Navigation Behavior

In a first experiment, we extracted actions describing the *Navigation Behavior* of children in *Orthograph*. *Navigation Behavior* captures all events that cause the displayed content to change. During game play, children collect points for correct responses as well as for time spent in the training in general. These points can be used to buy different visual perks for the game in the shop. Children can also analyze their performance (e.g. progress in the current module) in the progress view. The resulting Markov chain (see Figure 7.6, left) consists of three possible states: *Game*, *Shop*, and *Performance*.

Figure 7.6, right, shows the relative cluster sizes for the *Navigation Behavior* Markov Chain over the first 15 sessions of the training. The different colors denote different clusters. At the beginning of the training ($t = 0$), our pipeline detects seven different clusters, however, three of these clusters (colored in pink, brown, and orange) die within the first three training sessions. Children in these clusters spent more than 50% of their time browsing the shop and checking their performance (orange: 46% *Game*, 31% *Shop*, 23%

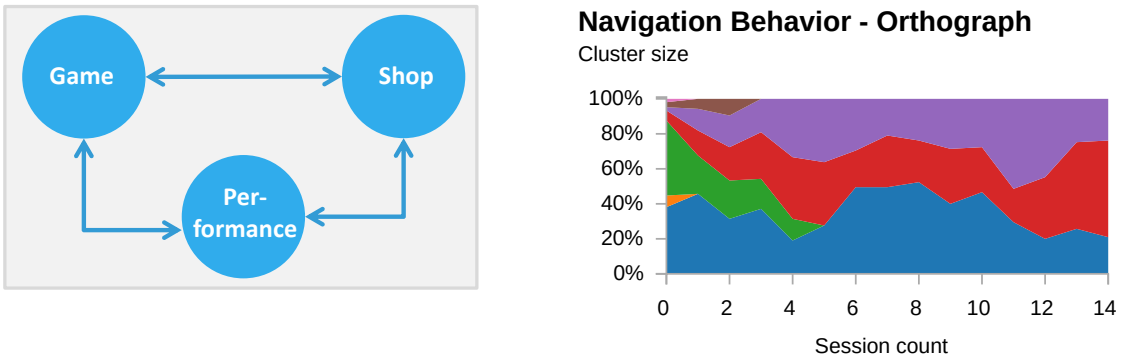


Figure 7.6: *Markov Chain for actions that capture the navigation behavior of students in Orthograph (left) and the relative cluster sizes (right) based on these actions.*

Performance; brown: 43% *Game*, 22% *Shop*, 35% *Performance*; pink: 40% *Game*, 32% *Shop*, 28% *Performance*) at the beginning of the training. We therefore hypothesize that children in these clusters tried out and played with the different views before getting used to the navigation possibilities of the system.

After $t = 5$ time steps, a further cluster (colored in green) dissolves before the clustering stabilizes to three main groups (colored in blue, red, and purple). These three clusters remain relatively stable until the end of the investigated time frame (split-merge measure $\rho = 0.06$). Figure 7.7 (top) shows the transition probabilities of the Markov Chains for the different clusters before the clusters dissolve (after $t = 3$ sessions). Children in the blue cluster are very focused on training, they spend 82% of their time in the *Game*. Once in the *Shop* or *Performance* state (18% of their time) they tend to select the following view with equal probabilities. Children in the red cluster like to browse the shop, a behavior that is visible from the high transition probabilities to the *Shop* state (*Game*→*Shop*: 0.41; *Performance*→*Shop*: 0.39), resulting in 34% of the training time spent browsing the shop. The purple cluster consists of children, who like to navigate to the shop and performance overview between solving the different tasks (*Game*→*Shop*: 0.41, *Game*→*Performance*: 0.44). However, these tend to be short visits as they will return to playing the game right after with high probability (*Performance*→*Game*: 0.58, *Shop*→*Game*: 0.77). Finally, children in the green cluster tend to select the next view randomly when playing the game. Once in the *Performance* state, they have a probability of 0.30 to browse the shop right after. The analysis of this time step illustrates that the different clusters differentiate well between focused children not making use of the navigation possibilities (blue cluster), children who frequently (but reasonably) use the different views (pur-

Figure 7.7: *Transition probabilities of the average Markov Chain for each cluster present in session $t = 3$ (top) and session $t = 6$ (bottom) for* Navigation Behavior *in* Orthograph. *The arrows indicate students transferring from the green cluster to the blue and red clusters between session $t = 3$ and session $t = 6$. Darker colors correspond to larger transition probabilities.*

ple and green cluster), and distracted children who spend long amounts of time off-task (red cluster).

After $t = 6$ training sessions, the green cluster dissolves and students from this cluster change to the red and blue clusters. The transition probabilities of the Markov Chains for these stable main clusters are illustrated in Figure 7.7 (bottom). The children in the blue cluster are still focused on training, spending 76% of their time solving tasks. However, they also check their training progress from time to time (14% of the time spent in the *Performance* state). After checking training progress, they tend to also browse the shop (*Performance*→*Shop*: 0.27). The children in the purple cluster have stopped navigating to the performance overview between different tasks (*Game*→*Performance*: 0.17) and instead visit the shop more frequently (*Game*→*Performance*: 0.58) and longer (35% of time spent in the *Shop* state). The red cluster still consists of children who like browsing the shop, a behavior that is visible from the high transition probabilities to the *Shop* state (*Game*→*Shop*: 0.33; *Performance*→*Shop*: 0.31). However, they also tend to

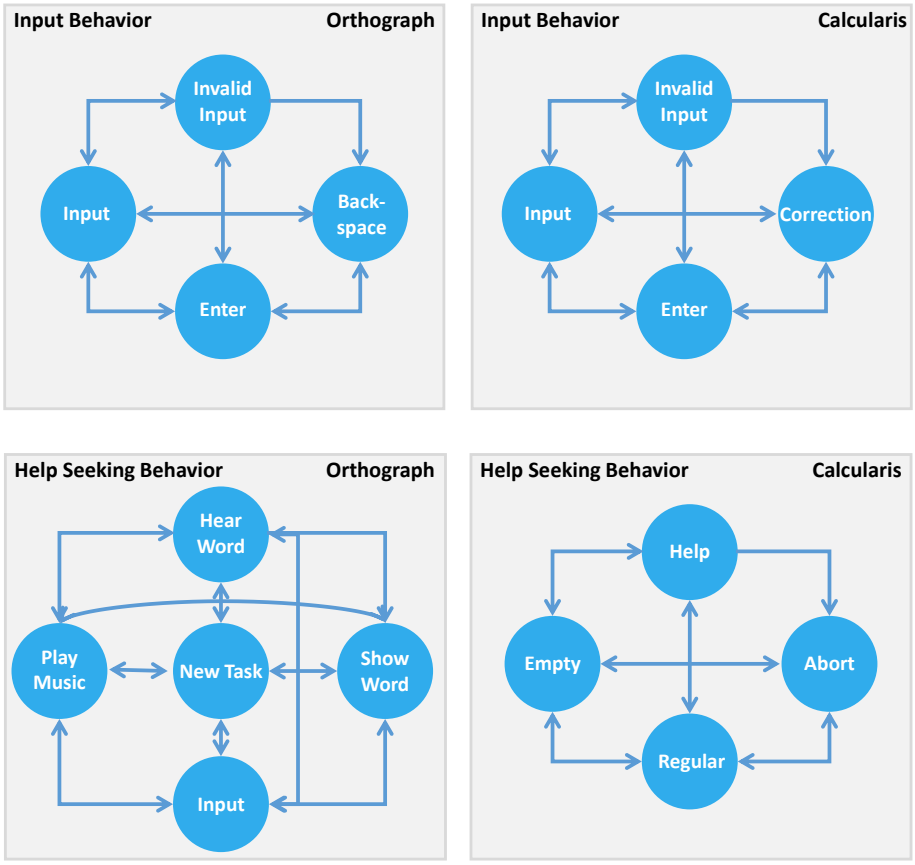


Figure 7.8: *Markov Chains for the* Input Behavior *and the* Help-Seeking Behavior *in* Orthograph *and* Calcularis.

spend time checking their progress, resulting in 47% of the training time spent off-task. Students from the green cluster, therefore, changed their behavior from frequent, but short off-task navigation to a more focused training style (change to the blue cluster) or to be completely distracted and spend significant amounts of times off-task (change to the red cluster).

7.3.3 Input & Help-Seeking Behavior

Our method can be used as a black box for any ITS and therefore also allows for comparison of behavior patterns across different ITS. The only user input needed is the definition of possible actions. To illustrate this possibility, we extracted two different sets of actions *Input Behavior* and *Help-Seeking Behavior* from data collected with *Orthograph* and with *Calcularis*.

Input Behavior captures all possible inputs. Implicitly these actions capture

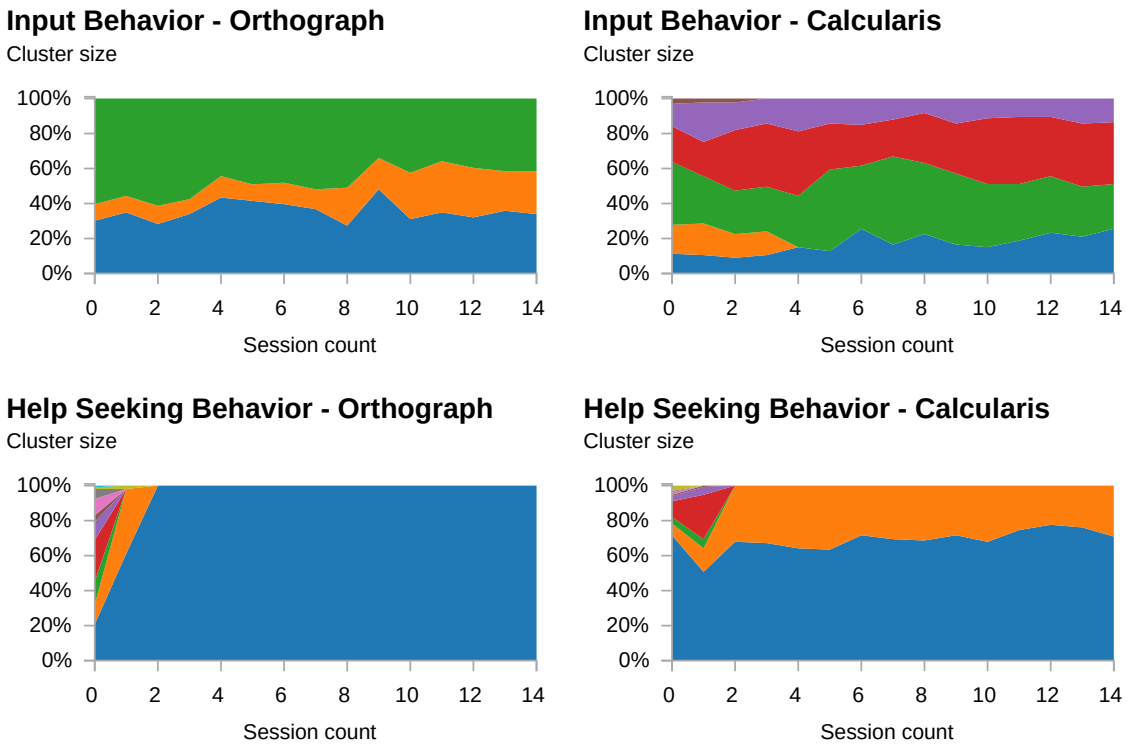


Figure 7.9: *Relative cluster sizes over the first* 15 *sessions based on the clustering of* Input Behavior *(left) and* Help-Seeking Behavior *(right) for students training with* Orthograph *and* Calcularis.

the performance of students, as e.g. a bad-performing student is likely to commit more mistakes. In *Orthograph*, children train spelling by writing words that are played back by the system. Therefore, the *Input Behavior* Markov Chain for *Orthograph* (see Figure 7.8) consists of four states: Children can type a letter (*Input*), correct themselves by deleting a letter (*Backspace*), provide invalid input such as typing a number (*Invalid Input*), or submit their solution (*Enter*). For *Calcularis*, we investigated calculation games. In these games, children need to solve different mental addition and subtraction tasks. We again define four states for the *Input Behavior* Markov Chain (see Figure 7.8): children can type a digit (*Input*), correct themselves by deleting a digit (*Correction*), provide invalid input such as random mouse clicks (*Invalid Input*), or set their answer (*Enter*).

Figure 7.9 shows the relative cluster sizes for the *Input Behavior* action set from *Orthograph* over 15 training sessions. Our method identifies three stable clusters. Investigating the stationary distributions of the Markov Chains reveals that students in the orange cluster show the highest probabilities for committing invalid inputs over all sessions ($t = 3 : 0.15$; $t = 7$: 0.23; $t = 13$:

0.16). The green cluster consists of focused students who consistently produce a low percentage of invalid inputs ($t = 3$: 0.06; $t = 7$: 0.04; $t = 13$: 0.05). Students in the blue cluster also tend to show low probabilities for invalid inputs across the different sessions ($t = 3$: 0.11; $t = 7$: 0.09; $t = 13$: 0.08). The orange cluster is an example of a *forming cluster* growing in size over the course of the training. We hypothesize that this event marks the increasing difficulty of the tasks and is caused by a downwards drift of students from the clusters with good performing students to the clusters with students showing worse performance. Further analysis of cluster transfers reveals that students indeed are never switching directly from the green (best performance) to the orange cluster (worst performance).

For *Calcularis*, the *Input Behavior* clusters are relatively stable over the course of the training (see Figure 7.9). There is one distinct *dissolve event* in the first four sessions: the orange cluster is dissolving into the blue and green clusters. Investigating the stationary distributions of the Markov chains of the three clusters reveals that all clusters have a relatively low probability for invalid inputs (t = 2: 0.17 (blue), 0.12 (orange), 0.08 (green)). However, students belonging to the blue cluster tend to perform multiple consecutive corrective actions in a row (*Correction*→*Correction*: 0.25 (blue), 0.13 (orange), 0.13 (green)). Students in the orange cluster are most likely to enter a valid input after a correction (*Correction*→*Input*: 0.68 (orange), 0.57 (blue), 0.65 (green)).

In *Orthograph*, differences in *Input Behavior* are mainly expressed by the percentage of invalid inputs provided. We observe a more distinct picture for *Calcularis*. While the invalid inputs are still an important indicator, children also exhibit different corrective behaviors.

Help-Seeking Behavior captures the use of hints available in the training environment. In *Orthograph*, children can re-play the given word (*Hear Word*), play the melody of the word (*Play Music*) and show the correct spelling of the word (*Show Word*). The corresponding Markov Chain is displayed in Figure 7.8. The states *New Task* and *Input* denote the play-back of a new word and a user input (keyboard), respectively. The development of the relative cluster sizes for these action sequences (see Figure 7.9) reveals a surprisingly large variance in student behavior (the clustering algorithm finds nine different clusters in the first two training sessions). However, the diversity in student behavior disappears through a large *cluster merge* after $t = 3$ sessions. Investigating the transition probabilities between the different actions, we observe that while students are experimenting with the three different help systems at the beginning of the training, the final cluster of students gave up on using the help functions. This drop in the frequency and diversity of

help usage indicates that the help functionality provided in *Orthograph* is not useful for most of the students.

Calcularis provides a limited help functionality. Children can require explanations for games (*Help*). Furthermore, they can directly require the solution of a task (*Empty*), if the task seems too difficult. Further states of the Markov Chain (displayed in Figure 7.8) are the setting of a complete answer (*Regular*) and the abortion of a task (*Incomplete*). We again observe a large *cluster merge* at the beginning of the training leading into two stable clusters. Investigating the stationary distributions of the Markov Chains of the two clusters reveals that students in the orange cluster are more likely to perform a help request compared to the blue cluster (t = 6: 0.03 (blue), 0.13 (orange)).

The *Help-Seeking Behavior* of the children is more difficult to compare across different ITS because the available hints are very different. However, our experiment shows that both learning environments do not provide ideal help options.

7.4 Discussion

In this chapter, we presented a complete pipeline for the evolutionary clustering of student behavior. This pipeline can be used as a black box for any ITS, requiring only the extraction of action sequences as input. Therefore, our pipeline can provide insights into user behaviors in many different scenarios. Our pipeline can allow teachers to quickly assess the dominant learning behaviors within a group even if the patterns of these behaviors are not yet known. The temporal analysis further allows practitioners to assess how certain behaviors develop over the course of the training, which can allow for earlier targeted interventions for students at risk of dropping out or getting lost within a course. Our clustering method can further be used to personalize student models to specific subgroups of students.

We demonstrated that enforcing temporal coherency between consecutive clusterings is beneficial for the detection of student behavior as well as the stable detection of *cluster events*. Our method outperforms previous work on synthetic data regarding clustering quality and stability. We applied our pipeline to different types of action sequences collected from two different ITS. The exploratory analysis demonstrates that our method can reveal interesting properties about the behavior of students and potential deficiencies of the learning environments.

Limitations. Data clustering in educational data mining is inherently an exploratory endeavor in which the type, number, and characteristics of a

clustering depend heavily on the insights one wants to gain or the research questions one wants to answer. A consequence of this is that there is no single true clustering of a data set unless the grouping is trivial. In this chapter, we therefore used data from a simulated mastery learning environment to compare our clustering pipeline to previous work. While all parameters of our simulation have been carefully chosen based on findings from real-world data sets, superior performance of our method on synthetic data does not directly imply superior performance on real-world data.

CHAPTER 8

Conclusion

Intelligent tutoring systems have been successful in adapting the difficulty level and selecting tasks according to the knowledge of learners in many different applications. To provide optimal adaptivity, these systems require detailed information about the student that is engaging with the system. Ideally, student models provide detailed information about the cognitive, metacognitive and affective states and traits of learners. Student models typically infer state and trait information from interaction events, which is challenging as interaction events provide only partial and ambiguous information on a student's true state [Conati and Kardan, 2013].

In this thesis, we contributed to this challenge. Namely, we developed techniques to leverage interaction data from intelligent tutoring systems to build models of student personality traits and characteristics. By focusing on student traits, our data-driven models provide insights into student learning that go beyond more traditional students models that are mostly concerned with modeling cognitive states.

Since gathering data from students with accurate student profiles is a time- and cost-intensive task, we presented two different frameworks for the prediction of student traits. First, we developed a fully data-driven pipeline for the automatic detection of student traits that can be seamlessly embedded into an intelligent tutoring system (detailed in Chapter 5). We demonstrated for the specific case of developmental dyscalculia that the outcome of neuropsychological testings could be accurately predicted using interaction events alone (sensitivity of 0.91 and specificity of 0.91). We performed an initial pilot study with 156 children in Swiss schools, in which we applied

our method to a classroom setting where children are much easier distracted and the data set is heavily imbalanced (only 5% of the children with developmental dyscalculia). The predictions of our method still exhibited moderate correlations to the neuropsychological assessments, but were overall worse than on the data set from children training at home.

To improve model quality, we explored the use of unlabeled interaction data that is typically readily available in high volumes. We therefore presented a second framework for the detection of student traits: a semi-supervised classification pipeline that employs variational auto-encoders to make effective use of unlabeled data (see Chapter 6 for details). Our developed variational auto-encoders learns efficient feature embeddings that improve the performance of standard classifiers by up to 28% compared to completely supervised training. Our pipeline shows improvements in classification accuracy of 15% compared to our fully supervised method on the classroom data set. Our method is data independent and classifier-agnostic. Hence it provides the ability to improve performance on a variety of other classification tasks in the field of educational data mining.

Finally, we developed a complete clustering pipeline to help practitioners explore and discover new student characteristics from unlabeled data sets (details can be found in Chapter 7). Our pipeline can be used as a black box for any intelligent tutoring system, requiring only the extraction of action sequences as input. Using synthetic data, we demonstrated that enforcing temporal coherency between consecutive clusterings is beneficial for the detection of student behavior as well as the stable detection of changes in the number of clusters. Exploratory analysis of different action sequences from *Orthograph* and *Calcularis* shows that our method is able to reveal interesting properties about the behavior of students and potential deficiencies of the learning environments.

8.1 Limitations & Future work

Woolf et al. [2013] formulated the five grand challenges for AI in educational systems: 1) creating mentors for every learner, 2) providing learning systems for 21st century skills, 3) making efficient use of interaction data to support learning, 4) providing universal access to global classrooms and 5) supporting lifelong and life-wide learning (see Chapter 1 for more details on these challenges). The work described in this thesis focused on advancing AI in educational systems for the first and third challenge. While the presented frameworks are effective in identifying and discovering student traits, we

are yet far from solving these two grand challenges. In the following, we list specific limitations of the presented work as well as directions for future work.

Generalizability. Our frameworks for supervised and semi-supervised classification of student traits were evaluated using data from a single intelligent tutoring system and using a single (but well-defined) student trait: developmental dyscalculia. While learning disabilities are an important and an interesting example of a student trait, more experiments with different student traits would allow to refine the presented methods and assess the generalizability of our method for the detection of other traits. Other important student traits include learning styles [Fleming, 2001; Butler and Gregorc, 1988], learner persistence [Tinto, 1997] or level of self-efficacy [Bandura, 1977]. Automatic identification of these student traits has the potential to refine intelligent tutoring systems further to meet the needs of diverse learners.

Learning from semi-labeled data sets. Based on the feature embedding computed by our variational auto-encoders, we demonstrated improved classification results with simple classifiers such as Logistic Regression. These results might indicate that variational auto-encoders learn feature embeddings that are interpretable by human experts. In the future, we want to explore the learned representations and compare them to traditional categorizations of students (skills, performance, etc.). Additionally, we want to extend our results to include additional feature types. Although we trained our networks on comparatively small sample sizes, the presented method scales (due to mini-batch learning) to much larger data sets ($> 100K$ users) allowing the training of more complex variational auto-encoders. Moreover, the generative model $p_\theta(\mathbf{x}|\mathbf{z})$ that is part of any variational auto-encoder can be used to produce realistic data samples [van den Oord et al., 2016]. Up-sampling of the minority class provides a potential way to improve the decision boundaries for classifiers. In contrast to common up-sampling methods such as ADASYN [He et al., 2008], sampling based on variational auto-encoders does not require nearest neighbor computations which makes them better applicable to small data sets. Preliminary results showed improvements in AUC by up-sampling based on variational auto-encoders of 2-3% compared to ADASYN.

Missing data. A common problem in educational data mining is missing data. Missing data is caused by different factors of which some are even desired properties of the intelligent tutoring system. Since intelligent tutoring systems provide an adaptive learning experience, not all students are working on the same tasks which leads to missing information about a student's

performance for many tasks. Further, due to student drop-out (a similar problem to attrition in longitudinal studies), we might have systematically missing data for certain subgroups of students (e.g. low performing students). While our supervised classification pipeline based on a Naive Bayes classifier is able to handle missing data, we currently only use students with complete data for our semi-supervised classification pipeline. To the best of our knowledge, variational auto-encoders have no natural extension to handle missing data. Traditional pre-processing methods such as data imputation could be used. However, these techniques make additional assumptions about the data generating process [Cheema, 2014]. For the future, we would like to explore options to handle missing data directly at the network level of variational auto-encoders.

Interpretability of clusters. While we demonstrated that the clusters found by our clustering pipeline could reveal interesting properties about the behavior of students and potential deficiencies of the learning environment, the interpretation of the clusters required the careful analysis of transition probabilities from extracted Markov chains. Better visualization and information summarization techniques of the characteristic behavior of students in a particular cluster would have the potential to allow teachers and educators to use the clustering information for improving the curriculum. Textual summaries of the important characteristics as produced by natural language generation systems [Reiter et al., 2000] would have the potential to make insights into student behavior accessible to non-experts in data analysis. Potential visualization techniques include techniques from simple adapted Sankey charts that highlight common action sequences within a group to methods for dynamic network visualization [Xu et al., 2013b].

Clustering approach. In this thesis, we improved on clustering techniques for action streams from intelligent tutoring systems by enforcing temporal coherence of clusters. Our approach shows a close resemblance to Kalman filters, which use time series of measurements to produce more accurate variable estimates by a joint probability distribution over the variables for each time step. Rephrasing the optimization in the framework of Kalman filters would allow us to directly model cluster drift (by choosing an appropriate state transition model, e.g. a Taylor approximation [Senesh and Wolf, 2009]) and adapt to the reliability of different extracted Markov Chain (e.g. transition probability estimates based on a short action sequence exhibit larger variance) by recursive covariance estimation [Feng et al., 2014] which would allow to include shorter sessions into the analysis (including unreliable data e.g. has been successfully incorporated for online learners [Minku and Yao, 2012]). Extending the framework in this way has the potential to produce more accurate similarity estimates between users. Similar ap-

proaches have already been successfully employed for clustering sequences from motion capture [Li and Prakash, 2011].

Closing the loop. In this thesis, we presented various methods for identifying and discovering student traits and characteristics. A crucial next step is to incorporate these methods into existing cognitive student models. Recent student models such as the Feature Aware Student Model (FAST) or Deep Knowledge Tracing (DKT) provide a principled way to incorporate this additional information about the students. In the future, we would like to evaluate the performance of such extended models on large data sets from different learning domains and compare these extended models against the current state-of-the art in student modeling.

Instructional policies. A large part of the research on intelligent tutoring systems has focused on constructing student models (see Figure 2.1 for an overview of the important student model components) to accurately predict student actions and to infer student knowledge or student traits (as in this work). However, the development of efficient and universal instructional policies has only recently gained attention [Käser et al., 2016]. The design of instructional policies includes the optimization of teaching sequences [Muldner and Conati, 2007; Murray et al., 2004; Clement et al., 2015; Rafferty et al., 2011; Mandel et al., 2014] as well as the design of policies on when to stop teaching a skill to avoid under or over-practicing a skill [Cen et al., 2007; Lee and Brunskill, 2012]. In web-based learning, course sequences and pathways through the learning content are adapted based on the learner's goals and previous knowledge [Brusilovsky and Vassileva, 2003; Chen et al., 2006]. Instructional policies are therefore an important component of any intelligent tutoring systems an we would like to explore how optimal instructional policies are affected by student traits and how data-driven instructional policies can be derived.

APPENDIX A

Instruments

ZAREKI-R

The mathematical abilities and performance in number processing was assessed using the ZAREKI-R test [von Aster et al., 2006]. ZAREKI-R consists of subtests for counting, writing numbers, mental calculation, reading numbers, comparison of numbers and memorizing numbers. Internal test reliability is high ($\alpha = 0.89$) and the validity of the test has been assessed using a factor analysis. Construct validity was indicated by demonstrating high correlations ($r = 0.73, p < 0.05$) between the total test score and the arithmetic subtest of SAT [Santos et al., 2012; Santos and Silva, 2008; Stein, 1994]. Furthermore, correlations between subtests of ZAREKI-R with the arithmetic subtest of WISC-III [Woolger, 2001] were in the range of [0.22, 0.62] [Santos et al., 2012].

HRT

We assessed arithmetic performance based on the addition and subtraction subtests of the HRT [Haffner et al., 2005]. HRT is a speed test, participants are asked to solve as many addition (subtraction) tasks as possible within two minutes. Tasks are ordered by difficulty. The two subtests show a high re-test reliability (addition: $r_{tt} = 0.84$, subtraction: $r_{tt} = 0.86$) over a two-weeks period. HRT exhibits a high criterion-related validity with the school grades in mathematics ($r = 0.72$).

BUEGA

Verbal and nonverbal intelligence as well as reading and writing performance were assessed using BUEGA [Esser et al., 2008a]. In tasks for determining verbal intelligence, participants are asked to finish sentences by using logic (e.g. "I play during the day, I sleep at ..."). For nonverbal intelligence, a logical continuation of a matrix has to be chosen. Internal consistencies for each school grade are sufficient to high ($\alpha = 0.81$ to $\alpha = 0.95$).

HAWIK IV

We used the HAWIK IV [Esser et al., 2008b; Petermann and Petermann, 2008] to assess speech comprehension and logical reasoning of the children. Speech comprehension was measured using the subtest where children had to find commonalities between two terms (i.e. 'red' and 'blue' are both colors). Logical reasoning was measured with the mosaic subtest. In this test, children have to replicate given patterns with cubes. Reliabilites for the subtests of the HAWIK IV vary between $r = .76$ and $r = .91$.

CFT 1 and CFT 20-R

The CFT 1 [Cattell et al., 1997] and CFT 20-R [Weiss, 2006] are instances of culture fair intelligence tests, i.e. the goal of these tests is to provide equal opportunities for people from different social and different cultures. The subtests of the CFT 1 and CFT 20-R assess logical reasoning with language- and number-free tasks. In the classification subtest of the CFT 1 for example, children are presented five images (e.g. four houses and one shoe) and have to select the one that does not fit. For children aged up to 9;5 years, we used the subtests 3, 4 and 5 (classification, similarities, matrices) of the CFT 1 to estimate their general intelligence. Children older then 9;5 years solved all subtests of part 1 of the CFT 20-R.

Harter

We assessed the self-perception of participants using two subtests "Peer acceptance" and "Sport competence" from the German version of the Harter self competence test [Asendorpf and Van Aken, 1993]. The test consists of set of Likert scaled questions for the two subtests accompanied by supporting pictures for participants in first and second grade. The two subtests were shown to have moderate internal reliability ($\alpha = 0.71$ and $\alpha = 0.85$).

Math anxiety

Math anxiety was assessed using the FRA test [Krinzinger et al., 2007] which is the German translated version of the MAQ test [Thomas and Dowker, 2000]. The test measures four emotional components regarding mathematics, namely, self-perceived performance, attitude, poor-performance unhappiness and anxiety. The internal consistency of the test is high (between $\alpha = 0.83$ and $\alpha = 0.91$).

Test of Attentional Performance (TAP)

We used the subtests "Alertness" and "Go/NoGo" of the computerized TAP [Zimmermann and Fimm, 2009] to assess the attentional performance of participants. Re-test reliabilities for the subtests are $r_{tt} = 0.61$ and $r_{tt} = 0.56$, respectively.

Working memory

The working memory was assessed by the word span and backward digit tasks of the AGTB 5-12 test [Hasselhorn, 2012]. The internal consistency of the test varies between $\alpha = 0.67$ and $\alpha = 0.99$ over different age groups. Retest reliability over a 1-2 week period was found to be in the range of $r_{tt} = 0.66$ to $r_{tt} = 0.89$.

References

[Abadi et al., 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Agrawal and Srikant, 1995] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering*, pages 3–14. IEEE, 1995.

[Aleven et al., 2006] Vincent Aleven, Bruce Mclaren, Ido Roll, and Kenneth Koedinger. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16(2):101–128, 2006.

[Andres et al., 2015] Juan Miguel L Andres, Ma Mercedes T Rodrigo, Ryan S Baker, Luc Paquette, Valerie J Shute, and Matthew Ventura. Analyzing Student Action Sequences and Affect While Playing Physics Playground. In *International Workshop on Affect, Meta-Affect, Data and Learning*, page 24, 2015.

[Arroyo and Woolf, 2005] Ivon Arroyo and Beverly Park Woolf. Inferring learning and attitudes from a Bayesian Network of log file data. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 33–40, 2005.

[Asendorpf and Van Aken, 1993] Jens B Asendorpf and Marcel AG Van Aken. Deutsche Versionen der Selbstkonzeptskalen von Harter. *Zeitschrift für Entwicklungspsychologie und pädagogische Psychologie*, 25(1):64–96, 1993.

[Attali, 2015] Yigal Attali. Reliability-Based Feature Weighting for Automated Essay Scoring. *Applied Psychological Measurement*, 39(4):303–313, 2015.

[Baker et al., 2004] R. S. Baker, A. T. Corbett, and K. R. Koedinger. Detecting Student Misuse of Intelligent Tutoring Systems. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 531–540, 2004.

[Baker et al., 2008] R. S. Baker, A. T. Corbett, and V. Aleven. 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, 2008.

[Baker et al., 2010a] R. S. Baker, A. T. Corbett, S. M. Gowda, A. Z. Wagner, B. A. MacLaren, L. R. Kauffman, A. P. Mitchell, and S. Giguere. Contextual Slip and Prediction of Student Performance after Use of an Intelligent Tutor. In *Proceedings of the Conference on User Modelling, Adaptation and Personalization*, pages 52–63, 2010.

[Baker et al., 2010b] Ryan SJd Baker, Adam B Goldstein, and Neil T Heffernan. Detecting the moment of learning. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 25–34, 2010.

[Baker et al., 2012] Ryan Baker, Sujith Gowda, Albert Corbett, and Jaclyn Ocumpaugh. Towards automatically detecting whether student learning is shallow. In *Intelligent Tutoring Systems*, pages 444–453. Springer, 2012.

[Baldeweg et al., 1999] Torsten Baldeweg, Alexandra Richardson, Sarah Watkins, Christine Foale, and John Gruzelier. Impaired auditory frequency discrimination in dyslexia detected with mismatch evoked potentials. *Annals of neurology*, 45(4):495–503, 1999.

[Bandura, 1977] Albert Bandura. Self-efficacy: toward a unifying theory of behavioral change. *Psychological review*, 84(2):191, 1977.

[Baschera et al., 2011] G. M. Baschera, A. G. Busetto, S. Klingler, J. Buhmann, and M. Gross. Modeling Engagement Dynamics in Spelling Learning. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 31–38, 2011.

[Baschera, 2011] Gian-Marco Baschera. *Modeling and Evaluation of Computer-Assisted Spelling Learning in Dyslexic Children*. PhD thesis, ETH Zurich, 2011.

[Beacham and Trott, 2005] Nigel Beacham and C. Trott. Screening for dyscalculia within HE. *MSOR Connections*, 5:1–4, 2005.

[Beal and Cohen, 2008] Carole R Beal and Paul R Cohen. Temporal data mining for educational applications. In *PRICAI 2008: Trends in Artificial Intelligence*, pages 66–77. Springer, 2008.

[Beck and Chang, 2007] Joseph Beck and Kai-min Chang. Identifiability: A fundamental problem of student modeling. *User Modeling 2007*, pages 137–146, 2007.

[Beck and Gong, 2013] Joseph E. Beck and Yue Gong. Wheel-spinning: Students who fail to master a skill. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 431–440, 2013.

[Beck and Xiong, 2013] Joseph Beck and Xiaolu Xiong. Limits to accuracy: how well can we do at student modeling? In *Proceedings of the International Conference on Educational Data Mining*, pages 4–11, 2013.

[Beck, 2005] Joseph E. Beck. Engagement tracing: Using response times to model student disengagement. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 88–95, 2005.

[Benevenuto et al., 2009] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 49–62. ACM, 2009.

[Bengio and others, 2009] Yoshua Bengio et al. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[Bengio et al., 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[Bengio, 2012] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478, 2012.

[Berg-Kirkpatrick et al., 2010] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics, 2010.

[Bergner et al., 2014] Yoav Bergner, Zhan Shu, and Alina A. Von Davier. Visualization and Confirmatory Clustering of Sequence Data from a Simulation-

Based Assessment Task. In *Proceedings of the International Conference on Educational Data Mining*, pages 177–184, 2014.

[Biswas et al., 2010] Gautam Biswas, Hogyeong Jeong, John S Kinnebrew, Brian Sulcer, and ROD ROSCOE. Measuring self-regulated learning skills through social interactions in a teachable agent environment. *Research and Practice in Technology Enhanced Learning*, 5(02):123–152, 2010.

[Bloom, 1984] Benjamin S Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4–16, 1984.

[Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[Bowman et al., 2016] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *Proceedings of SIGNLL Conference on Computational Natural Language Learning (CONLL)*, page 10, 2016.

[Boyer et al., 2000] Kristy Elizabeth Boyer, Robert Phillips, Amy Ingram, Eun Young Ha, Michael Wallis, Mladen Vouk, and James Lester. Characterizing the Effectiveness of Tutorial Dialogue with Hidden Markov Models. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 55–64, 2000.

[Brusilovsky and Vassileva, 2003] Peter Brusilovsky and Julita Vassileva. Course sequencing techniques for large-scale web-based education. *International Journal of Continuing Engineering Education and Life Long Learning*, 13(1-2):75–94, 2003.

[Butler and Gregorc, 1988] Kathleen Ann Butler and Anthony F Gregorc. *It's all in your mind: A student's guide to learning style*. Learner's Dimension, 1988.

[Butterworth et al., 2011] Brian Butterworth, Sashank Varma, and Diana Laurillard. Dyscalculia: From brain to education. *Science*, 332(6033):1049–1053, 2011.

[Butterworth, 2003] Brian Butterworth. *Dyscalculia screener*. Nelson Publishing Company Ltd., 2003.

[Butterworth, 2005] B. Butterworth. Developmental dyscalculia. In J. D. Campell, editor, *The Handbook of Mathematical Cognition*, pages 455–469. Taylor&Francis, 2005.

[Campbell et al., 2013] Trevor Campbell, Miao Liu, Brian Kulis, Jonathan P How, and Lawrence Carin. Dynamic clustering via asymptotics of the dependent

dirichlet process mixture. In *Advances in Neural Information Processing Systems*, pages 449–457, 2013.

[Cattell et al., 1997] R.B. Cattell, R.H. Weiss, and J. Osterland. *Grundintelligenztest Skala 1 (CFT)*. Hogrefe, Göttingen, 1997.

[Cen et al., 2007] H. Cen, K. R. Koedinger, and B. Junker. Is Over Practice Necessary? - Improving Learning Efficiency with the Cognitive Tutor through Educational Data Mining. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 511–518, 2007.

[Cen et al., 2008] H. Cen, K. R. Koedinger, and B. Junker. Comparing Two IRT Models for Conjunctive Skills. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 796–798, 2008.

[Chakrabarti et al., 2006] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560. ACM, 2006.

[Chang et al., 2006] Kai-min Chang, Joseph Beck, Jack Mostow, and Albert Corbett. A bayes net toolkit for student modeling in intelligent tutoring systems. In *Intelligent tutoring systems*, volume 2006, pages 104–113. Springer, 2006.

[Cheema, 2014] Jehanzeb R Cheema. A review of missing data handling methods in education research. *Review of Educational Research*, 84(4):487–508, 2014.

[Chen et al., 2006] Chih-Ming Chen, Chao-Yu Liu, and Mei-Hui Chang. Personalized curriculum sequencing utilizing modified item response theory for web-based instruction. *Expert Systems with applications*, 30(2):378–396, 2006.

[Chollet, 2015] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[Cisero et al., 1997] C.A. Cisero, J.M. Royer, H.G. Marchant, and S.J. Jackson. Can the computer-based academic assessment system (CAAS) be used to diagnose reading disability in college students? *Journal of Educational Psychology*, 89(4):599–620, 1997.

[Clement et al., 2015] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. Multi-Armed Bandits for Intelligent Tutoring Systems. *Journal of Educational Data Mining*, 7, 2015.

[Cohen et al., 1982] Peter A Cohen, James A Kulik, and Chen-Lin C Kulik. Educational outcomes of tutoring: A meta-analysis of findings. *American educational research journal*, 19(2):237–248, 1982.

[Cohen Kadosh et al., 2007] R. Cohen Kadosh, K. Cohen Kadosh, T. Schuhmann, A. Kaas, R. Goebel, A. Henik, and A. T. Sack. Virtual dyscalculia induced by parietal-lobe TMs impairs automatic magnitude processing. *Current Biology*, 17:689–693, 2007.

[Conati and Kardan, 2013] Cristina Conati and Samad Kardan. Student modeling: Supporting personalized instruction, from problem solving to exploratory open ended activities. *AI Magazine*, 34(3):13–26, 2013.

[Conati et al., 2002] Cristina Conati, Abigail Gertner, and Kurt Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, 12(4):371–417, 2002.

[Cooper et al., 2010] David G. Cooper, Kasia Muldner, Ivon Arroyo, Beverly Park Woolf, and Winslow Burleson. *Ranking Feature Sets for Emotion Models Used in Classroom Based Intelligent Tutoring Systems*, pages 135–146. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[Corbett and Anderson, 1994] Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *Journal of User Modeling and User-Adapted Interaction*, 4:253–278, 1994.

[Corbett et al., 1997] Albert T Corbett, Kenneth R Koedinger, and John R Anderson. Intelligent tutoring systems. *Handbook of human-computer interaction*, 5:849–874, 1997.

[Cortina, 1993] Jose M Cortina. What is coefficient alpha? An examination of theory and applications. *Journal of applied psychology*, 78(1):98, 1993.

[Cowley and Charles, 2016] Benjamin Cowley and Darryl Charles. Behavlets: a method for practical player modelling using psychology-based player traits and domain specific features. *User Modeling and User-Adapted Interaction*, 26(2-3):257–306, 2016.

[Dehaene and Cohen, 1995] S. Dehaene and L. Cohen. Towards an anatomical and functional model of number processing. *Mathematical Cognition*, 1:82–120, 1995.

[Démonet et al., 2004] Jean-François Démonet, Margot J Taylor, and Yves Chaix. Developmental dyslexia. *The Lancet*, 363(9419):1451–1460, 2004.

[Desmarais and Lemieux, 2013] Michel Desmarais and François Lemieux. Clustering and visualizing study state sequences. In *Proceedings of the International Conference on Educational Data Mining*, pages 224–227, 2013.

[Desoete and Grégoire, 2006] Annemie Desoete and Jacques Grégoire. Numerical competence in young children and in children with mathematics learning disabilities. *Learn. Individ. Differ.*, 16(4):351–367, 2006.

[Dhanani et al., 2014] Asif Dhanani, Seung Yeon Lee, Phitchaya Phothilimthana, and Zachary Pardos. A comparison of error metrics for learning model parameters in Bayesian knowledge tracing. Technical report, UCB/EECS-2014-131, EECS Department, University of California, Berkeley, 2014.

[Esser et al., 2008a] G. Esser, A. Wyschkon, and K. Ballaschk. *BUEGA: Basisdiagnostik Umschriebener Entwicklungsstörungen im Grundschulalter*. Hogrefe, Göttingen, 2008.

[Esser et al., 2008b] Günter Esser, Anne Wyschkon, and Katja Ballaschk. Basisdiagnostik umschriebener Entwicklungsstörungen im Grundschulalter. *Hogrefe, Göttingen*, 2008.

[Fabius and van Amersfoort, 2015] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *International Conference on Learning Representations*, 2015.

[Fancsali et al., 2013] Stephen Fancsali, Tristan Nixon, and Steven Ritter. Optimal and worst-case performance of mastery learning assessment with Bayesian knowledge tracing. In *Proceedings of the International Conference on Educational Data Mining*, pages 35–42, 2013.

[Felder and Brent, 2005] Richard M Felder and Rebecca Brent. Understanding student differences. *Journal of engineering education*, 94(1):57–72, 2005.

[Feng et al., 2014] Bo Feng, Mengyin Fu, Hongbin Ma, Yuanqing Xia, and Bo Wang. Kalman filter with recursive covariance estimation—sequentially estimating process noise covariance. *IEEE Transactions on Industrial Electronics*, 61(11):6253–6263, 2014.

[Fischer and Molenaar, 1995] Gerhard H Fischer and Ivo W Molenaar. *Rasch models: Foundations, recent developments, and applications*. Springer Science & Business Media, 1995.

[Fleming, 2001] Neil D Fleming. *Teaching and learning styles: VARK strategies*. IGI Global, 2001.

[Galaburda et al., 1985] Albert M Galaburda, Gordon F Sherman, Glenn D Rosen, Francisco Aboitiz, and Norman Geschwind. Developmental dyslexia: four consecutive patients with cortical anomalies. *Annals of neurology*, 18(2):222–233, 1985.

[Galaburda et al., 2006] Albert M Galaburda, Joseph LoTurco, Franck Ramus, R Holly Fitch, and Glenn D Rosen. From genes to behavior in developmental dyslexia. *Nature neuroscience*, 9(10):1213–1217, 2006.

[Geary et al., 1991] David C. Geary, Sam C Brown, and V. A. Samaranayake. Cognitive addition: A short longitudinal study of strategy choice and speed-of-processing differences in normal and mathematically disabled children. *Developmental psychology*, 27(5):787–797, 1991.

[Gong et al., 2010] Yue Gong, Joseph E Beck, and Neil T Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *International Conference on Intelligent Tutoring Systems*, pages 35–44, 2010.

[Gonzalez-Brenes and Huang, 2015] Jose Gonzalez-Brenes and Yun Huang. Your model is predictive—but is it useful? theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation. In *Proceedings of the International Conference on Educational Data Mining*, pages 187–194. University of Pittsburgh, 2015.

[González-Brenes et al., 2014] J. P. González-Brenes, Yun Huang, and Peter Brusilovsky. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *Proceedings of the International Conference on Educational Data Mining*, pages 84–91, 2014.

[Goodfellow et al., 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Graf and Fife, 2013] E. A. Graf and J. H. Fife. Difficulty Modeling and Automatic Generation of Quantitative Items: Recent Advances and Possible Next Steps. In *Automatic Item Generation: Theory and Practice*, pages 157–179. Routledge, 2013.

[Gross and Vögeli, 2007] Markus Gross and Christian Vögeli. A multimedia framework for effective language training. *Computers & Graphics*, 31(5):761–777, 2007.

[Gu et al., 2014] Junjie Gu, Hang Cai, and Joseph E Beck. Investigate performance of expected maximization on the knowledge tracing model. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 156–161. Springer, 2014.

[Hadwin et al., 2007] Allyson F Hadwin, John C Nesbit, Dianne Jamieson-Noel, Jillianne Code, and Philip H Winne. Examining trace data to explore self-regulated learning. *Metacognition and Learning*, 2(2-3):107–124, 2007.

[Haffner et al., 2005] J. Haffner, K. Baro, P. Parzer, and F. Resch. Heidelberger Rechentest: Erfassung mathematischer Basiskomptenzen im Grundschulalter, 2005.

[Hao et al., 2015] Jiangang Hao, Zhan Shu, and Alina Davier. Analyzing Process Data from Game/Scenario- Based Tasks: An Edit Distance Approach. *Journal of Educational Data Mining*, 7(1):33–50, 2015.

[Harris, 1989] Deborah Harris. Comparison of 1-, 2-, and 3-parameter IRT models. *Educational Measurement*, 8(1):35–41, 1989.

[Hasselhorn, 2012] Marcus Hasselhorn. *Arbeitsgedächtnistestbatterie für Kinder von 5 bis 12 Jahren: AGTB 5-12*. Hogrefe, 2012.

[Hattie, 2003] John Hattie. Teachers make a difference, what is the research evidence? In *Building Teacher Quality: What does the research tell us ACER Research Conference*, volume 36, pages 27–38, 2003.

[He et al., 2008] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Proceddings of the IEEE International Joint Conference on Neural Networks*, pages 1322–1328, 2008.

[He et al., 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[Herold et al., 2013] James Herold, Alex Zundel, and Thomas F Stahovich. Mining meaningful patterns from students' handwritten coursework. In *Proceedings of the International Conference on Educational Data Mining*, pages 67–73, 2013.

[Higgins et al., 2016] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.

[Hoffmann, 2007] Heiko Hoffmann. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3):863–874, 2007.

[Hofmann and Buhmann, 1997] Thomas Hofmann and Joachim M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

[Hurvich and Tsai, 1989] Clifford M Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, pages 297–307, 1989.

[Imam et al., 2006] Tasadduq Imam, Kai Ting, and Joarder Kamruzzaman. z-svm: an svm for improved classification of imbalanced data. *AI 2006: Advances in Artificial Intelligence*, pages 264–273, 2006.

[Jarušek and Pelánek, 2012] Petr Jarušek and Radek Pelánek. Analysis of a simple model of problem solving times. In *Intelligent Tutoring Systems*, pages 379–388. Springer, 2012.

[Jenkins et al., 2007] Joseph R Jenkins, Roxanne F Hudson, and Evelyn S Johnson. Screening for at-risk readers in a response to intervention framework. *School Psychology Review*, 36(4):582, 2007.

[Jiang et al., 2016] Zhuoxuan Jiang, Peng Li, Yan Zhang, and Xiaoming Li. Generating Semantic Concept Map for MOOCs. In *Proceedings of the International Conference on Educational Data Mining*, pages 595–596, 2016.

[Jing, 2015] Shumin Jing. Automatic Grading of Short Answers for MOOC via Semi-supervised Document Clustering. In *Proceedings of the International Conference on Educational Data Mining*, pages 554–555, 2015.

[Johns and Woolf, 2006] Jeff Johns and Beverly Woolf. A Dynamic Mixture Model to Detect Student Motivation and Proficiency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 163–168, 2006.

[Käser et al., 2013a] T. Käser, A. G. Busetto, B. Solenthaler, G.-M. Baschera, J. Kohn, K. Kucian, M. von Aster, and M. Gross. Modelling and Optimizing Mathematics Learning in Children. *International Journal of Artificial Intelligence in Education*, 2013.

[Käser et al., 2013b] T. Käser, A. G. Busetto, B. Solenthaler, J. Kohn, M. von Aster, and M. Gross. Cluster-based prediction of mathematical learning patterns. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 389–399, 2013.

[Käser et al., 2013c] Tanja Käser, Gian-Marco Baschera, Juliane Kohn, Karin Kucian, Verena Richtmann, Ursina Grond, Markus Gross, and Michael von Aster. Design and evaluation of the computer-based training program calcularis for enhancing numerical cognition. *Frontiers in Developmental Psychology*, 4:489, 2013.

[Käser et al., 2016] Tanja Käser, Severin Klingler, and Markus Gross. When to stop?: Towards universal instructional policies. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, LAK '16, pages 289–298, New York, NY, USA, 2016. ACM.

[Käser, 2014] Tanja Käser. *Modeling and Optimizing Computer-Assisted Mathematics Learning in Children*. PhD thesis, Diss., ETH Zürich, Nr. 22145, 2014.

[Kast et al., 2007] M. Kast, M. Meyer, C. Vögeli, M. Gross, and L. Jäncke. Computer-based Multisensory Learning in Children with Developmental Dyslexia. *Restorative Neurology and Neuroscience*, 25(3-4):355–369, 2007.

[Kast et al., 2011] Monika Kast, Gian-Marco Baschera, Markus Gross, Lutz Jäncke, and Martin Meyer. Computer-based learning of spelling skills in children with and without dyslexia. *Annals of Dyslexia*, 61(2):177–200, 2011.

[Khajah et al., 2014a] Mohammad Khajah, Rowan Wing, Robert Lindsey, and Michael Mozer. Integrating latent-factor and knowledge-tracing models to predict individual differences in learning. In *Proceedings of the International Conference on Educational Data Mining*, pages 99–106, 2014.

[Khajah et al., 2014b] Mohammad M Khajah, Yun Huang, José P González-Brenes, Michael C Mozer, and Peter Brusilovsky. Integrating knowledge tracing and item response theory: A tale of two frameworks. *Proceedings of Workshop on Personalization Approaches in Learning Environments (PALE 2014) at the 22th International Conference on User Modeling, Adaptation, and Personalization*, pages 7–12, 2014.

[Khajah et al., 2016] Mohammad Khajah, Robert V. Lindsey, and Michael C. Mozer. How Deep is Knowledge Tracing? In *Proceedings of the International Conference on Educational Data Mining*, pages 94–101, 2016.

[King et al., 2007] Wayne King, Sally Giess, and Linda Lombardino. Subtyping of children with developmental dyslexia via bootstrap aggregated clustering and the gap statistic: comparison with the double-deficit hypothesis. *Int J Lang Comm Dis*, 42(1):77–95, 2007.

[Kingma and Ba, 2015] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*, 2015.

[Kingma and Welling, 2014] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Proceedings of the International Conference on Learning Representations*, 2014.

[Kingma et al., 2014] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 3581–3589, 2014.

[Kinnebrew and Biswas, 2012] John S Kinnebrew and Gautam Biswas. Identifying learning behaviors by contextualizing differential sequence mining with action features and performance evolution. In *Proceedings of the International Conference on Educational Data Mining*, pages 57–64, 2012.

[Kinnebrew et al., 2013] John S Kinnebrew, Daniel LC Mack, and Gautam Biswas. Mining temporally-interesting learning behavior patterns. In *Proceedings of the International Conference on Educational Data Mining*, pages 252–255, 2013.

[Kirkpatrick, 2009] Mark Kirkpatrick. Patterns of quantitative genetic variation in multiple dimensions. *Genetica*, 136(2):271–284, 2009.

[Klingler, 2013] Severin Klingler. A screening tool for children at risk of developmental dyscalculia, 2013. Master thesis, ETH Zurich.

[Köck and Paramythis, 2011] Mirjam Köck and Alexandros Paramythis. Activity sequence modelling and dynamic clustering for personalized e-learning. *Journal of User Modeling and User-Adapted Interaction*, 21(1):51–97, 2011.

[Koedinger et al., 1997] K. R. Koedinger, J. R. Anderson, W. H. Hadley, and M. A. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8(1):30–43, 1997.

[Koedinger et al., 2013] KennethR. Koedinger, JohnC. Stamper, ElizabethA. McLaughlin, and Tristan Nixon. Using Data-Driven Discovery of Better Student Models to Improve Student Learning. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 421–430, 2013.

[Kostopoulos et al., 2015] Georgios Kostopoulos, Sotiris Kotsiantis, and Panagiotis Pintelas. Predicting student performance in distance higher education using semi-supervised techniques. In *Model and Data Engineering*, pages 259–270. Springer, 2015.

[Krinzinger et al., 2007] Helga Krinzinger, Liane Kaufmann, Ann Dowker, Gemma Thomas, Martina Graf, Hans-Christoph Nuerk, and Klaus Willmes. Deutschsprachige Version des Fragebogens für Rechenangst (FRA) für 6-bis 9-jährige Kinder. *Zeitschrift für Kinder-und Jugendpsychiatrie und Psychotherapie*, 35(5):341–367, 2007.

[Kucian et al., 2006] K. Kucian, T. Loenneker, T. Dietrich, M. Dosch, E. Martin, and M. von Aster. Impaired neural networks for approximate calculation in dyscalculic children: a functional MRI study. *Behavioral and Brain Functions*, 2(1):31, 2006.

[Kucian et al., 2011] K. Kucian, U. Grond, S. Rotzer, B. Henzi, C. Schönmann, F. Plangger, M. Gälli, E. Martin, and M. von Aster. Mental Number Line Training in Children with Developmental Dyscalculia. *NeuroImage*, 57(3):782–795, 2011.

[Kulik and Fletcher, 2016] James A Kulik and JD Fletcher. Effectiveness of intelligent tutoring systems a meta-analytic review. *Review of Educational Research*, 86(1):42–78, 2016.

[Labutov and Lipson, 2016] Igor Labutov and Hod Lipson. Web as a textbook: Curating Targeted Learning Paths through the Heterogeneous Learning Resources on the Web. In *Proceedings of the International Conference on Educational Data Mining*, pages 110–118, 2016.

[Landerl et al., 2004] K. Landerl, A. Bevan, and B. Butterworth. Developmental dyscalculia and basic numerical capacities: a study of 8-9-year-old students. *Cognition*, 93:99–125, 2004.

[LeCun et al., 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Lee and Brunskill, 2012] Jung In Lee and Emma Brunskill. The Impact on Individualizing Student Models on Necessary Practice Opportunities. In *Proceedings of the International Conference on Educational Data Mining*, pages 118–125, 2012.

[Lehmann and Murray, 2005] Sandra Lehmann and Micah M Murray. The role of multisensory memories in unisensory object discrimination. *Cognitive Brain Research*, 24(2):326–334, 2005.

[Lepper and Chabay, 1988] Mark R. Lepper and Ruth W. Chabay. *Socializing the Intelligent Tutor: Bringing Empathy to Computer Tutors*, pages 242–257. Springer US, New York, NY, 1988.

[Lester et al., 2013] James C Lester, Eun Y Ha, Seung Y Lee, Bradford W Mott, Jonathan P Rowe, and Jennifer L Sabourin. Serious games get smart: Intelligent game-based learning environments. *AI Magazine*, 34(4):31–45, 2013.

[Lewis et al., 1994] C. Lewis, G. J. Hitch, and P. Walker. The prevalence of specific arithmetic difficulties and specific reading difficulties in 9- to 10-year old boys and girls. *Journal of Child Psychology and Psychiatry*, 35:283–292, 1994.

[Li and Biswas, 2000] Cen Li and Gautam Biswas. A Bayesian Approach to Temporal Data Clustering using Hidden Markov Models. In *Proceedings of the International Conference on Machine Learning*, pages 543–550, 2000.

[Li and Prakash, 2011] Lei Li and B Aditya Prakash. Time series clustering: Complex is simpler! In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 185–192, 2011.

[Liao et al., 2013] Hank Liao, Erik McDermott, and Andrew Senior. Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 368–373. IEEE, 2013.

[Lovegrove et al., 1990] William J Lovegrove, Ralph P Garzia, and Steven B Nicholson. Experimental evidence for a transient system deficit in specific reading disability. *Journal of the American Optometric Association*, page 137, 1990.

[Ma et al., 2014] Wenting Ma, Olusola O Adesope, John C Nesbit, and Qing Liu. Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology*, 106(4):901, 2014.

[Mandel et al., 2014] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline Policy Evaluation Across Representations with Applications to Educational Games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1077–1084, 2014.

[Martinez-Maldonado et al., 2013] Roberto Martinez-Maldonado, Kalina Yacef, and Judy Kay. Data mining in the classroom: Discovering groups' strategies at a multi-tabletop environment. In *Proceedings of the International Conference on Educational Data Mining*, pages 121–128, 2013.

[Mazzocco and Thompson, 2005] Michele MM Mazzocco and Richard E Thompson. Kindergarten predictors of math learning disability. *Learning Disabilities Research & Practice*, 20(3):142–155, 2005.

[Meila, 2005] Marina Meila. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd international conference on Machine learning*, pages 577–584. ACM, 2005.

[Meyer and Hailey, 2012] JP Meyer and E Hailey. A study of Rasch, partial credit, and rating scale model parameter recovery in WINSTEPS and jMetrik. *Journal of Applied Measurement*, 13(3):248–258, 2012.

[Miller and Soh, 2013] Lee Dee Miller and Leen-Kiat Soh. Meta-Reasoning Algorithm for Improving Analysis of Student Interactions with Learning Objects using Supervised Learning. In *Proceedings of the International Conference on Educational Data Mining*, pages 129–136, 2013.

[Min et al., 2014] Wookhee Min, Bradford W. Mott, Jonathan P. Rowe, and James C. Lester. Leveraging semi-supervised learning to predict student problem-solving performance in narrative-centered learning environments. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 664–665, 2014.

[Minku and Yao, 2012] Leandro L Minku and Xin Yao. Using unreliable data for creating more reliable online learners. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[Mnih et al., 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, An-

dreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Muldner and Conati, 2007] Kasia Muldner and Cristina Conati. Evaluating a decision-theoretic approach to tailored example selection. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 483–488, 2007.

[Murray et al., 2004] R. Charles Murray, Kurt Vanlehn, and Jack Mostow. Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence in Education*, 14(3-4):235–278, 2004.

[Nelder and Mead, 1965] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308–313, 1965.

[Nesbit et al., 2007] John C Nesbit, Mingming Zhou, Yabo Xu, and P Winne. Advancing log analysis of student interactions with cognitive tools. In *12th Biennial Conference of the European Association for Research on Learning and Insruction (EARLI)*, 2007.

[Nicolson and Fawcett, 1990] RI Nicolson and AJ Fawcett. Automaticity: A new framework for dyslexia research? *Cognition*, 35(2):159–182, 1990.

[Noël and Rousselle, 2011] Marie-Pascale Noël and Laurence Rousselle. Developmental changes in the profiles of dyscalculia: an explanation based on a double exact-and-approximate number representation model. *Frontiers in Human Neuroscience*, 5:165, 2011.

[Ostad, 1997] S. A. Ostad. Developmental differences in addition strategies: A comparison of mathematically disabled and mathematically normal children. *British Journal of Education Psychology*, 67:345–357, 1997.

[Ostad, 1999] S. A. Ostad. Developmental progression of subtraction strategies: A comparison of mathematically normal and mathematically disabled children. *European Journal of Special Needs Education*, 14:21–36, 1999.

[Pardo, 2005] Leandro Pardo. *Statistical inference based on divergence measures*. CRC Press, 2005.

[Pardos and Heffernan, 2010a] Z. A. Pardos and N. T. Heffernan. Modeling Individualization in a Bayesian Networks Implementation of Knowledge Tracing. In *Proceedings of the Conference on User Modelling, Adaptation and Personalization*, pages 255–266, 2010.

[Pardos and Heffernan, 2010b] Z. A. Pardos and N. T. Heffernan. Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm. In *Proceedings of the International Conference on Educational Data Mining*, pages 161–170, 2010.

[Pardos and Heffernan, 2011] Z. A. Pardos and N. Heffernan. Introducing item difficulty to the knowledge tracing model. In *Proceedings of the Conference on User Modelling, Adaptation and Personalization*, pages 243–254. Springer, 2011.

[Pardos and Yudelson, 2013] Zachary A Pardos and Michael Yudelson. Towards moment of learning accuracy. In *Workshops Proceedings of the International Conference on Artificial Intelligence in Education*, page 3, 2013.

[Pardos et al., 2012] Zachary A. Pardos, Shubhendu Trivedi, Neil T. Heffernan, and Gábor N. Sárközy. Clustered knowledge tracing. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 405–410, 2012.

[Pashler et al., 2009] Harold Pashler, Nicholas Cepeda, Robert V Lindsey, Ed Vul, and Michael C Mozer. Predicting the optimal spacing of study: A multiscale context model of memory. In *Advances in Neural Information Processing Systems 22*, pages 1321–1329. Curran Associates, Inc., 2009.

[Pavlik and Anderson, 2005] Philip I Pavlik and John R Anderson. Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science*, 29(4):559–586, 2005.

[Pavlik et al., 2009] Philip I. Pavlik, Hao Cen, and Kenneth R. Koedinger. Performance Factors Analysis - A New Alternative to Knowledge Tracing. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 531–538, 2009.

[Peckham and McCalla, 2012] Terry Peckham and Gord McCalla. Mining Student Behavior Patterns in Reading Comprehension Tasks. In *Proceedings of the International Conference on Educational Data Mining*, pages 87–94, 2012.

[Pedregosa and others, 2011] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

[Pelánek, 2014] Radek Pelánek. A brief overview of metrics for evaluation of student models. In *Approaching Twenty Years of Knowledge Tracing Workshop of the 7th International Conference on Educational Data Mining*, 2014.

[Pelleg and Moore, 2000] Dan Pelleg and Andrew Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the International Conference on Machine Learning*, volume 1, pages 727–734, 2000.

[Perera et al., 2009] Dilhan Perera, Judy Kay, Irena Koprinska, Kalina Yacef, and Osmar R Zaïane. Clustering and sequential pattern mining of online collaborative learning data. *IEEE Transactions on Knowledge and Data Engineering*, 21(6):759–772, 2009.

[Petermann and Petermann, 2008] Franz Petermann and Ulrike Petermann. HAWIK-IV. *Kindheit und Entwicklung*, 17(2):71–75, 2008.

[Piech et al., 2015] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J. Guibas, and Jascha Sohl-Dickstein. Deep Knowledge Tracing. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 505–513, 2015.

[Rafferty et al., 2011] Anna N. Rafferty, Emma Brunskill, Thomas L. Griffiths, and Patrick Shafto. Faster Teaching by POMDP Planning. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 280–287, 2011.

[Rebolledo-Mendez et al., 2013] Genaro Rebolledo-Mendez, Benedict Du Boulay, Rosemary Luckin, and Edgard Ivan Benitez-Guerrero. Mining data from interactions with a motivational-aware tutoring system using data visualization. *Journal of Educational Data Mining*, 5(1):72–103, 2013.

[Reiter et al., 2000] Ehud Reiter, Robert Dale, and Zhiwei Feng. *Building natural language generation systems*. Cambridge University Press, 2000.

[Reshef et al., 2011] David N Reshef, Yakir A Reshef, Hilary K Finucane, Sharon R Grossman, Gilean McVean, Peter J Turnbaugh, Eric S Lander, Michael Mitzenmacher, and Pardis C Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[Reye, 2004] J. Reye. Student Modelling Based on Belief Networks. *International Journal of Artificial Intelligence in Education*, 14(1):63–96, 2004.

[Rezende et al., 2014] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the International Conference on Machine Learning*, pages 1278–1286, 2014.

[Ritter et al., 2009] Steven Ritter, Thomas K. Harris, Tristan Nixon, Daniel Dickison, R. Charles Murray, and Brendon Towle. Reducing the knowledge tracing space. In *Proceedings of the International Conference on Educational Data Mining*, pages 151–160, 2009.

[Rollinson and Brunskill, 2015] Joseph Rollinson and Emma Brunskill. From Predictive Models to Instructional Policies. *Proceedings of the International Conference on Educational Data Mining*, pages 179–186, 2015.

[Rüsseler et al., 2006] Jascha Rüsseler, Ivonne Gerth, and Thomas F Münte. Implicit learning is intact in adult developmental dyslexic readers: Evidence from the serial reaction time task and artificial grammar learning. *Journal of Clinical and Experimental Neuropsychology*, 28(5):808–827, 2006.

[Santos and Silva, 2008] FH Santos and PA Silva. Avaliação da discalculia do desenvolvimento: uma questão sobre o processamento numérico e o cálculo. *Transtornos de Aprendizagem: da Avaliação à reabilitação*, pages 125–137, 2008.

[Santos et al., 2012] Flávia Heloísa Dos Santos, Paulo Adilson Da Silva, Fabiana Silva Ribeiro, Ana Luiza Ribeiro Pereira Dias, Michele Cândida Frigério, Georges Dellatolas, and Michael von Aster. Number processing and calculation in Brazilian children aged 7-12 years. *The Spanish journal of psychology*, 15(02):513–525, 2012.

[Scheffe, 1999] Henry Scheffe. *The analysis of variance*, volume 72. John Wiley & Sons, 1999.

[Schmid et al., 2014] Richard F Schmid, Robert M Bernard, Eugene Borokhovski, Rana M Tamim, Philip C Abrami, Michael A Surkes, C Anne Wade, and Jonathan Woods. The effects of technology use in postsecondary education: A meta-analysis of classroom applications. *Computers & Education*, 72:271–291, 2014.

[Schölkopf et al., 1997] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 583–588, 1997.

[Schulte-Körne et al., 2004] G Schulte-Körne, W Deimel, J Bartling, and H Remschmidt. Neurophysiological correlates of word recognition in dyslexia. *Journal of Neural Transmission*, 111(7):971–984, 2004.

[Senesh and Wolf, 2009] M Senesh and A Wolf. Motion estimation using point cluster method and kalman filter. *Journal of biomechanical engineering*, 131(5):051008, 2009.

[Shalev and von Aster, 2008] R. Shalev and M. G. von Aster. Identification, classification, and prevalence of developmental dyscalculia. *Encyclopedia of language and literacy development*, 2008.

[Shalev, 2004] R. S. Shalev. Developmental dyscalculia: Review. *Journal of Child Neurology*, 19:765–771, 2004.

[Shams and Seitz, 2008] Ladan Shams and Aaron R Seitz. Benefits of multisensory learning. *Trends in cognitive sciences*, 12(11):411–417, 2008.

[Shaywitz, 1998] Sally E Shaywitz. Dyslexia. *New England Journal of Medicine*, 338(5):307–312, 1998.

[Shute, 2011] Valerie J Shute. Stealth assessment in computer-based games to support learning. *Computer games and instruction*, 55(2):503–524, 2011.

[Silver et al., 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[Solenthaler et al., under review] Barbara Solenthaler, Severin Klingler, Tanja Käser, and Markus Gross. Under Review: An Architecture for Intelligent Educational Games. *IEEE Computer Graphics and Applications*, under review.

[Soller and Lesgold, 2007] Amy Soller and Alan Lesgold. Modeling the process of collaborative learning. In *The role of technology in CSCL*, pages 63–86. Springer, 2007.

[Sønderby et al., 2016] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 3738–3746, 2016.

[Srivastava et al., 2014] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Stamper and Koedinger, 2011] John C. Stamper and Kenneth R. Koedinger. Human-machine Student Model Discovery and Improvement Using DataShop. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 353–360, 2011.

[Stein, 1994] Lílian Milnitsky Stein. TDE: teste de desempenho escolar: manual para aplicação e interpretação. *São Paulo: Casa do Psicólogo*, pages 1–17, 1994.

[Tam et al., 2015] Vincent Tam, Edmund Y Lam, ST Fung, WWT Fok, and Allan HK Yuen. Enhancing educational data mining techniques on online educational resources with a semi-supervised learning approach. In *Proceedings of the IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 203–206, 2015.

[Thomas and Dowker, 2000] G Thomas and A Dowker. Mathematics anxiety and related factors in young children. In *British Psychological Society Developmental Section Conference*, 2000.

[Tinto, 1997] Vincent Tinto. Classrooms as communities: Exploring the educational character of student persistence. *The Journal of higher education*, 68(6):599–623, 1997.

[Trivedi et al., 2011] Shubhendu Trivedi, Zachary A. Pardos, and Neil T. Heffernan. Clustering students to generate an ensemble to improve standard test

score predictions. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 377–384, 2011.

[Trivedi et al., 2012] Shubhendu Trivedi, Zachary A. Pardos, Gábor N. Sárközy, and Neil T. Heffernan. Co-clustering by bipartite spectral graph partitioning for out-of-tutor prediction. In *Proceedings of the International Conference on Educational Data Mining*, pages 33–40, 2012.

[Turian et al., 2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394, 2010.

[Van de Sande, 2013] Brett Van de Sande. Properties of the Bayesian knowledge tracing model. *Journal of Educational Data Mining*, 5(2):1–10, 2013.

[van den Oord et al., 2016] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional Image Generation with PixelCNN Decoders. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 4790–4798, 2016.

[van der Linden, 2006] Wim J van der Linden. A lognormal model for response times on test items. *Journal of Educational and Behavioral Statistics*, 31(2):181–204, 2006.

[Vanlehn et al., 2005] Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3):147–204, 2005.

[von Aster and Shalev, 2007a] M. G. von Aster and R. Shalev. Number development and developmental dyscalculia. *Developmental Medicine and Child Neurology*, 49:868–873, 2007.

[Von Aster and Shalev, 2007b] Michael G Von Aster and Ruth S Shalev. Number development and developmental dyscalculia. *Developmental Medicine & Child Neurology*, 49(11):868–873, 2007.

[von Aster et al., 2006] Michael von Aster, Monika Weinhold Zulauf, and Ralf Horn. *Neuropsychologische Testbatterie für Zahlenverarbeitung und Rechnen bei Kindern: ZAREKI-R*. Pearson, 2006.

[Von Aster et al., 2015] M. Von Aster, L. Rauscher, K. Kucian, T. Käser, U. McCaskey, and J. Kohn. Calcularis - Evaluation of a computer-based learning program for enhancing numerical cognition for children with developmental dyscalculia, 2015. 62nd Annual Meeting of the American Academy of Child and Adolescent Psychiatry.

[von Aster, 2000] Michael von Aster. Developmental cognitive neuropsychology of number processing and calculation: varieties of developmental dyscalculia. *European Child & Adolescent Psychiatry*, 9:41–57, 2000.

[Wang and Beck, 2012] Yutao Wang and Joseph E Beck. Using student modeling to estimate student knowledge retention. *International Educational Data Mining Society*, pages 200–203, 2012.

[Wang and Beck, 2013] Y. Wang and J. Beck. Class vs. Student in a Bayesian Network Student Model. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 151–160, 2013.

[Wang and Heffernan, 2012] Y. Wang and N. T. Heffernan. The student skill model. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 399–404. Springer, 2012.

[Weiss, 2006] R.H. Weiss. *Grundintelligenztest Skala 2 (CFT 20-R)*. Hogrefe, Göttingen, 2006.

[Wilson and De Boeck, 2004] M. Wilson and P. De Boeck. Descriptive and explanatory item response models. In *Explanatory item response models: A generalized linear and nonlinear approach*, pages 43–74. Springer, 2004.

[Woolf et al., 2009] Beverly Woolf, Winslow Burleson, Ivon Arroyo, Toby Dragon, David Cooper, and Rosalind Picard. Affect-aware tutors: recognising and responding to student affect. *International Journal of Learning Technology*, 4(3-4):129–164, 2009.

[Woolf et al., 2013] Beverly Park Woolf, H Chad Lane, Vinay K Chaudhri, and Janet L Kolodner. AI Grand Challenges for Education. *AI Magazine*, 34(4):66–84, 2013.

[Woolger, 2001] Christi Woolger. *Wechsler Intelligence Scale for Children-Third Edition (wisc-iii)*, pages 219–233. Springer US, Boston, MA, 2001.

[World Health Organization, 1993] World Health Organization. *The ICD-10 classification of mental and behavioural disorders: diagnostic criteria for research*, volume 10. Geneva: World Health Organization, 1993.

[Xu and Mostow, 2011] Yanbo Xu and Jack Mostow. Using logistic regression to trace multiple subskills in a dynamic Bayes net. In *Proceedings of the International Conference on Educational Data Mining*, pages 241–246, 2011.

[Xu and Mostow, 2013] Yanbo Xu and Jack Mostow. Using item response theory to refine knowledge tracing. In *Proceedings of the International Conference on Educational Data Mining*, pages 356–357, 2013.

[Xu et al., 2013a] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[Xu et al., 2013b] Kevin S Xu, Mark Kliger, and Alfred O Hero. A regularized graph layout framework for dynamic network visualization. *Data Mining and Knowledge Discovery*, pages 1–33, 2013.

[Xu et al., 2014] Kevin S Xu, Mark Kliger, and Alfred O Hero Iii. Adaptive evolutionary clustering. *Data Mining and Knowledge Discovery*, 28(2):304–336, 2014.

[Yudelson et al., 2013] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized Bayesian Knowledge Tracing Models. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 171–180, 2013.

[Zhang, 2004] Harry Zhang. The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, pages 562–567, 2004.

[Zhou and Li, 2005] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[Zhu, 2006] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2006.

[Ziegler et al., 2005] Andreas Ziegler, Inke R König, Wolfgang Deimel, Ellen Plume, Markus M Nöthen, Peter Propping, André Kleensang, Bertram Müller-Myhsok, Andreas Warnke, Helmut Remschmidt, et al. Developmental dyslexia–recurrence risk estimates from a german bi-center study using the single proband sib pair design. *Human heredity*, 59(3):136–143, 2005.

[Zimmermann and Fimm, 2009] Peter Zimmermann and Bruno Fimm. *Testbatterie zur Aufmerksamkeitsprüfung-Version 2.2 (TAP)*. Psytest, 2009.