DISS. ETH NO. 22145

Modeling and Optimizing Computer-Assisted Mathematics Learning in Children

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

Tanja Käser Jacober

Master of Science ETH in Computer Science, ETH Zurich, Switzerland
born on 28.06.1982
citizen of Dürrenroth (BE), Switzerland

accepted on the recommendation of

Prof. Dr. Markus Gross, examiner
Prof. Dr. Kenneth R. Koedinger, co-examiner
Prof. Dr. Michael von Aster, co-examiner

2014

Abstract

Arithmetic abilities are essential in modern society. However, many children suffer from difficulties in learning mathematics, ranging from mild to severe numeracy problems. The prevalence of developmental dyscalculia is about 3% – 6% in German speaking countries. Children with developmental dyscalculia often develop anxiety and aversion against the subject and experience difficulties in school and later in profession. Despite the relatively high prevalence, few targeted interventions for children with developmental dyscalculia exist and only a fraction of these programs is computer-based.

In this thesis, we present a complete loop in the data-driven development of an intelligent tutoring system for mathematics learning that overcomes the limitations of previous work. This process consists of three steps: The development of a first training environment, its evaluation in user studies and the data-driven validation and improvement of the system.

We first develop `Calcularis`, a computer-based training program for children with difficulties in learning mathematics. The curriculum and concepts of the system are theory-based: The program transforms current neuro-cognitive findings into the design of different instructional games. A Bayesian network student model representing different mathematical skills and their dependencies, and a non-linear control algorithm ensure adaptation of the training to the mathematical abilities of the individual child. Furthermore, the program features a bug library allowing recognition and remediation of specific errors.

In a second step, we evaluate `Calcularis` in two user studies to prove its effectiveness. Based on the input data collected in these studies, we perform a data-driven validation and improvement of the program in the third step.

We assess student model and controller properties and analyze the quality of our model via logistic regression. The data-driven investigations lead to the development and extensive analysis of techniques for model validation. We improve prediction accuracy of the student model by introducing a *constrained latent structured prediction* method for efficient parameter learning in Bayesian networks. By applying a clustering and classification approach, we are able to predict the mathematical characteristics of the children. Moreover, we also explore the possible addition of an engagement model to `Calcularis`.

Finally, we develop a data-driven diagnosis tool for developmental dyscalculia based only on input data. The integration of this tool into `Calcularis` closes the loop of data-driven development.

Zusammenfassung

In der heutigen Gesellschaft sind mathematische Fähigkeiten sehr wichtig. Viele Kinder haben jedoch grosse Schwierigkeiten mit der Zahlenverarbeitung oder beim Rechnen. In deutschsprachigen Ländern leiden etwa 3% – 6% der Kinder unter Dyskalkulie. Kinder mit Dyskalkulie entwickeln oft eine Abneigung gegen die Mathematik oder sogar Mathematikangst und haben Schwierigkeiten in der Schule und später im Berufsleben. Trotzdem existieren nur wenige Therapieprogramme für Kinder mit Dyskalkulie und nur ein Bruchteil dieser Programme ist computerbasiert.

In dieser Dissertation beschreiben wir einen kompletten Zyklus der datengestützten Entwicklung einer intelligenten Lernumgebung. Dieser Zyklus besteht aus drei Schritten: Der Entwicklung einer ersten Lernumgebung, der Evaluation dieser Umgebung in Benutzerstudien sowie der datengestützten Validierung und Verbesserung des Systems.

In einem ersten Schritt entwickeln wir `Calcularis`, ein intelligentes Trainingsprogramm für Kinder mit mathematischen Schwierigkeiten. Die verschiedenen Spiele von `Calcularis` basieren auf aktuellen Erkenntnissen aus der Neuropsychologie. Ein Bayes-Netz, das verschiedene mathematische Fähigkeiten und deren Abhängigkeiten repräsentiert, sowie ein nicht-linearer Kontrollalgorithmus ermöglichen eine Anpassung des Trainings an die mathematsichen Fähigkeiten der einzelnen Kinder.

In einem zweiten Schritt evaluieren wir die Effektivität von `Calcularis` in zwei Benutzerstudien. Basierend auf den Logfiles dieser Benutzerstudien validieren und verbessern wir das Programm in einem dritten Schritt.

Wir untersuchen die Eigenschaften des adaptiven Modells und des Kontrollalgorithmus und analysieren die Qualität des Modells mittels einer logistischen Regression. Ausserdem führen wir eine umfassende Analyse von verschiedenen Techniken zur Modellvalidierung durch. Wir verbessern die Genauigkeit des adaptiven Modells durch einen Algorithmus zum effizienten Erlernen der Parameter eines Bayes-Netzes. Mittels einer Clustering- und Klasssifikationsmethode sagen wir die mathematischen Eigenschaften der Kinder voraus. Ausserdem untersuchen wir die Möglichkeit, ein Motivationsmodell zu `Calcularis` hinzuzufügen.

Schlussendlich entwickeln wir ein datenbasiertes Diagnosetool für Dyskalkulie. Die Erweiterung von `Calcularis` durch dieses Tool schliesst den datenbasierten Entwicklungszyklus.

Acknowledgments

My sincere thanks go to my advisor Prof. Markus Gross. He initiated the research in intelligent learning environments and gave me the opportunity to work in this exciting field. I am grateful for his great trust and the freedom to investigate new research fields. His experience and his unconditional support were invaluable during my Ph.D.

I am also deeply thankful to Prof. Michael von Aster, who guided me closely during the last four years. He ignited my interest in the fascinating area of human learning. His boundless enthusiasm and profound scientific experience in the field of mathematics learning were invaluable for my work. Special thanks go to Prof. Kenneth R. Koedinger, who hosted me at Carnegie Mellon University. I am thankful for the great amount of time he dedicated to advising me. I was able to profit a lot from his long experience in intelligent tutoring systems and educational data mining.

Furthermore, I would like to thank my close collaborators. It was a pleasure to work with Dr. Alberto Giovanni Busetto, who offered great help for many of the technical aspects of this thesis. Moreover, I also want to thank Dr. Alexander Schwing for the fruitful collaboration and the many inspiring discussions. I am grateful for the support of Dr. Barbara Solenthaler whose advice and scientific intuition were of great value. Many thanks also go to my collaborators Dr. Juliane Kohn, Dr. Karin Kucian, Ursina Grond, Verena Richtmann and Larissa Rauscher who contributed to the psychological part of this thesis. It was a great pleasure to work with them and I am thankful for the great commitment and support offered during my Ph.D.

I would like to thank Christian Vögeli, Matthias Ringwald and all the other coworkers at Dybuster AG. It was a pleasure to work with them and I wish them all the best for their future with Dybuster and Calcularis.

Special thanks go to all current and former members of CGL, DRZ and IGL, which made the time at the lab fun, diverting and a great place to spend the many days and nights.

I am deeply thankful to my friends and family, particularly to my parents, for their great support during my Ph.D. Last, I am most thankful to my husband René for going with me through the ups and downs of my Ph.D. This work would not have been possible without your enduring love, understanding and support.

This thesis was funded by the CTI-grant 11006.1.

Contents

CHAPTER 1

Introduction

Arithmetic skills are essential in modern society. However, many children experience difficulties in learning mathematics, ranging from mild to severe numeracy problems. The prevalence of developmental dyscalculia is estimated to about $3-6\%$ in German and English speaking countries (Lewis et al., 1994; Shalev and von Aster, 2008). Learning disabilities often lead to anxiety and aversion against the subject (Rubinsten and Tannock, 2010) and to underperformance in school and later in profession (Bynner, 1997).

Despite the relatively high prevalence of developmental dyscalculia, only a few scientifically evaluated interventions exist, and only a fraction of these programs is computer-based. And yet, the computer presents an inexpensive extension to conventional one-to-one therapy. Computers are an attractive medium for children and can provide intensive training in a stimulating environment. The playful environment in combination with the fact that the computer is an emotionally neutral medium may also lead to increased motivation and enhance positive self-concepts. Most importantly, educational software can be designed to adapt to an individual child's abilities, behavior or affective states. Existing computer-based interventions for number processing (Wilson et al., 2006a; Fuchs et al., 2006; Lenhard et al., 2011; Kucian et al., 2011; Butterworth et al., 2011) are based on neuro-cognitive models of number processing and numerical development. This theoretical basis is an important criterion for a sound targeted intervention. However, the existing computer-based interventions provide only limited user adaptability.

The field of intelligent tutoring systems provides a large body of research in terms of adaptivity and user modeling. A fundamental property of these systems is their 'intelligence', *i.e.*, the adaptation to the student. Popular techniques for modeling student knowledge include Bayesian Knowledge Tracing (Corbett and Anderson, 1994) or Performance Factors Analysis (Pavlik et al., 2009). Furthermore, Markov Decision Processes are used for teaching planning (Brunskill and Russell, 2011; Rafferty et al., 2011) or diagnosing misconceptions (Rafferty et al., 2012). Bayesian network models are employed to model and predict students' learning styles (Kim et al., 2012), engagement states (Baschera et al., 2011) and goals (Conati et al., 2002). Intelligent tutoring systems have been successfully employed in different learning domains, amongst others also for learning mathematics (Koedinger et al., 1997; Arroyo et al., 2004; Rau et al., 2009). Existing systems in the domain of mathematics are, however, designed for normal learning children and focus on specific aspects of the domain.

The work presented in this thesis is concerned with the development and the efficacy of computer-assisted therapy approaches for developmental dyscalculia. It spans several areas of interest, including elements of psychology, student modeling and data mining. We address the limitations of previous work by combining knowledge about developmental dyscalculia and mathematical understanding with state of the art modeling and data mining techniques. We first develop the therapy software `Calcularis` for developmental dyscalculia, drawing from the fields of developmental and neuro-psychology as well as intelligent tutoring systems. In a second step, the software is evaluated in two large user studies to prove its effectiveness. The input data collected in these user studies is then used for model validation. We investigate how existing models can be validated and improved based on log file data. The data-driven investigations lead to the development and extensive analysis of techniques for model validation. Based on the collected log file data, we also address the question of how a computer-based system can identify, represent and predict the knowledge and affective states of the user. We propose a mathematical knowledge representation along with efficient learning and inference methods that outperform existing student models in prediction of knowledge. By employing a clustering and classification algorithm, we are able to predict mathematical learning characteristics of the children. Furthermore, we explore the addition of a framework for engagement modeling to `Calcularis`. Finally, we complete `Calcularis` with a data-driven diagnosis tool, which is able to classify children as being at risk for developmental dyscalculia based only on their input data.

In the following, we will give an overview of the performed research activities during this thesis. We then present the principal contributions of the work, before outlining the structure of the thesis. Finally, we list the publications that have been accepted in the context of this thesis.

1.1 Overview

In this thesis, we describe one complete loop in the data-driven development of an intelligent tutoring system: From the development of a first system over the evaluation in user studies to the data-driven validation and improvement of the system. A conceptual overview of the performed research activities and resulting publications can be found in Fig. 1.1.

Calcularis. Computer-based therapy systems present inexpensive extensions to conventional one-to-one therapy by providing an adaptive and fear-free learning environment. The effectiveness of computer-based therapy programs has been proven by several user studies targeting children with dyslexia (Gross and Vögeli, 2007; Kast et al., 2007) or ADHD (Klingberg et al., 2005).

In this thesis, we develop the computer-based therapy system `Calcularis` for elementary school children with developmental dyscalculia or difficulties in learning mathematics. The training program combines knowledge about developmental dyscalculia and mathematical understanding with state of the art modeling techniques. We transform current neuro-cognitive findings into the design of different instructional games. Furthermore, we use a special design for numerical stimuli, encoding the properties of numbers using visual cues such as colors, forms and topologies. This special design aims at enhancing the different properties of number and hence facilitating number understanding. The transfer of information through different channels also stimulates perception and facilitates the retrieval of memory (Lehmann and Murray, 2005; Shams and Seitz, 2008). We employ concepts from student modeling and machine learning, enabling the system to adapt to the knowledge level of the user. A bug library, allowing recognition and remediation of specific errors of the children, completes the program.

We model the mathematical knowledge of the user with a dynamic Bayesian network (Murphy, 2002). This network is a directed acyclic graph representing different mathematical skills as well as their hierarchy and interrelationships. Compared to previous approaches employing Hidden Markov Models (Corbett and Anderson, 1994; Reye, 2004), our knowledge representation has three main advantages. First, we are able to consider all skills jointly within one model. Second, the ability to model the hierarchy and the dependencies between different skills of a learning domain increases the representational power of the model. And third, the non-linear structure conveys a more complex and adaptive control algorithm than a simple linear hierarchy.

Based on the non-linear knowledge representation, we introduce a control algorithm that allows movement into different directions. Besides advancing to more difficult skills, we allow the controller to select easier skills for training or pick remediation

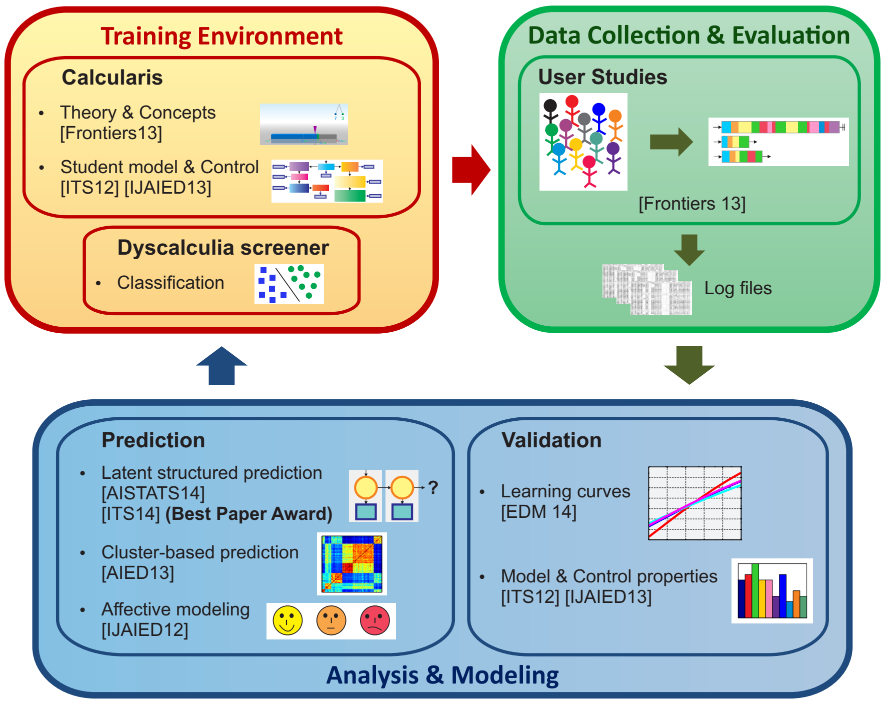

Training Environment
Calcularis
Theory & Concepts [Frontiers13]
Student model & Control [ITS12] [IJAIED13]
Dyscalculia screener
Classification
Data Collection & Evaluation
User Studies
[Frontiers 13]
Log files
Prediction
Latent structured prediction [AISTATS14] [ITS14] (Best Paper Award)
?
Cluster-based prediction [AIED13]
Affective modeling [IJAIED12]
Validation
Learning curves [EDM 14]
Model & Control properties [ITS12] [IJAIED13]
Analysis & Modeling

Figure 1.1: Thesis Overview: This work describes one complete loop in the data-driven development of a computer-based training system. We first develop **Calcularis** (red), a computer-based training program for learning mathematics. In a second step, the system is evaluated in two **user studies** (green), proving its effectiveness. The collected input data is used for model **validation** (blue). We perform a data-driven assessment of student model and controller properties and analyze the quality of our model via logistic regression. We also improve the system regarding **prediction** (blue). We propose a constrained structured prediction method for increasing the accuracy of the model. By applying a clustering and classification approach, we are able to predict students' mathematical characteristics. In addition, we explore the possibility of a general framework for modeling affective states. In a final step, we close the loop of the data-driven development: Our **Dyscalculia screener** (red) is able to classify children as having developmental dyscalculia based only on their input data.

skills for specific errors. By doing so, the path through the skill network is different for each child and targets the needs of the individual user. Furthermore, forgetting and knowledge gaps are captured by the possibility to perform skill retrocession.

User studies. When developing an intervention program, assessment of the actual clinical effectiveness by means of evaluation studies is essential. We evaluate the effectiveness of `Calcularis` in two user studies in Germany and Switzerland with elementary school children. The results demonstrate that children improve significantly in addition and subtraction regarding accuracy and solution times. Furthermore, they also exhibit a refined spatial number representation after training. Our results confirm the findings of previous studies (Siegler and Booth, 2004; Booth and Siegler, 2006, 2008; Halberda et al., 2008), which demonstrated significant correlations between arithmetical learning and the quality of numerical magnitude representation.

Validation. An important part when developing an intelligent tutoring system is the data-driven validation of the student model and the control algorithm. We validate our skill model by applying the often used approach of Additive Factors Models (Cen et al., 2007, 2008). This type of logistic regression model might, however, suffer from underestimation of student learning when applied to a mastery-learning data set. We therefore extensively investigate, how student learning can be measured in a mastery-based system. We suggest a variety of logistic regression models and analyze their properties. Furthermore, we also compare prediction accuracy of the different modeling techniques on unseen data.

In a second step, we perform a data-driven assessment of the developed student model and control algorithm according to different quality criteria. We demonstrate that students show an increased mathematical performance over the training period within the system using logistic regression. Furthermore, we assess the controller design and show that the possibility of going back to easier skills speeds up learning. We do so by using a logistic regression approach with bootstrapping. Finally, we also demonstrate that the system adjusts rapidly to the knowledge state of the user.

Prediction. Prediction is a fundamental task of an intelligent tutoring system. The quality of the student model can be measured by its prediction accuracy, *i.e.*, how good the model is at prediction on unseen data. Most previous work has focused on what we will call *short-term prediction*: Given the outcomes of tasks $1,\dots,n-1$, what will be the outcome of task n? In this thesis, we investigate *short-term prediction* as well as *long-term prediction*. *Long-term prediction* is for example the prediction of the overall training outcome, knowledge gaps of the student or performance in external training assessments. Another aspect of prediction that we will explore is the possibility of a general framework for predicting engagement states of the user.

Prediction of task outcomes of the student directly influences task selection and therefore training efficiency. One of the most used approaches to model student knowledge is Bayesian Knowledge Tracing (Corbett and Anderson, 1994), a special case of a Hidden Markov Model (Reye, 2004). Prediction accuracy of this method has been improved using clustering approaches (Pardos et al., 2012b) or individualization techniques, such as learning student- and skill-specific parameters (Pardos and Heffernan, 2010a; Wang and Heffernan, 2012; Yudelson et al., 2013) or modeling the parameters per school class (Wang and Beck, 2013). In this thesis, we will exploit the potential of dynamic Bayesian networks (Murphy, 2002). As these models usually do not exhibit a tree structure, they impose challenges for inference and learning. We introduce a method called *constrained structured prediction with latent variables* for efficient parameter learning, yielding accurate and interpretable models. We apply a constrained optimization and demonstrate that this regularization through constraints improves prediction accuracy and guarantees model interpretability. Furthermore, we also show that the increased representational power of our models yields significant improvements in prediction accuracy over Bayesian Knowledge Tracing (Corbett and Anderson, 1994).

Existing models are mostly focused on predicting student knowledge, *i.e.*, task outcomes. In this thesis, we propose a method which predicts learning characteristics of students such as knowledge gaps and overall training achievement. We will use an approach consisting of an offline clustering, followed by an online classification of children. At the end of the training, children are clustered into subgroups of similar mathematical performance based on their training characteristics. The resulting subgroups can be interpreted according to theory and concepts about the development of mathematical understanding. During training, children are classified to a specific subgroup based on the training information available so far. Prediction of future performance and knowledge gaps is then performed using subgroup information.

Affective modeling is receiving increasing attention due to its recognized relevance in learning. In general, affective models can be inferred from several sources, such as sensor data (Cooper et al., 2010; Heraz and Frasson, 2009) and user input data (Baker et al., 2004; Johns and Woolf, 2006; Arroyo and Woolf, 2005). In previous work, Baschera et al. (2011) have developed an engagement dynamics model in spelling learning that can adapt the training to individual students based on data-driven identification of engagement states from student input. Building upon this model, we explore whether we can transfer the existing framework to a more general engagement dynamics model for multiple learning domains. In particular, we focus on developmental dyslexia and dyscalculia. We argue that the assumption of similar engagement patterns in the two cases is justified and, thus, that a similar engagement model would be beneficial. This work is a purely theoretical exploration. We provide a detailed assessment of similarities and dissimilarities of the two cases of developmental dyslexia and dyscalculia in terms of learning domain and student model and analyze the re-usability of the engagement model for spelling learning.

Dyscalculia screener. The diagnosis of developmental dyscalculia (or a learning disability in general) involves a range of standardized tests assessing children's domain specific as well as domain general abilities. A computer-based diagnosis tool would allow for an inexpensive, nationwide screening. In this thesis, we develop a diagnosis tool, classifying children as being at risk for developmental dyscalculia based on their input data. We extract task dependent (such as the answer time) as well as game dependent features from the log files collected in the user studies. Those features will be pre-processed by applying kernel transformations to make them comparable. A pairwise clustering allows us to group features according to their similarities and to therefore reduce the number of selected features for classification. Classification is then performed using a Support Vector Machine (Caruana and Niculescu-Mizil, 2006; Statnikov et al., 2008) or a probabilistic classifier allowing for an adaptive test time and providing information about the certainty of the predicted label. The development of the diagnosis tool and its integration into `Calcularis` constitute the final step of this thesis and close the loop of data-driven development.

1.2 Principal Contributions

In the following, we summarize the principle contributions of the work presented in this thesis:

- *Computer-based therapy system for dyscalculia*: We introduce `Calcularis`, a new computer-based training program for children with difficulties in learning mathematics. The program combines theory and concepts of numerical development with state of the art modeling techniques. Its structure and games are based on neuro-cognitive models of number processing and numerical development. Adaptivity is ensured through a Bayesian network model representing the mathematical skills of the user and a non-linear control algorithm. Furthermore, the system features a bug library allowing adaptation to specific problems of the user. The effectiveness of the training program has been demonstrated in two user studies.
- *Dynamic Bayesian network model*: We model the mathematical knowledge of the user with a dynamic Bayesian network (Murphy, 2002) representing 100 different mathematical skills and their relationships. Compared to previous work employing Bayesian Knowledge Tracing (Corbett and Anderson, 1994; Koedinger et al., 1997), we are able to model the different skills of a learning domain jointly within a single model, which increases the representational power of the model. We present an approach called *constrained structured prediction with latent variables* for efficient parameter learning in general graphical models and show that our regularization via parameter constraints improves prediction accuracy on unseen data. We furthermore perform experiments on large-scale data sets from different learning domains such as mathematics, spelling learning and physics, demonstrating that the modeling of skill hierarchies increases the predictive performance of a model.
- *Non-linear control algorithm*: We propose a non-linear control algorithm for task selection. In contrast to other systems (Conati et al., 2002; Koedinger et al., 1997; Gross and Vögeli, 2007), we allow forward (advancing to more difficult skills) and backward (going back to easier skills) movements along the edges of the skill model's graph structure as well as 'jumps' to remediation skills for specific errors. This control design allows for an adaptation of the training sequence to the individual child. Furthermore, forgetting and knowledge gaps are implicitly captured.

We provide a data-driven validation of the control design and demonstrate that this non-linear design is beneficial for learning.

- *Clustering and classification*: We introduce a clustering and classification approach for predicting external assessment results as well as learning characteristics such as knowledge gaps and overall training achievement of the children. In a first step, we cluster children according to individual learning trajectories. Compared to previous approaches, we use the subgroup information not only to improve prediction accuracy, but also to provide a valuable tool for experts to analyze individual learning patterns. The second step consists of a supervised online classification during training, enabling prediction of future performance. We show that prediction accuracy of learning characteristics can be significantly improved by taking subgroup information into account.

- *Learning curve analyses*: Improving cognitive models of learners based on log file data is a common approach. Prior work (Murray et al., 2013) began to explore the potential parameter estimate biases that may result from data from tutoring systems that employ a mastery-learning mechanism whereby poorer students get assigned tasks that better students do not. We extend this work by exploring a wider set of modeling techniques and by using a data set with additional observations of longer-term retention that provide a check on whether judged mastery is maintained. We investigate variations of logistic regression models including the Additive Factors Model (Cen et al., 2007, 2008) and others explicitly designed to adjust for mastery-based data. We extensively analyze properties and prediction accuracy of the different models and discuss implications for use and interpretation.

- *Dyscalculia screener*: About 3 – 6% of the children in German and English speaking countries suffer from developmental dyscalculia (Lewis et al., 1994; Shalev and von Aster, 2008). Diagnosis is conducted in a one-to-one setting with an expert, using standardized tests. We introduce a computer-based screener for developmental dyscalculia, indicating whether children are at risk for this learning disability. Classification is based on features extracted from log file data of `Calcularis`. By applying a probabilistic classifier, we are able to quantify the uncertainty of the label and to adapt the test duration. Initial cross validation on user study data yields a classification accuracy of about 90% and an average test time of 14 minutes.

1.3 Thesis outline

In this thesis, we first give an overview of related work in the fields of number processing and numerical development as well as student modeling (Chapter 2). We then introduce the three main parts of this work (see overview in Fig. 1.1): *Training Environment*, *Data Collection & Evaluation* and *Analysis & Modeling*. We conclude by reviewing the main contributions of the thesis and suggesting potential future work (Chapter 11). In the following, the structure of the three main parts of the thesis is described.

In the first part, we introduce the dyscalculia therapy software `Calcularis` in detail (Chapter 3). We describe the general concepts, the different games as well as the student model and control algorithm of the system.

In the second part, we give the details of the two user studies conducted in Germany and Switzerland (Chapter 4). We describe the data collected in these studies. Furthermore, we present the design of the user study conducted in Switzerland in 2012 along with first results and case studies.

The third part details the analyses, validation and improvements conducted based on the available user data. In Chapter 5, we validate the skill model of `Calcularis` using learning curves. Furthermore, we analyze, assess and compare a variety of techniques for model validation. Chapter 6 describes the data-driven assessment of the quality of the student model and the control mechanism as well as the analyses of specific problems of the students. Chapters 7-9 deal with prediction: We go from short-term prediction of student answers (Chapter 7) to long-term prediction of training achievement and knowledge gaps (Chapter 8) over to the prediction of students' engagement states (Chapter 9).

Chapter 10 closes the loop of data-driven development. Based on their training data, we classify children into having developmental dyscalculia or not and integrate the developed dyscalculia screener into the training environment.

1.4 Publications

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

- **T. KÄSER**, K. KOEDINGER, and M. GROSS (2014). Different parameters - same prediction: An analysis of learning curves. *Proceedings of EDM (London, UK, 4-7 July, 2014)*, pp. 52-59.
  This paper provides an extensive analysis of modeling techniques for fitting learning curves. It assesses the properties of different models as well as their prediction accuracy on unseen data.
- **T. KÄSER**, S. KLINGLER, A. G. SCHWING, and M. GROSS (2014). Beyond Knowledge Tracing: Modeling Skill Topologies with Bayesian Networks. *Proceedings of ITS (Honolulu, Hawaii, 5-9 June, 2014)*, pp. 188-198. **[Best Paper Award]**
  This paper aims at increasing the representational power of the student model by employing dynamic Bayesian networks that are able to represent skill topologies. The

performance of this approach is evaluated on five large-scale data sets of different learning domains such as mathematics, spelling learning and physics.

- **T. KÄSER**, A. G. SCHWING, T. HAZAN, and M. GROSS (2014). Computational Education using Latent Structured Prediction. *Proceedings of AISTATS (Reykjavik, Iceland, 22-25 April, 2014)*, pp. 540-548.
  This paper employs a constrained latent structured prediction approach for parameter learning and demonstrates the benefits of regularization through constraints.
- **T. KÄSER**, A. G. BUSETTO, B. SOLENTHALER, G.-M. BASCHERA, J. KOHN, K. KUCIAN, M. VON ASTER, and M. GROSS (2013). Modelling and Optimizing Mathematics Learning in Children. *IJAIED: "Best of ITS 2012"*, 23(1-4): 115-135.
  This paper is an extended version of the ITS 2012 paper. It provides a detailed description of the student model and control algorithm of `Calcularis` as well as an extensive assessment of model and control properties.
- **T. KÄSER**, A. G. BUSETTO, B. SOLENTHALER, J. KOHN, M. VON ASTER, and M. GROSS (2013). Cluster-Based Prediction of Mathematical Learning Patterns. *Proceedings of AIED (Memphis, USA, 9-13 July, 2013)*, pp. 389-399.
  This paper uses a two-step approach consisting of clustering and classification to predict learning characteristics of students.
- **T. KÄSER**, G.-M. BASCHERA, J. KOHN, K. KUCIAN, V. RICHTMANN, U. GROND, M. GROSS, and M. VON ASTER (2013). Design and evaluation of the computer-based training program Calcularis for enhancing numerical cognition. *Frontiers in Developmental Psychology*, 4: 489.
  This paper introduces the theory and concepts behind `Calcularis` as well as the different games and describes the design and results of a user study conducted in Switzerland along with two case studies.
- **T. KÄSER**, G.-M. BASCHERA, A. G. BUSETTO, S. KLINGLER, B. SOLENTHALER, J. M. BUHMANN, and M. GROSS (2012). Towards a Framework for Modelling Engagement Dynamics in Multiple Learning Domains. *IJAIED: "Best of AIED 2011 - Part 2"*, 22(2): 42-70.
  This paper is an extended version of the AIED 2011 paper by Baschera et al. (2011). It explores to possibility of a joint framework for engagement modeling in developmental dyslexia and dyscalculia.
- **T. KÄSER**, A. G. BUSETTO, G.-M. BASCHERA, J. KOHN, K. KUCIAN, M. VON ASTER, and M. GROSS (2012). Modelling and Optimizing the Process of Learning Mathematics. *Proceedings of ITS (Chania, Greece, 14-18 June, 2012)*, pp. 389-398.
  This paper describes the student model and control algorithm of `Calcularis` along with first evaluations of model properties and training effectiveness.

During the course of this thesis, the following peer-reviewed papers have been accepted which are not directly related to the presented work.

- J. KOHN, V. RICHTMANN, L. RAUSCHER, K. KUCIAN, **T. KÄSER**, U. GROND, G. ESSER, and M. VON ASTER (2013). Das Mathematikangstinterview (MAI): Erste psychometrische Gütekriterien. *Lernen und Lernstörungen*, 2(3): 177-189.
  This paper describes the design and first results of an interview for assessing math fear, developed in the context of the user studies with `Calcularis`.
- K. KUCIAN, J. KOHN, V. RICHTMANN, U. GROND, **T. KÄSER**, M. M. HANNULA-SORMUNEN, G. ESSER, and M. VON ASTER (2012). Kinder mit Dyskalkulie fokussieren spontan weniger auf Anzahligkeiten. *Lernen und Lernstörungen*, 1(4): 241-253.
  This paper evaluates the SFON effect on data collected from user studies with `Calcularis`.

Additional publications and book chapters during the time period of this thesis:

- **T. KÄSER** and M. VON ASTER (2013). Computerbasierte Lernprogramme für Kinder mit Rechenschwäche. In von Aster, M., & Lorenz, J. (Eds.), *Rechenstörungen bei Kindern. Neurowissenschaft, Psychologie, Pädagogik, 2. Auflage*. Göttingen: Verlag Vandenhoek & Rupprecht, pp. 259-276.
  This book chapter provides an overview of computer-based training programs for children with developmental dyscalculia and introduces `Calcularis` in detail.
- **T. KÄSER**, K. KUCIAN, M. RINGWALD, G.-M. BASCHERA, M. VON ASTER, and M. GROSS (2011). Therapy Software for Enhancing Numerical Cognition. In J. Özyurt, A. Anschütz, S. Bernholt & J. Lenk (Eds.), *Interdisciplinary perspectives on cognition, education and the brain - Hanse-Studies* (Vol. 7, pp. 207-216). Oldenburg: BIS-Verlag.
  This extended abstract gives an introduction into the development of `Calcularis`.

CHAPTER 2

Related Work

This chapter describes the related work in the different areas of research influencing this thesis. First, it covers work in the field of mathematics learning, including models of number processing and numerical development as well as characteristics of developmental dyscalculia. Second, the potential of computer-based training programs for mathematics learning is analyzed and existing conventional and computer-based interventions for developmental dyscalculia are discussed. Third, the area of intelligent tutoring systems is introduced. The different components of an intelligent tutoring system are described with a focus on student modeling. The student modeling part gives an overview of different knowledge representations and modeling techniques. Furthermore, an introduction to affective modeling, *i.e.*, modeling the student's engagement states is given. The final part of this chapter discusses the student modeling techniques relevant for this thesis in detail.

2.1 Development of mathematical understanding

The computer-based training program `Calcularis` developed in this thesis is designed for children with developmental dyscalculia or difficulties in learning mathematics. It is therefore important to understand how number processing and numerical understanding normally develop and what the characteristics of developmental dyscalculia are. In the following, we describe the cognitive models and concepts that are relevant for our work and give a short introduction to developmental dyscalculia.

2.1.1 Neuro-cognitive models of number processing and numerical development

Current neuropsychological models postulate distinct representational modules, located in different brain areas, which are relevant for adult cognitive number processing and calculation. One of the first models, the 'triple-code model' (Dehaene and Cohen, 1995) comprises a verbal module supporting counting and number fact retrieval, a visual-Arabic module required for solving written arithmetic and an analogue magnitude module (mental number line) for semantic number processing. Lately, an fMRI meta-analysis enabled further insights into supporting and domain-general functions involved in solving arithmetic tasks and suggested a modification and extension of the triple-code model (Arsalidou and Taylor, 2011). Results from functional brain imaging in adults and children indicate that the representation of the mental number line emerges during the first years of school in the parietal lobe due to practice and experiences (Rivera et al., 2005; Ansari and Dhital, 2006; Kucian et al., 2008). The initial assumption of the analogue magnitude representation being notation-independent was challenged in 2007 (Cohen Kadosh et al., 2007a). Nieder (2012) recently showed that there are indeed notation-dependent as well as notation-independent neurons responding to numerosity.

While the triple-code model denotes the end state of numerical development, the four-step developmental model (von Aster and Shalev, 2007) describes the path to this end state. It divides the semantic representation (analogue magnitude representation) into an implicit core representation of magnitude and an explicit mental number line, the latter considered as being a 'representational redescription' of the former (Karmiloff-Smith, 1992). The (inherited) core-system representation of cardinal magnitude provides the basic meaning of numbers (step 1). Based on this representation, children learn to associate a perceived number with spoken and later written and Arabic symbols. The process of linguistic (step 2) and Arabic (step 3) symbolization is in turn a precondition for the development of a mental number line (step 4). The different representations develop depending on the growing capacity of domain-general functions like working memory.

Lately, other authors have suggested different models of numerical development (Carey, 2001, 2004; Kucian and Kaufmann, 2009; Kaufmann et al., 2011; Noël and Rousselle, 2011; Kaufmann and von Aster, 2012; Vogel and Ansari, 2012). Some authors argue that developmental dyscalculia is mainly caused by an early, probably genetic, deficit of the basic non-symbolic magnitude system (Butterworth et al., 2011), while others suggest that problems may arise from different developmental reasons, including maladaptive learning experiences and math anxiety. To summarize, there is still an open debate about developmental trajectories and reasons for failure in learning mathematics. However, there seems to be agreement that based on early non-symbolic abilities to access and compare numerical magnitudes, different components of semantic and symbolic representations are developing during childhood and school years. These components develop based on the increasing capacity of domain-general functions and enable a child to successively acquire arithmetic skills.

2.1.2 Developmental dyscalculia

Developmental dyscalculia (DD) is a specific learning disability affecting the acquisition of arithmetic skills (von Aster and Shalev, 2007). Genetic, neurobiological, and epidemiological evidence indicates that DD is a brain-based disorder, although poor teaching and environmental deprivation have also been discussed in its etiology (Shalev, 2004).

DD is thought to have its neuropsychological basis due to a limited ‘number sense’, which implies a deficit in very basic numerical skills such as number comparison (Landerl et al., 2004; Rubinsten and Henik, 2005; Butterworth, 2005a,b). Besides exhibiting fundamental deficits in number processing (Cohen Kadosh et al., 2007b; Mussolin et al., 2010; Kucian et al., 2006; Price et al., 2007), children with DD also tend to suffer from difficulties in acquiring simple arithmetic procedures and exhibit a deficit in fact retrieval (Ostad, 1997, 1999). The prevalence of DD is estimated to about 3-6% (Shalev and von Aster, 2008; Badian, 1983; Lewis et al., 1994) in English and German speaking countries.

Children with DD often show comorbidities with dyslexia (von Aster and Shalev, 2007; Ostad, 1998; Lewis et al., 1994; Badian, 1999; Barbaresi et al., 2005; Dirks et al., 2008; Ackerman and Dykman, 1995) and ADHD (Shaywitz et al., 1994; Fletcher, 2005; Barbaresi et al., 2005). In addition, learning disabilities frequently lead to anxiety and aversion against the subject (Rubinsten and Tannock, 2010) and to underperformance in school and later in profession (Bynner, 1997).

2.2 Existing computer-based interventions

Using computer-based interventions for DD seems promising. When teaching mathematics, the highly complex processes of domain-specific cognitive development need to be taken into account. The development of each child’s numerical abilities often follows a different speed and is intertwined with the development of other cognitive domains and domain-general abilities (von Aster and Shalev, 2007; Kucian and Kaufmann, 2009; Kaufmann et al., 2011), leading to different mathematical performance profiles (von Aster, 2000; Geary, 2004; Wilson and Dehaene, 2007). Computer-based trainings can be designed to adapt to an individual child’s abilities and provide intensive training in a stimulating environment (Kullik, 2004). The training can for example adapt to cognitive (Naglieri and Johnson, 2000) or to performance profiles of the children (von Aster, 2000; Geary, 2004; Wilson and Dehaene, 2007). This individualization in combination with the fact that the computer is an emotionally neutral medium may also lead to increased motivation and enhance positive self-concepts as every learner gains feelings of success (Ashcraft and Faust, 1994; Spitzer, 2009).

In the past years, different meta-analyses have assessed the effects of computer-based instruction for mathematics learning, revealing positive results. Kulik and colleagues (Kulik and Kulik, 1991; Kulik, 1994) computed an average effect size of 0.47 in elementary school. Other studies reported effect sizes ranging from 0.13 to 0.8 (Khalili and Shashaani, 1994; Fletcher-Flinn and Gravatt, 1995). Li and Ma (2010) found larger effects for elementary school than for higher education and showed that special needs students especially benefit from computer-based instruction.

Interventions specifically targeting children with DD are mostly conventional. Techniques include training programs for preschool children at risk of developing mathematical difficulties (Griffin et al., 1994; Van De Rijt and Van Luit, 1998; Arnold et al., 2002; Wright, 2003) as well as remedial programs for elementary school children (Van Luit and Naglieri, 1999; Dowker, 2001, 2003; Fuchs et al., 2006; Wilson et al., 2006a; Butterworth et al., 2011; Lenhard et al., 2011; Kucian et al., 2011). Programs designed for preschool children mostly focus on building basic-numerical skills, whereas elementary school trainings target a broader range of skills. Some interventions address basic numerical skills and the establishment of the mental number line (Wilson et al., 2006a), while others train arithmetic fact knowledge (Van Luit and Naglieri, 1999; Fuchs et al., 2006) or are aligned to scholar curricula (Lenhard et al., 2011). Other effective approaches combine the training of basic-numerical capacities with the training of arithmetical knowledge (Dowker, 2001, 2003; Kucian et al., 2011).

There exist a few computer-based interventions in number processing. The computer-based intervention `Number Race` for children with DD trains number comparisons and enhances the links between number and space (Wilson et al., 2006a). Evaluation of the training revealed significant improvements in basic numerical cognition, but the effects did not generalize to counting or arithmetic (Wilson et al., 2006b; Räsänen et al., 2009; Wilson et al., 2009). `Rescue Calcularis` is another computer-based intervention for children with DD. It aims to improve the construction and access to the mental number line. The evaluation of the program showed that children with and without DD could benefit from the training (Kucian et al., 2011). `Elfe and Mathis` is a computer-based training aligned to the German scholar curriculum (Lenhard et al., 2011). Its evaluation demonstrated significant effects. Fuchs et al. (2006) presented a computer-based program to acquire fact knowledge, reporting significant effects in addition. Butterworth et al. (2011) suggest the use of adaptive interactive games for remediation. The proposed games train basic-numerical skills (number comparisons and counting) as well as the spatial number representation and simple arithmetic facts.

The introduced previous studies demonstrate the efficacy of computer-based intervention in number processing. The presented programs, however, mostly focus on specific skills and provide only limited adaptability.

On the other hand, existing adaptive computer-based training programs (called intelligent tutoring systems) for learning mathematics are mostly designed for normal performing children and focus on specific aspects of the domain. The `Cognitive Tutor` (Koedinger et al., 1997) is an intelligent tutoring system for teaching algebra to high-school students. Other work includes a program for fraction learning (Rau et al., 2009) or a web-based math test for high-school students (Arroyo et al., 2004).

2.3 Intelligent Tutoring Systems (ITS)

Computer-assisted learning is gaining importance in children's education. Intelligent tutoring systems (ITS) are successfully employed in different fields of education, such as physics (Conati et al., 2002), algebra (Koedinger et al., 1997) and reading (Mostow et al., 1993). Computer-based therapy systems for learning disabilities have gained particular attention. Such systems present inexpensive extensions to conventional one-to-one therapy by providing an adaptive and fear-free learning environment. The effectiveness of computer-based therapy programs has been proven by several user studies targeting children with dyslexia (Gross and Vögeli, 2007; Kast et al., 2007), DD (Wilson et al., 2006a; Lenhard et al., 2011; Kucian et al., 2011), and ADHD (Klingberg et al., 2005).

In this section, we first describe the different components of a generic ITS. We then detail the most important component for this work, the *Student Model*, by introducing prior work on student modeling and giving an overview of the most popular techniques. Finally, we introduce a newer field of ITS, which deals with modeling not only the knowledge state of the student, but also his (or her) affective states.

2.3.1 Principal components of an ITS

An overview of ITS was presented by Shute and Psotka (1994), introducing the main components of a generic ITS: Knowledge of the domain (*Domain Expert*), knowledge of the learner (*Student Model*) and knowledge of teaching strategies (*Tutor*). These components and their relations are illustrated in Fig. 2.1 (Shute and Psotka, 1994).

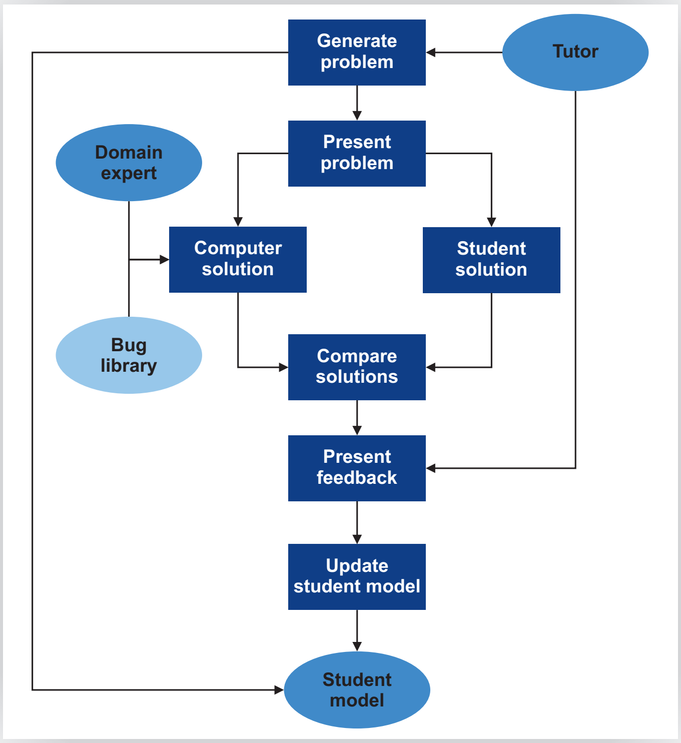


Figure 2.1: Program flow and principal components (denoted by ellipses) of a generic ITS (Shute and Psotka, 1994): The system incorporates knowledge about the domain (*Domain Expert*), knowledge of the learner (*Student Model*) and knowledge about teaching strategies (*Tutor*). The (optional) *Bug Library* contains a list of typical misconceptions for the domain. The rectangles describe program decisions or actions.

In an ITS, a student learns mainly from solving problems that are adjusted to the student's knowledge state. To select appropriate tasks, the system needs to assess the current knowledge state of a student. Therefore, the initial knowledge of the student needs to be modeled by the system and updated based on the interactions of the student (for example the solved problems) with the training program. These tasks are handled by the *Student Model*. Furthermore, the program also considers, what the student needs to know. This knowledge about the domain is given by the *Domain Expert*. Finally, the system needs to decide how the selected problem is presented. Information about the teaching strategy is contained in the *Tutor*.

Based on all these components, the training program selects a problem (task) and presents it to the student. The solution of the student and the expert solution are then compared and the system gives a feedback to the student. The manner of feedback is defined in the *Tutor*. Some systems also compare the student solution to the content of a *Bug Library*. The *Bug Library* contains a list of typical errors (or misconceptions) of the learning domain. Finally, the *Student Model* is updated based on the solved problem, and the next problem is generated.

2.3.2 Student modeling

The *Student Model* is a central component of an ITS. To improve the training outcome, knowledge of performance profile, knowledge gaps and learning behaviors of the student as well as an accurate performance prediction are essential. This is particularly important for students suffering from learning disabilities as the heterogeneity of these children requires a high grade of individualization.

A student model can be characterized by its form of representing the knowledge of the learning domain, *e.g.*, mathematics. The most popular representations are overlay models, perturbation models, and cognitive models (Kass, 1989). These three categories are illustrated in Fig. 2.2 (Baschera, 2011).

One of the first knowledge representations employed in student modeling was the overlay model (Barr et al., 1976). The overlay concept assumes, that student knowledge is a subset of the expert knowledge. The goal of the training is to extend the student knowledge until it conforms to the expert knowledge. The overlay approach assumes that all differences in knowledge between the student and the expert stem from a lack of student knowledge. This technique is therefore not able to model student misconceptions.

A technique that tackles the disadvantages of the overlay approach is the perturbation model. In contrast to the overlay concept that models only correct knowledge, the perturbation model takes faulty knowledge into account. An early and popular example for a perturbation model is `DEBUGGY` (Burton, 1982), which models students' misconceptions or bugs in their basic mathematical skills and thus provides a mechanism for explaining why a student is making a mistake. More recently, Baschera and Gross (2010b) presented a Poisson-based perturbation model for representing word-spelling errors.

The third main category for knowledge representation is the cognitive model. This model represents the student knowledge as a subset of the cognitive model of the learning domain. It does not directly model domain knowledge, but independent production rules or skills which allow to solve the exercises of the domain. One of the most popular approaches for building a cognitive model is Bayesian Knowledge Tracing (Corbett and Anderson, 1994).

Current tutoring systems use a variety of approaches to model student learning. Most of the techniques make use of a cognitive knowledge representation, sometimes combined with a perturbation model. Markov Decision Processes are used for teaching

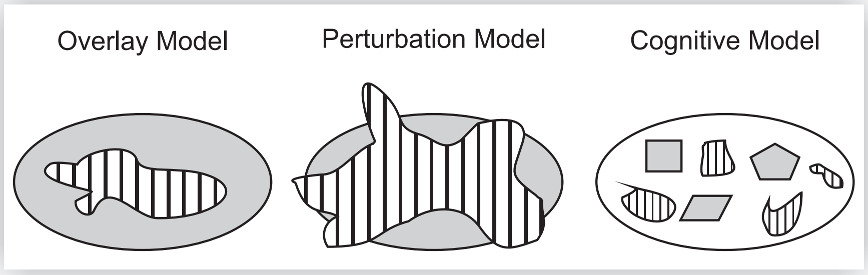


Figure 2.2: Different student representations (illustration by Baschera (2011)): Expert knowledge is shaded gray, while the student knowledge is illustrated by the ruled areas. The overlay and perturbation models are expert models, *i.e.*, student knowledge is modeled as a subset of the expert knowledge. In the cognitive model, student and expert knowledge are represented as subsets of skills of the learning domain.

planning (Brunskill and Russell, 2011; Rafferty et al., 2011) or diagnosing misconceptions (Rafferty et al., 2012). A popular approach to represent student learning is Performance Factors Analysis (Pavlik et al., 2009). Logistic regression was proposed for modeling student learning (Rafferty and Yudelson, 2007; Yudelson and Brunskill, 2012; Rafferty et al., 2013). Furthermore, student knowledge and learning can be represented by Hidden Markov Models (Piech et al., 2012), Bayesian networks (Brunskill and Russell, 2011; González-Brenes and Mostow, 2012b,a) or Bayesian Knowledge Tracing (Corbett and Anderson, 1994). Bayesian Networks are also employed to model and predict students' learning styles (Kim et al., 2012), engagement states (Baschera et al., 2011) and goals (Conati et al., 2002).

2.3.3 Affective modeling

Due to its relevance in learning, affective modeling is gaining increasing importance. Motivation and positive self-concepts for example influence the learning outcome (Ashcraft and Faust, 1994; Spitzer, 2009). The goal of an affective model is to represent, identify and predict affective states of the student such as emotions, motivation or attention.

Previous work in affective modeling can be divided into two groups. The first group of models utilizes sensor data, while the second group relies on student input data only. Sensor measurements have the potential to directly measure a large number of affective features. However, the measurements are usually limited to laboratory experimentation due to the expensive equipment needed. The measurement of student interaction data on the other hand provides the opportunity to obtain large data sets from different experimental conditions. However, measurements of affect based on student interaction are indirect, *i.e.*, the true affective state needs to be inferred from the input data.

Models from the first group use a variety of sensors. Eye tracking data was for example used to analyze students' attention to hints (Muir and Conati, 2012). A combination of eye tracking and interaction data was utilized to classify whether a student's behavior is conducive for learning (Kardan and Conati, 2013). Conati (2002) presented a model based on heart-rate data to assess student emotional reaction during interaction with an educational game. Other authors (Heraz and Frasson, 2009) utilized a combination of brainwave data (measuring the learner's mental state) and user input (indicating the learner's affective state) to predict the correctness of user answers. Furthermore, camera data is another option in affective modeling (Cooper et al., 2010).

The second group contains models relying on student input data only. Beck (2005) suggested a model based on Item Response Theory (Wilson and De Boeck, 2004), utilizing students' response times to predict their engagement. By employing a Hidden Markov Model, Johns and Woolf (2006) predicted student motivation. Furthermore, Arroyo and Woolf (2005) presented a Bayesian network based on features extracted from log file data to infer students' attitudes, perceptions and learning. A Bayesian network model was also used to identify the attentional state of the student (Baschera et al., 2011).

2.4 Modeling techniques

In this thesis, we will draw from the fields of ITS, machine learning and educational data mining to represent and predict student knowledge, learning, characteristics and affective states. In section 2.3.2, we have given an overview of popular techniques for modeling student knowledge. In this section, we provide a detailed discussion of the modeling techniques relevant for this thesis.

Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994) is an example for a cognitive model, representing the knowledge of the learning domain as a set of skills or production rules. A BKT model is a special case of a Hidden Markov Model (HMM) (Reye, 2004). The hidden variable of the model denotes the student knowledge, *i.e.*, one skill. This variable is assumed to be binary as the skill can either be mastered by the student or not. The observations represent tasks associated with the respective skill. Observations are also binary: A student solves a task correctly or not. From this description follows, that one BKT model per skill is needed to

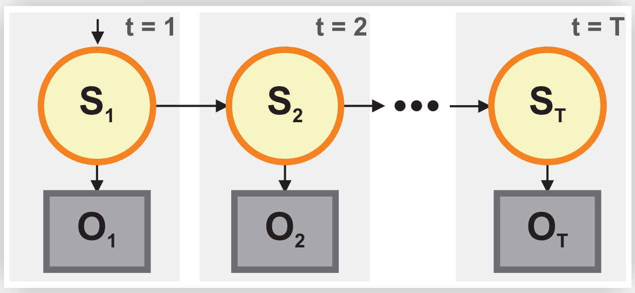


Figure 2.3: Structure of a BKT model for skill S over T time steps. The hidden variables S_t (denoted by circles) represent the state of skill S over time, while the observed variables O_t (denoted by rectangles) describe observations of skill S at time t.

represent the knowledge of the domain. Figure 2.3 illustrates the BKT model for an example skill S.

The transition probabilities $p(s_t|s_{t-1})$ of the network can be described by two parameters: p_L, the probability of a skill changing from the unknown to the known state and p_F, the probability of forgetting a previously known skill. Also the emission probabilities $p(o_t|s_t)$ are specified using two parameters. The guess probability p_G of getting a task correct despite not knowing the respective skill and the slip probability p_S of making a mistake when applying a known skill. The initial probability $p(s_1)$ of knowing a skill a-priori is described by the parameter p_0. In traditional BKT (Corbett and Anderson, 1994), the forget probability p_F is assumed to be zero. Therefore, a BKT model can be completely specified by the parameter set $\theta = \{p_0, p_L, p_G, p_s\}$.

An important task when using a BKT model is *inference*: Given the BKT parameters θ and a sequence of observations $\mathbf{o_m} = (o_{m,1}, ..., o_{m,t})$ with $o_{m,t} \in \{0,1\}$ and time $t \in \{0, ..., T\}$ for a student m, what is the probability $p(S_t = 1|o_{m,1}, ..., o_{m,t})$ that the skill S is in the known state at time t? The *inference* task is usually evaluated after each solved problem of the student. Based on the posterior probability $p(S_t = 1|o_{m,1}, ..., o_{m,t})$ the system decides on the next task to be solved.

The second important task, the *learning* task amounts to estimating the parameters θ of the BKT model given some observations: Given a sequence of observations $\mathbf{o_m} = (o_{m,1}, ..., o_{m,t})$ with time $t \in \{0, ..., T\}$ for the m-th student with $m \in \{1, ...M\}$, what are the parameters θ that maximize the likelihood $\prod_m p(\mathbf{o_m}|\theta)$ of the available data. Fitting the parameter θ of a BKT model increases prediction accuracy and therefore allows for a better adaptation to the student's knowledge.

Exhibiting a tree structure, BKT allows for efficient parameter learning and accurate inference. Popular techniques for learning in BKT include expectation maximization (Pardos and Heffernan, 2010b; Wang and Heffernan, 2012), brute-force grid search (Baker et al., 2010) or gradient descent (Yudelson et al., 2013).

Dynamic Bayesian Networks

Bayesian networks (Pearl, 1988) are directed acyclic graphs, where the nodes represent random variables and the arcs specify the relationships between these random variables. These relationships can be specified using conditional probability tables (CPT). A Bayesian network therefore describes a probability distribution. The random variables can be discrete or continuous. In the following, we will only discuss the case, where a random variable can take two states: `true` or `false`. If there is a directed connection between a variable V_i and a variable V_j, V_i is called a parent of V_j. The belief of a node V_i of the network (probability that the random variable takes the state `true`) is conditioned over its parents $pa(V_i)$ and therefore the joint probability of a Bayesian network with N variables can be specified as follows:

$$p(v_1,...,v_N) = \prod_i p_{v_i} \text{ where } p_{v_i} := p(v_i|pa(V_i)). \tag{2.1}$$

As in BKT, variables can be hidden or observed. Therefore, an important task when using Bayesian networks is *inference*: Given some observations, what are the beliefs of the hidden variables. As opposed to BKT, Bayesian networks do not necessarily exhibit a tree structure and therefore the *inference* task cannot be solved accurately. However, there exists a variety of algorithms for approximate inference in Bayesian networks such as loopy belief propagation (Kschischang et al., 2006), fractional belief propagation (Wiegerinck and Heskes, 2003) or tree-reweighted belief propagation (Minka and Qi, 2003). As in BKT, the second important task is *learning*.

Dynamic Bayesian networks (DBN) (Murphy, 2002) are needed to model sequential data. A DBN can be seen as the extension of a Bayesian network over time: The structure of the Bayesian network in time step $t = 1$ is copied to all time steps t with $t = \{2,...,T\}$. Furthermore, directed connections between the nodes of different time slices are added. A HMM is the most simple example of a DBN. Algorithms for *inference* and *learning* for Bayesian networks can be directly applied to DBNs. We again interpret the variables of the DBN in terms of a learning context: The latent variables represent skills S_i and the observable variables O_i denote the associated task outcomes. An example DBN over T time steps is given in Fig. 2.4, where the circles denote the hidden variables and the rectangles represent observable nodes.

Employing DBNs (instead of BKT) in ITS has the potential to increase the representational power of the student model and hence further improve prediction accuracy.

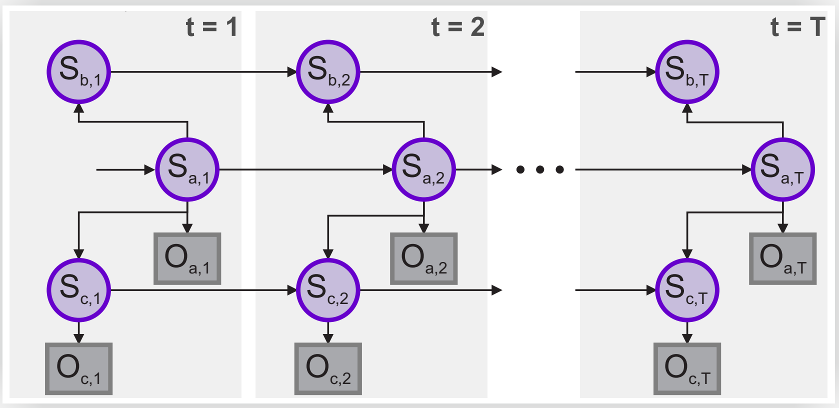


Figure 2.4: Graphical model of an example DBN over T time steps. The hidden variables (denoted by circles) model the states of the three skills S_a, S_b and S_c over time. The rectangles represent observations associated with the skills S_a (variable O_a) and S_c (variable O_c). Skill S_b cannot be observed.

In contrast to BKT, DBNs are able to represent the hierarchy and relationships between the different skills of a learning domain. In ITS, DBNs have been used to model and predict students' performance (Conati et al., 2002; Mayo and Mitrovic, 2001), engagement states (Baschera et al., 2011), and goals (Conati et al., 2002). DBNs are also employed in user modeling (Horvitz et al., 1998). In cognitive sciences, DBNs are applied to model human learning (Frank and Tenenbaum, 2010) and understanding (Baker et al., 2005). Despite their beneficial properties to represent knowledge, DBNs have received less attention in student modeling as they impose challenges for learning and inference.

In this thesis, we propose a DBN student model for representing different mathematical skills (see Sec. 3.3). Furthermore, we demonstrate how to solve the *learning* task for DBNs efficiently (see Chapter 7). Our method guarantees plausible parameter estimates and shows a higher prediction accuracy on unseen data than BKT.

Logistic regression models

Logistic regression models are used in Item Response Theory (IRT) (Wilson and De Boeck, 2004) to model the response (correct/wrong) of a student to an item. IRT is based on the idea that the probability of a correct response to an item is a mathematical function of student and item parameters.

Although mainly used in computerized adaptive testing to predict the probability of a correct answer (Baker, 2001), IRT models (or models inspired by IRT) have been applied for various purposes. Jarušek and Pelánek (2012) describe a model which assumes an exponential relationship between problem solving ability and time - based on IRT and collaborative filtering - to predict problem solving times. Response times of a person on a set of test items were also predicted using a lognormal model (van der Linden, 2006). Answer times have also been used to predict students' engagement states (Beck, 2005). Furthermore, Johns and Woolf (2006) proposed the combination of an IRT model to predict student proficiency and a HMM to infer students' motivation.

One of the most popular regression models for student modeling is the Additive Factors Model (AFM) (Cen et al., 2007, 2008). It is widely used to fit learning curves and to analyze and improve student learning. AFMs help identify flat or ill-fitting learning curves that indicate opportunities for tutor or model improvement. Consistently low error curves indicate opportunities to reallocate valuable student time (Cen et al., 2007). Consistently high error curves with poor fit indicate a miss-specified skill model that can be improved (Koedinger et al., 2013; Stamper and Koedinger, 2011), and used to design better instruction (Koedinger and McLaughlin, 2010).

The AFM is a generalized linear mixed model (GLMM) (Boeck, 2008) applying a logistic regression. In a logistic regression model, the observations of the students follow a Bernoulli distribution. A Bernoulli distribution is a binomial distribution with $n = 1$. Letting $y_{pi} \in \{0, 1\}$ denote the response of student p on item i, we obtain $y_{pi} \sim \text{Binomial}(1, \pi_{pi})$. The linear component π_{pi} of the AFM can then be formulated as follows:

$$\text{logit}(\pi_{pi}) = \theta_p + \sum_k q_{ik} \cdot (\beta_k + \gamma_k \cdot T_{pk}), \tag{2.2}$$

with $\theta_p \sim \mathcal{N}(0, \sigma_\theta^2)$. The AFM is a GLMM with a random effect θ_p for student proficiency and fixed effects β_k (difficulty) and γ_k (learning rate) for the skill S_k (knowledge component). The learning rate γ_k is constrained to be greater than or equal to zero for AFMs. q_{ik} is 1, if item i uses skill S_k and 0 otherwise. Finally, T_{pk} denotes the number of practice opportunities student p had at skill S_k. The AFM is related to the linear logistic test model (LLTM) (Wilson and De Boeck, 2004) and the Rasch model (Wilson and De Boeck, 2004). When removing the third term $(\gamma_k \cdot T_{pk})$ of Eq. (2.2), we obtain an LLTM. Additionally assuming a unique-step skill model (one skill per step) results in the Rasch model. The intuition of the AFM is that the probability of a student getting a step correct is proportional to the amount of required knowledge of the student θ_p, plus the difficulty of the involved skills β_k and the amount of learning gained from each practice opportunity γ_k.

As the AFM is a GLMM, it can be fit using maximum likelihood, which involves integration over the random effects (Breslow and Clayton, 1993). Integration is performed using methods such as numeric quadrature or Markov Chain Monte Carlo (MCMC).

CHAPTER 3

Calcularis

The developed software `Calcularis` is an intelligent tutoring system (ITS) for children with developmental dyscalculia (DD) or difficulties in learning mathematics. It constitutes the core of this thesis as well as its first step. All developed models and analyses are based on log files collected in user studies conducted with `Calcularis` (described in detail in Chapter 4) or from the first product version sold since the beginning of 2013. This chapter describes the development of `Calcularis` in detail.

3.1 Design Principles

The system of `Calcularis` is based on the special needs of children with DD and aims at supporting the development of mathematical understanding in general. The program transforms current neuro-cognitive findings into the design of different instructional games. It combines the training of basic numerical cognition with the training of different number representations and their interrelations, and with the training of arithmetic abilities. The intervention relies on three design principles:

1. *Design of numerical stimuli*: A special number design enhancing the three different number representations (as specified by the triple-code model (Dehaene and Cohen, 1995) introduced in Sec. 2.1.1) is consistently used throughout the training program. Furthermore, the three different number modalities are shown simultaneously at the end of each trial. The encoding of different properties of a number through different information channels supports the acquisition of the different number representations and facilitates number understanding.

2. *Adaptability and scaffolding*: The development of each child's numerical abilities often follows a different speed and is intertwined with the development of other cognitive domains and domain-general abilities (von Aster and Shalev, 2007; Kucian and Kaufmann, 2009; Kaufmann et al., 2011), leading to different mathematical performance profiles (von Aster, 2000; Geary, 2004; Wilson and Dehaene, 2007). Our training system contains a *Student Model* (described in Sec. 3.3) that optimizes the learning process by providing a hierarchically structured learning environment teaching fundamental knowledge first (scaffolding). Furthermore, task selection and difficulty are adapted to the knowledge level of the child. The other component important for adaptivity is the *Bug Library* (detailed in Sec. 3.4): It enables the system to recognize and address specific problems of a child.

3. *Different types of knowledge*: The intervention program aims to balance the acquisition of conceptual knowledge with automation training. Children are taught conceptual knowledge before going over to automation training. An arithmetic operation is for example first introduced and explained. The arithmetic operation and its solution are then modeled using stimuli and finally, mental calculation is trained.

In the following, we will explain the different components of `Calcularis` in detail, following the general structure of a learning program described in Sec. 2.3. We first introduce the *Tutor* of the system by describing the curriculum (the structure) and the games of the program as well as the special design for numerical stimuli. In a second step, we present the *Student Model* of the program consisting of a dynamic Bayesian network (DBN) (Murphy, 2002) representing mathematical skills and a control algorithm. Finally, the *Bug Library* is specified.

3.2 Tutor

The *Tutor* defines the curriculum of the program as well as how domain knowledge is represented, utilized and communicated. In `Calcularis`, a special design for numerical stimuli is used to represent the properties of a number. The *Tutor* also specifies how the material should be instructed. `Calcularis` consists of multiple games in a hierarchical structure that follows the natural development of mathematical understanding.

3.2.1 Design for numerical stimuli

The special design for numerical stimuli is intended to enhance the different number modalities and to strengthen the links between them. Properties of numbers are encoded with visual cues such as color, form and topology. The digits of a number

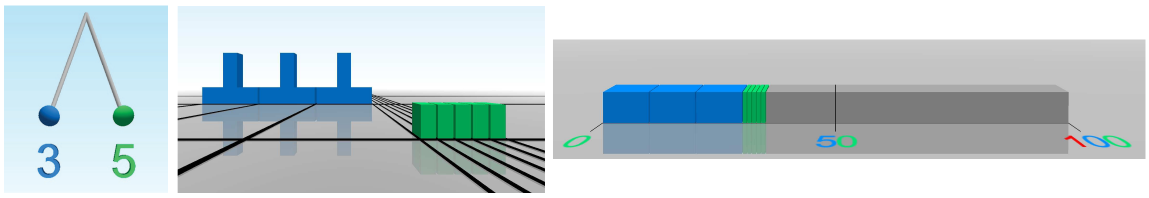


Figure 3.1: Numerical stimuli for the number 35. Units are colored in green, tens in blue and hundreds in red. The digits of a number are attached to the branches of a number graph (left) to facilitate the acquisition of the Arabic notation. Numbers are constructed using colored blocks (center) to emphasize cardinality. The number line representation with integrated blocks (right) enhances the ordinality of numbers.

are attached to the branches of a graph and are represented with different colors according to their positions in the place-value system: Units are colored in green, tens in blue and hundreds in red (see Fig. 3.1 (left)). We assume that this representation facilitates the acquisition of the Arabic notation as well as the translation between verbal and Arabic notation. The cardinal magnitude of number is emphasized by representing the number as an assembly of one, ten and hundred blocks. This representation illustrates the fact that numbers are composed of other numbers. The blocks are linearly arranged from left to right (see Fig. 3.1 (center)) or are directly integrated in the number line (see Fig. 3.1 (right)) to make the connection to the analogue magnitude module (Dehaene and Cohen, 1995).

3.2.2 Curriculum

The training program is composed of multiple games in a hierarchical structure. Figure 3.2 shows the different areas of the training program. The version of the training program employed in the user studies as well as in the actual product version is constrained to specific areas of this structure: *intuitive number understanding*, *number representations* and *arithmetic operations* with natural numbers up to 1000. Figure 3.2 therefore illustrates the final vision of the training program with solid and dashed boxes representing the already integrated and the planned components, respectively.

Games are structured along number ranges and are further divided into hierarchically ordered areas:

1. *Number representations*: This area focuses on different number modalities and number understanding in general. It trains transcoding between the different number representations. Furthermore, the three interpretations of number are established: Cardinality (quantity), ordinality (position in a sequence) and relativity (difference between two numbers). Games in this area are hierarchically ordered according to the four-step developmental model (von Aster and Shalev, 2007).

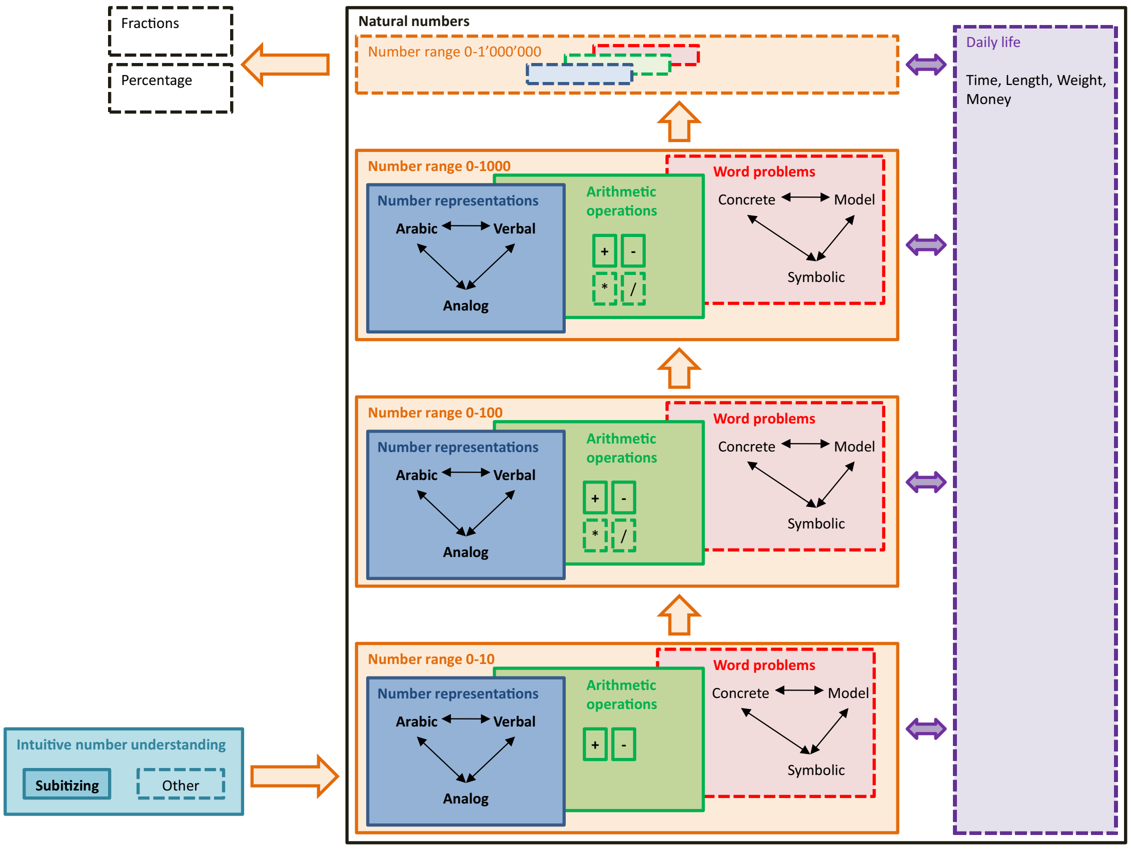


Figure 3.2: The training program is structured into the three main areas of *number representations* (blue), *arithmetic operations* (green) and *word problems* (red). These areas can be further divided into different number ranges. The areas of *intuitive number understanding* (purple) and *daily life* (turquoise) complete the curriculum. Parts that have not yet been implemented are marked with dashed lines.

2. *Arithmetic operations*: This area trains arithmetic operations at different difficulty levels. Task difficulty is determined by task complexity, the magnitude of numbers involved and the means (visual aids) available to solve the task. At the moment, the program contains only addition and subtraction tasks.

3. *Word problems*: A complete understanding of mathematical operations requires the ability to associate a described situation with a mathematical operation and vice versa. This also presumes an understanding of the actual meaning of the operation. The importance of word problems was confirmed by the LOGIK user study (Weinert and Schneider, 1999): Mathematical performance in the 11^{th} grade was highly correlated to performance in word problems in the 2^{nd} grade.

Each area builds up on knowledge gained in previous areas and therefore deepens the previously acquired knowledge.

An additional forth area (*intuitive number understanding*) serves as a precondition for the three areas described above. This area focuses on important precursor abilities (Landerl et al., 2004; Hannula and Lehtinen, 2005; Mazzocco and Thompson, 2005; Krajewski and Schneider, 2009) such as subitizing or counting. The fifth area of the structure will teach mathematical concepts (such as time or money) important for *daily life*.

All games can also be categorized based on their complexity and relative importance. Main games are complex games requiring a combination of abilities to solve them. Support games train specific skills and serve as a prerequisite for the main games. Each area features exactly one main game and several support games. The main games are the same for each number range; they just differ by the cardinal magnitude of numbers used. The training path through the structure traverses each number range from left to right starting with the number range from 0-10.

3.2.3 Games

The training program consists of ten different types of games that are associated with the different areas of the training program. By varying the numbers used in the games, we obtain 81 different types of tasks (task difficulty levels).

Subitizing. Subitizing refers to the rapid and accurate judgment of number performed for small numbers of items (up to four). In the `Subitizing` game (see Fig. 3.3(a)), children are presented a number verbally as well as in Arabic notation. A box of items (or a number of fingers) shows up for a limited amount of time (200 ms) on the left side of the screen. Children have to click when the number of items (or fingers) corresponds to the presented number. The game belongs to the area of *intuitive number understanding* and is classified as a support game.

Estimation. The `Estimation` game (see Fig. 3.3(b)) is a support game in the area of *number representations*. In this game, a number in the range from 0-100 (or 0-1000) as well as three squares containing point sets are displayed. Children need to decide which point set corresponds to the given number. The amount of time to solve the task is limited to ten seconds in order to prevent counting.

Transfer. The `Transfer` game exists in different modes and aims to train transcoding between different number representations. Children have to translate a spoken number to the Arabic notation (as displayed in Fig. 3.3(c)) or they have to model a number presented in Arabic notation using colored blocks. The `Transfer` game again belongs to the area of *number representations* and is classified as a support game.

Distance. The `Distance` game (see Fig. 3.3(d)) enhances the relative aspect of number (number as a difference between two numbers). Children have to find the numbers that are larger (or smaller) by a given number *x* than the displayed number. In the number range from 0-10, *x* can be 1, 2 or 3. In higher number ranges, *x* is a multiple of ten or hundred. This game is again a support game in the area of *number representations*.

Ordering. The `Ordering` game (see Fig. 3.3(e)) is a support game in the area of *number representations*, training ordinal number understanding. A sequence of numbers is displayed for a period of five seconds. Children need to decide if the sequence was sorted in ascending order.

Secret Number. The `Secret Number` game (see Fig. 3.3(f)), a support game in the area of *number representations*, trains the ability to assign a number to an interval. Children have to guess a number in as few steps as possible. After each guess, they are told if the secret number is smaller or larger than the guessed number.

Landing. The `Landing` game (see Fig. 3.3(g)) is the main game of the area of *number representations* aimed at training spatial number representation. A purple cone must be directed to the position of a given number on a number line (with indicated center) using a joystick. Numbers are presented in verbal or Arabic notation. In another game setting the cardinality of a given point set and the position of this quantity on the number line have to be estimated. The required accuracy for a correct solution is a deviance of less than 5%.

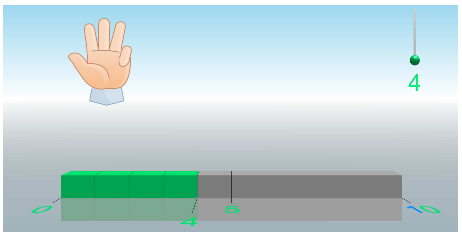
Slide Rule. The `Slide Rule` game (see Fig. 3.3(h)) is a support game belonging to the area of *arithmetic operations*, providing an introduction to addition and subtraction using the part-whole scheme (Resnick, 1984). An operation task is presented to the child, as well as a number line and a glass case containing a number of unit blocks (according to the first number of the task). The size of the glass case must be changed such that it contains the result of the task.

Plus-Minus. In the `Plus-Minus` game (see Fig. 3.3(i)), an arithmetic operation given in Arabic notation must be modeled using colored blocks (one, ten, hundred). Different strategies are allowed to find the result. This game is associated with the area of *arithmetic operations* and is classified as a support game.

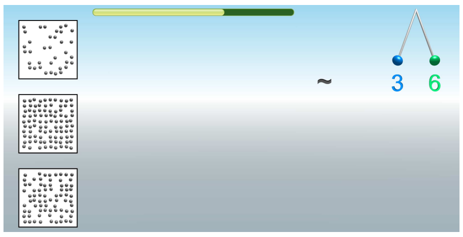
Calculator. In the `Calculator` game (see Fig. 3.3(j)), mental addition and subtraction are trained. The child needs to type the result of an addition (or subtraction) task presented in Arabic notation. The `Calculator` game is the main game of the area of *arithmetic operations*.

3.3 Student Model

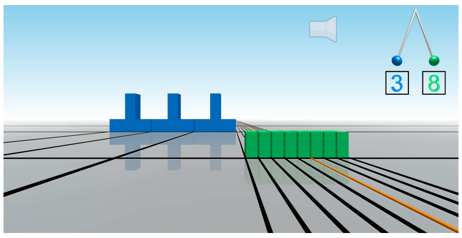
A fundamental component of an ITS is its student model: the subsystem making the teaching decisions. It selects the skills for training and determines the actions for the



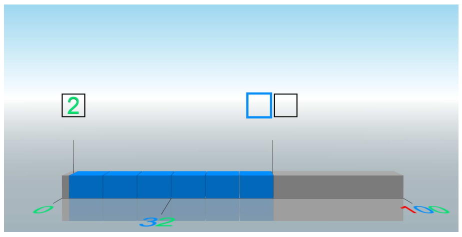
(a) `Subitizing` game.



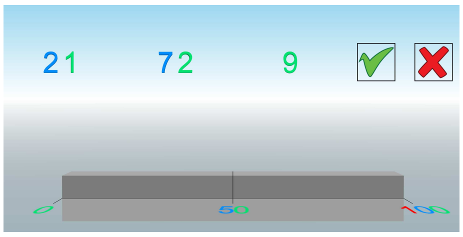
(b) `Estimation` game.



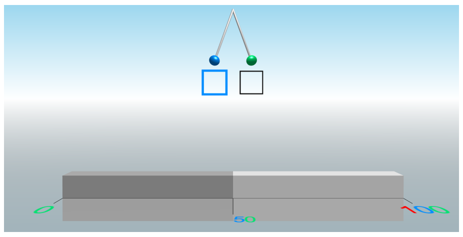
(c) `Transfer` game.



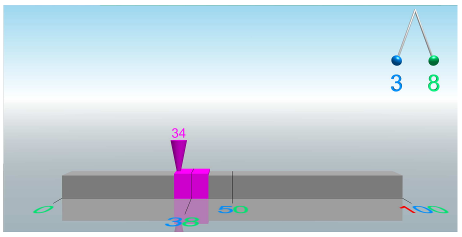
(d) `Distance` game.



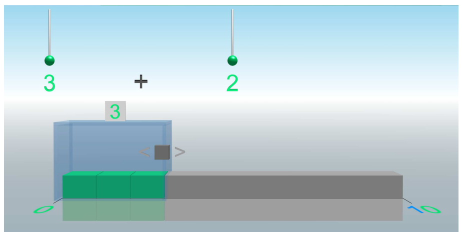
(e) `Ordering` game.



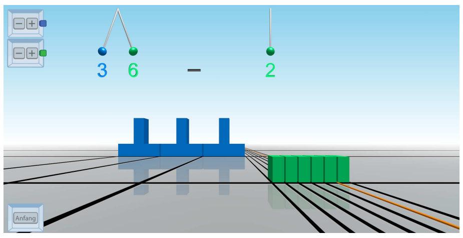
(f) `Secret Number` game.



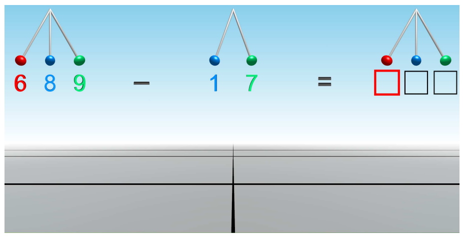
(g) `Landing` game.



(h) `Slide Rule` game.



(i) `Plus-Minus` game.



(j) `Calculator` game.

Figure 3.3: The first version of `Calcularis` consists of ten different types of games for the areas of *intuitive number understanding* (a), *number representations* (b-g) and *arithmetic operations* (h-j).

selected skill. `Calcularis` employs a dynamic Bayesian network (DBN) to represent the knowledge of the student and applies a specially designed control algorithm for task selection. In this section, we will explain the knowledge representation as well as the control algorithm in detail.

3.3.1 Dynamic Bayesian network

The mathematical knowledge of the learner is modeled using a DBN (see Sec. 2.4 for an introduction). The network consists of a directed acyclic graphical model representing different mathematical skills and their dependencies. Two skills S_a and S_b have a (directed) connection if mastering skill S_a is a prerequisite for mastering skill S_b. As the skills cannot be directly observed, the system infers them by posing tasks and evaluating user actions. Such observations $\mathbf{o}$ indicate the presence of a skill probabilistically. The posteriors $p(\mathbf{s_t}|\mathbf{o_t})$ of the net are updated after each time step t, *i.e.*, each solved task, using the sum-product algorithm (libDAI (Mooij, 2010)). $\mathbf{s_t}$ denotes the states of all skills S_i of the network at time t. The DBN has a memory of 5, *i.e.*, posteriors are calculated over the last five time steps and therefore $\mathbf{o_t} = (o_{t-4}, o_{t-3}, o_{t-2}, o_{t-1}, o_t)$ includes only the observations of the last five time steps. We initialize all probabilities to 0.5 as we do not have any knowledge about the mathematical proficiency of a learner at the beginning of the training (the students are of different age and have different mathematical skill levels). This initialization is in accordance with the principle of maximum entropy.

The skill net representation is ideal for modeling mathematical knowledge as the learning domain exhibits a distinctively hierarchical structure. The structure of the net was designed using experts' advice and incorporates domain knowledge. The design of the net was inspired by the work from Falmagne et al. (1990). Like in knowledge space theory, we order skills hierarchically and assume that some skills can be surmised by others. The basic assumption is that to know a skill S_a, the child needs to know all the precursor skills of S_a. However, in our case, each skill is assigned to exactly one task. Our work can also be related to partial order knowledge structures (Desmarais et al., 1995) which also model dependencies between skills as conditional probabilities.

Our resulting student model contains 100 different skills as illustrated in Fig. 3.4. Table 3.1 explains the different skills of the skill net and their notation used in Fig. 3.4. The presented skill net is the student model developed for the actual version of the training program and therefore covers only numbers up to 1000 and the areas of *intuitive number understanding*, *number representations* and *arithmetic operations* of the target structure (illustrated in Fig. 3.2). All games of the training program (introduced in Sec. 3.2.2) are associated with one or several skills of the student model.

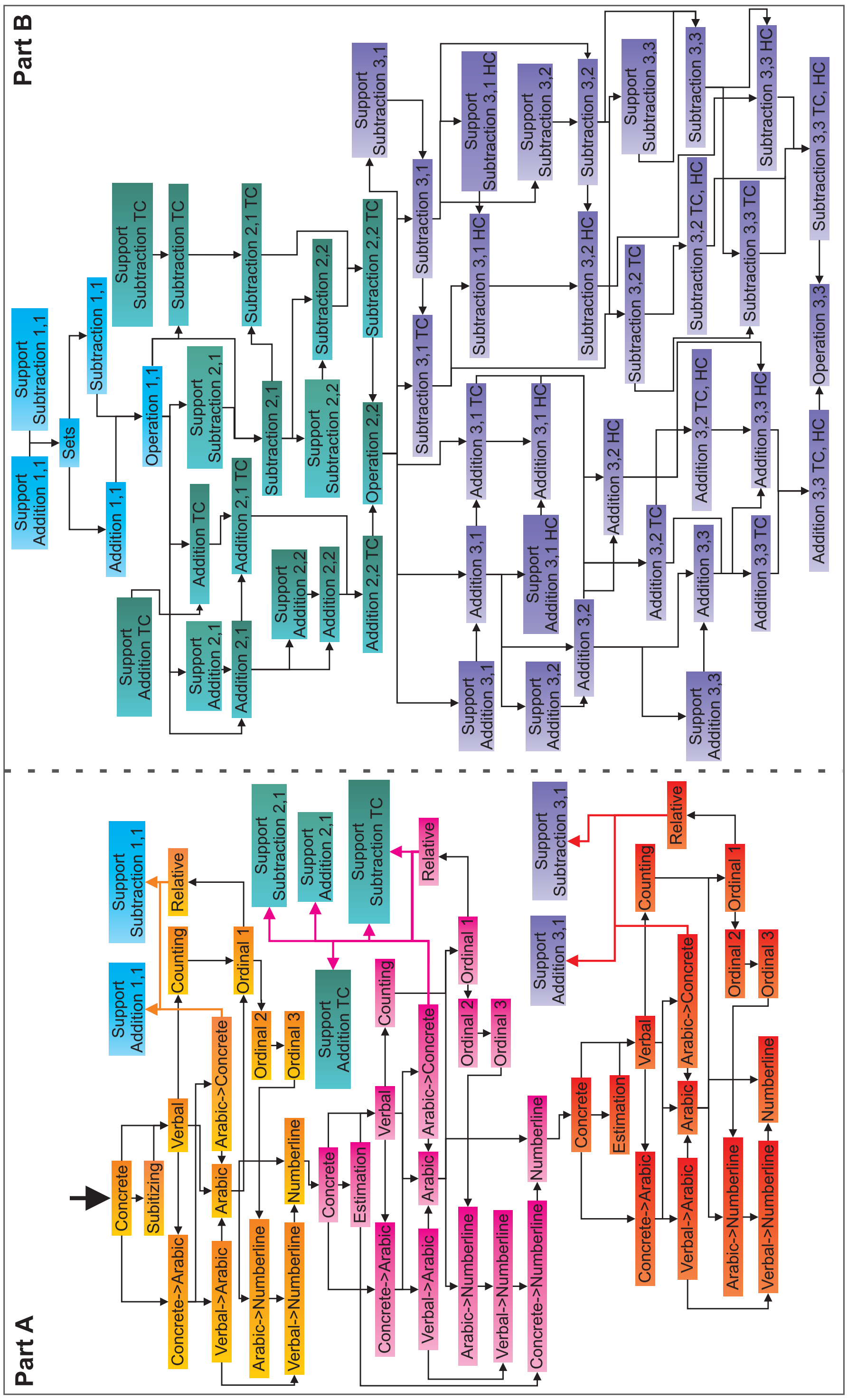


Figure 3.4: Skill net containing the skills of the area of *number representations* and *intuitive number understanding* (*Part A*, left) and *arithmetic operations* (*Part B*, right). The different colors denote the different number ranges 0-10 (yellow, blue), 0-100 (pink, green), and 0-1000 (red, purple). Some skills from *Part B* have been duplicated to *Part A* to denote the connections between the two areas.

Part A comprises the areas of *intuitive number understanding* and *number representations* of the curriculum (detailed in Sec. 3.2.2). The skills in *Part A* are ordered and color-coded according to the different number ranges 0-10, 0-100, and 0-1000. Within each number range, the hierarchy follows the four-step developmental model (von Aster and Shalev, 2007): The linguistic symbolization (step 2), Arabic symbolization (step 3), and analogue magnitude representation (step 4) develop based on a (probably) inherited representation of cardinal magnitude of numbers (step 1). Following this model, the transcoding between the linguistic and Arabic symbolization (*Verbal→Arabic*) is trained before giving the position of a written number on a number line (*Arabic→Numberline*). The skill *Subitizing* is associated with the `Subitizing` game, while the *Estimation* skill belongs to the `Estimation` game. Furthermore, the `Transfer` game trains the transcoding skills *Concrete→Arabic*, *Verbal→Arabic* and *Arabic→Concrete*. Skills *Ordinal 1* and *Relative* are affiliated with the `Distance` game, *Ordinal 2* represents the `Ordering` game and *Ordinal 3* is associated with the `Secret Number` game. Finally, the `Landing` game covers the skills *Arabic→Numberline*, *Verbal→Numberline* and *Concrete→Numberline*.

Part B covers the area of *arithmetic operations* of the curriculum (see Sec. 3.2.2). Skills in *Part B* can also be divided into the number ranges 0-10, 0-100 and 0-1000 (color-coded in Fig. 3.4). Furthermore, they are ordered according to their difficulties. The difficulty of a task depends not only on the magnitude of the numbers included in the task and the complexity of the task, but also on the representation of the task and the means allowed to solve it. A task such as '65+22=87' (*Addition 2,2*) is considered more difficult than computing '13+5=18' (*Addition 2,1*). On the other hand, modeling '65+22=87' with one, ten and hundred blocks (*Support Addition 2,2*) is easier than calculating it mentally. And finally, tasks involving carrying such as '65+27=92' (*Addition 2,2 TC*) are more complex to solve than tasks without a carry. All skills training mental calculation (*e.g.*, *Addition 2,2*) are covered by the `Calculator` game. Skills in the number range from 0-10 involving the use of material (such as *Support Addition 1,1*) are associated with the `Slide Rule` game, while the `Plus-Minus` game comprises such skills (for example *Support Addition 3,1 HC*) in higher number ranges.

In general, each skill of the hierarchical network is associated with a task, *i.e.*, there exists a game type for each skill in the network, as already detailed above. However, some skills such as for example *Counting* are not associated with any game and can therefore not be observed.

Table 3.1: Explanation of skills (by area) and notations used in the skill net (see Fig. 3.4).

Area	Notation	Definition
Part A		
Number Representations	Concrete	Number represented as a set of objects.
	Verbal	Spoken number.
	Arabic	Written number.
	Numberline	Number represented as a position on a number line.
Transcoding	r1→r2	Translation of number from representation r1 to r2.
Ordinality	Ordinal 1	Precursor and successor of a number need to be given.
	Relative	Calculate indirect (+/-2, +/-3) precursor and successors of a given number.
	Ordinal 2	Are the given numbers sorted in ascending order?
	Ordinal 3	Guess a secret number.
Other	Subitizing	Simultaneous perception of numbers up to four.
	Estimation	Which of three displayed point sets corresponds to the given number?
	Counting	Forward (and backward) counting.
Part B		
Mental calculation	Addition a1, a2	Addition of two numbers. a1 and a2 denote the number of digits of the addends. TC denotes a ten crossing and HC a hundred crossing.
	Subtraction s1,s2	Subtraction of two numbers. s1 and s2 denote the number of digits of the minuend and the subtrahend. TC denotes a ten crossing and HC a hundred crossing.
	Addition TC	Addition with carrying in the range from 0-20.
	Subtraction TC	Subtraction with borrowing in in the range from 0-20.
	Operation o1, o2	Addition or subtraction of two numbers used for repetition. o1 and o2 denote the number of digits of the operation. Operation 2,2 for example denotes any addition or subtraction skill in the number range 0-100.
Calculation concepts	Support Addition	Addition of two numbers. The task can be solved using one, ten and hundred blocks.
	Support Subtraction	Subtraction of two numbers. The task can be solved using one, ten and hundred blocks.
	Sets	Understanding of operations on sets.

3.3.2 Control algorithm

Based on the estimation of student knowledge delivered by the student model, *i.e.*, the posterior probabilities of the skills, the control algorithm needs to make the teaching decisions. The selection of actions is rule-based and non-linear, the algorithm allows forward (training of a more difficult skill) as well as backward movements (training of an easier skill). This controller design increases the set of possible actions (due to multiple precursors and successors). Therefore, rather than following a specified sequence to the goal, learning paths are adapted individually, *i.e.*, each child trains different skills and hence plays different games during training, as illustrated in Fig. 3.6. After each solved task, the controller selects one of the following options based on the current state:

1. **Stay**: Continue the training of the current skill;
2. **Go back**: Train a precursor skill;
3. **Go forward**: Train a successor skill;

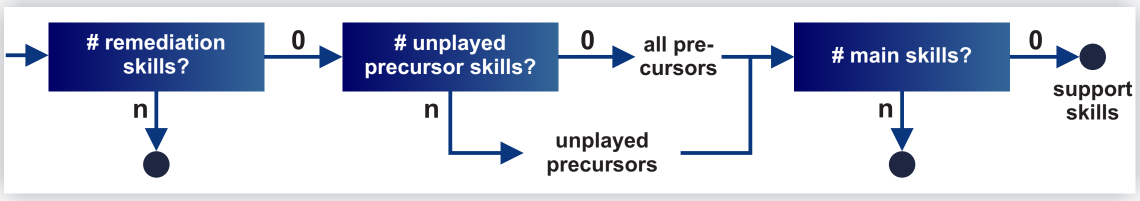
The decision is based on the posterior probabilities delivered by the student model. After each solved task, the controller fetches the posterior probability $p(s_{i,t}|\mathbf{o}_t)$ of the skill S_i being trained at time t. Then, $p(s_{i,t}|\mathbf{o}_t)$ is compared against a lower and an upper threshold, denoted by $p^l_{S_i}(t)$ and $p^u_{S_i}(t)$. The resulting interval defines the optimal training level: if the probability lies between the thresholds, 'Stay' is selected. In contrast, 'Go Back' and 'Go forward' are selected if $p(s_{i,t}|\mathbf{o}_t) < p^l_{S_i}(t)$ and if $p(s_{i,t}|\mathbf{o}_t) > p^u_{S_i}(t)$, respectively. Thresholds are not fixed, they converge with the number of played samples n_{S_i} at skill S_i:

$$p^l_{S_i}(t) = p^{l0}_{S_i} \cdot {l_c}^{n_{S_i}} \text{ and } p^u_{S_i}(t) = p^{u0}_{S_i} \cdot {u_c}^{n_{S_i}}. \tag{3.1}$$

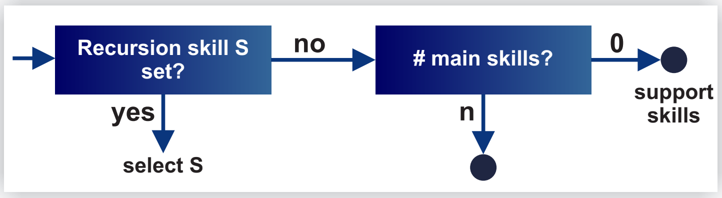
Initial values of the upper ($p^{u0}_{S_i}$) and lower ($p^{l0}_{S_i}$) thresholds as well as the change rates (l_c, u_c) are heuristically determined. The convergence of the thresholds ensures a sufficiently large number of solved tasks per skill and prevents training the same skill for too long without passing it.

When 'Stay' is selected, a new appropriate task (associated with the same skill) is built. Otherwise, a precursor (or successor) skill is selected by fetching all precursor (successor) skills of the current skill and feeding them into a decision tree. The nodes of these trees encode selection rules that were designed using experts' advice.

The decision tree for the 'Go Back' option is displayed in Fig. 3.5(a). For this option, remediation skills are preferred: If errors matching patterns of the bug library (see section Sec. 3.4) are detected, the relevant remediation skill is trained. A typical mistake in addition involving two-digit numbers would be to sum up all the digits, *i.e.*, '23 + 12 = 8' (*Addition 2,2*). This mistake indicates that the child has not yet



(a) Decision tree for the 'Go back' option.



(b) Decision tree for the 'Go forward' option.

Figure 3.5: Decision trees for the 'Go back' (a) and 'Go forward' (b) options. The rectangles denote decision nodes, while the circles represent the end nodes. At the end nodes, the candidate skill with lowest posterior probability ('Go back' option) or with posterior probability closest to 0.5 ('Go forward' option) is selected. The selection rules encoded in the trees were designed using experts' advice.

understood the Arabic notation system in the number range from 0-100. A remediation skill for this error is the training of the Arabic notation system in this range, *i.e.*, decomposing numbers between 0 and 100 into tens and units and thus learning the meaning of the digit position of a number (*Arabic*→*Concrete*).

If the child did not commit any of the typical errors, the controller prefers unplayed precursor skills. The hierarchical skill model assumes that the precursor skills of a skill S_a are a prerequisite for knowing S_a. If the child fails skill S_a, the controller tries to find the particular precursor skill that might cause the problem. For the played precursor skills, the controller assumes that the child already knows them (since they have been played and passed) and hence an unplayed precursor skill is selected. Finally, main skills are preferred over support skills. Therefore, if a child fails in solving addition problems with two-digit numbers (*Addition 2,2*) the controller first checks if the child can do mental calculation (= main skill) of simpler addition problems (for example *Addition 2,1*). If this is the case, the support skill modeling the operation with material is picked. If, however, the child also fails in solving the simpler addition problem, this easier skill needs to be trained first. Hence, the main skills are always checked first.

If there is more than one candidate precursor skill after crossing the decision tree (*i.e.*, going through all the rules), the candidate skill with the lowest posterior probability is selected. Therefore, the controller selects the skill where the child has the lowest (estimated) proficiency.

The decision tree for the ‘Go Forward’ option is displayed in Fig. 3.5(b). For this option, recursion skills are preferred. If a user fails to master skill S_a and goes back to S_b, S_a is set as a recursion skill. After passing S_b, the controller will return to S_a. If a child for example fails solving addition problems with two-digit numbers (*Addition 2,2*) and goes back to train an easier skill (*Addition 2,1*), the child will return to the addition problems with two-digit numbers (*Addition 2,2*) after passing that easier skill.

If no recursion skill is set, the controller again prefers main skills over support skills. If the child masters solving addition problems with a two-digit and a one-digit number (*Addition 2,1*) the controller will go further to ask addition problems involving two two-digit numbers (*Addition 2,2*). This rule ensures that children having a good mathematical knowledge take the fastest path through the skill net. The support skill modeling the same task using material (*Support Addition 2,2*) will only be played if the child does not master the mental calculation.

If there is more than one candidate successor skill at the end of the decision tree, the candidate skill with posterior probability closest to 0.5 (maximization of entropy) is selected. This final rule ensures that the gain of knowledge about the child is maximized. To consolidate less sophisticated skills and to increase variability, the controller uses selective recalls.

This control design exhibits the following advantages:

1. *Adaptability*: the network path targets the needs of the individual user (Fig. 3.6).
2. *Memory modeling*: forgetting and knowledge gaps are addressed by going back.
3. *Locality*: the controller acts upon current nodes and neighbors, avoiding unreliable estimates of far nodes.
4. *Generality*: the controller is domain-model independent: it can be used on arbitrary discrete structures.

3.4 Bug Library

The program has access to a bug library storing typical error patterns (Gerster, 1982) and is therefore able to adapt to specific errors of the children. If a child commits a typical error several times, the controller systematically selects actions for remediation. Table 3.2 lists the typical error patterns stored in the bug library, along with examples and remediation tasks. For the area of *number representations*, only one pattern is stored for the `Landing` game: Positioning the cone on the wrong

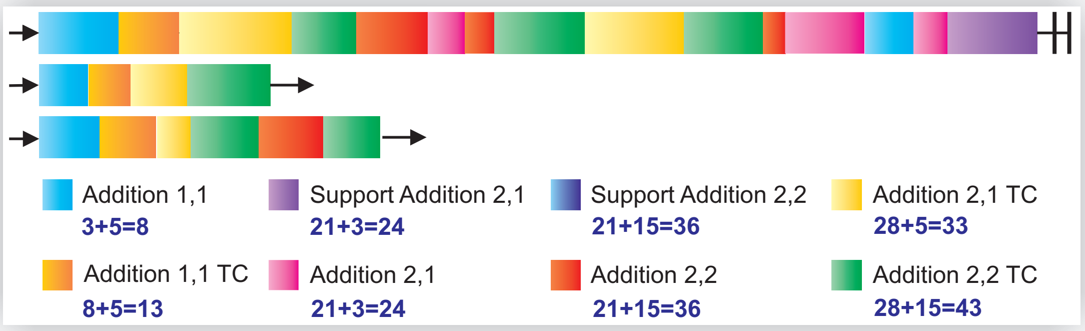


Figure 3.6: Skill sequences of three children in addition in the number range from 0 – 100. The notation is consistent with Fig. 3.4 and is explained in Tab. 3.1. User 2 (middle sequence) and user 3 (bottom sequence) passed all skills in the range, while user 1 (top sequence) did not pass this range within the training period. The length of the rectangles indicates the number of played samples at the respective skill.

side of the indicated center of the number line, *i.e.*, positioning the cone at a number < 50 when the given number is > 50. For the area of *arithmetic operations*, a range of error patterns are stored in the bug library. Some of these patterns can be attributed to problems in counting or understanding the basic concepts of addition and subtraction. Remediation skills for these error patterns train simple addition and subtraction tasks with colored blocks (*Support Addition/Subtraction 1,1*). Other error patterns probably occur due to a lack in understanding the Arabic notation system, *i.e.*, the meaning of the different positions of the digits. A selected remediation action for these patterns is the training of the Arabic notation system (*Arabic*→*Concrete*). Another typical error is the switching of digits (twenty-five is written as '52') which is remediated by training transcoding from spoken to written numbers (*Verbal*→*Arabic*). Finally, problems with carrying or borrowing are also addressed (*Support Addition/Subtraction TC*). The bug library was built based on previous work identifying typical error patterns and their causes (Gerster, 1982).

Table 3.2: Description of typical errors along with examples and remediation skills for the domains of *number representations* (NR) and *arithmetic operations*, separate for addition (A) and subtraction (S).

	Description	Example	Remediation
NR	Landing game: The child lands the cone on the wrong side of the center (5, 50, or 500).	-	Training of the ordering of numbers according to their magnitude (*Ordinal 1*).
A, S	Correct result is missed by 1 (+/- 1).	5 + 3 = 7	Training of addition or subtraction with colored blocks (*Support Addition/Subtraction 1,1*).
A, S	Addition instead of subtraction (or vice versa).	5 + 3 = 2	Training of addition or subtraction with colored blocks (*Support Addition/Subtraction 1,1*).
A	Addition of all digits.	12 + 24 = 9	Training of the Arabic notation system (*Arabic→Concrete*).
A, S	Switching of digits when reading/writing a number.	24 - 3 = 12	Transcoding from spoken to written notation (*Verbal→Arabic*).
A, S	Use of wrong digit order.	63 - 5 = 13	Training of the Arabic notation system (*Arabic→Concrete*).
A, S	Forgetting the carry when bridging to ten.	34 + 7 = 31	Training of bridging to ten using colored blocks (*Support Addition/Subtraction TC*).
A, S	Addition/Subtraction of inner and outer digits.	34 + 13 = 56	Training of the Arabic notation system (*Arabic→Concrete*).
S	Building the difference between digits.	34-17 = 23	Training of the Arabic notation system (*Arabic→Concrete*).

CHAPTER 4

User studies

When developing a training or therapy program, assessment of the actual clinical effectiveness by means of evaluation studies is essential. User studies are, however, not only important for evaluation of effectiveness, but also enable data collection: During training with `Calcularis`, all user actions (such as keyboard input and keystrokes) along with respective timestamps are saved to log files. Such log files allow for further analysis of user behavior and the applied models (see Chapter 5 and Chapter 6) and refinement of the models based on the collected data (as described in chapters 7- 9).

In the first part of this thesis, the computer-based training program `Calcularis` (detailed in Chapter 3) along with a knowledge representation for mathematical skills was developed. In a second step, `Calcularis` was evaluated in two user-studies. One study was conducted in Germany and Switzerland by our collaboration partners (University of Potsdam, University Children's Hospital Zurich) and included 134 children from $2^{nd} - 5^{th}$ grade of elementary school: 64 children were diagnosed with developmental dyscalculia (DD) and 70 were control children. Evaluation of training effects is still under progress and therefore a detailed description of this study and its results is not provided in this thesis. In the following, we will refer to this evaluation study as *BMBF-study*. The second study, referred to as the *SWISS-study*, was conducted in Switzerland and included 41 children with difficulties in learning mathematics.

In this chapter, we will describe the evaluation study conducted in Switzerland, the *SWISS-study*, in detail. We will first introduce the study design and the participants, before describing the external test measures. After that, we will present and discuss the results and limitations of the study.

4.1 Study design and participants

The *SWISS-study* included 41 children with difficulties in learning mathematics. Participants were divided into a training group ($n = 20$, 65% females) completing a 12-weeks training and a waiting group ($n = 21$, 66.6% females) starting with a 6-weeks rest period before executing a 6-weeks training. This study design has two advantages. First, the effects of a 12-weeks training can be compared to those of a 6-weeks training, assessing if children profit from a prolonged training period. Second, comparing the training effects of the training group to those of a waiting group allows controlling for developmental and schooling effects.

Mathematical performance of both groups was evaluated at the beginning of the study (t_1), after six weeks (t_2) and after 12 weeks (t_3). Children were required to train with the program five times per week, with daily training sessions of 20 minutes. The groups were matched according to age (training group: 9.96 years (SD $\sigma = 1.35$); waiting group: 9.98 (SD $\sigma = 1.33$); $t(39) = -0.04$, $p = .96$), gender and intelligence (training group CFT-score: 93.8 (SD $\sigma = 11.9$); waiting group CFT-score: 93.5 (SD $\sigma = 14.1$); $t(39) = 0.07$, $p = .95$) (Cattell et al., 1997; Weiss, 2006). Groups were built by forming matched pairs of kids, followed by a quasi-random assignment to either the training or waiting group (ensuring that the number of males was balanced between the groups).

All participants were German-speaking and visited the 2nd-5th grade of elementary school. Children were indicated by parents and teachers as exhibiting difficulties in learning mathematics. On average, arithmetic performance (measured with the *Heidelberger Rechentest* HRT (Haffner et al., 2005)) of the participants was around the 10th percentile, corresponding to a T-score of 37 (HRT addition T-score: 37.15 (SD $\sigma = 7.69$); HRT subtraction T-score: 37.29 (SD $\sigma = 8.77$)). At the beginning of the study (t_1), there was no significant difference in arithmetic performance between the groups (HRT addition: $t(39) = 0.59$, $p = .55$; HRT subtraction: $t(39) = -0.63$, $p = .53$).

Children performed the training at home with the exception of one mandatory training session per six weeks at our laboratory. Children received a sticker per completed training session that they could put on their training progress sheet. During the training period, all the input data of the children was saved. Therefore, the exact training time of the children could be determined at the end of the study and children with an insufficient number of sessions were excluded from the analysis.

Parents gave informed consent and children received a small gift for their participation. The *SWISS-study* was conducted in context of the *BMBF-study*, which was approved by the ethics committee of the University of Potsdam.

4.2 Instruments

All children underwent a series of mathematical performance and number processing tests at t_1, t_2 and t_3, detailed below. Furthermore, they completed a questionnaire after the training, including questions on difficulty, motivation, and personal evaluation of the training.

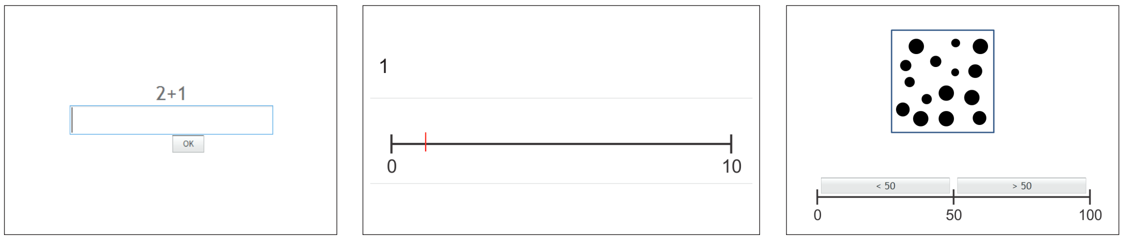
4.2.1 Heidelberger Rechentest (HRT)

Arithmetic performance was assessed using the addition and subtraction subtests of the HRT (re-test reliability: addition $r_{tt} = .82$, subtraction $r_{tt} = .86$). In these subtests, children are presented a list of addition (subtraction) tasks ordered by difficulty. The goal is to solve as many tasks as possible within two minutes. The maximum number of correct tasks is 40. During the test sessions, the addition subtest of the HRT was always solved first, followed by the subtraction subtests and the computer-based tests described below.

4.2.2 Computer-based tests

Children also solved a series of computer-based mathematical tests, illustrated in Fig. 4.1:

- **Arithmetic (AC)** (see Fig. 4.1(a)): In this test, children solve a series of addition (subtraction) tasks. Trials are ordered by difficulty and presented serially. The time to solve the tasks is ten minutes. The maximum number of solved tasks is 76.
- **Number line (NL)** (see Fig. 4.1(b)): In this test, children need to indicate the position of a given number (presented in Arabic notation as well as verbally) on a number line. The number line is represented on the screen as a one-dimensional black line with labeled end points. The position of the number can be indicated by mouse-click. There are ten tasks in the number range from 0-10 (NL 10), 20 tasks between 0-100 (NL 100) and ten tasks between 0-1000 (NL 1000).
- **Non-symbolic magnitude comparison (NC)**: In this test, children are presented ten sets with $1 - 9$ black dots (excluding 5) for a period of 120 milliseconds. Children need to indicate if the presented number of dots was smaller or larger than 5. The representation of the dots is balanced according to spatial distribution and area



(a) Arithmetic test (AC). (b) Number line test (NL 10). (c) Estimation test.

Figure 4.1: Example tasks for the different subtests of the computer-based assessment. In the AC, children solve a series of addition (a) and subtraction tasks. The NL 10 consists of indicating the position of a given number on a number line (b). In the Estimation test (c), children need to judge if the given point set is smaller or larger then 50 by clicking on the according buttons.

properties as described by Rubinsten and Sury (2011). The black area is the same for all trials. Half of the trials have a small extension (high density), while the other half is spread out (low density). The layout of the tasks is the same as for Estimation (see Fig. 4.1(c)) with the exception that the number line goes from 0 to 10 with the position of the 5 indicated.

- **Estimation** (see Fig. 4.1(c)): In this task, children are presented twenty sets with $1 - 99$ black dots (excluding 50). Children need to decide, if the presented sets are smaller or larger than 50. Numbers are equally distributed over the range. Confounding visual factors are controlled as described in the NC task. Stimuli are shown for a period of five seconds.

During the test sessions, the different tests were solved in the following order: AC addition, NL 10, NC, AC subtraction, NL 100, estimation, NL 1000. The computer-based tests exist in three paralleled versions (one per measurement point). The versions were paralleled according to content and item difficulty. Each version of the addition and subtraction tests for example contains the same number of tasks between 0-10 and the same number of tasks involving carrying or borrowing.

4.2.3 Feedback questionnaire

Children completed a training evaluation questionnaire at the end of the study (t_3). Children indicated for each game, how much they liked it. The scale was represented through smileys, going from a laughing (4) to a crying (0) smiley. The difficulty of the training was judged on a scale from very easy (0) to very difficult (4). And finally, children needed to indicate if the training helped them on a scale from not true (0) to absolutely correct (3).

4.3 Results

To ensure a sufficient number of training sessions, only children with at least 24 complete sessions after the 6-weeks training period were included in the evaluation of the training. Thus, five children from the training group (4: technical challenges, 1: $<$24 training sessions) and four children from the waiting group (1: abort of study, 3: $<$ 24 training sessions) were excluded from the analysis. The exclusions did not change the matching of the groups. Table 4.1 gives an overview of the training statistics. Over the course of the 6-weeks training, there were no significant differences between the training group and the waiting group regarding the number of training sessions ($t(30) = -1.40$, $p = .17$) or the number of solved tasks ($t(30) = -1.03$, $p = .31$). The two groups also showed a very similar average speed, *i.e.*, number of solved tasks per session ($t(30) = 0.03$, $p = .98$). Furthermore, the training group and the waiting group progressed equally fast during the 6-weeks: There were no significant differences in the highest reached skills in *number representations* ($t(30) = -0.04$, $p = .97$) and *arithmetic operations* ($t(30) = 0.26$, $p = .79$). Table 4.1 also illustrates a (probably) decreasing motivation over time: While the children of the training group played on average 30.2 sessions during the first six weeks, this number dropped to an average of 19 sessions in the second part of the training. Children also got a bit slower in the second part, *i.e.*, they solved fewer tasks per session. This slowdown is, however, probably due to the increasing difficulty of the tasks.

4.3.1 Quantitative analyses

A repeated measures general linear model (GLM) analysis was conducted to evaluate training effects ($t_1 - t_2$) as a within-subject factor and group (Training/Waiting) as a between-subject factor. Post-hoc paired-sample t-tests were used to test for differences in performance for consecutive testing periods ($t_1 - t_2$, $t_2 - t_3$). Effect sizes r were computed according to Field (2009). According to Field (2009), $r = 0.1$ is a small effect, $r = 0.3$ a medium effect and $r = 0.5$ a large effect. No corrections for multiple testing were applied. Table 4.2 summarizes the means and standard deviations of the behavioral measures for all measurement points, including calculated statistical results. There were no between-group performance differences prior to the intervention (at t_1).

Arithmetic (AC addition and subtraction). The interaction between training and group was significant for subtraction ($p = .028$) and showed a trend for addition ($p = .081$). Both operations demonstrated medium effect sizes (subtraction: $r = 0.39$, addition: $r = 0.31$). The prolongation of the training from 6 to 12 weeks

Table 4.1: Training statistics (standard deviations σ in brackets) of the training group (n = 15) and the waiting group (n = 17). For the 6-weeks training period, there were no significant differences between the training group and the waiting group.

	6-weeks period Training group ($t_1 - t_2$)	**6-weeks period** Waiting group ($t_2 - t_3$)	**12-weeks period** Training group ($t_1 - t_3$)
# training sessions	30.2 (3.2)	32.4 (5.2)	49.2 (2.6)
# totally solved tasks	1635.0 (293)	1737.0 (266)	2575.0 (414)
# solved tasks per session	54.0 (7.2)	54.0 (5.7)	52.2 (7.0)
Highest skill reached[a] (*Number representations*)	38.6 (8.3)	38.7 (8.4)	40.5 (6.8)
Highest skill reached[a] (*Arithmetic operations*)	40.5 (14.7)	39.1 (15.5)	43.0 (15.1)

[a] The skills of the adaptive model are divided into the content areas of the training program described in Sec. 3.2.2. Skills in each area are ordered by their number, with the easiest skill having the lowest number.

($t_2 - t_3$) yielded an additional trend of improvement (addition: $p = .072$; subtraction: $p = .066$).

HRT (addition and subtraction). The interaction between training and group was significant only for subtraction (subtraction: $p = .002$; addition $p = .375$), where children showed a large effect size ($r = 0.52$). The prolongation of the training yielded an additional improvement, which was significant only for addition ($p = .004$).

Number line (NL). The quality of the spatial number representation was measured by calculating the distance (percentage) and the variance of the distance between the correct and the indicated location of the number on the number line. In the number range from 0-10, children tended to locate the correct position on the number line more accurately after training ($p = .058$) and showed decreased variance ($p = .022$). The interaction between training and group was significant only for the variance (mean: $p = .12$; variance: $p = .034$). Children demonstrated medium effect sizes for both measures (mean: $r = 0.28$, variance: $r = 0.38$). The prolongation of the training did not yield any further benefit. In the number range from 0-100, interaction between training and group was not significant (mean: $p = .33$; variance: $p = .50$). The prolongation of the training had a beneficial effect (mean: $p = .042$;

Table 4.2: Training effects of training group TG (n = 15) and waiting group WG (n = 17) on mathematical performance (Means μ (SD σ)).

Mathematical performance		t_1	t_2	t-score ($t_2 - t_1$)	F-score[d]	t_3	t-score ($t_3 - t_2$)	ES[e]
AC Addition[a]	TG	25.9 (10.8)	30.0 (14.1)	2.38*	3.26^{+}	34.4 (17.7)	1.95^{+}	0.31
	WG	27.4 (10.7)	26.1 (12.0)	-0.56		29.4 (11.7)	1.90^{+}	
AC Subtraction[a]	TG	19.2 (12.7)	24.7 (17.1)	2.77*	5.32*	28.8 (17.9)	1.99^{+}	0.39
	WG	19.6 (10.2)	18.9 (10.7)	-0.39		26.3 (14.5)	4.11**	
NL10, mean[b]	TG	13.2 (10.1)	9.3 (10.6)	-2.06^{+}	2.56	7.5 (5.0)	-0.81	0.28
	WG	10.3 (10.4)	12.3 (11.6)	0.65		6.1 (4.2)	-2.70*	
NL10, var[c]	TG	10.2 (6.3)	6.9 (6.5)	-2.58*	4.92*	6.7 (4.6)	-0.15	0.38
	WG	7.4 (6.4)	9.4 (7.6)	1.02		4.9 (3.3)	-3.10**	
NL100, mean[b]	TG	10.2 (4.8)	9.6 (6.4)	-0.62	0.98	7.6 (3.3)	-2.24*	0.18
	WG	13.5 (6.0)	11.3 (5.7)	-1.72		9.3 (7.0)	-1.35	
NL100, var[c]	TG	7.7 (3.4)	8.2 (5.1)	0.39	0.47	6.2 (3.0)	-2.15*	0.12
	WG	9.7 (4.3)	9.1 (5.2)	-0.59		8.2 (5.8)	-0.79	
NL1000, mean[b]	TG	18.5 (10.9)	16.1 (7.5)	-1.61	0.70	12.9 (6.7)	-1.79^{+}	0.15
	WG	18.0 (7.3)	17.4 (8.1)	-0.44		12.6 (5.6)	-4.20**	
NL1000, var[c]	TG	13.3 (7.1)	11.9 (5.7)	-0.84	0.00	10.0 (6.4)	-1.01	0.00
	WG	13.5 (5.6)	12.0 (5.7)	-1.13		10.0 (4.4)	-1.87^{+}	
Estimation[a]	TG	15.1 (3.9)	14.9 (3.2)	-0.12	2.85	15.8 (2.1)	1.25	0.29
	WG	13.6 (4.5)	16.3 (2.2)	2.25*		15.9 (2.9)	-0.61	
NC[a]	TG	7.9 (1.9)	8.1 (1.5)	0.39	0.21	8.4 (1.6)	0.59	0.08
	WG	7.4 (2.3)	7.8 (2.2)	0.94		8.0 (1.8)	0.19	
HRT Addition[a]	TG	18.7 (5.4)	20.4 (5.6)	2.47*	0.81	22.5 (5.4)	3.46**	0.16
	WG	18.5 (4.8)	19.4 (4.3)	1.27		20.4 (5.7)	1.5	
HRT Subtraction[a]	TG	15.3 (6.1)	19.8 (5.3)	4.85***	11.38**	20.2 (6.2)	0.59	0.52
	WG	16.9 (6.3)	16.9 (5.6)	0.06		18.4 (5.7)	1.5	

$^{+}$ $p < .1$, * $p < .05$, ** $p < .01$, *** $p < .001$
[a] number of correctly solved tasks
[b] distance (percentage) from correct position
[c] variance of distance (percentage) from correct position
[d] time (t_1-t_2) x group
[e] Effect sizes of interaction time (t_1-t_2) x group. r = .10: small effect, r = .30: medium effect, r = .50: large effect

Table 4.3: Characteristics of the three example cases. The children included in the case studies were girls of similar age with a number of complete training sessions close to 30.

	Sex	**Age**	**Class**	**Played sessions**	**Solved tasks**	**Tasks per Session**
Anne	female	8;11	3	28	1272	45.4
Eva	female	9;8	4	33	1803	54.6
Jane	female	9;10	4	33	1795	54.4

variance: $p = .05$). In the number range from 0-1000, children tended to locate the numbers more accurately only after 12 weeks (mean: $p = .096$; variance: $p = .331$).

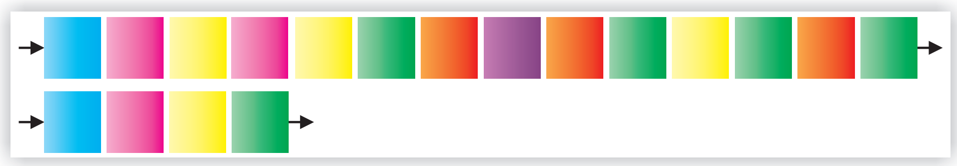
NC and Estimation. In these two tasks, the interaction between training and group was not significant (Estimation: $p = .11$; NC: $p = .65$). Unexpectedly, the waiting group showed a significant improvement in the Estimation test ($p = .039$). This significant result stems from outliers with large improvement (children with two correct answers at t_1 and 17 correct answers at t_2) due to not understanding the task at t_1.

Feedback questionnaire. Children generally liked the training (average score: 3.0 (SD $\sigma = 0.55$), scale: $0-4$) and rated its difficulty as appropriate (average rating: 1.7 (SD $\sigma = 0.74$), scale: $0-4$). They also reported that the training helped them to improve in mathematics (average score: 2.1 (SD $\sigma = 0.89$), scale: $0-3$). For both groups, there was no correlation between the ratings for the difficulty and the liking (training group: $\rho = -0.07$, $p = .82$; waiting group: $\rho = 0.25$, $p = .38$). Furthermore, there were no significant differences between the scores of the training and the waiting group.

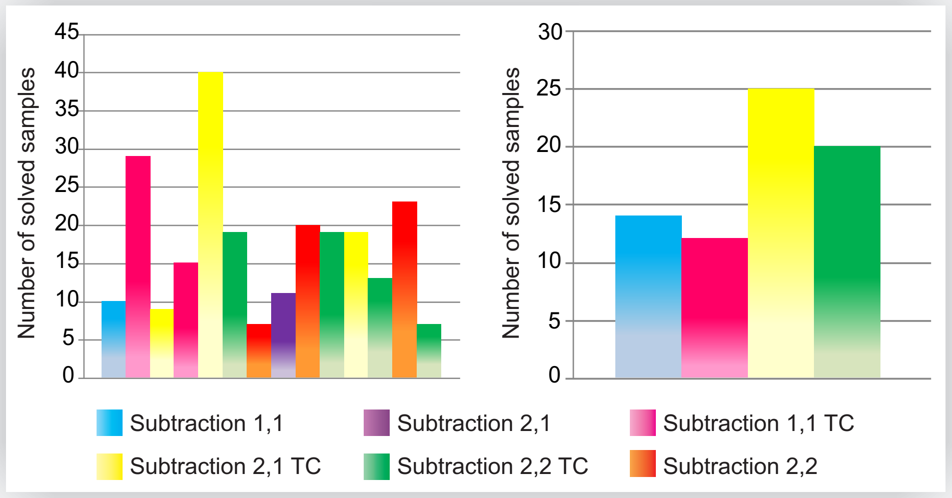
4.3.2 Qualitative analyses

To (qualitatively) assess the concept of the student model and the controller, we conducted case studies with three children for different domains of the learning program. We were particularly interested in analyzing the path through the skill net for different children and the association of training success within the program with external pre- and post-test results. Furthermore, the case studies provide an illustration of the concept of the learning program and the operation of the controller. In the following, the path through the skill net and the training success of a few selected children is described. The children and their training characteristics are depicted in Tab. 4.3. The analyses stem from the 6-weeks training period.

Subtraction 0 – 100. The first case study includes Anne and Jane. We analyzed



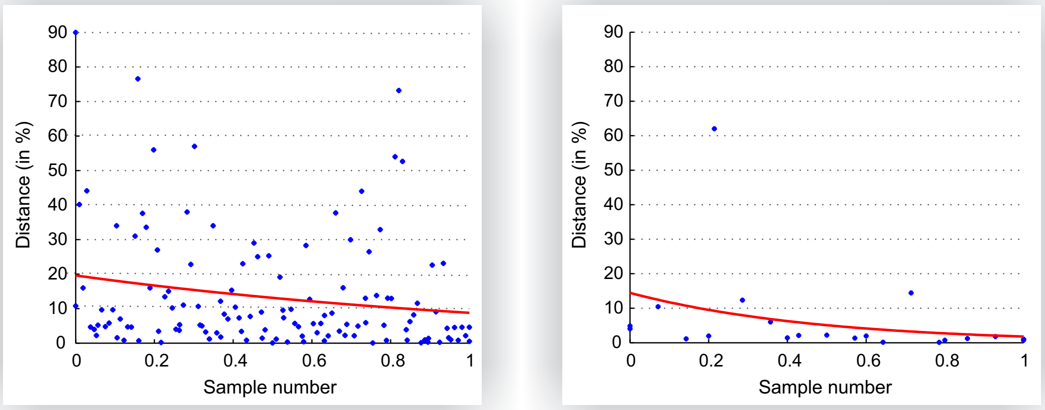
(a) Skill sequences of Anne (top) and Jane (bottom).



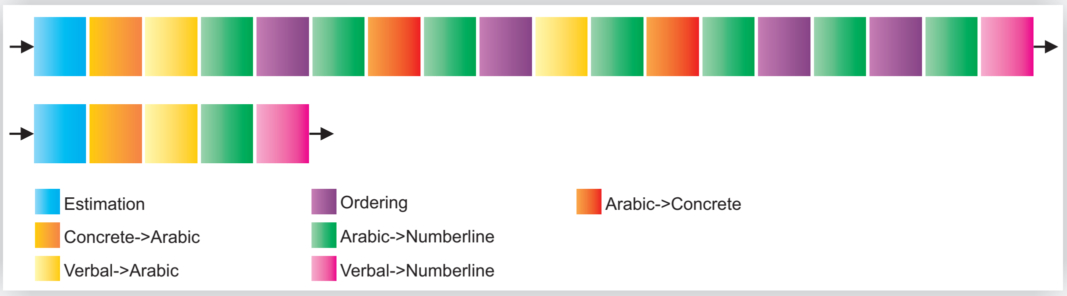
(b) Number of played samples for Anne (left) and Jane (right).

Figure 4.2: Skill sequences (a) and number of played samples (b) in subtraction from 0-100 for Anne and Jane. The skills correspond to the subtraction skills of the student model illustrated in Fig. 3.4 and explained in Tab. 3.1. While Jane took the direct path through the section and needed few samples to master the subtraction skills, Anne's sequence exhibits several branches and she therefore spent more time in subtraction.

their learning paths (path through the skill net) for subtraction in the range from 0-100. Figure 4.2 illustrates the sequence of skills of the two children and the respective numbers of played samples. It becomes clear, that the path through the skill net is different for each child (see Fig. 4.2(a)). While Jane took the straight path through the subtraction section, the path of Anne exhibits several branches as she had to go back and consolidate more basic skills. This fact is also demonstrated in Fig. 4.2(b). Jane needed in total only 71 samples to pass the subtraction 0 – 100 section, whereas Anne solved 241 samples to work through this part of the skill net. The external training effects in subtraction from 0-100 (measured by the AC subtraction test, Sec. 4.2.2) support this result. In the initial measurement before the training, Jane solved in total 40 tasks, 39 of them correct. She was already proficient in subtraction tasks between 0-100 before the training. In contrast, Anne solved in total 26 tasks, 10 of them correct. After the training, Anne managed to solve 23



(a) Average distance (in %) from correct position over time for Eva (left) and Jane (right).



(b) Skill sequences of Eva (top) and Jane (bottom).

Figure 4.3: Average distance (in %) from correct position over the course of the training (a) as well as skill sequences in the area of *number representations* from 0-100 (b) for Eva and Jane. The skills correspond to the number representation skills (*Part A*) of the student model illustrated in Fig. 3.4 and explained in Tab. 3.1. Jane mastered the skills directly, while Eva had to go back and rehearse easier skills.

tasks correctly; she especially improved in subtraction involving borrowing. Also Jane showed an improvement after the training, she solved 49 tasks correctly. However, most of her improvement stemmed from subtraction tasks in the range from 0-1000 (the AC subtraction test contains 32 tasks between 0-100; the rest of the tasks is in the range from 0-1000).

Number line 0-100. For Eva and Jane, the ability to place a number on a number line (between 0-100) was compared. This ability is trained by the skills *Arabic*$\rightarrow$*Numberline* and *Verbal*$\rightarrow$*Numberline* in the number range from 0-100 of the skill net (see Fig. 3.4). Before the training, Eva managed the task with an average distance of 11.4% (measured by the NL 100 test, Sec. 4.2.2). In contrast, Jane reached an average distance of 5.4%. Thus, Jane was already more accurate than Eva at the beginning of the training. This fact was confirmed during the course of the training. While Eva needed 127 samples, to pass the respective skill, Jane passed with only 21 samples. The maximum deviation for a sample to be rated as correct was 5%. Figure 4.3(a) displays the improvement curves over the course of the training. Recorded input data from all children shows that most samples exhibit an error of 0 – 20% with only a few samples lying above this range. Therefore, fitting has been done using a generalized linear regression model, assuming a Poisson distribution of the data. The sample indices have been normalized between 0 and 1.

The training sequences in the area of *number representations* in the number range from 0-100 of the two children show the same picture (see Fig. 4.3(b)). Jane took the direct path through the skill net, whereas Eva had to go back several times. After the training, Eva achieved an average deviation of 6.5% in the NL 100 test and Jane's average deviation was 5.1%. While Eva improved significantly, Jane stagnated on a high level.

4.4 Discussion

Although many children experience difficulties in learning mathematics, few studies have investigated targeted interventions based on neuro-cognitive findings of the typical and atypical development of mathematical abilities. In this thesis, we developed a computer-based intervention targeting children with difficulties in learning mathematics and performed a first evaluation of its effectiveness. The results achieved are promising and show significant improvements in subtraction and number representation. Moreover, they confirm the behavioral effects obtained in a previous study employing the computer-based training program `Rescue Calcularis` (Kucian et al., 2011).

Training. The first pilot study was conducted not only to assess the efficacy of the training program but also the practicality and adaptability of the learning environment. Feedback from children who have completed the training and rated the difficulty level of the learning program as appropriate, confirms that the quality of the adaptation and the estimation of the children's knowledge were sufficient. The evaluation of the feedback questionnaire also supports the improvement of mathematical performance measured in the external tests: On average, children reported that the training had improved their mathematical performance. This subjective feeling of improvement and learning success might also enhance positive self-concepts (Ashcraft and Faust, 1994; Spitzer, 2009). Moreover, children also indicated that they liked to train with the program. The popularity of the learning environment is beneficial as training can only be successful if the children are motivated. Furthermore, the finding demonstrates that the computer is an attractive medium for children and is in line with previous studies (Kulik and Kulik, 1991; Schoppek and Tulis, 2010; Kucian et al., 2011).

Behavioral effects. The results of the user study reveal positive training effects in mathematical skills after completion of the training. Children significantly improved their subtraction skills over the course of the 6-weeks-training: They were not only able to solve more complex subtraction problems (medium-large effect in the AC subtraction test) but also solved subtraction tasks faster (large effect in HRT). This improvement in subtraction supports the notion of a better mathematical understanding as subtraction is considered as a main indicator for the development of the spatial number line representation (Dehaene, 2011). Furthermore, the decrease in problem solution times can be seen as a shift to increased fact retrieval (Geary et al., 1991; Lemaire and Siegler, 1995; Barrouillet and Fayol, 1998; Jordan et al., 2003). Compared to subtraction, children demonstrated smaller effects in addition (medium effect in the AC addition test). This may be due to the adaptive nature of the intervention: Addition and subtraction tasks are trained in parallel for each difficulty level. As children performed better in addition in the pre-test, they received more training in subtraction during the intervention. Interestingly, the waiting group did not show significant training effects in the HRT subtests after their 6-weeks training ($t_2 - t_3$). This fact might stem from the low number of participants or from the adaptability of the training program leading to a different training trajectory for each child.

Children were also able to locate the position of a number on a number line more accurately after training. In the number range between 0-10, the deviation from the correct position was reduced by 33% after six weeks. Children especially also reduced the variance (medium-large effect size). No further improvement was yield by the prolongation of the training. Yet, most children passed the skills in the number range from 0-10 in the first few weeks and thus did not train in this range anymore in the second part of the training. In the number range between 0-100, there was no significant interaction. However, the training effect was significant after three months (reduction of deviation about 30%). This delay is probably due to the fact, that some of the children arrived at this level only in the second part of the training. Better performance in the number line task indicates refinement of the internal mental number line and more accurate access to it and confirms the results of previous studies (Siegler and Booth, 2004; Booth and Siegler, 2006, 2008; Halberda et al., 2008) which demonstrated significant correlations between arithmetical learning and the quality of numerical magnitude representation.

No significant training effects were observed in the NC and Estimation tests. These results however need to be interpreted with caution because of ceiling effects. At the pre-tests, children solved on average 80% of the NC and 75% of the Estimation tasks correctly. Furthermore, some children even reached the maximum score. This result is in line with previous findings (Noël and Rousselle, 2011).

For most of the tasks tested before and after training, prolongation of the training from six weeks to 12 weeks yielded a beneficial effect. The improvement of the training group over the whole training period ($t_1 - t_3$) was significant for all tests except the Estimation and NC tests. In some tasks (for example NL 100 and NL 1000), the effects of the second part of the training were similar or higher to those of the first part. This may be due to two facts: First, as the training covers the number range from 0-1000, most children had not worked through the whole training after the first six weeks. Second, the intervention trains different abilities, whose effects support each other. It has been shown that the training of conceptual knowledge and number representations leads to an improvement also in mental calculation (Kaufmann et al., 2003, 2005). Furthermore, the training of arithmetic operations implicitly deepens the knowledge of number representations. These supporting effects between the different abilities, however, need time to develop (Kaufmann et al., 2003, 2005). The prolongation of the training time from six to 12 weeks thus probably led to a strengthening of the mutual effects between the training in number representations and the training in arithmetic operations.

Although a training program focusing on a broad range of mathematical skills and showing a high degree of individualization seems beneficial, it also poses challenges for evaluation. First, training a variety of skills shortens the training time of each specific skill and thus leads to smaller training effects, as mentioned above. Second, due to the high adaptability of the program, each child pursues a different training trajectory, *i.e.*, the children train different skills over the course of the training. Therefore, training progress is difficult to compare and inconsistencies in training effects may be observed.

Limitations. Some limitations regarding the participants and the study design have to be considered. First, there were no measurements done after a 12-weeks rest period. Thus, for the 12-weeks training period, the training effects could not be compared to the effects of a rest period. Regarding the significant effects of the 6-weeks training, we conclude that also the effects of the 12-weeks training period can be plausibly attributed to the training.

Second, children were not tested according to common criteria of DD. Children were indicated by parents and teachers as exhibiting difficulties in learning mathematics. Generally, participants indeed demonstrated a mathematical performance below the 25th percentile in the pre-tests (the four children performing above the 25th percentile had insufficient grades in math). As described in Sec. 4.1, the participants' mean score even demonstrated an arithmetic performance around the 10th percentile. It has been shown that the cognitive characteristics of low performing children are indeed dependent on the cut-off criterion used. However, children fulfilling a softer criterion exhibit similar difficulties to those fulfilling stronger criteria, but to a smaller extent (Murphy et al., 2007). Therefore, our less strict criterion for deficits in mathematical performance seems also informative.

Third, the effects of the training period were only compared to those of a rest period. No comparison to a control training was conducted. As this first pilot study was designed to evaluate the concepts used in the training program and to assess its adaptability, the design used seems sufficient. In the *BMBF-study*, we also compare the effects to a control training.

CHAPTER 5

Learning curve analysis

When building a computer-based learning program, it is not only essential to assess the effectiveness of the training, but also the quality and characteristics of its components. In a first step (detailed in Chapter 4), we have evaluated the effectiveness of `Calcularis` in two user-studies: The *BMBF-study* and the *SWISS-study*. Based on the log file data collected in these two studies - containing all user actions (such as keyboard input and keystrokes) along with respective timestamps - we validated the skill model of `Calcularis` by employing an Additive Factors Model (AFM) (Cen et al., 2007, 2008).

AFMs are popular models for analyzing and improving student learning. However, applying such models to data from tutoring systems that employ a mastery-learning mechanism whereby poorer students get assigned tasks that better students do not may result in potential parameter estimate biases. We therefore propose a range of alternative logistic regression models for model validation and extensively analyze and evaluate them on the data collected in the user studies.

To facilitate the model validation conducted in this chapter as well as the data-driven analyses performed in Chapter 6, we introduce the concept of 'key skills'. Key skills are defined in terms of subject-dependent difficulty, they are the most difficult skills for the user to pass. More formally,

Definition 5.1. *A skill S_a is a* ***key skill*** *for a user U, that is $S_a \in \mathcal{K}_U$, if the user went back to a precursor skill S_b at least once before passing S_a.*

From this follows that the set of key skills $\mathcal{K}_U$ may be different for each user U (and it typically is). In the sequence shown in Fig. 3.6, user 2 has no key skills, while user 3 has one key skill (colored in green) and user 1 has several key skills.

In the following, we first explain the validation of the skill model used in `Calcularis` by employing the widely used approach of AFMs. We then propose alternate logistic regression models, explicitly designed to adjust for mastery-learning. Finally, we provide a detailed evaluation and comparison of the different models regarding their properties as well as prediction accuracy on new data.

5.1 Validation of skill model using AFM

AFMs are widely used for model validation. Flat learning curves can for example be an indication for an underestimation of student knowledge (Cen et al., 2007). Based on AFMs, skill models can be refined and improved (Koedinger et al., 2013; Stamper and Koedinger, 2011), which inspired the design of novel instructional tasks (Koedinger et al., 2010). In the following analyses, we assess addition and subtraction skills of `Calcularis` using AFMs.

The data set used for the experimental evaluation was collected in the *BMBF-study* (see Chapter 4). It contains recorded log files from 134 participants (69% females). 64 participants (73% females) were diagnosed with developmental dyscalculia (DD) and 70 participants (66% females) were control children (CC). The collected log files contain six weeks of training with at least 24 complete sessions (of 20 minutes) per child. On average, children completed 28.9 sessions (SD $\sigma = 3.3$). Over the course of the training, each child solved 1521 tasks (SD $\sigma = 269$), while the number of solved tasks per session corresponded to 52.7 (SD $\sigma = 7.2$).The following analyses as well as the experimental evaluation include all addition and subtraction skills in the number range from 0-100 of the skill model of `Calcularis`, resulting in a total of 20 skills included. These skills are colored in dark turquoise in Fig. 3.4. The notation of the different skills is explained in Tab. 3.1. For our analyses, we include only so called ‘regular’ samples, *i.e.*, random re-tests of already mastered skills are excluded, which results in a data set containing $36'350$ tasks. We assume that the random re-tests help to prevent forgetting, but do not induce further learning.

To validate the skill model of `Calcularis`, we fit a standard AFM (see Eq. (2.2)) with the following parameters to our data set: the random effect θ_p for student proficiency, the fixed effect β_k (difficulty) and the fixed effect γ_k (learning rate) for the skills S_k. The fitted random effect θ_p amounts to 0.751 (SD $\sigma = 0.867$), the fixed effects β_k and γ_k are displayed in Tab. 5.1. Only seven skills (*i.e.*, 35% of the skills) show significantly positive slopes ($\gamma_k > 0$). Skill *Support Subtraction 2,1* exhibits the largest learning rate with $\gamma_k = 0.836$ (SD $\sigma = 0.411$), while *Support Addition 1,1* possesses a large negative learning rate with $\gamma_k = -0.042$ (SD $\sigma = 0.001$). Therefore, the model predicts little or no learning for most of the skills.

The high intercepts β_k - the intercept of $\beta_k = 3.26$ for *Addition 1,1* is equivalent to an initial probability of 0.96 of being correct - suggest that some of the skills might be very easy for the students, which causes ceiling effects. We therefore tried to explain the flat learning curves with the following two hypotheses:

1. Students might not have improved in all skills present in the analysis, but only in skills they had problems with.
2. Due to ceiling effects, students might not have lowered their error rate, but became faster (smaller answer times) in answering the tasks.
3. The skill specification of the model might be suboptimal.

In our first hypothesis, we only assessed the key skills (see Def. 5.1) of the children. Fitting an AFM over the key skills of the children naturally prevents ceiling effects. Not unexpectedly, the fitted model exhibits a lower student variance with $\theta_p = 0.11$ (SD $\sigma = 0.33$) as all children tend to start with lower intercepts in their problem skills. The fitted fixed effects are both significant with $\beta_k = 0.33$ (SD $\sigma = 0.03$) and $\gamma_k = 0.006$ (SD $\sigma = 0.0004$). The significantly positive slope indicates that on average children improved with every opportunity they had to apply a key skill, which confirms our first hypothesis.

To investigate the second hypothesis, we fit an AFM over all addition and subtraction skills, but instead of using the answers of the children (correct/wrong), we used their answer times (in milliseconds). To fit this model, we adapted the AFM to apply a Poisson (instead of a logistic) regression. Letting $y_{pi} \in [0, \infty[$ denote the answer time of student p on item i, we obtain $y_{pi} \sim \text{Pois}(\lambda_{pi})$. The linear component λ_{pi} of the Poisson-AFM can then be formulated as

$$\log(\lambda_{pi}) = \theta_p + \sum_k q_{ik} \cdot (\beta_k + \gamma_k \cdot T_{pk}). \tag{5.1}$$

The fitted random effect θ_p amounts to 0.21 (SD $\sigma = 0.45$), the fixed effects β_k and γ_k are displayed in Tab. 5.2. In the case of the Poisson-AFM, negative slopes (γ_k) indicate that children got faster with each opportunity. For skill *Addition 1,1* children for example started with an average answer time of 8.69 seconds and improved to an average of 6.45 seconds after ten tasks. Of the 20 investigated skills, 15 show significantly negative slopes. The skill *Support Subtraction 2,2* exhibits a significantly positive slope with $\gamma_k = 0.0004$, however, the learning rate of this skill was positive ($\gamma_k = 0.006$) when applying the standard AFM (see Tab. 5.1). For *Addition 1,1 TC*, *Subtraction 1,1*, *Subtraction 2,1* and *Support Addition 2,1* children neither improved in learning rate (see Tab. 5.1) nor in answer times (see Tab. 5.2). Our second hypothesis therefore holds true for the majority of the investigated skills.

Table 5.1: Fixed effects β_k and γ_k of the AFM for the different skills S_k along with standard deviations (in brackets) and significance values. Only 35% of the skills show significantly positive learning rates γ_k.

Skill	$\boldsymbol{\beta_k}$ **(SD σ)**	$\boldsymbol{\gamma_k}$ **(SD σ)**
Support Addition 1,1	2.77 (0.13)***	-0.042 (0.001)***
Support Subtraction 1,1	3.21 (0.20)***	-0.007 (0.022)
Addition 1,1	3.26 (0.22)***	-0.012 (0.030)
Subtraction 1,1	3.11 (0.15)***	-0.020 (0.008)*
Support Addition 2,1	1.38 (0.34)***	-0.011 (0.009)
Support Addition 1,1 TC	1.20 (0.26)***	0.024 (0.011)*
Support Subtraction 2,1	-1.56 (1.45)	0.836 (0.411)*
Support Subtraction 1,1 TC	0.93 (0.11)***	0.009 (0.001)***
Addition 2,1	1.99 (0.11)***	0.006 (0.002)**
Addition 1,1 TC	1.93 (0.11)***	0.001 (0.004)
Subtraction 2,1	2.04 (0.10)***	-0.001 (0.001)
Subtraction 1,1 TC	1.65 (0.01)***	-0.007 (0.002)***
Addition 2,1 TC	1.42 (0.10)***	-0.002 (0.001)
Subtraction 2,1 TC	1.04 (0.10)***	0.002 (0.002)
Support Addition 2,2	2.20 (0.45)***	0.016 (0.018)
Support Subtraction 2,2	0.53 (0.17)**	0.006 (0.003)$^+$
Addition 2,2	1.54 (0.12)***	0.014 (0.004)**
Subtraction 2,2	0.97 (0.10)***	0.004 (0.002)*
Addition 2,2 TC	1.23 (0.10)***	-0.002 (0.003)
Subtraction 2,2 TC	0.17 (0.09)$^+$	-0.003 (0.002)*

$^+$ p <.1, * p < .05, ** p < .01, *** p < .001

Table 5.2: Fixed effects β_k and γ_k of the Poisson-AFM for the different skills S_k along with standard deviations (in brackets) and significance values. Negative rates γ_k indicate a decrease of answer times. Children significantly improved their answer times for 75% of the skills.

Skill	$\boldsymbol{\beta_k}$ **(SD σ)**	$\boldsymbol{\gamma_k}$ **(SD σ)**
Support Addition 1,1	9.40 (0.0393)***	-0.0230 (0.000037)***
Support Subtraction 1,1	9.05 (0.0393)***	-0.0054 (0.000057)***
Addition 1,1	9.07 (0.0393)***	-0.0298 (0.000079)***
Subtraction 1,1	8.93 (0.0393)***	0.0033 (0.000028)***
Support Addition 2,1	9.99 (0.0393)***	0.0007 (0.000025)***
Support Addition 1,1 TC	1.01 (0.0393)***	-0.0052 (0.000027)***
Support Subtraction 2,1	1.05 (0.0392)***	-0.2042 (0.000673)***
Support Subtraction 1,1 TC	1.03 (0.0392)***	-0.0030 (0.000003)***
Addition 2,1	9.34 (0.0393)***	-0.0002 (0.000007)***
Addition 1,1 TC	9.26 (0.0393)***	0.0000 (0.000013)
Subtraction 2,1	9.49 (0.0392)***	0.0018 (0.000007)***
Subtraction 1,1 TC	9.55 (0.0392)***	-0.0007 (0.000005)***
Addition 2,1 TC	9.85 (0.0392)***	-0.0019 (0.000005)***
Subtraction 2,1 TC	1.01 (0.0392)***	-0.0056 (0.000007)***
Support Addition 2,2	1.07 (0.0393)***	-0.0199 (0.000032)***
Support Subtraction 2,2	1.06 (0.0393)***	0.0004 (0.000009)***
Addition 2,2	9.73 (0.0393)***	-0.0042 (0.000013)***
Subtraction 2,2	1.00 (0.0393)***	-0.0033 (0.000006)***
Addition 2,2 TC	1.02 (0.0392)***	-0.0041 (0.000007)***
Subtraction 2,2 TC	1.04 (0.0392)***	-0.0043 (0.000005)***

$^{+}$ p <.1, * p < .05, ** p < .01, *** p < .001

The conducted analyses confirm our first hypothesis: Over the course of the training, children improved on their key skills. However, as the set of keys skills tends to be different for every child, we cannot make statements about the improvement on specific skills. Our second hypothesis was only partially confirmed: Some of the skills show neither positive learning rates nor decreasing answer times with increasing opportunity count. These flat learning curves do, however, not come unexpected. From previous work (Murray et al., 2013), we know that learning curves are prone to student attrition when applied to data generated by a mastery-learning algorithm such as Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994): As students are aligned by opportunity count, the right hand side of the learning curve fitted by an AFM is dominated by students, who require a large number of opportunities to master a skill, which might in turn lead to underestimation of learning rates γ_k. Before considering a revision of the skill model as suggested by the third hypothesis, we therefore investigated the use of alternate logistic regression models for validation. In the next sections, we introduce variations of logistic regression models and discuss their parameter estimations and prediction of learning in the data.

5.2 Alternative logistic regression models

Learning curves are averaged over many students. The AFM aligns the students by opportunity count. When applied to mastery-learning data, it may suffer from student attrition with increasing numbers of opportunities. We therefore propose alternative logistic regression models that adjust for mastery-based data and therefore reduce the introduced bias.

Learning Gain Model (LG). With the LG model, we introduce a new alternative to the AFM. The LG model avoids student attrition by aligning the students at their first sample (when they start the training) and at their last sample, i.e., when they end the training (independent of whether they mastered the skill or not). The LG model is a variation of the AFM (see Eq. (2.2)). Letting $y_{pi} \in \{0,1\}$ denote the response of student p on item i, we obtain $y_{pi} \sim \text{Binomial}(1, \pi_{pi})$. The linear component π_{pi} of the LG model is then defined as

$$\text{logit}(\pi_{pi}) = \theta_p + \sum_k q_{ik} \cdot (\beta_k + \gamma_k \cdot N_{pk}), \tag{5.2}$$

where the random effect θ_p denotes the student proficiency. The fixed effect β_k describes the difficulty and the fixed effect γ_k the learning rate of a skill S_k (knowledge component). q_{ik} is 1, if item i uses skill S_k and 0 otherwise. $N_{pk} \in [0,1]$ denotes the (normalized) number of practice opportunities student p had at skill S_k., i.e., we normalize over all opportunities student p had at skill S_k during the training. Rather than measuring the amount learned per opportunity, this model estimates the learning gain of the students over the course of the training.

Alternative logistic regression models. To adjust for mastery-based data, alternative ways to fitting the curves have been proposed in previous work (Murray et al., 2013) for BKT. In the following, we reformulate these suggestions and apply them to logistic regression models. The **Mastery-Aligned Model (MA)** can be formulated using Eq. (2.2) for the linear component of the AFM, but with a different definition of T_{pk}. For the MA model, we count backwards: T_{pk} is the number of opportunities student p had at skill S_k as seen from mastery. T_{pk} is 0 at mastery, 1 at one opportunity before mastery and so on. Thus, the MA model aligns students at mastery, which solves the problem of student attrition. A different way to deal with student attrition is to group students by the number of opportunities needed to first master a skill. The linear component of this **Disaggregated Model (DIS)** can be defined as follows:

$$\operatorname{logit}(\pi_{pi}) = \theta_p + \sum_{k,m} q_{ik} \cdot (\beta_{k,m} + \gamma_{k,m} \cdot T_{pk}), \tag{5.3}$$

where the difficulty $\beta_{k,m}$ and the learning rate $\gamma_{k,m}$ are fit by skill S_k and mastery group m. By combining the MA and the DIS models, the **Mastery-Aligned and Disaggregated Model (DISMA)** can be constructed. This model disaggregates students into groups based on the number of opportunities needed until mastering the skill and furthermore aligns the students at mastery.

All the newly introduced models are again generalized linear mixed models (GLMM) (Boeck, 2008) as the linear predictor π_{pi} contains random effects (for the students) in addition to the fixed effects (for the skills).

5.3 Comparison of model properties

To evaluate the different logistic regression models and their properties, and to assess which modeling technique should be used to quantify learning in mastery-based data sets, we conducted two experiments on the data set also used for the model validation via AFMs (described in Sec. 5.1). In a first experiment, we analyzed the parameter fit of different regression models and evaluated their performance in prediction of new items. Furthermore, we compared prediction accuracy of regression models to that of traditional BKT. We used all the samples until the children mastered a skill and predicted the outcome of the first re-test. In a second experiment, we evaluated the prediction accuracy of regression models as well as BKT when generalizing to new students. We fitted the model based on a subset of students and predicted the outcome for the rest of the students.

Prediction accuracy for both experiments was measured using the root mean squared error (RMSE), the accuracy (number of correctly predicted student successes/failures based on a threshold of 0.5) and the area under the ROC curve (AUC). Prediction accuracy was computed using bootstrap aggregation with re-sampling

($n = 200$) in the first experiment and a student-stratified 10-fold cross validation in the second experiment.

Fitting for the regression models was done in `R` using the `lme4` package. To be able to compare the parameter fit of the different models, we did not constrain γ_k to be greater than or equal to zero. Parameters for BKT were estimated by maximizing the likelihood of the observed data (see Sec. 2.4) using a Nelder Mead simplex optimization (Nelder and Mead, 1965). This minimization technique does not require the computation of gradients and is for example available in `fminsearch` of `Matlab`. The following constraints were imposed on the parameters: $p_g \leq 0.3$ and $p_s \leq 0.3$.

For our analyses, we used two versions of the data set described in Sec. 5.1. The first version (denoted as *Version 1* in the following) contains the samples of all children at the respective skills (original data set), and the second version (denoted as *Version 2* in the following) includes only children that mastered the respective skills. *Version 2* of the data set makes the inclusion of the MA and DISMA models possible. However, it excludes students not mastering a skill from the analysis, which leads to a more homogeneous data set, but due to the drop-out of many children with DD, also to a less interesting data set. While the original data set (*Version 1*) contains $36'350$ tasks, *Version 2* of the data set contains only $20'784$ tasks.

5.3.1 Analysis of the parameter fit

In this experiment, we investigated the parameter fit of three regression models on the data set *Version 1*: The AFM, the LG model and the DIS model. The three models obtain very different parameter estimations for the same data. As already observed in Sec. 5.1 (detailed results in Tab. 5.1), the AFM predicts learning (positive γ_k) for 35% of the skills. The LG model on the other hand fits positive learning rates γ_k for all skills and the DIS model obtains positive learning rates $\gamma_{k,m}$ for 92% of the cases. Hence, the models predict very different amounts of learning in the data and we therefore analyze their residuals and prediction accuracies in the following.

Residual analyses

All three models tend to overestimate the outcome for badly performing students and underestimate the outcome for well performing students. This finding is also visible in Fig. 5.1, which displays the mean residuals r with $r =$ *fitted outcome - true outcome* by estimated student proficiency θ_p. Furthermore, the residuals r are strongly correlated to student proficiency ($\rho_{AFM} = -0.9621$, $\rho_{LG} = -0.9612$, $\rho_{DIS} = -0.9532$). These results are as expected, because the models' predictions are averaged over all the students. While the residuals r are very similar for the AFM and the LG models, the DIS model exhibits less variance in student proficiency. As

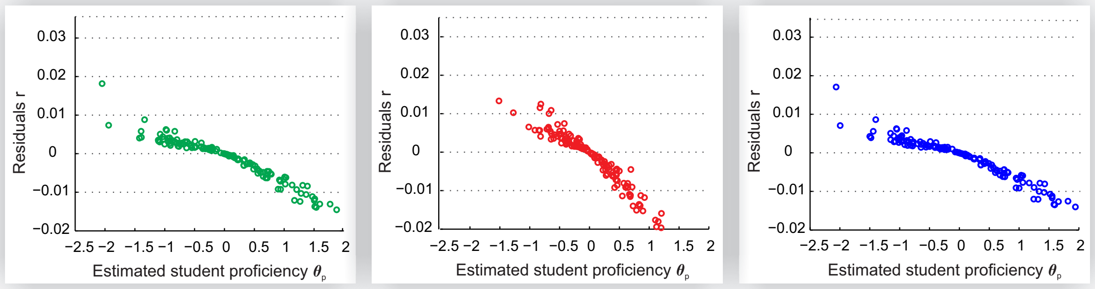
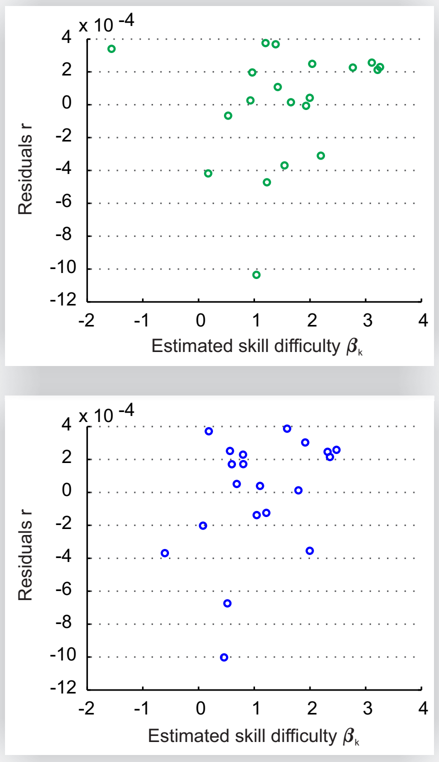


Figure 5.1: Mean residuals r by estimated student proficiency θ_p for the AFM (left), the DIS (center) and the LG (right) model. All three models tend to overestimate the outcome for weak performers and underestimate for well performing students.

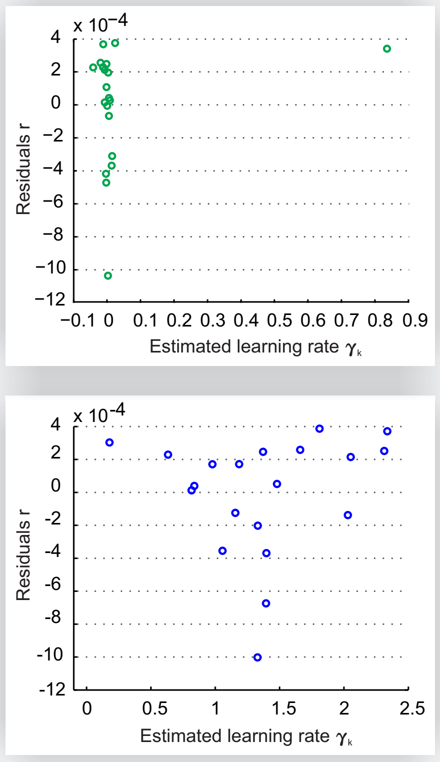
the students are grouped by the number of opportunities needed to master a skill, student proficiency within a group is more homogeneous.

For the AFM and the LG model, we also analyzed the mean residuals r regarding the skill parameters β_k and γ_k from the models. There are no significant correlations between skill difficulty β_k and mean residuals r, neither for the AFM ($\rho_{AFM} = 0.1677, p_{AFM} = .4798$) nor for the LG model ($\rho_{LG} = 0.3777$, $p_{AFM} = .1066$). From Fig. 5.2(a), which displays the mean residuals r by estimated skill difficulty β_k, it is also obvious that these measures are not related for both models. The residuals r are also not correlated to the estimated learning rate γ_k ($\rho_{AFM} = 0.2058$, $p_{AFM} = .3840$; $\rho_{LG} = 0.1051$, $p_{LG} = .6592$) as displayed in Fig. 5.2(b). Figure 5.2(b) demonstrates how different the parameter fits of the two models are regarding the learning rates γ_k. The AFM fits learning rates γ_k in a very small range around 0 and 45% of the learning rates are not significantly different from zero. The outlier stems from a skill played by only two students resulting in a total of 14 solved tasks for this skill. Learning rates γ_k fitted by the LG model are all positive and exhibit a larger variance.

The mean residuals r over time are displayed in Fig. 5.3. For the AFM and the DIS model, an averaging window ($n = 10$) was used to compute the mean residuals r with increasing opportunity count. Both models underestimate the outcome for less than 20 opportunities and overestimate it for larger numbers. For the AFM, this observation is confirmed by the significant positive correlation between the opportunity count and the mean residuals r ($\rho_{AFM} = 0.3950$, $p_{AFM} < .001$). This result probably stems from the fact that the well performing students master the skills much faster and therefore student numbers drop with higher opportunity counts. The DIS model exhibits a lower variance, as this model groups the students by the number of opportunities needed to master a skill and thus student performance within a group is more homogeneous ($\rho_{DIS} = 0.0860$, $p_{DIS} = .4785$). For the LG model, the mean residuals r are plotted by the normalized opportunity count in Fig. 5.3 (right). The LG model



(a) Mean residuals r by estimated skill difficulty β_k for the AFM (top) and the LG model (bottom).



(b) Mean residuals r by estimated learning rates γ_k for the AFM (top) and the LG model (bottom).

Figure 5.2: Mean residuals r by estimated skill difficulty β_k (a) and learning rates γ_k (b) for the AFM and LG models. For both models, residuals r are neither correlated to the skill difficulty β_k nor to the learning rates γ_k.

underestimates the outcome in the beginning and in the end and overestimates in-between. Through normalizing the opportunity count, we align the beginning and the end of the training for each student. We therefore end up with more observations from low performing students in the middle and the model overestimates the outcome in this part.

Re-test prediction

The residual analyses demonstrate that the models interpret the same data very differently, i.e., the parameter fit and properties of the models vary a lot. To validate these different parameter fits, we computed the prediction accuracy for the first re-test (data set *Version 1*) and compared it to a BKT model. The observed mean outcome

Table 5.3: Prediction accuracy of first re-test for data set *Version 1* and *2*. The values in brackets denote the standard deviation. The best model per error measure is marked bold. Performance of the AFM, LG and MA models is very similar, while the DIS and DISMA models exhibit lower prediction accuracy. The AFM, LG and MA models also outperform BKT regarding predictive performance.

		RMSE	**Accuracy**	**AUC**
Data set: Version 1	AFM	**0.3562 (0.0101)**	0.8391 (0.0119)	**0.6825 (0.0230)**
	LG	0.3587 (0.0125)	**0.8451 (0.0113)**	0.6778 (0.0250)
	DIS	0.3780 (0.0140)	0.8394 (0.0122)	0.6054 (0.0255)
	BKT	0.3614 (0.0111)	0.8428 (0.0118)	0.6033 (0.0250)
Data set: Version 2	AFM	**0.3563 (0.0114)**	**0.8474 (0.0123)**	**0.6622 (0.0250)**
	LG	0.3666 (0.0124)	0.8416 (0.0107)	0.6602 (0.0245)
	DIS	0.3765 (0.0141)	0.8416 (0.0120)	0.5998 (0.0290)
	MA	0.3633 (0.0117)	0.8401 (0.0114)	0.6508 (0.0255)
	DISMA	0.3783 (0.0133)	0.8396 (0.0116)	0.6011 (0.0256)
	BKT	0.3613 (0.0111)	0.8423 (0.0115)	0.6102 (0.0302)

over all re-tests is high with 0.8419. The AFM underestimates the true outcome with an average prediction of 0.8287, while the LG (average prediction 0.9108) and DIS models (average prediction 0.9488) overestimate the true outcome. Prediction accuracy for the different models is listed in Tab. 5.3. The AFM shows the best RMSE ($RMSE_{AFM} = 0.3562$) and AUC ($RMSE_{AUC} = 0.6825$), while the LG model exhibits the highest accuracy ($Accuracy_{LG} = 0.8451$). As the performance of students is generally high, RMSE and AUC are, however, better quality measures than accuracy. The LG model performs second best in RMSE ($RMSE_{LG} = 0.3587$) and AUC ($AUC_{LG} = 0.6778$). However, the small differences between the AFM and the LG model along with the high variances of the error measures indicate that there are no significant differences between the two models. The DIS model on the other hand demonstrates a considerably higher RMSE ($RMSE_{DIS} = 0.3780$) and also exhibits a low AUC ($AUC_{DIS} = 0.6054$) compared to the two other regression models. The DIS model estimates the parameters $\beta_{k,m}$ and $\gamma_{k,m}$ by skill and mastery group. The resulting large number of parameters produces overfitting. Performance on the training data set supports the overfitting hypothesis: The DIS model outperforms the AFM and the LG model in RMSE, accuracy and AUC on the training data set.

Interestingly, the AFM and the LG model also outperform the BKT model. The RMSE of BKT ($RMSE_{BKT} = 0.3614$) is higher than those of the two regression

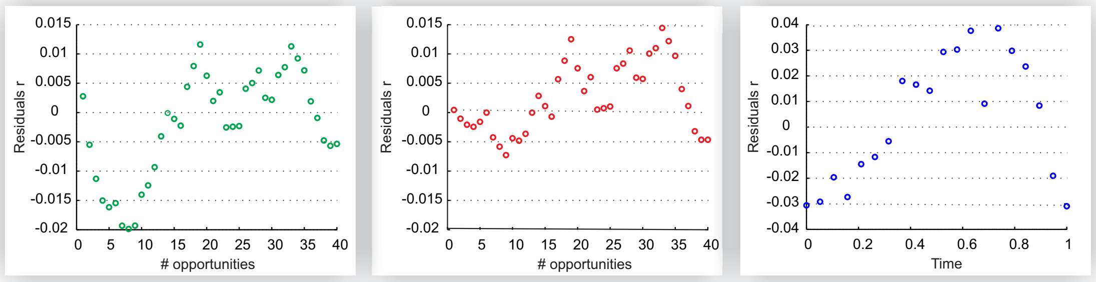


Figure 5.3: Mean residuals r by opportunity count for the AFM (left) and the DIS model (center) and by normalized opportunity count for the LG model (right). The AFM and DIS models underestimate the outcome for small opportunity counts and overestimate for larger numbers. The LG model underestimates the outcome in the beginning and in the end and overestimates in-between.

models, but standard deviations are again large. BKT exhibits especially a lower performance in AUC ($AUC_{BKT} = 0.6033$). The better performance of the regression models might come from two facts: First, the regression models fit the parameter θ_p for the individual student's proficiency, while traditional BKT does not do any student individualization. Second, BKT assumes that there is no forgetting, while the regression models are allowed to fit negative learning rates γ_k. However, the time between mastering a skill and the first re-test tends to be long. On average, the first re-test was done after 140 opportunities. A logistic regression analysis shows, that there is indeed a small, but significant amount of forgetting (intercept = 1.8545, slope = -0.0012) in the data. The probability of being correct at mastery amounts to 0.8647 and decreases to 0.8419 after 140 opportunities. Note, however, that the forgetting hypothesis is only valid for the AFM, as learning rates γ_k are all positive for the LG model.

Experiments on data set *Version 2*

To be able to include the MA and DISMA models in our analyses, we also evaluated prediction accuracy for the first re-test based on data set *Version 2*.

For this version of the data set, the LG and MA models predict positive learning rates γ_k for 100% of the skills, while the AFM fits positive learning rates γ_k for 54% of the skills. The DIS and DISMA models show positive learning rates $\gamma_{k,m}$ for 90% of the mastery groups. Residuals r of the DISMA model are very similar to those of the DIS model and we therefore only discuss the mean residuals r for the MA model. Figure 5.4 displays the mean residuals r by estimated student proficiency θ_p (left), skill difficulty β_k (center left), learning rates γ_k (center right) and opportunity

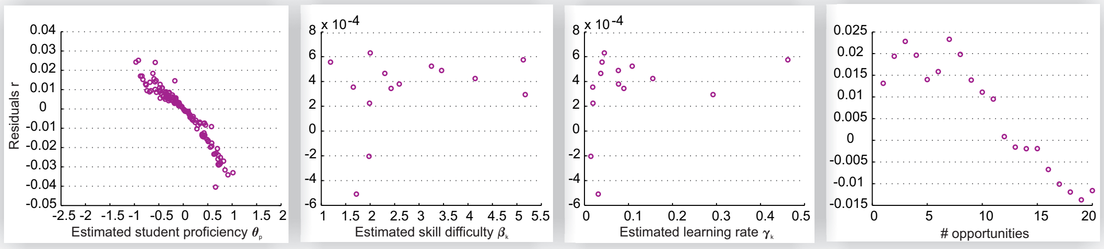


Figure 5.4: Mean residuals r by estimated student proficiency θ_p (left), skill difficulty β_k (center left), learning rates γ_k (center right) and opportunity count (right) for the MA model. The MA model tends to overestimate weak performers and underestimate well performing students. Residuals r are again uncorrelated to β_k and γ_k. The MA model overestimates the outcome in the beginning and underestimates it with increasing opportunity count.

count (right). Similarly to the other models, the MA model tends to overestimate the weaker students and underestimate the well performing students (see Fig. 5.4 (left)). The correlation between estimated student proficiency θ_p and mean residuals r is again strong ($\rho_{MA} = -0.9497$, $p_{MA} < .001$). As for the other models, mean residuals r are uncorrelated to skill difficulty β_k ($\rho_{MA} = 0.2916$, $p_{MA} = .3118$) and to learning rates γ_k ($\rho_{MA} = -0.2993$, $p_{MA} = .2986$). The MA model fits positive learning rates γ_k for all skills S_k (see Fig. 5.4 (center right)). To compute the mean residuals r by opportunity count, we again used an averaging window ($n = 10$). Unlike the other models, the MA model overestimates the outcome in the beginning and underestimates it with increasing opportunity count. This result is due to the mastery alignment of the model: As well performing students need less opportunities to master a skill, student attrition occurs in the beginning, where only weaker students remain in the analysis.

We again validated the parameter fit of the different models by predicting the first re-test and comparing prediction accuracy to BKT. Prediction accuracy for the different models is listed in Tab. 5.3. The AFM performs best for all error measures ($RMSE_{AFM} = 0.3563$, $AUC_{AFM} = 0.6622$). The performance of the LG model ($RMSE_{LG} = 0.3666$, $AUC_{LG} = 0.6602$) is again very close to that of the AFM. Interestingly, the MA model performs well in RMSE ($RMSE_{MA} = 0.3633$) and also exhibits a large AUC ($AUC_{MA} = 0.6508$). The high variances again indicate that differences between the AFM, the LG and the MA models are not significant. The DIS and DISMA models perform considerably worse in RMSE and AUC than the best three regression models. The performance of BKT is similar to the first version of the data set, with an RMSE ($RMSE_{BKT} = 0.3613$) in the range of the best regression models and a significantly lower AUC ($AUC_{BKT} = 0.6102$).

Table 5.4: Prediction accuracy of student-stratified cross-validation for data set *Version 1* and *2*. The values in brackets denote the standard deviations. The best model per error measure is marked bold. Interestingly, the AFM and LG models again outperform BKT regarding prediction accuracy.

		RMSE	**Accuracy**	**AUC**
Data set: Version 1	AFM	0.4200 (0.0184)	0.7525 (0.0300)	0.6693 (0.0222)
	LG	0**.4164 (0.0175)**	**0.7583 (0.0248)**	**0.6931 (0.0211)**
	BKT	0.4236 (0.0216)	0.7546 (0.0304)	0.6688 (0.0244)
Data set: Version 2	AFM	0.4008 (0.0247)	0.7850 (0.0296)	0.6755 (0.0335)
	LG	**0.3936 (0.0241)**	**0.7859 (0.0295)**	**0.7199 (0.0260)**
	BKT	0.4032 (0.0241)	0.7849 (0.0297)	0.6810 (0.0289)

5.3.2 Generalization to new students

In a second experiment, we investigated how well the different regression models generalize to new students using a student-stratified 10-fold cross validation. We also compared prediction accuracy of the logistic regression models to those of BKT, as this is an often used approach in student modeling. For new students (i.e., the students in the test set), the number of opportunities to mastery is not known, therefore only the AFM and the LG model were included in this analysis. Prediction accuracy along with standard deviations for the regression models as well as BKT is listed in Tab. 5.4. The LG model shows the best performance in all error measures for *Version 1* of the data set. The performance of the AFM is very close to that of the LG model in RMSE ($RMSE_{LG} = 0.4164$, $RMSE_{AFM} = 0.4200$). The high variance indicates that there are no significant differences between the two models regarding RMSE. The AUC of the LG model is, however, considerably higher than that of the AFM ($AUC_{LG} = 0.6931$, $AUC_{AFM} = 0.6693$).
Both regression models again outperform BKT in RMSE ($RMSE_{BKT} = 0.4236$) and AUC ($AUC_{BKT} = 0.6688$), but the high variance indicates that there are no significant differences in RMSE between all three models and also not in AUC between the AFM and the BKT model.
The results for *Version 2* of the data set show a similar picture. As expected, all models demonstrate a higher prediction accuracy for *Version 2* of the data set. As this version of the data set includes only students that mastered a skill, overall performance is more homogeneous and therefore prediction is easier.

5.4 Discussion

AFMs are widely used to analyze and improve student learning (Cen et al., 2007; Koedinger et al., 2013; Stamper and Koedinger, 2011). However, AFMs are prone to student attrition when applied to data from mastery learning: As students are aligned by opportunity count, the right hand side of the learning curve fitted by an AFM is dominated by students, who require a large number of opportunities to master a skill, which might in turn lead to underestimation of learning rates γ_k. Indeed, Murray et al. (2013) observed that averaging over different students with different initial knowledge states and learning rates may result in aggregated learning curves that appear to show little student learning, even though a mastery-learning student model such as BKT identified the students as mastering the skills at run time. This issue can be solved by using alternative models for fitting the learning curves (Murray et al., 2013). Our experiments on data from a mastery-learning student model (DBN skill model) with confirmed learning (significant improvement in external post-tests) support these results: AFM fitted positive learning rates γ_k for about half of the skills and only 70% of the positive γ_k were significantly different from zero. We therefore extensively evaluated alternative modeling techniques to analyze mastery-learning data. Indeed, alternative models such as the LG and MA models predicted positive learning for all skills and learning rates γ_k and generally showed a higher variance, i.e., learning rates differed from skill to skill. Our results demonstrate that different (although very similar) regression models explain the same data in a different way and that alternative regression models predict different patterns of learning. When applying AFM to mastery-learning data sets, flat learning curves might therefore not necessarily represent an insufficient skill model (which does not mean that the skill model cannot be improved), but suggest the use of an alternative modeling technique for validation of results.

Despite the different parameter fits, prediction accuracy of the regression models is very similar. When it comes to generalizing to new students, the LG model shows the most accurate prediction. However, as we observe a high variance in accuracy measures, there is most likely no significant difference in prediction accuracy between the AFM and the LG model. Although the AFM performs best in predicting the first re-test, the high variance of the error measures indicates that there is no significant difference between the AFM, the LG and the MA models. The disaggregated models (DIS, DISMA) perform significantly worse than the other regression models. As the disaggregation into different subpopulations increases the number of parameters, the lower performance of these models might be due to overfitting. This hypothesis is supported by the fact that the disaggregated models outperform the other regression models on the training data set in all error measures. Nonetheless, Murray et al. (2013) demonstrated the potential of disaggregated models. Prediction accuracy of these models should therefore be evaluated on larger data sets.

We also compared the regression models to BKT, an approach that is traditionally used for student modeling. BKT models are outperformed by most of the regression models when it comes to prediction accuracy on unseen data. The AFM and the LG model show a higher accuracy when predicting the first re-test, while the AFM, the LG and the MA model generalize better to new students than BKT. Although these differences are probably not significant (due to the high variance in the error measures), they are still interesting. One reason for this observation might be that BKT does not model forgetting. Our analyses have, however, shown that there is forgetting in the data. As the LG and MA models fit only positive learning rates γ_k, this explanation is only valid for the AFM model. Another reason for the superiority of the logistic regression models could be that traditional BKT does not have any student individualization. However, Yudelson et al. (2013) demonstrated on a different data set that a student individualized parameter p_0 (probability of knowing a skill a-priori) does not lead to significant improvements. The reason for the difference in prediction accuracy between BKT and logistic regression models therefore needs to be investigated further.

CHAPTER 6

Evaluation of student model and controller properties

In the previous chapter (Chapter 5), we performed a data-driven validation of the skill model of `Calcularis`. In this chapter, we extend this data-driven validation by assessing the quality of the student model and the control mechanism based on the input data collected in the user studies. Furthermore, we also use the collected data to understand properties of the users and the different mathematical skills. To conduct the analyses, we define important and desirable criteria for quality assessment:

1. *Effectiveness of the training program*: With no doubt, a learning program should be effective. We have already demonstrated that participants improve over the course of the training in external mathematical performance tests (detailed in Chapter 4). In this chapter, we show that participants demonstrated an increased mathematical performance over the training period within the system (see Sec. 6.1).

2. *Assessment of controller design*: The controller design of `Calcularis` allows forward and backward movements within the skill net. In Sec. 6.2, we show that this control mechanism significantly speeds up learning.

3. *Adaptability*: Another desirable property is the fast adaptation of the program to the mathematical knowledge of the user. In Sec. 6.3, we demonstrate that `Calcularis` adapts rapidly to the knowledge level of the children.

In the second part of this chapter (see Sec. 6.4), we analyze the performance of the users in the program as well as properties of skills. Such analyses can lead to a better understanding of the mathematical knowledge of the users.

All the analyses conducted in the following are based on the same data set. Input data consists of input logs recorded from 63 participants (71% females) of the *BMBF-study* and the *SWISS-study* (see Chapter 4). All children exhibited developmental dyscalculia (DD) or difficulties in learning mathematics. The log files contain six weeks of training with at minimum 24 complete sessions (of 20 minutes) per user. On average, each user completed 29.8 sessions (SD $\sigma = 2.4$). The total number of solved tasks was 1540 (SD $\sigma = 276$), while the number of solved tasks per session corresponded to 51.7 (SD $\sigma = 7.9$).

6.1 System-internal improvement analysis

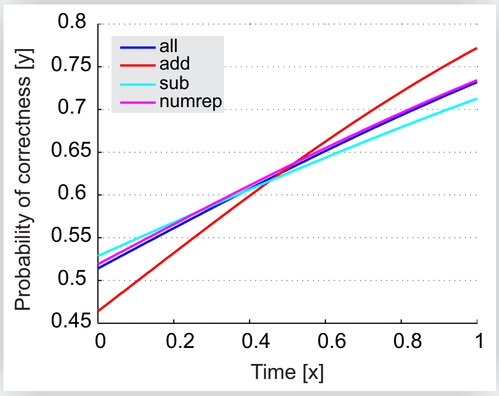
To quantify the improvement within the system, we conducted two different analyses, one analysis using the error rates of the children as a performance measure and a second analysis (only for the `Landing` game, see Sec. 3.2.2) assessing the spatial number representation of the children.

In the first analysis, we used the error rates of the children as a performance measure. We estimated the learning rate over the key skills $\mathcal{K}_U$ (see Def. 5.1) from all available samples (both if the participant mastered the key skills during training or not) by applying a variation of the Learning Gain model (LG) defined in Sec. 5.2. Letting $y_{pi} \in \{0,1\}$ denote the response of child p on a task i associated with a key skill S_k, we obtain $y_{pi} \sim \text{Binomial}(1, \pi_{pi})$. We adjust the LG model to only include key skills and formulate the linear component of the model as

$$\text{logit}(\pi_{pi}) = \theta_p + \sum_k q_{ik} \cdot (\beta + \gamma \cdot x_{pk}), \tag{6.1}$$

with $S_k \in \mathcal{K}_U$ for every child p. Furthermore, $\theta_p \sim \mathcal{N}(0, \sigma_\theta^2)$ represents the proficiency of child p, while $x_{pk} \in [0,1]$ denotes the (normalized) number of practice opportunities child p had at key skill S_k. q_{ik} is 1, if item i uses skill S_k and 0 otherwise. The fixed effect β describes the average key skill difficulty and the fixed effect γ the learning rate. We fit a model for all key skills. Furthermore, we also divide the key skills in categories C (*Addition*, *Subtraction*, *Number representations*) and fit a separate model per category. The resulting model over all key skills ($C = All$) exhibits an estimated mean improvement of 21.8% (95% confidence interval = $[0.21, 0.23]$). The learning curves for the different categories along with the exact coefficients for the fixed effects β and γ are displayed in Fig. 6.1.

In a second analysis, we investigated the accuracy of children in the `Landing` game (skills *Arabic*$\rightarrow$*Numberline*, *Verbal*$\rightarrow$*Numberline* and *Concrete*$\rightarrow$*Numberline* in the skill net illustrated in Fig. 3.4): In this game, we can not only measure the error rate (correct or wrong) but also the distance (in %) from the correct position to



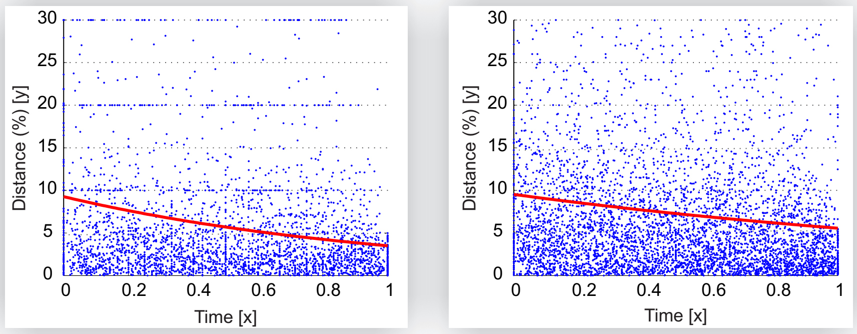
Category C	**Coefficient**	**Estimate (SD σ)**	**sig.**	**95% ci**
All	β	0.06 (0.05)	0.22	[-0.03 0.14]
	γ	0.95 (0.04)	$<$ 1e-4	[0.87 1.03]
Addition	β	-0.14 (0.09)	0.13	[-0.33 0.04]
	γ	1.36 (0.11)	$<$ 1e-4	[1.14 1.59]
Subtraction	β	0.11 (0.08)	0.15	[-0.04 0.27]
	γ	0.79 (0.07)	$<$ 1e-4	[0.66 0.93]
Number representations	β	0.08 (0.07)	0.26	[-0.06 0.21]
	γ	0.94 (0.06)	$<$ 1e-4	[0.82 1.06]

Figure 6.1: The percentage of correctly solved tasks (of key skills) increases over the training period by 21.8% for all skills (top). The normalized sample indices (Time[x]) are displayed on the x-axis, while the y-axis shows the ratio of correct solutions. Improvements for *addition* (add), *subtraction* (sub) and *number representations* (numrep) are in the same range. Exact coefficients for the fixed effects of the fitted model along with standard deviation (in brackets) are plotted by respective significance (sig.) and confidence intervals (ci) (bottom).

assess performance. We again employ a variation of the LG model to estimate the improvement of the children. Letting $y_{pi} \in [0, 100]$ denote the distance (from the correct position) child p achieved in a task i associated with a landing skill S_l, we obtain $y_{pi} \sim \text{Pois}(\lambda_{pi})$. We therefore define the linear component of the model as

$$\log(\lambda_{pi}) = \theta_p + \sum_l q_{il} \cdot (\beta + \gamma \cdot x_{pl}), \tag{6.2}$$

where $S_l \in \{Arabic{\rightarrow}Numberline,\ Verbal{\rightarrow}Numberline,\ Concrete{\rightarrow}Numberline\}$.



Number range	Coefficient	Estimate (SD σ)	sig.	95% ci
0-10	β	2.14 (0.06)	$< 1e-4$	[2.03 2.25]
	γ	-0.98 (0.02)	$< 1e-4$	[-1.02 -0.94]
0-100	β	2.25 (0.04)	$< 1e-4$	[2.17 2.34]
	γ	-0.54 (0.02)	$< 1e-4$	[-0.57 -0.51]

Figure 6.2: Landing accuracy in the number range 0-10 (left) and 0-100 (right) increases over time (top). The x-axis denotes the normalized sample indices (Time[x]), while the y-axis displays the deviance from the correct position. Exact coefficients of the fitted model along with standard deviation (in brackets) are plotted by respective significance (sig.) and confidence intervals (ci) (bottom).

Furthermore, $\theta_p \sim \mathcal{N}(0, \sigma_\theta^2)$ represents the proficiency of child p (in terms of a distance). $x_{pl} \in [0,1]$ denotes the normalized number of practice opportunities child p had at skill S_l. q_{il} is 1, if item i uses skill S_l and 0 otherwise. The fixed effect β denotes the initial average distance from the correct position, while γ describes the improvement in accuracy. We fit a separate model for the number ranges 0-10 and 0-100. The resulting models for the number ranges 0-10 and 0-100 along with the exact coefficients for the fixed effects are displayed in Fig. 6.2.

Over time, children achieved greater accuracy when giving the position of a number on a number line (Fig. 6.2 (top)). The significance of γ in both number ranges demonstrates the significant improvement in accuracy (Fig. 6.2 (bottom)).

The clear lines of points visible at 10%, 20% and 30% in Fig. 6.2 (top left) arise from the nature of the `Landing` game: Children need to indicate the position of a given number by steering a falling cone with the joystick. If the joystick is not moved, the cone will always land at the position of the five (in the number range 0-10), which leads to deviations of exactly 10% (if the given number was 4 or 6), 20% (if the given number was 3 or 7) or 30% (if the given number was 2 or 8).

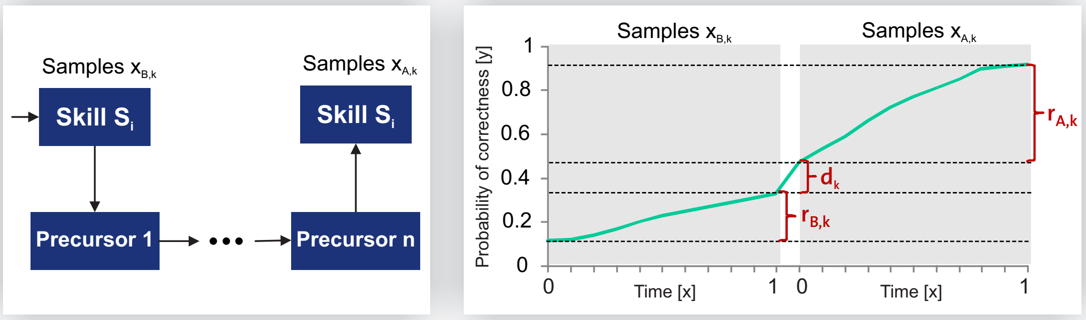


Figure 6.3: Illustration of a going back case (left): A user fails a skill S_i and has to rehearse one or several easier skills before coming back to train S_i. The direct improvement d_k (right) is the difference between the initial correct rate (at $x_{A,k} = 0$) after going back and the achieved correct rate (at $x_{B,k} = 1$) before going back. $r_{A,k}$ and $r_{B,k}$ (right) denote the learning rates before and after going back.

6.2 Assessment of control mechanism

Employing a dynamic Bayesian network (DBN) (Murphy, 2002) student model has also implications on the control mechanism: Due to the graph structure of the skill hierarchy, most skills have several precursor and successor skills. This structure implies that in order to master a skill S_a, all precursor skills of S_a need to be known and therefore it seems to be natural to allow forward (advancing to more difficult skills) and backward (going back to easier skills) movements of the controller.

In this analysis, we demonstrate that the possibility to go back to easier (played or unplayed) skills yields a significant beneficial effect. We show that the children not only immediately start reducing the rate of mistakes, but that they also learn faster. The log files recorded 973 individual cases of going back. On average, 20.6 cases (SD $\sigma = 12.1$) of going back are recorded per user. Figure 6.3 (left) illustrates the definition of a going back case k: All cases k in which users play a certain skill (samples $x_{B,k}$), go back to one or several easier skills, and finally pass them to come back to the current skill (samples $x_{A,k}$) are incorporated in the analysis. The variable $x_{B,k}$ therefore denotes all tasks before going back, while $x_{A,k}$ stands for the tasks solved after going back. We normalize $x_{A,k}$ and $x_{B,k}$ and therefore $x_{A,k} \in [0,1]$ and $x_{B,k} \in [0,1]$. From Fig. 6.4 it can be seen that the number of going back cases varies a lot among the users, *i.e.*, the users exhibit very different levels of mathematical knowledge.

For our analysis, we proceed as follows: Per each going back case k, we estimate the correct rate over time $c_{A,k}$ ($c_{B,k}$) separately for the samples before going back $x_{B,k}$ and the samples after going back $x_{A,k}$. Fitting is performed via logistic regression us-

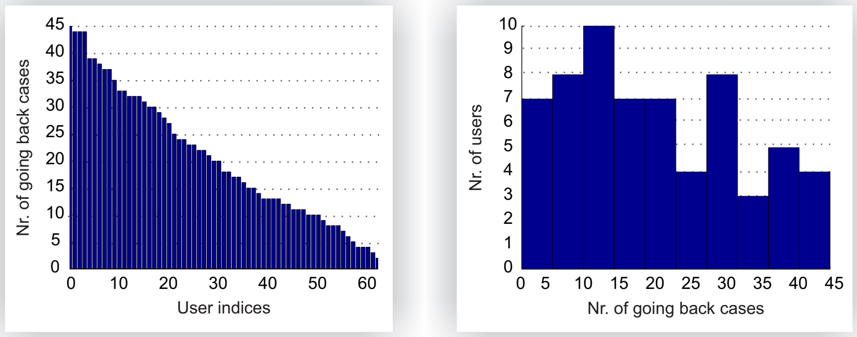


Figure 6.4: Number of going back times per user sorted in ascending order (left) and distribution over number of going back cases (right). The equal distribution of going back numbers demonstrates the heterogeneity of mathematical knowledge of the children.

ing bootstrap aggregation (Breiman, 1996) with resampling ($B = 200$). We therefore obtain learning curves for each going back case k and measure the following properties of the curves (illustrated in Fig. 6.3 (right)): The direct improvement d_k is the difference between the initial correct rate $c_{A,k}$ (at $x_A = 0$) after going back and the achieved correct rate $c_{B,k}$ (at $x_B = 1$) before going back. The improvement in learning rate r_k is the difference in learning rate over $c_{A,k}$ and $c_{B,k}$ (*i.e.*, $r_k = r_{A,k} - r_{B,k}$).

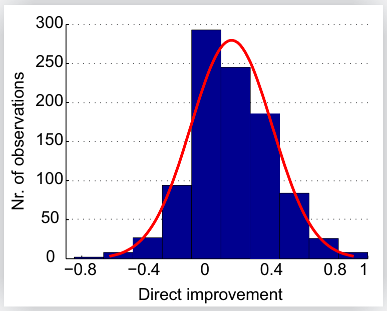
From Fig. 6.5 (top), we can see that the distributions over $\bar{d}$ (mean over d_k) and $\bar{r}$ (mean over r_k) are well approximated by normal distributions with means greater than 0. The rate of correct tasks $\bar{d}$ is increased by 0.14 while the learning rate $\bar{r}$ is even increased by 0.36 after going back. Both measurements are positive on average and a two-sided t-test indicates their statistically significant difference from 0 (statistics detailed in Fig. 6.5 (bottom)).

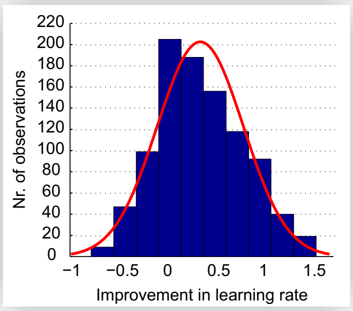
6.3 Controller adaptability

To provide an effective training, fast adaptation to the knowledge level of the student is important. At the beginning of the training period, all participants start with the lowest (easiest) skill of the skill net (*Subitizing*, see Fig. 3.4) and then advance through the skill net depending on their performance.

We therefore define the adaptation time $[t_0, t_{\mathcal{K}_U}]$ as the period between the start t_0 of the training and the first time the user hits one of his key skills $t_{\mathcal{K}_U}$ (see Def. 5.1). Thus, this analysis measures how long it took the user to arrive in a training area, where she or he exhibits difficulties.

On average, the participants reached their $t_{\mathcal{K}_U}$ after solving 148.3 tasks (SD $\sigma = 122.6$). The number of complete sessions played up to this point was 2.1 (SD





	Mean μ	sig.	99% ci of μ	SD σ	99% ci of σ
$\bar{\mathbf{d}}$	0.1411	<1e-3	[0.1248 0.1574]	0.2570	[0.2459 0.2690]
$\bar{\mathbf{r}}$	0.3618	<1e-3	[0.3330 0.3907]	0.4543	[0.4348 0.4756]

Figure 6.5: Distributions over direct improvement $\bar{d}$ (top left) and improvement in learning rate $\bar{r}$ (top right). Both measures are well approximated by a normal distribution with $\mu > 0$. Statistics for the improvement after going back (bottom): Mean improvement μ, significance of mean (sig.), standard deviation (σ), and confidence intervals (ci).

$\sigma = 1.97$). These results show that the model rapidly adjusts to the state of knowledge of the user. The fast adaptability is also confirmed by the fact that 52.4% of the children hit their first key skill already in the number range 0-10, 38.1% of children in the number range 0-100 and only 9.5% of the children in the number range 0-1000. The fast adaptation to the child's knowledge ensures that each child trains at the optimal difficulty level already after a few days of training.

6.4 Analysis of key skills

As children pursue different trajectories through the skill net during the training period, they tend to show various patterns of key skills. This variety is evidenced in Fig. 3.6: While user 2 has no key skills in the displayed number range, user 3 has one key skill (*Addition 2,2 TC*) and user 1 has three key skills (*Addition 2,1*, *Addition 2,2* and *Addition 2,2 TC*) in addition between 0-100. Despite this variety, some skills seem to be difficult for most of the children and thus more likely to be key skills: Nine skills were key skills for more than one third of the children. Of these skills, five were subtraction skills, four number representation skills and one an addition skill. Even more than 50% of the children had problems with the top three key skills: Indicating the position of a number on a number line from 0-100 (*Arabic*$\rightarrow$*Numberline* in Fig. 3.4) was difficult for 52% of the children. This result is

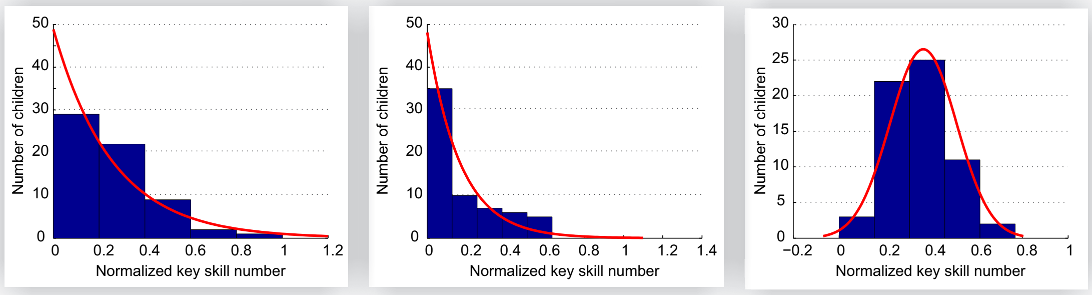


Figure 6.6: Distribution over normalized number of key skills for *number representations* skills (left), *addition* skills (center) and *subtraction* skills (right). For *number representations* and *addition* skills, normalized key skill numbers follow an exponential distribution. For *subtraction* skills, they follow a normal distribution.

in line with previous work, which observed deficits of mental number representation in children with DD (Kucian et al., 2006; Mussolin et al., 2010; Price et al., 2007). More than 50% of the children also had problems in subtraction with borrowing in the number range from 0-100 (*Subtraction 2,1 TC* and *Subtraction 2,2 TC* in Fig. 3.4). This result again confirms the link between subtraction and spatial number representation (Dehaene, 2011).

By definition, key skills demonstrate in which areas the users exhibit difficulties. Therefore, we can also assess the performance of the users within the system by their number of key skills. The normalized number of key skills is computed as the number of key skills divided by the number of totally played skills. On average, the normalized number of key skills per user was 0.27 (SD $\sigma = 0.14$). This number can be interpreted as follows: On average, the children had difficulties with 27% of the skills that they played. When breaking this number down into the different categories of the training program (*number representations*, *addition* and *subtraction*), it can be seen that most problems arose in *subtraction*. The normalized number of key skills was 0.26 (SD $\sigma = 0.19$) in *number representations*, 0.17 (SD $\sigma = 0.2$) in *addition* and 0.37 (SD $\sigma = 0.15$) in *subtraction*. The distributions over the normalized key skill numbers in the different categories are displayed in Fig. 6.6. Interestingly, we observe that the normalized key skill numbers for *addition* and *number representations* skills follow an exponential distribution. The long tail of the distribution demonstrates that most children did not have difficulties in these categories. Rather, only few children had strong difficulties in these categories. On the other hand, the normalized number of key skills in *subtraction* is significantly higher than in the two other categories (indicated by a two-sided t-test: $p < .001$ for both categories).

6.5 Discussion

In this chapter, we have assessed the quality of the student model and the control mechanism based on the log file data collected in the user studies (see Chapter 4). The analyses were conducted to demonstrate the effectiveness of the program and the controller design as well as its adaptability. Furthermore, we also analyzed properties of users and key skills.

In a first analysis, we estimated the learning rate of the children over their key skills $\mathcal{K}_U$. Our suggested logistic regression model normalizes the opportunity count over time (*i.e.*, start of the training $= 0$, end of the training $= 1$) and therefore measures the learning gain over the course of the training. The estimated mean improvement of 21.8% over $\mathcal{K}_U$ demonstrates that children were able to improve in areas where they had problems. Interestingly, subtraction exhibits a lower improvement than addition. Given the external training effects (detailed in Sec. 4.3), we would expect the opposite. However, children have a lot more subtraction key skills than addition key skills. Therefore, despite the average improvement per skill being higher for addition, the total improvement is still higher for subtraction. Furthermore, the higher number of key skills in subtraction leads to more practice in subtraction skills.

In a second analysis, we estimated the improvement in accuracy when positioning a number on a number line (skills *Arabic*$\rightarrow$*Numberline*, *Verbal*$\rightarrow$*Numberline* and *Concrete*$\rightarrow$*Numberline* in the skill net illustrated in Fig. 3.4). Children improved significantly in both number ranges, which demonstrates a refined spatial number representation. The improved number line representation is consistent with the significant improvement in subtraction, as subtraction is considered the main indicator for numerical understanding (Dehaene, 2011). Furthermore, this result also confirms previous studies (Siegler and Booth, 2004; Booth and Siegler, 2006, 2008; Halberda et al., 2008) which demonstrated significant correlations between arithmetic abilities and the quality of numerical magnitude representation.

In contrast to previous work (Corbett and Anderson, 1994; Beck and Sison, 2006; Koedinger et al., 1997) employing a linear skill hierarchy (such as Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994)), our DBN structure is non-linear and we therefore also allow backward movements (to easier skills) of the controller. Our analysis demonstrates that children reduce the rate of mistakes immediately after going back to an easier skill and also exhibit a higher learning rate. Our model therefore implicitly addresses forgetting and knowledge gaps.

Fast adaptability to the knowledge state of the user is important for effective teaching, as children tend to exhibit very different mathematical performance profiles (von Aster, 2000; Geary, 2004; Wilson and Dehaene, 2007). The student model of `Calcularis` is able to adapt to the knowledge state of the user within 2.1 sessions on average. Therefore, the users train in areas where they have problems already in the first week of the training.

The results of our key skills analyses demonstrate that users exhibit different mathematical problems and very different numbers of key skills. These results are in line with literature (von Aster, 2000; Geary, 2004; Wilson and Dehaene, 2007), which demonstrates that children show different mathematical performance profiles. Despite this variety, some skills are difficult for all children. They tend to have most difficulties in subtraction (as illustrated in Fig. 6.6), which again confirms the external training effects measured in the user studies (see Sec. 4.3).

CHAPTER 7

Latent structured prediction

A key feature of an intelligent tutoring system (ITS) is the adaptation of the learning content and the difficulty level to the individual student. The selection of problems is based on the estimation and prediction of the student's knowledge by the student model. Therefore, modeling and predicting student knowledge accurately is a fundamental task of an ITS.

Probabilistic models are widely used for representing, estimating and predicting student knowledge. One of the most popular approaches is Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994), a special case of a Hidden Markov Model (HMM) (Reye, 2004). As the prediction accuracy of a probabilistic model is dependent on its parameters, an important task when using BKT is parameter learning. Recently, the prediction accuracy of BKT models has been improved using clustering approaches (Pardos et al., 2012b) or individualization techniques, such as learning student- and skill-specific parameters (Pardos and Heffernan, 2010a; Wang and Heffernan, 2012; Yudelson et al., 2013) or modeling the parameters per school class (Wang and Beck, 2013).

In this chapter, we present a different approach for improving prediction accuracy of probabilistic models. We increase the representational power of the model by employing more complex dynamic Bayesian networks (DBN) (Murphy, 2002), representing the different skills of a learning domain as well as their dependencies jointly in one model. Utilizing models without a tree structure, however, imposes challenges for inference and learning. We therefore use constrained structured prediction with latent variables to learn the parameters of the network. The regularization via constraints naturally enforces model interpretability and we demonstrate, that the constraint setting also improves the prediction accuracy of the model. Furthermore, we compare the performance of the more complex hierarchical models to that of BKT on large-scale data sets from different learning domains.

7.1 Structured learning for data-driven education

When employing DBNs, we consider the different skills of a learning domain jointly within a single model. Student knowledge is represented using binary latent variables, *i.e.*, each variable represents knowledge about one specific skill (for example addition). We further assume that a skill can either be mastered by the student or not. The latent variables are updated based on the correctness of students' answers to questions that test the skill under investigation, hence observations are also binary. We also model the dependencies between the different skills, *e.g.*, two skills S_a and S_b are conditionally dependent, if S_a is a prerequisite for mastering S_b. In the following subsections, we describe the parameter learning task in detail.

7.1.1 Probabilistic Notation

The learning task of a DBN model can be described as follows: let the set of N variables of the model be denoted by $V = \{V_i \mid i \in \{1,\ldots,N\}\}$. In addition, consider an input space object $\mathcal{X}$ denoting the set of skills of the model and the corresponding task specific output space $\mathcal{Y}$ representing a sequence of answers. Furthermore, we let $\mathcal{H}$ denote the domain of the unobserved variables, *i.e.*, missing answers and the unobserved binary variables denoting whether a skill is mastered. Moreover, we let $\mathbf{y_m} = (y_{m,1},\ldots,y_{m,T})$ represent a sequence of T binary answers from the m-th student. As T is student dependent, *i.e.*, every student completes a different number of tasks during the training, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{H}$ also depend on the student. During learning, we are interested in finding the parameters θ that maximize the likelihood of the observed data, *i.e.*, the likelihood of a training set $\mathcal{D}$ consisting of $|\mathcal{D}|$ input- and output-space object pairs $(\mathbf{x_m}, \mathbf{y_m}) \in \mathcal{X}_m \times \mathcal{Y}_m$. The log likelihood of the model is then given by

$$L(\theta) = \sum_{(\mathbf{x_m},\mathbf{y_m})} \ln \left(\sum_{\mathbf{h_m}} p(\mathbf{y_m}, \mathbf{h_m} \mid \mathbf{x_m}, \theta) \right), \tag{7.1}$$

where we marginalize over the states of the latent variables $\mathbf{h_m}$ for student m. The joint probability $p(\mathbf{y_m}, \mathbf{h_m} \mid \mathbf{x_m}, \theta)$ of the model for student m is defined as

$$p(\mathbf{y_m}, \mathbf{h_m} \mid \mathbf{x_m}, \theta) = \prod_i p(V_{m,i} = v_{m,i} \mid pa(V_{m,i}) = \mathbf{v_{m,pa(V_{m,i})}}) = \prod_i p_{i j_{m,i} \mathbf{k_{m,i}}}, \tag{7.2}$$

where $pa(V_{m,i})$ are the parents of $V_{m,i}$, while $v_{m,i}$ and $\mathbf{v_{m,pa(V_{m,i})}}$ are the realizations of the random variables $V_{m,i}$ and $pa(V_{m,i})$, *i.e.*, the states assigned to $V_{m,i}$ and $pa(V_{m,i})$

given by $(\mathbf{y_m}, \mathbf{h_m})$. Furthermore, we let $j_{i,m} := v_{m,i}$ and $\mathbf{k_{m,i}} := \mathbf{v_{m,pa(V_{m,i})}}$ to simplify the notation. Therefore, $p_{i j_{m,i} \mathbf{k_{m,i}}}$ denotes exactly one entry in the conditional probability table (CPT) of $V_{m,i}$.

7.1.2 Log-linear formulation

The log-likelihood of a DBN can alternatively be formulated using a log-linear model. This formulation is flexible and predominantly used in recent literature (Lafferty et al., 2001; Schwing et al., 2012). Therefore, we reformulate the learning task in the following. Let $\phi : \mathcal{Y} \times \mathcal{H} \to \mathbb{R}^F$ denote a mapping from the latent space $\mathcal{H}$ and the observed space $\mathcal{Y}$ to an F-dimensional feature vector. The log likelihood from Eq. (7.1) can then be reformulated to

$$L(\mathbf{w}) = \sum_{(\mathbf{x_m},\mathbf{y_m})} \ln \left(\sum_{\mathbf{h_m}} \exp \left(\mathbf{w}^\top \phi(\mathbf{x_m}, \mathbf{y_m}, \mathbf{h_m}) - \ln(Z) \right) \right), \tag{7.3}$$

where Z is a normalizing constant and $\mathbf{w}$ denotes the weights of the model. Next, we show that this log-linear formulation of the log-likelihood is equivalent to the traditional probabilistic notation. Comparing Eq. (7.3) to Eq. (7.1), it follows that

$$\prod_i p_{i j_{m,i} \mathbf{k_{m,i}}} = \frac{1}{Z} \exp \mathbf{w}^\top \phi(\mathbf{x_m}, \mathbf{y_m}, \mathbf{h_m}) = \frac{1}{Z} \exp \sum_i w_i^\top \phi_i(\mathbf{x_m}, \mathbf{y_m}, \mathbf{h_m}), \tag{7.4}$$

and therefore

$$\forall i, j, \mathbf{k} : p_{ij\mathbf{k}} = \frac{1}{Z} \exp w_i^\top \phi_i(\mathbf{v}), \tag{7.5}$$

where $\mathbf{v}$ are the realizations of all random variables in V with $j \in \mathbf{v}$ and $\mathbf{k} \subset \mathbf{v}$. A feature vector ϕ and weights $\mathbf{w}$ that fulfill Eq. (7.5) can be specified as follows: consider the CPT describing the relationship between a node V_a and its $n-1$ parent nodes $pa(V_a)$. The CPT for these n nodes contains 2^n entries. Let $\mathbf{k} \in \{0,1\}^{n-1}$ denote one possible assignment of states to the parent nodes $pa(V_a)$. We can therefore define $p(V_a = 1 \mid pa(V_a) = \mathbf{k}) = 1 - p(V_a = 0 \mid pa(V_a) = \mathbf{k}) = 1 - p_{a,0,\mathbf{k}}$. To continue, let $p_{a,v_a,\mathbf{k}} = \frac{1}{Z} \exp w_{a,\mathbf{k}}(1 - 2v_a)$, which leads to the feature function $\phi_a(v) = 1 - 2v_a$ and normalization $Z = \exp w_{a,\mathbf{k}}(1-2v_a)/(\exp w_{a,\mathbf{k}}(1-2v_a) + \exp(-w_{a,\mathbf{k}}(1-2v_a)))$. The probabilities $p_i \in \theta$ are therefore proportional (in the log domain) to the weights $w_i \in \mathbf{w}$ and we can easily switch between the two notations. We obtain the joint distribution as a product of the exponential terms which translates to a weighted linear combination of feature vector entries in the exponent and thus fulfills Eq. (7.5). From this formulation also follows that we need 2^{n-1} parameters to specify a CPT including n skills.

7.1.3 Optimization

We subsequently solve the learning problem by optimizing the log-linear model of the data. Performing maximum likelihood we choose the weights $\mathbf{w}$ such that the model assigns highest probability to the training set $\mathcal{D}$. Note that for clarification of notation we neglect dependence of variables V and spaces $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{H}$ on the student m in the following. Furthermore, we will also explicitly indicate estimations, *i.e.*, $\hat{\mathbf{y}}$ and $\hat{\mathbf{h}}$ denote estimations for $\mathbf{y}$ and $\mathbf{h}$. We therefore reformulate Eq. (7.3) as

$$L(\mathbf{w}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \ln p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}), \tag{7.6}$$

with $p(\hat{\mathbf{y}} \mid \mathbf{x}, \mathbf{w}) \propto \sum_{\hat{\mathbf{h}}\in\mathcal{H}} \exp \mathbf{w}^\top \phi(\mathbf{x}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$. If the data is independent and identically distributed (i.i.d.), minimization of the negative log likelihood $-\ln[p(\mathbf{w}) \prod_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{w})]$ yields the following optimization

$$\min_{\mathbf{w}} \frac{C}{2} \|\mathbf{w}\|_2^2 - \sum_{\mathbf{y}\in\mathcal{D}} \ln p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}),$$

with a log-quadratic prior function $p(\mathbf{w})$.

Considering optimization of the aforementioned non-convex cost function we commonly follow the expectation maximization (EM) approach (Dempster et al., 1977) or more generally the concave convex procedure (CCCP) (Yuille and Rangarajan, 2003). We linearize the concave term by computing its gradient at the current iterate and subsequently minimize a convex objective. This step, identical to optimizing HMMs via EM, is guaranteed to converge to a stationary point (Sriperumbudur and Lanckriet, 2009).

But contrasting HMMs, neither linearization of the concave part nor minimization of the resulting convex objective is computationally tractable for general models. To our benefit and as indicated before and detailed below, the elements of the feature vector $\phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ typically decompose into functions depending only on a small fraction of variables. This can be employed to approximate the objective. Recently, Schwing et al. (2012) showed that a convex approximation admits more efficient learning of parameters than its non-convex counterpart. Note that interpretability of the parameters $\mathbf{w}$ is not guaranteed, particularly since guarantees exist for only converging to a local optimum. However, interpretability implies some form of expectation regarding the parameters. In the following, we therefore propose to constrain the parameter space. This is useful since domain experts are capable of restricting the range of acceptable parameters, *e.g.*, it is reasonable to assume the guess probability p_{guess} to be less than 0.3 (Corbett and Anderson, 1994; Yudelson et al., 2013).

7.2 Learning with constrained parameters

To formulate the constrained optimization, we let $\bar{\ell}(\mathbf{y},\mathbf{x},\mathbf{w}) = -\ln p(\mathbf{y} \mid \mathbf{x},\mathbf{w})$, *i.e.*, explicitly,

$$\bar{\ell}(\mathbf{x},\mathbf{y},\mathbf{w}) = \ln \sum_{\hat{\mathbf{y}},\hat{\mathbf{h}}} \exp \hat{\phi}(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}},\mathbf{w}) - \ln \sum_{\hat{\mathbf{h}} \in \mathcal{H}} \exp \hat{\phi}(\mathbf{x},\mathbf{y},\hat{\mathbf{h}},\mathbf{w})$$

while the potential is given as $\hat{\phi}(\mathbf{x},\mathbf{y},\mathbf{h},\mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x},\mathbf{y},\mathbf{h})$. Then we augment the learning task to read as the *constrained* optimization problem

$$\min_{\mathbf{w}} \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \bar{\ell}(\mathbf{x},\mathbf{y},\mathbf{w}) \quad \text{s.t.} \quad \mathbf{w} \in \mathcal{C}, \tag{7.7}$$

with $\mathcal{C}$ denoting a convex set. Leaving the constraint set aside, this program possesses the same difficulty as the original task, *i.e.*, we minimize a non-convex objective operating on exponentially sized sets. Being interested in the quality of duality based approximations, we subsequently follow Schwing et al. (2012).

We first note that an upper-bound to the program given in Eq. (7.7) is stated by the following cost function:

$$\frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{(\mathbf{x},\mathbf{y})} \left(\ln \sum_{\hat{\mathbf{y}},\hat{\mathbf{h}}} \exp(\hat{\phi}(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}},\mathbf{w})) - H(q_{(\mathbf{x},\mathbf{y})}) - \mathbb{E}_{q_{(\mathbf{x},\mathbf{y})}}[\hat{\phi}(\mathbf{x},\mathbf{y},\hat{\mathbf{h}},\mathbf{w})] \right), \tag{7.8}$$

with H denoting the entropy and $\mathbb{E}$ indicating computation of the expectation. Importantly, the upper bound allows dividing the program into two parts which are iterated alternating when following the CCCP procedure: on the one hand a minimization w.r.t. the distribution $q_{(\mathbf{x},\mathbf{y})}$ ranging over the latent space $\hat{\mathbf{h}} \in \mathcal{H}$ for every sample $(\mathbf{x},\mathbf{y})$. This task is often referred to as 'latent variable prediction task'. On the other hand a minimization w.r.t. the weight vector $\mathbf{w}$ subject to constraints $\mathcal{C}$. Both problems remain intractable without further modifications. However, we notice that minimization to find the distributions $q_{(\mathbf{x},\mathbf{y})}$ directly follows Schwing et al. (2012) and we can incorporate their approximation without further modification.

Due to the additional constraint set it is the second task which requires specific attention. The relevant excerpt from the linearized program given in Eq. (7.7) reads as follows:

$$\min_{\mathbf{w}\in\mathcal{C}} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \ln \sum_{\hat{\mathbf{y}},\hat{\mathbf{h}}} \exp \mathbf{w}^\top \phi(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}}) - \mathbf{w}^\top \mathbf{d} + \frac{C}{2}\|\mathbf{w}\|_2^2. \tag{7.9}$$

We note that the vector of empirical means $\mathbf{d} \in \mathbb{R}^F$ contains information from the observed variables as well as information from the linearization of the concave part.

```
Algorithm 1 (Structured Prediction with Constrained Parameter Spaces).
Let $\tilde{\phi}_{(\mathbf{x},\mathbf{y}),r}((\hat{\mathbf{y}},\hat{\mathbf{h}})_r) = \sum_{k:r\in\mathcal{R}_k} w_k\phi_{k,r}(\mathbf{x},(\hat{\mathbf{y}},\hat{\mathbf{h}})_r)$.

Repeat until convergence:

1. Update Lagrange multipliers: $\forall(\mathbf{x},\mathbf{y}),r,p\in P(r),(\mathbf{y},\mathbf{h})_r$
   $\mu_{(\mathbf{x},\mathbf{y}),p\to r}((\mathbf{y},\mathbf{h})_r) = \ln \sum_{(\mathbf{y},\mathbf{h})_p\setminus(\mathbf{y},\mathbf{h})_r} (\exp(\tilde{\phi}_{(\mathbf{x},\mathbf{y}),r}((\mathbf{y},\mathbf{h})_r) - \sum_{p'\in P(p)} \lambda_{(\mathbf{x},\mathbf{y}),p\to p'}((\mathbf{y},\mathbf{h})_{p'}) + \sum_{r'\in C(p)\setminus r} \lambda_{(\mathbf{x},\mathbf{y}),r'\to p}((\mathbf{y},\mathbf{h})_{r'})))$
   $\lambda_{(\mathbf{x},\mathbf{y}),r\to p}((\mathbf{y},\mathbf{h})_r) \propto \frac{1}{1+|P(r)|}\left(\hat{\phi}_{(\mathbf{x},\mathbf{y}),r}((\mathbf{y},\mathbf{h})_r) + \sum_{p'\in P(r)} \mu_{(\mathbf{x},\mathbf{y}),p'\to r}((\mathbf{y},\mathbf{h})_r)\right) - \mu_{(\mathbf{x},\mathbf{y}),p\to r}((\mathbf{y},\mathbf{h})_r)$

2. Perform a gradient step and project the result onto the constraint set $\mathcal{C}$:
   $\mathbf{w} \leftarrow P_{\mathcal{C}}[\mathbf{w} - \gamma\nabla_w f(\lambda,\mathbf{w})]$
```

Figure 7.1: An algorithm for learning parameters of structured models within constrained parameter spaces.

This task differs from the standard structured prediction program in an additional regularization w.r.t. the constraint set $\mathcal{C}$. Although assumed to be convex subsequently, this additional regularization makes the program more challenging to solve in general. We subsequently show the approximations required to obtain an efficient algorithm based on projected gradients. To this end, we first state the dual program of the task given in Eq. (7.9).

Claim 1. *The dual program of the constrained structured prediction task (see Eq.* (7.9)*) reads as*

$$\max_{p_{(\mathbf{x},\mathbf{y})}\in\Delta} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} H(p_{(\mathbf{x},\mathbf{y})}(\hat{\mathbf{y}},\hat{\mathbf{h}})) + \frac{C}{2}\|P_{\mathcal{C}}[z]\|_2^2 - Cz^\top P_{\mathcal{C}}[z],$$

where we maximize the entropy H of distributions $p_{(\mathbf{x},\mathbf{y})}$ restricted to the probability simplex $\Delta_{\mathcal{Y}\times\mathcal{H}}$ over the complete data space. The projection of $z = \frac{1}{C}\left(d - \sum_{(\mathbf{x},\mathbf{y}),\hat{\mathbf{y}},\hat{\mathbf{h}}} p_{(\mathbf{x},\mathbf{y})}(\hat{\mathbf{y}},\hat{\mathbf{h}})\phi(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}})\right)$ onto the constraint set $\mathcal{C}$ is denoted by $P_{\mathcal{C}}[z]$ and $d\in\mathbb{R}^F$ refers to the vector of empirical means.

Proof: We introduce a temporary variable $g(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}}) = \mathbf{w}^\top\phi(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}})$ to decouple the soft-max function from the norm minimization in Eq. (7.9). Optimizing w.r.t. both,

w and g, we obtain the entropy as the conjugate dual of the soft-max. Minimizing the norm subject to constraints yields the projection of the difference between the empirical means vector **d** and its estimate onto the constraint set $\mathcal{C}$. We note that $\mathcal{C} = \mathbb{R}^F$ yields the solution given by Hazan and Urtasun (2010), which concludes the proof. $\square$

The aforementioned summation over exponentially sized sets within the primal problem manifests itself in distributions $p_{(\mathbf{x},\mathbf{y})}$ over respective simplexes $\Delta_{\mathcal{Y}\times\mathcal{H}}$. Instead of working with a full joint distribution over the set of all possible solutions $\mathcal{Y}\times\mathcal{H}$, we operate with corresponding marginals $b_{(\mathbf{x},\mathbf{y})}$ for sample $(\mathbf{x},\mathbf{y})$ and respective marginalization constraints. The marginals are chosen according to the variable dependence structure introduced within the feature vector $\phi(\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}})$.

More formally, let the k-th element of the feature vector be given by $\phi_k(\mathbf{x},\mathbf{y},\mathbf{h}) = \sum_{r\in\mathcal{R}_k}\phi_{k,r}(\mathbf{x},(\mathbf{y},\mathbf{h})_r)$ where r specifies a restriction of the function to a subset of the observed and unobserved variables. The set of all restrictions for the k-th element of the feature vector is referred to via $\mathcal{R}_k$. All in all we therefore consider the marginals $b_{(\mathbf{x},\mathbf{y}),r}((\mathbf{y},\mathbf{h})_r)$ which are required to fulfill the marginalization constraints, *i.e.*, we enforce them to be consistent amongst each other. Importantly, this means that we neglect the exponential number of constraints within the marginal polytope by adopting its local approximation (Wainwright and Jordan, 2008). In addition to usage of marginals, we approximate the joint entropy $H(p_{(\mathbf{x},\mathbf{y})}) \approx \sum_r H(b_{(\mathbf{x},\mathbf{y}),r})$.

To obtain an approximated convex primal, we introduce Lagrange multipliers $\lambda_{(\mathbf{x},\mathbf{y}),r\to p}((\mathbf{y},\mathbf{h})_r)$ for each marginalization constraint that ties together two restrictions r and p. We obtain the approximated, convex and constrained primal as follows:

$$\min_{\mathbf{w}\in\mathcal{C},\lambda} \sum_{(\mathbf{x},\mathbf{y}),r} \ln \sum_{(\hat{\mathbf{y}},\hat{\mathbf{h}})_r} \exp \hat{\phi}_{(\mathbf{x},\mathbf{y}),r}((\hat{\mathbf{y}},\hat{\mathbf{h}})_r) - d^\top w + \frac{C}{2}\|\mathbf{w}\|_2^2, \tag{7.10}$$

where we denote the re-parameterized potential via

$$\begin{aligned}\hat{\phi}_{(\mathbf{x},\mathbf{y}),r}((\hat{\mathbf{y}},\hat{\mathbf{h}})_r) &= \sum_{k:r\in\mathcal{R}_k} w_k\phi_{k,r}((\mathbf{x},\hat{\mathbf{y}},\hat{\mathbf{h}})_r) + \sum_{p\in P(r)} \lambda_{r\to p}((\hat{\mathbf{y}},\hat{\mathbf{h}})_r)\\ &\quad - \sum_{c\in C(r)} \lambda_{c\to r}((\hat{\mathbf{y}},\hat{\mathbf{h}})_c),\end{aligned}$$

where $P(r)$ is the set containing the parent regions of r in the region graph, while $C(r)$ contains the child regions of r (Schwing et al., 2012). The Lagrange multipliers $\lambda_{r\to p}$ and $\lambda_{c\to r}$ denote the messages that are sent along the edges of the region graph (Schwing et al., 2012). The derivation follows Hazan and Urtasun (2010) and Schwing et al. (2012) and we recover the constraint set $\mathcal{C}$ by computing the dual for the projection $P_{\mathcal{C}}$. Intuitively we push energy λ between different restrictions such that we can find a weight vector **w** which minimizes the objective subject to $\mathcal{C}$.

Constrained structured prediction with latent variables.
Repeat until convergence:

1. Solve the approximate 'latent variable prediction' until convergence and update the empirical means $\mathbf{d}$.
2. Perform a single iteration of 'constrained structured prediction' as detailed in Fig. 7.1.

Figure 7.2: Algorithm for constrained structured prediction with latent variables.

This formulation differs from Hazan and Urtasun (2010) in that the domain for the parameters $\mathbf{w}$ is constrained by the convex set $\mathcal{C}$. We proceed by iterating between updates for the Lagrange multipliers λ and the model parameters $\mathbf{w}$ which guarantees convergence for the convex cost function. Note that optimization of the program given in Eq. (7.10) w.r.t. λ is unconstrained. Therefore we follow a block-coordinate descent scheme.

Let $f(w, \lambda)$ denote the cost function of the program given in Eq. (7.10). Fixing λ, f is a smooth, convex but non-linear function in $\mathbf{w}$ and a well-known method to address the constraint minimization of f w.r.t. $\mathbf{w}$ is the projected gradient algorithm (Rockafellar, 1970). We use the gradient of the smooth cost-function as a descent direction, perform a step and project the result onto the constraint set $\mathcal{C}$.

It is important to note that a single projection step is sufficient for convergence guarantees since block-coordinate descent methods only require to decrease the cost function at every iteration which is ensured after a single projection. We summarize this observation in the following claim.

Claim 2. *The algorithm outlined in Fig. 7.1 guarantees convergence of the constrained structured prediction program given in Eq.* (7.10).

Proof: Strong convexity admits block-coordinate descent updates (Tseng, 1993), *i.e.*, iterating between updates for weights $\mathbf{w}$ and Lagrange multipliers λ. The requirement of decreasing the cost function is met for the updates w.r.t. λ and also ensured by a single projection of $\mathbf{w}$ onto $\mathcal{C}$, which consequently proves the claim. $\square$

Combining the structured prediction algorithm outlined in Fig. 7.1 with the 'latent variable prediction task' we obtain the algorithm given in Fig. 7.2 which we will refer to as *constrained structured prediction with latent variables*.

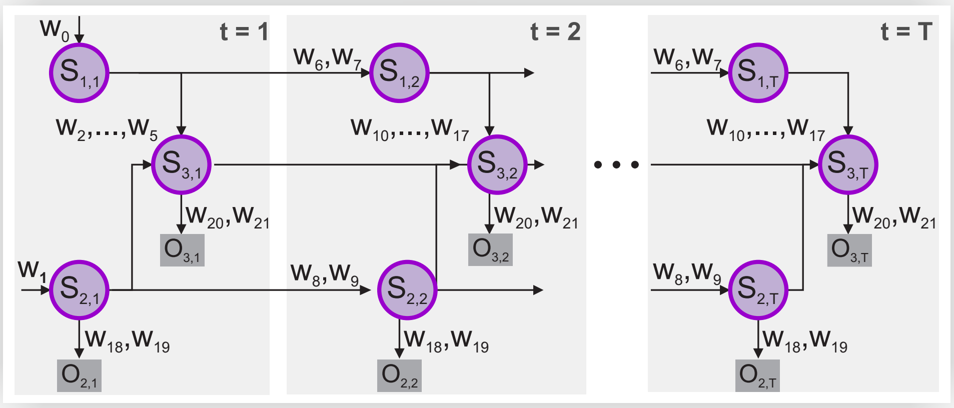


Figure 7.3: Structure of the graphical model for an example DBN with T time steps. Circular nodes represent skills, while the rectangles denote the tasks associated with those skills. The weights of the model are assumed to be stationary (time-invariant), *i.e.*, the parameters are shared over the different time slices.

7.3 Model specification and parametrization

DBNs can be specified and parametrized in different ways. In this section, we introduce the parametrization used for the experimental evaluation of our method (described in Sec. 7.4 and Sec. 7.5). Note, however, that the proposed algorithm is independent of the used parametrization. Therefore, the parametrization introduced in the following can be easily extended.

As in BKT, we can interpret the parameters of a DBN in terms of a learning context. Figure 7.3 illustrates the graphical model of a simple DBN with three skills S_1, S_2 and S_3 over T time steps. Two of the skills (S_2 and S_3) have associated tasks represented by gray rectangles, while skill S_1 cannot be observed. To specify the CPTs of this example DBN, we employ $F = 22$ weights that can be associated with a parameter set θ. We subsequently use $\simeq$ to denote proportionality in the log domain, *i.e.*, $w_i \simeq p_i$ is equivalent to $w_i \propto \exp p_i$.

Let O_3 denote the task associated with skill S_3. Then the parameters $w_{20} \simeq p(O_{3,t} = 0 \mid S_{3,t} = 0) = 1 - p_G$ and $w_{21} \simeq p(O_{3,t} = 0 \mid S_{3,t} = 1) = p_S$ represent the guess and slip probabilities. Similarly, w_{18} and w_{19} are associated with p_G and p_S as evident from Fig. 7.3. This association with guess and slip probabilities is important for choosing appropriate constraints: p_G and p_S have been constrained in previous work (Corbett and Anderson, 1994; Yudelson et al., 2013), a usual bound is $p_G \leq 0.3$ and $p_S \leq 0.3$. Furthermore, parameters $w_6 \simeq p(S_{1,t} = 0 \mid S_{1,t-1} = 0) = 1 - p_L$ and $w_7 \simeq p(S_{1,t} = 0 \mid S_{1,t-1} = 1) = p_F$

are associated with learning and forgetting; the same holds true for w_8 and w_9. It seems appropriate to limit these probabilities to be less than 0.5 - a forget probability $p_F > 0.5$ would lead to a model assuming a student that constantly forgets previously learned content.

Skills S_1 and S_2 are prerequisites for knowing skill S_3, *i.e.*, the probability that skill S_3 is mastered in time step t depends not only on the state of skill S_3 in the previous time step, but also on the states of S_1 and S_2 in the current time step. Therefore $w_{10} \simeq p(S_{3,t} = 0 \mid S_{3,t-1} = 0, S_{1,t} = 0, S_{2,t} = 0) = 1 - p_{L0}$, where p_{L0} denotes the probability that the student learns S_3 despite not knowing S_1 and S_2. We will again constrain p_{L0} as we assume an *AND* relationship for precursor skills in the model: In order to master S_3, S_1 and S_2 need to be known. The skill model of `Calcularis` (see Fig. 3.4) is also based on this assumption. In addition, $w_{17} \simeq p(S_{3,t} = 0 \mid S_{3,t-1} = 1, S_{1,t} = 1, S_{2,t} = 1) = p_{F1}$, the probability of forgetting a previously learned skill. We will constrain this probability for the reasons stated in the paragraph above.

Furthermore, we set $w_l \simeq 1 - p_{LM}$ if $l \in \{11, 12, 13\}$ and $w_l \simeq 1 - p_{FM}$ if $l \in \{14, 15, 16\}$, where p_{LM} denotes the probability that the student learns S_3 given that he knows at least one of the precursor skills of S_3. Moreover, p_{FM} is the probability that the student forgets the previously known skill S_3, when either S_1 or S_2 or none of them are known. Note that this parametrization is a simplification as it will allow us to set one bound for several weights.

Finally, the parameters w_l with $l \in \{2, 3, 4, 5\}$ describe the dependencies between the different skills. We let $w_l \simeq 1 - p_{P0}$, if $l \in \{2, 3, 4\}$ and $w_5 \simeq p_{P1}$, where p_{P0} is the probability of knowing a skill despite having mastered only part of the prerequisite skills and p_{P1} denotes the probability of failing a skill given that all precursor skills have been mastered already. This parametrization is again derived from the *AND* relationship assumed for precursor skills, as described above. Moreover, we refer to the probability of knowing a skill a-priori via p_0. Note that w_0 and w_1 are associated with p_0. The DBN illustrated in Fig. 7.3 can therefore be described by the parameter set $\theta = \{p_0, p_G, p_L, p_F, p_{L0}, p_{F1}, p_{LM}, p_{FM}, p_{P0}, p_{P1}\}$.

7.4 Evaluation of regularization

To assess the influence of the regularization with constraints on the prediction accuracy of the model, we evaluated our constrained approach in two (small) real data experiments. In particular, we compared the accuracy of our approach to that of an unconstrained setting. Furthermore, we also checked against a model using parameters chosen by experts: As the constraints are selected based on domain knowledge, we expect the expert parameters to be in the same range as the parameters learned by our approach.

For our experiments, we used log file data from 126 participants of the *BMBF-study*. From the 126 children (69% females), 57 were diagnosed as having developmental dyscalculia (DD) and 69 were control children (CC). On average children completed 28.9 sessions (SD $\sigma = 3.3$). The number of solved tasks was 1523 (SD $\sigma = 270$) and the number of solved tasks per session corresponded to 52.8 (SD $\sigma = 7.2$).

The prediction accuracy was computed as follows: given a set of observations for the DBN, we predicted the state of the unobserved nodes and provided the root mean squared error (RMSE), the classification error (CE, i.e., frequency of predicted state not equaling true state) and the area under the ROC curve (AUC). If not noted otherwise, convex learning stops when the improvement of the primal is less than 10^{-9} or the maximum number of iterations exceeds 500. In case of constraints the stopping criterion is met if the primal improves by less than $5 \cdot 10^{-6}$ or 300 iterations are exceeded. For inference, we limited the number of message passing iterations to 100.

7.4.1 Number understanding

In a first experiment, we looked at two skills taught in the number range from 0-100. Figure 7.4 illustrates the model, which is an extract of the skill model of `Calcularis` (illustrated in Fig. 3.4). Skill S_1 (*Ordinal 1* in Fig. 3.4) represents knowledge of the concept of ordinal number understanding, *i.e.*, understanding a number as a position in a sequence. There exists no exercise for this skill, hence no observations are available. The concept of relative number understanding is represented by skill S_2 (*Relative* in Fig. 3.4). Relative number understanding denotes the ability to understand a number as a difference between two numbers. We cannot directly observe this ability, but the results of an exercise associated with it. These results are referred to by rectangles which denote the outcome of a particular 'task'. For this experiment, we used a maximum of $T = 50$ time-steps (task outcomes) per child (mean: 22.16 (SD 9.98)). One child with no observations at skill S_2 was excluded from the analysis.

The model representing this task employs $F = 11$ parameters to specify the conditional probabilities that define the network illustrated in Fig. 7.4. The parametrization of the graphical model is performed as described in Sec. 7.3. Following this section, parameters w_9 and w_{10} are associated with the guess probability p_G and the slip probability p_s, which are commonly assumed to be lower than 0.3 (Corbett and Anderson, 1994; Yudelson et al., 2013). This upper bound translates to the constraints $w_9 \geq 0.4236$ and $w_{10} \leq -0.4236$. Furthermore, from Sec. 7.3 we know that parameters w_5 and w_8 are associated with learning and forgetting. We limit these

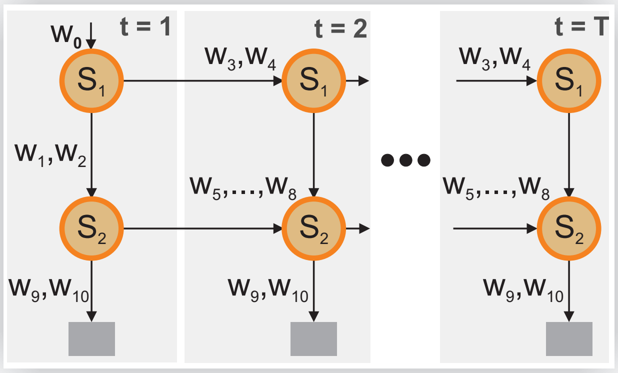


Figure 7.4: Structure of the graphical model used for the number understanding experiment for T time steps. Skill S_1 denotes knowledge about the ordinality of numbers (*Ordinal 1* in Fig. 3.4), while S_2 represents knowledge about the concept of relative number understanding (*Relative* in Fig. 3.4).

probabilities to be lower than 0.3, yielding $w_5 \geq 0.4236$ and $w_8 \leq -0.4236$. The aforementioned constraints define the set $\mathcal{C}_1$.

We refer to set $\mathcal{C}_2$ as the constraints within the set $\mathcal{C}_1$, augmented by the following restrictions. Since w_3 and w_4 are also related to learning and forgetting (see again Sec. 7.3), we utilize constraints identical to those for w_5 and w_8: $w_3 \geq 0.4236$ and $w_4 \leq -0.4236$. Similarly, we define $w_6 \geq 0.4236$ and $w_7 \leq -0.4236$. In addition, the hierarchical skill model of `Calcularis` assumes that the number understanding ability S_1 is a prerequisite for relative number understanding S_2 (von Aster and Shalev, 2007). Hence we restrict w_1 and w_2 by assuming that the probability of knowing S_2 given S_1 is larger than 0.7, while we let the probability of knowing S_2 despite not knowing S_1 be smaller than 0.3, which yields $w_1 \geq 0.4236$ and $w_2 \leq -0.4236$. Configurations $\mathcal{C}_3$ and $\mathcal{C}_4$ constrain the same parameters as $\mathcal{C}_1$ and $\mathcal{C}_2$, but are more restrictive by replacing 0.3 and 0.7 with 0.2 and 0.8.

After learning the model parameters using only the observed training data, prediction on the test data is performed as follows: we assume 'Task 1' to be given and predict the outcome of 'Task 2'. Afterward, we employ results from both 'Task 1' and 'Task 2' to predict the outcome of 'Task 3' and continue to predict 'Task k', $k \in \{4,5,\ldots,50\}$ assuming the preceding task outcomes to be given.

The performance results provided in Tab. 7.1 are computed using 10-fold cross validation. The most accurate results (per error measure) are marked in bold. We observe our constrained learning approach to outperform expert parameters as well as the unconstrained solution for most error metrics. Also the unconstrained optimization $\mathcal{C} = \emptyset$ yields good prediction results with the following parameter values:

	Expert	$\mathcal{C} = \emptyset$	$\mathcal{C} = \mathcal{C}_1$	$\mathcal{C} = \mathcal{C}_2$	$\mathcal{C} = \mathcal{C}_3$	$\mathcal{C} = \mathcal{C}_4$
RMSE	0.464	0.393	0.382	0.379	0.374	**0.373**
CE	0.346	0.213	0.213	0.213	**0.202**	**0.202**
AUC	**0.625**	0.500	0.606	0.591	0.615	0.615

Table 7.1: Different error measures for the number understanding experiment. Comparison of unconstrained and constrained conditions to previous work using a domain expert (Käser et al., 2012). The best model per error measure is marked in bold. The regularization through constraints improves prediction accuracy (as compared to $\mathcal{C} = \emptyset$).

$w_1, ..., w_8$ are set to 0, which results in uniform distributions for the according CPTs. The parameters w_9 and w_{10} are set to values smaller than -1 (over all folds). The model therefore predicts a correct outcome with a probability higher than 0.88, independent of previous observations and the state of the hidden nodes. As the investigated skill was easy to solve for most children, this model exhibits a high prediction accuracy. It is, however, not interpretable with respect to human learning. Note that expert parameters generally yield a good AUC, but exhibit a high RMSE and CE. This result is not unexpected, as the expert parameters are not fit to the training data.

7.4.2 Subtraction

The four skills investigated in this experiment are different subtraction skills in the number range from $0-100$. The graphical model, which is again an extract of the skill model used in `Calcularis` (see Fig. 3.4), is illustrated in Fig. 7.5. The notation of the different skills is explained in Tab. 3.1. Skill S_1 denotes a subtraction task without borrowing and a single-digit number as the subtrahend (*Subtraction 2,1*) while skill S_3 also represents a subtraction task without borrowing, but with a two-digit subtrahend (*Subtraction 2,2*). S_2 denotes subtraction with borrowing and a single-digit subtrahend (*Subtraction 2,1 TC*) and S_4 denotes the ability to do subtraction with borrowing and two two-digit numbers (*Subtraction 2,2 TC*). The rectangles denote results of an exercise associated with the skills S_2, S_3 and S_4. Again, we used a maximum of $T = 50$ time-steps (task outcomes) per child (mean: 43.59 (SD $\sigma = 10.47$)). To specify the conditional probabilities of the graphical model (Fig. 7.5), we employed $F = 33$ parameters.

The constrained configurations for this experiment follow the domain knowledge introduced in the first experiment. More specifically, $\mathcal{C}_1$ denotes the following constraints: $w_i \geq 0.4236$, $\forall i \in \{9, 11, 15, 19, 27, 29, 31\}$ while $w_i \leq -0.4236$,

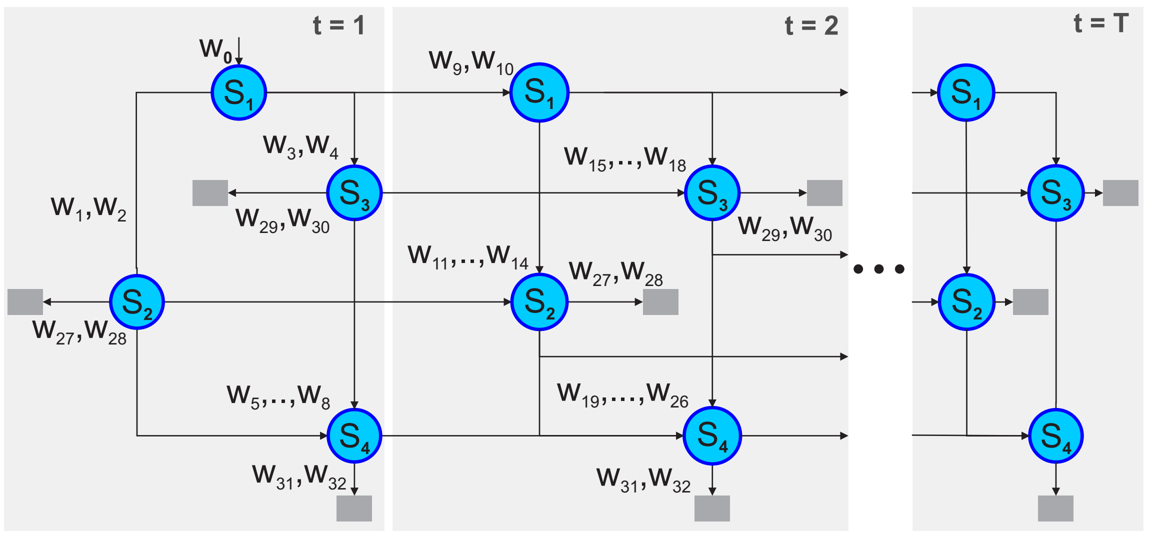


Figure 7.5: Structure of the graphical model over T time steps used for the subtraction experiment. Skill S_1 denotes skill *Subtraction 2,1*, S_2 represents *Subtraction 2,1 TC*, skill S_3 stands for *Subtraction 2,2* and S_4 describes skill *Subtraction 2,2 TC*. The model is an extract of the skill model illustrated in Fig. 3.4.

$\forall i \in \{10, 14, 18, 26, 28, 30, 32\}$. The second configuration $\mathcal{C}_2$ augments the set $\mathcal{C}_1$ by adding $w_i \geq 0.4236$, $\forall i \in \{1, 3, 5, 6, 7, 12, 16, 20, 21, 22\}$ and $w_i \leq -0.4236$, $\forall i \in \{2, 4, 8, 13, 17, 23, 24, 25\}$. Again, configurations $\mathcal{C}_3$ and $\mathcal{C}_4$ constrain the same parameters as $\mathcal{C}_1$ and $\mathcal{C}_2$, but are more restrictive by replacing 0.4236 and -0.4236 with 0.6913 and -0.6913.

Prediction was done as described in the first experiment and the performance results provided in Tab. 7.2 were again computed using 10-fold cross validation. We observe again significant improvements of our constraint approach compared to the expert parameters as well as the unconstrained setting in all error metrics. We highlight the improvement of the classification error by 5.9% when learning our computational education model within a constrained parameter space ($CE_{C_\emptyset} = 0.325$, $CE_{C_4} = 0.268$).

7.5 Comparison to non-hierachical models

In a second evaluation, we assessed the prediction accuracy of DBNs in comparison to non-hierarchical models, *i.e.*, BKT models representing only one skill. We performed this evaluation to demonstrate the benefits of models with higher representational power. We performed experiments on five data sets from various learning domains. The data sets were collected with different tutoring systems and contain data from elementary school students up to university students. We compare the

	Expert	$\mathcal{C} = \emptyset$	$\mathcal{C} = \mathcal{C}_1$	$\mathcal{C} = \mathcal{C}_2$	$\mathcal{C} = \mathcal{C}_3$	$\mathcal{C} = \mathcal{C}_4$
RMSE	0.489	0.469	0.453	0.436	0.446	**0.433**
CE	0.398	0.325	0.313	0.287	0.302	**0.268**
AUC	0.555	0.561	0.641	0.674	0.621	**0.682**

Table 7.2: Different error measures for subtraction. Comparison of different configurations to previous work using a domain expert (Käser et al., 2012). The best model per error measure is marked in bold. The constrained optimization outperforms the expert parameters as well as the unconstrained configuration regarding all error measures.

prediction accuracy of DBNs modeling skill topologies with the performance of traditional BKT models.

Fitting the BKT models was done using Yudelson et al. (2013), applying skill-specific parameters and using gradient descent for optimization. As described by Yudelson et al. (2013), we set the forget probability p_F to 0, while p_S and p_G were bounded by 0.3. In the following, we will denote this constrained BKT version as BKT$_C$.

We used constrained latent structured prediction (as described in Sec. 7.2) to learn the parameters of the DBN models. All models were parametrized according to Sec. 7.1 and we imposed the constraints described in the following on the parameter set θ of the different models to ensure interpretable parameters. For our first constraint set $\mathcal{C}_1$, we let $p_D \leq 0.3$ for $D \in \{G, S, L, F, L0, F1\}$ to ensure that parameters associated with guessing, slipping, learning and forgetting remain plausible. The constraints on θ can be directly turned into constraints on $\mathbf{w}$. For the example DBN (Fig. 7.3), the constraints translate into the following linear constraints on the weights for $\mathcal{C}_1$: $w_i \geq 0.4236$, if $i \in \{6, 8, 10, 18, 20\}$ and $w_i \leq -0.4236$, if $i \in \{7, 9, 17, 19, 21\}$. For the second constraint set $\mathcal{C}_2$, we augmented $\mathcal{C}_1$ by limiting $p_D \leq 0.3$ if $D \in \{LM, FM, P0, P1\}$, yielding $w_i \geq 0.4236$, if $i \in \{2, 3, 4, 11, 12, 13\}$ and $w_i \leq -0.4236$, if $i \in \{5, 14, 15, 16\}$ for the example DBN (Fig. 7.3). The additional constraints ensure that parameters are consistent with the hierarchy assumptions of the model. The constraint sets $\mathcal{C}_3$ and $\mathcal{C}_4$ bound the same parameters as $\mathcal{C}_1$ and $\mathcal{C}_2$, but are more restrictive by replacing 0.3 by 0.2. Note that constraints were selected according to Sec. 7.3.

Prediction was again performed as follows: we assumed the observation at time $t = 1$ to be given and predicted the outcome at time t with $t \in \{2, ..., T\}$ based on the previous $t - 1$ observations. The number of observations T for the different experiments is the minimal number of observations covering all skills of the according experiment. The exact number of observations T is given in the description of each experiment. To assess prediction accuracy, we again provide the following error measures: Root mean squared error (RMSE), classification error CE (ratio of incorrectly predicted student successes and failures based on a threshold of 0.5) and the area under the ROC curve (AUC). All error measures were calculated using cross-validation. Statistical significance was computed using a two-sided t-test, correcting for multiple comparisons (Bonferroni-Holm).

Note that we selected skills, where users showed low performance for our experiments, in order to make learning and prediction more challenging. In the following, we describe the DBNs for the five data sets and discuss the prediction accuracy for our models as well as for BKT$_C$.

7.5.1 Number representation

For the first experiment, we used data collected from `Calcularis`. The data set contains log files of 1581 children training with the product version of `Calcularis` with at least five sessions of 20 minutes per user. The log files of the product version were collected in an uncontrolled setting and therefore, demographic information about the children is not available. Furthermore, the training statistics of the users differ a lot. On average, children completed 20.7 sessions (SD $\sigma = 19.2$). The total number of solved tasks was 1395 (SD $\sigma = 1087$), while the number of solved tasks per session corresponded to 75.5 (SD $\sigma = 20.8$). The graphical model used in this experiment (see Fig. 7.3) is an excerpt of the skill model of `Calcularis` (illustrated in Fig. 3.4, see Tab. 3.1 for an explanation of the notation used). Skill S_1 (*Arabic*) represents knowledge about the Arabic notation system. `Calcularis` does not contain any tasks associated with this skill. The ability to assign a number to an interval is denoted by S_2 (*Ordinal 3*). The task associated with this skill is to guess a number in as few steps as possible. Finally, S_3 denotes the ability to indicate the position of a number on a number line (*Arabic*$\rightarrow$*Numberline*). We used a maximum of $T = 100$ observations per child for learning and prediction and specified the CPTs of the graphical model with $F = 22$ weights.

Prediction errors for the constraint sets $\mathcal{C}_1$ to $\mathcal{C}_4$ as well as BKT$_C$ are given in Tab. 7.3. The constrained DBN approach yields significant and large improvements in prediction accuracy compared to BKT$_C$. We highlight the improvement in accuracy by 11.4% (CE$_{BKT_C}$ = 0.3141, CE$_{\mathcal{C}_2}$ = 0.2783) and the reduction of the RMSE by 3.8% (RMSE$_{BKT_C}$ = 0.4550, RMSE$_{\mathcal{C}_4}$ = 0.4378). Also note the large improvement achieved in AUC (AUC$_{BKT_C}$ = 0.5975, AUC$_{\mathcal{C}_2}$ = 0.7093).

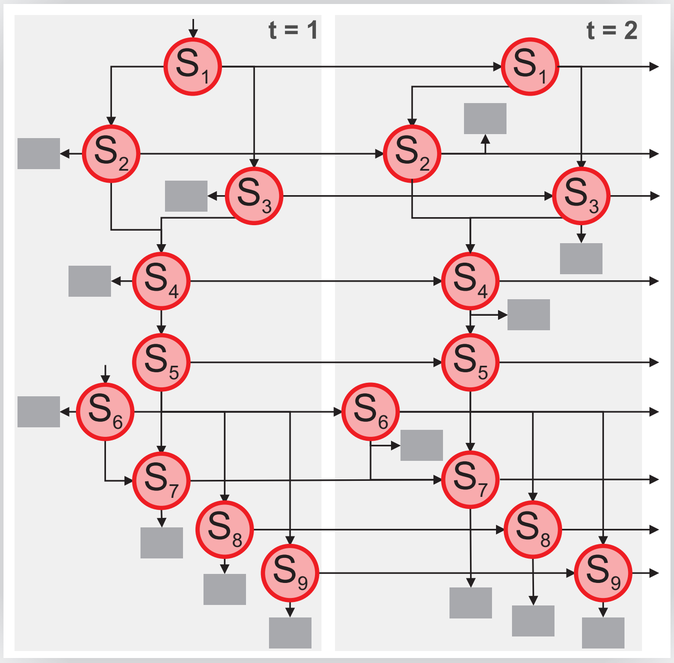


Figure 7.6: Graphical model for the subtraction experiment for the first two time steps. The model contains eight subtraction skills (S_1,...S_5 and S_7,...S_9) and one number representation skill (S_6) with associated tasks (denoted by rectangles). Two of the skills (S_1 and S_6) cannot be observed. The model is an excerpt of the skill model (see Fig. 3.4) of `Calcularis`.

.

7.5.2 Subtraction

The second experiment is based on the same data set as the first experiment (described in Sec. 7.5.1). This time, however, we investigated subtraction and number representation skills. The graphical model (see Fig. 7.6) is again an excerpt of the skill model of `Calcularis` (illustrated in Fig. 3.4, see Tab. 3.1 for an explanation of the notation used). Subtraction skills are ordered according to their difficulty, which is determined by the magnitude of involved numbers, task complexity and the means allowed to solve a task. Skills S_1 (*Subtraction 2,1*), S_2 (*Subtraction 2,1 TC*), S_3 (*Subtraction 2,2*), S_4 (*Subtraction 2,2 TC*) and S_5 (*Operation 2,2*) denote subtraction tasks in the number range $0-100$. We emphasize that there are no observation nodes associated with S_1 and S_5. The number representation skill S_6 (*Relative*) represents knowledge about the relational aspect of number (number as a difference between other numbers) in the number range $0-1000$. Finally, skills S_7 (*Support*

Table 7.3: Prediction accuracy of the experiments, comparing BKT_C with different constraint sets for the DBN models. The best result for each error measure is marked bold. Significant improvements compared to BKT_C are also indicated (*). Our hierarchical models outperform BKT_C over all data sets and in all error measures.

		$\mathbf{BKT_C}$	$\mathcal{C} = \mathcal{C}_1$	$\mathcal{C} = \mathcal{C}_2$	$\mathcal{C} = \mathcal{C}_3$	$\mathcal{C} = \mathcal{C}_4$
Number representation	RMSE	0.4550	0.4469*	0.4452*	0.4416*	**0.4378***
	CE	0.3141	0.3279	**0.2783***	0.3079	0.2831*
	AUC	0.5975	0.7072*	**0.7093***	0.7087*	0.7049*
Subtraction	RMSE	0.4368	0.4417	**0.4215***	0.4389	0.4216*
	CE	0.2818	0.2812	0.2588*	0.2757*	**0.2580***
	AUC	0.5996	0.6157*	0.6870*	0.6332*	**0.6916***
Physics	RMSE	0.4530	0.4521	0.4272*	0.4465*	**0.4244***
	CE	0.2930	0.2893*	0.2677*	0.2870*	**0.2616***
	AUC	0.5039	0.6511*	0.6971*	0.6795*	**0.7007***
Algebra	RMSE	0.3379	0.3335*	**0.3254***	0.3321*	0.3267*
	CE	0.1461	0.1466	0.1392*	0.1466	**0.1379***
	AUC	0.5991	0.6682*	0.7004*	0.6718*	**0.7007***
Spelling	RMSE	0.4504	0.4521	0.4495*	0.4492*	**0.4472***
	CE	0.2898	0.2893	0.2914	**0.2882**	0.2906
	AUC	0.5029	0.5695*	0.5771*	0.5735*	**0.5804***

* $p < .05$

Subtraction 3,1), S_8 (*Subtraction 3,1*) and S_9 (*Subtraction 3,1 TC*) represent subtraction in the number range 0-1000. A maximum of $T = 100$ observations per child was used for learning and prediction. Specification of the CPTs for the model requires $F = 86$ weights.

The resulting prediction accuracy for this experiment (see Tab. 7.3) again demonstrates that the DBN model outperforms BKT_C. With a reduction of the RMSE by 3.5% ($RMSE_{BKT_C}$ = 0.4368, $RMSE_{\mathcal{C}_2}$ = 0.4215) and an increase of the accuracy by 8.4% (CE_{BKT_C} = 0.2818, $CE_{\mathcal{C}_4}$ = 0.2580), improvements confirm the results observed in the first experiment. Also the growth in AUC (AUC_{BKT_C} = 0.5996, $AUC_{\mathcal{C}_4}$ = 0.6916) is again substantial.

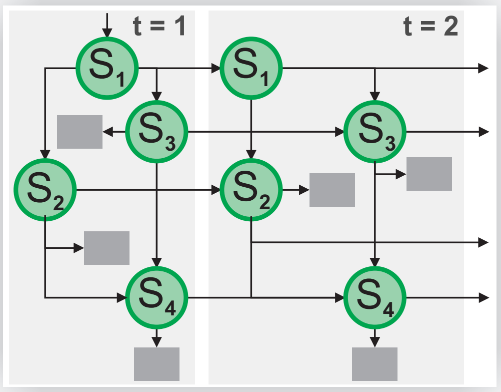


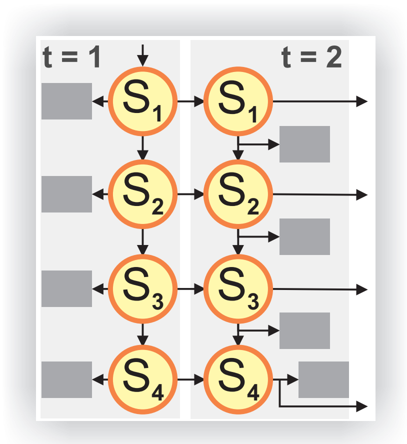
Figure 7.7: Graphical model for the physics experiment for the first two time steps. The model consists of four modules: *Vectors* (S_1), *Translational Kinematics* (S_2), *Statistics* (S_3) and *Dynamics* (S_4). The rectangles represent the tasks associated with the modules.

7.5.3 Physics

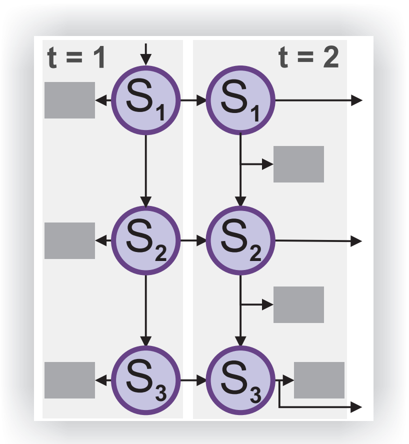
This experiment is based on the 'USNA Physics Fall 2005' data set accessed via DataShop (Koedinger et al., 2010). Data originate from 77 students of the United States Naval Academy and were collected from `Andes2`, an ITS for physics (Conati et al., 2002). The tutor uses rule-based algorithms to build solution graphs that identify all possible solutions of the different tasks. Based on these graphs, a Bayesian network is constructed to assess the general physics knowledge of the student as well as the progress for the problem at hand.

We used the different modules of the data set as skills for our experiment. The graphical model is depicted in Fig. 7.7. Note that we intentionally used a simplified skill model to avoid introducing incorrect assumptions and to assess if even non-experts can exploit skill structures using our proposed methods. The model consists of the following modules: *Vectors* (S_1), *Translational Kinematics* (S_2), *Statistics* (S_3) and *Dynamics* (S_4). These modules consist of more complex tasks for the given topic, *i.e.*, calculating total forces in a system (see example by Conati et al. (2002)). A maximum of $T = 500$ observations per student were considered for learning and prediction and the model is described by $F = 33$ weights.

In this experiment, the benefits of the DBN model are again high (see Tab. 7.3): the accuracy is increased by 10.7% ($\text{CE}_{BKT_C} = 0.2930$, $\text{CE}_{\mathcal{C}_4} = 0.2616$) while the RMSE is reduced by 6.3% ($\text{RMSE}_{BKT_C} = 0.4530$, $\text{RMSE}_{\mathcal{C}_4} = 0.4244$) and the AUC grows to 0.7007 ($\text{AUC}_{BKT_C} = 0.5039$).



(a) Graphical model for algebra experiment over the first two time steps.



(b) Graphical model for spelling experiment over the first two time steps.

Figure 7.8: Graphical models for the algebra (a) and spelling (b) experiments. The algebra model includes four skills dealing with word problems involving calculations with whole numbers. The spelling model consists of three modules containing increasingly difficult words.

7.5.4 Algebra

For this experiment, we used data from the KDD Cup 2010 Educational Data Mining Challenge (*http://pslcdatashop.web.cmu.edu/KDDCup*). The data set contains log files of 6043 students that were collected by the `Cognitive Tutor` (Koedinger et al., 1997), an ITS for mathematics learning. The student model applied in this system is based on BKT.

We used the units of the 'Bridge to Algebra' course as skills for our experiment and selected four units of increasing difficulty, where students have to solve word problems involving calculations with whole numbers. The graphical model for this experiment is illustrated in Fig. 7.8(a). Skill S_1 (*e.g.*, 728624 – 701312) denotes written addition and subtraction tasks without carrying/borrowing, while S_2 involves carrying/borrowing (*e.g.*, 728624 – 703303). S_3 (*e.g.*, 33564×18) and S_4 (*e.g.*, $10810 \div 46$) represent long multiplications and divisions. Note that the skill model is again simplified for the reasons explained in the Physics experiment. We used a maximum of $T = 500$ observations per student for learning and prediction and specified the CPTs of the model employing $F = 29$ weights.

Similarly to the previous experiments, the DBN model significantly outperforms BKT$_C$ (see Tab. 7.3). The RMSE is reduced by 3.7% (RMSE$_{BKT_C}$ = 0.3379, RMSE$_{\mathcal{C}_2}$ = 0.3254), while accuracy is increased by 5.6% (CE$_{BKT_C}$ = 0.1461, CE$_{\mathcal{C}_4}$ =

0.1379) and the AUC increases to 0.7007 (AUC$_{BKT_C}$ = 0.5991). Note that DBN and BKT$_C$ both perform better than in the other experiments as the high performance of students in the involved skills makes learning and prediction easier.

7.5.5 Spelling learning

The last experiment uses data collected from `Dybuster`, an ITS for elementary school children with dyslexia (Gross and Vögeli, 2007). The data set at hand contains data of 7265 German-speaking children. `Dybuster` groups the words of a language into hierarchically ordered modules with respect to their frequency of occurrence in the language corpus as well as a word difficulty measure. The latter is computed based on the word length, the number of dyslexic pitfalls and the number of silent letters contained in the word.

We used these modules as skills to build our graphical model (see Fig. 7.8(b)). Skills S_1, S_2 and S_3 denote the modules 3, 4 and 5 within `Dybuster`. Word examples are ‘warum’ (‘why’, S_1), ‘Donnerstag’ (‘Thursday’, S_2) and ‘Klapperschlange’ (‘rattlesnake’, S_3). We use a maximum of $T = 200$ observations per child for the learning and prediction tasks and parametrized the model using $F = 21$ weights.

While the DBN model still significantly outperforms BKT$_C$ in this experiment (see Tab. 7.3), the magnitudes of improvement are small: the RMSE is reduced by 0.7% (RMSE$_{BKT_C}$ = 0.4504, RMSE$_{\mathcal{C}_4}$ = 0.4472), the highest AUC amounts to 0.5804 (AUC$_{BKT_C}$ = 0.5029) and there is no significant improvement in CE.

7.6 Discussion

The goal of this work in *short-time prediction*, *i.e.*, predicting the outcome of task $t+1$ given the outcome of the t previous tasks, was to provide an efficient method for parameter learning that yields accurate prediction, while keeping parameters interpretable. We have solved this task by introducing an algorithm called *constrained structured prediction with latent variables* (described in Sec. 7.2).

The results of the first experiments (detailed in Sec. 7.4) demonstrate that introducing domain knowledge in the form of parameter constraints has a two-fold benefit. On one hand, the introduced parameter constraints guarantee an interpretable model. On the other hand, the proposed restrictions lead to improvement of the error metrics. Introducing restrictions on the parameter space is particularly beneficial for more complex models as well as for more difficult skills. For difficult skills where children change from the unlearnt to the learnt state after some training time, the unconstrained optimization converges to a solution closed to a uniform distribution (of correct and wrong outcomes), while the introduced domain knowledge enables more precise modeling of learning.

The results of the second evaluation (detailed in Sec. 7.5) demonstrate that more complex DBN models outperform BKT in prediction accuracy. For hierarchical learning domains, CE can be reduced by 10%, while improvements of RMSE by 5% are feasible. The DBN models generally exhibit a significantly higher AUC than BKT, which indicates that they are better at discriminating failures from successes. As expected, adding skill topologies has a much smaller benefit for learning domains that are less hierarchical in nature (such as spelling learning). The results obtained on the physics and algebra data sets show that even simple hierarchical models improve prediction accuracy significantly. A domain expert employing a more detailed skill topology and more complex constraint sets could probably obtain an even higher accuracy on these data sets. The use of the same parametrization and constraint sets for all experiments demonstrates that basic assumptions about learning hold across different learning domains and thus the approach is easy to use.

CHAPTER 8

Cluster-based prediction

In a computer-based therapy system, knowledge of performance profile, knowledge gaps and learning behaviors of the student as well as an accurate performance prediction are essential to improve diagnostics and intervention outcome. This is particularly important for students suffering from learning disabilities as the heterogeneity of these children requires a high grade of individualization. In Chapter 7, we have improved what we will call the *short-term prediction*: Using our latent structured prediction algorithm, we improved the accuracy when predicting the outcome of task $t+1$ given the outcome of the t previous tasks. In this chapter, we aim at improving the *long-term prediction* of the system: We try to predict external assessment results as well as learning characteristics of the students such as knowledge gaps and overall training achievement.

Given the high diversity of students using a computer-based training system, clustering approaches have proven to be useful to detect small and homogeneous groups of learners. In fact, amongst others, clustering approaches have been employed to improve *short-term prediction* accuracy. The precision of Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994) can be increased using clustering (Pardos et al., 2012b) and multiple classification models can also improve performance prediction within a system (Gong et al., 2012). Moreover, clustering (Trivedi et al., 2011) and co-clustering (Trivedi et al., 2012) approaches successfully improved post-test score predictions. Furthermore, ensemble methods offer a way to increase prediction accuracy by training different types of student models (Baker et al., 2011; Pardos et al., 2012a). Clustering can also be used to gain insight on learning characteristics of the students. Bootstrap aggregated clustering (King et al., 2007) identified different subtypes of children with dyslexia. Other authors used offline clustering followed by online classification to analyze and predict the students' input behaviors (Amershi and Conati, 2009; Kardan and Conati, 2011).

Our model uses online and offline cluster information for *long-term prediction*. The approach is articulated in three steps: In a first step, we cluster children according to individual learning trajectories. Compared to previous approaches, we use the subgroup information not only to improve prediction accuracy, but also to provide a valuable tool for experts to analyze individual learning patterns. The second step consists of a supervised online classification during training and in the third step, we predict future performance based on cluster assignment.

In the following, we first specify the three steps in detail before presenting the results of our experimental evaluation on a data set consisting of log files from the *BMBF-study* (described in Chapter 4).

8.1 Clustering, classification and prediction

The three steps of our approach are clustering (offline), classification (online) and prediction (online). To be able to perform the first two steps, we need to extract and process the features for clustering and classification, *i.e.*, features which are able to identify subgroups with similar mathematical patterns. In the following, we therefore first specify the extracted features as well as the feature processing pipeline used for clustering and classification. We then explain the clustering and classification steps in detail. Finally, we explain how performance prediction can be done based on cluster information.

8.1.1 Feature extraction and processing

From the log files, we identified a set of recorded features, which describe local and global properties of the user's training performance. The set contains cumulative as well as per skill measures, and covers performance, error behavior and timing. Table 8.1 lists the features, which are evaluated after each training session.

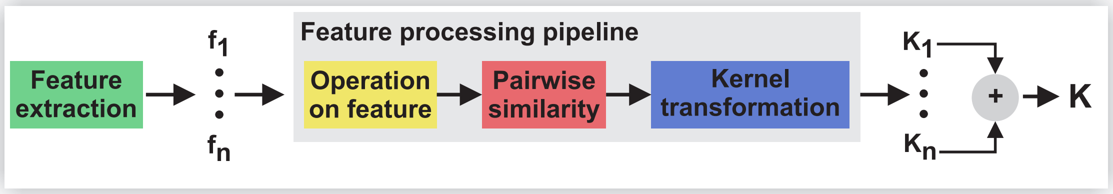
Having continuous and discrete feature types as well as different scales, we process the features to make them comparable. The processing pipeline illustrated in Fig. 8.1 (top) is used for the clustering as well as the classification features. Depending on their nature, features are processed before calculating pairwise similarities s_{ij} (between each pair of samples (students) i and j). The resulting similarity matrices $\mathbf{S_i}$ are transformed into a Kernel and summed up to obtain the similarity matrix $\mathbf{K}$. Finally, $\mathbf{K}$ is transformed to a distance matrix $\mathbf{D}$ using a constant shift ($\mathbf{D}$ = #features - $\mathbf{K}$).

Table 8.1: Extracted features and abbreviations (bold) used in the following. The feature set covers performance, error behavior and timing of the users and contains cumulative as well as per skill measures. *Part A* and *Part B* are consistent with Fig. 3.4.

Feature	Description
Highest **S**kills	Indices of highest skills for *Part A* and *Part B*.
Number of **P**assed **S**kills	Total number of skills passed.
Played **S**kills	Indices of played skills for *Part A* and *Part B*. Set feature.
Pass **T**imes	Accumulated time (from start of training) in seconds until passing a skill. Not passed skills are set to ∞.
Samples per **S**kill	Number of samples needed to pass a skill. Not passed skills are set to ∞.
Key **S**kills[a]	Indices of problem skills. Set feature.
Answer **T**imes	Mean answer time per skill. Not played skills are set to ∞.
Performance **P**er **S**kill	Mean performance (correct trials/all trials) per skill. Not played skills are set to 0.

[a] Key skill *S*: If a user went back to a precursor skill at least once before passing *S* (see Def. 5.1).

The employed processing modules are listed in Fig. 8.1 (bottom). For the *Operation* step (colored yellow in Fig. 8.1), we have three different pre-processing operations available: The logarithm (**LogInv**) naturally deals with outliers, while taking the inverse (**Inv**, **LogInv**) removes the ∞ values. The Beta cumulative distribution function (**Beta**) is applied to performance features: The range of this feature type is limited to the interval $[0,1]$. We hypothesize that performance differences between children are larger near the boundaries, *i.e.*, that it is more difficult to improve the ratio of correctly solved tasks from 0.8 to 0.9 than from 0.5 to 0.6. We also use three different *Similarity Measures* (colored red in Fig. 8.1): The L1-norm (**L1**) computes the absolute distance between two features and is applied for performance or time features. The Jaccard index (**JC**) is commonly applied to compute the similarity between two sets. And **SD** denotes the shortest path between two skills s_A and s_B on the skill net (illustrated in Fig. 3.4): The shortest path between skill *Arabic*$\rightarrow$*Concrete* and *Verbal*$\rightarrow$*Numberline* is for example **SD** $= 3$ as three edges of the graph need to be traversed to reach *Verbal*$\rightarrow$*Numberline*. In a third step, we apply *Kernel transformations* (colored blue in Fig. 8.1) to the features. **JK** is used for sets: It is invariant under set sizes and ensures that transformed data points have unit length. The standard Gaussian Kernel (**GK**) is shift invariant, its sensitivity can be influenced by



Operation		*Similarity measures*		*Kernel transformations*	
Inv:	$1/\mathbf{f}$	**JC:**	$(\mathbf{F_i} \cap \mathbf{F_j})/(\mathbf{F_i} \cup \mathbf{F_j})$	**JK:**	$2^S - 1$
LogInv:	$1/log(\mathbf{f})$	**L1:**	$\|\|\mathbf{f_i} - \mathbf{f_j}\|\|_1$	**GK:**	$exp(-S^2/(2 \cdot \sigma^2))$
Beta:	$betacdf(\mathbf{f}, .5, .5)^a$	**SD:**	$min(\mathbf{f_i} - \mathbf{f_j})^b$	**RK:**	$exp(-S/\sigma)$

[a] Cumulative distribution function of Beta distribution with $\alpha, \beta = 0.5$.
[b] Shortest path between skills on the skill net .

Figure 8.1: Feature processing pipeline (top) and processing modules employed on feature **f** (**F** in case of a set feature) (bottom). Features are pre-processed (yellow) before computing pairwise similarities (red). The resulting similarity matrices are transformed into a Kernel (blue). The different processing modules can be combined arbitrarily.

γ. The **RK** is also an exponential kernel, but more sensitive than the **GK**, which is useful to capture small differences in for example performance. The modules of the different steps of the processing pipeline can be combined arbitrarily.

8.1.2 Offline clustering

An inherent property of the controller design of `Calcularis` is its adaptability. Rather than following a specified sequence of skills to the goal, learning paths are individually adapted for each child. Form and maxima of the network paths vary depending on the learning characteristics of a student (see Fig. 8.5). These variations suggest that clustering the children on the basis of their trajectories identifies subgroups of children with similar mathematical learning profiles. Furthermore, the use of the trajectory features allows for modeling the development of mathematical learning over time. Clustering is performed offline, *i.e.*, taking into account all training sessions of the children.

Children are clustered at the end of the training using trajectory features. These features take into consideration how far the children came during the training (and how fast they arrived there) as well how they reached this point. The selected features are **PT** evaluated per part and number range (6 dimensions: A_{10}, B_{10}, A_{100}, B_{100}, A_{1000}, B_{1000}) and **PS** (set features for *Part A* and *Part B*). *Part A* and *Part B* are consistent with Fig. 3.4 and therefore correspond to the domains of *number* *representations* (*Part A*) and *arithmetic operations* (*Part B*). **PT** is processed using **LogInv** $\rightarrow$ **L1** $\rightarrow$ **GK** which yields the similarity matrix $\mathbf{K_1}$. For **PT**, we apply the inverse of the logarithm of the feature (**LogInv**) as a pre-processing step. The logarithm naturally removes outliers, while taking the inverse removes the ∞ values that occur as **PT** is set to ∞ for not passed (mastered) skills. As we are dealing with a time feature, we use the L1-norm **(L1**) as a distance measure and apply a Gaussian Kernel (**GK**) to obtain $\mathbf{K_1}$. The pipeline **JC** $\rightarrow$ **JK** used for **PS** results in $\mathbf{K_2}$ and $\mathbf{K_3}$. As we are dealing with a set feature (**PS** contains skill indices) we apply the similarity measure and kernel transformation defined for sets: **JC** and **JK**. The combined similarity matrix **K** ($\mathbf{K} = \mathbf{K_1} + \mathbf{K_2} + \mathbf{K_3}$) is finally transformed to the distance matrix **D** (**D** = 3 - **K**) used for clustering.

As the measurements are characterized by relations, *i.e.*, they represent dissimilarities between each pair of students i and j, we perform pairwise-clustering (PC) (Hofmann and Buhmann, 1997) on **D**. Through a kernel transformation, dissimilarity values can be interpreted as distances between points in a (usually higher-dimensional) Euclidean space. As shown by the Constant Shift Embedding transformation, PC exhibits a cost which is equivalent to that of K-means in the Euclidean embedding of the similarity data (Roth et al., 2003). The optimal number of clusters k^* can be determined by the Bayesian Information Criterion (BIC) (Pelleg and Moore, 2000), calculating the effective number of parameters as the normalized trace of the kernel transformation matrix (Haghir Chehreghani et al., 2012).

8.1.3 Online classification

We classify students after each training session and use the according cluster information for performance prediction. The tracked data allows assigning a student to the cluster of children showing similar knowledge and learning patterns. The similarities shared with other students are useful to predict the training performance of the subject, either within the tutoring system or by external assessments.

The features used for clustering represent global measures and are thus not optimized for early classification. As all children start the training at the lowest skill level (A_{10}), their trajectories tend to be similar during early training and do not provide information about future performance. This fact is also visible in Fig. 8.5: The trajectories of the two users look similar regarding their forms and maxima for about the first 400 tasks. Therefore, we use additional features (detailed in Tab. 8.1) taking into account local differences. While **HS**, **NPS**, **PS** and **KS** are cumulative features, **PT**, **SS**, **AT** and **PPS** are evaluated per skill. The features contain information about time, performance and specific problems of the children. Due to the different nature of the features - **AT** is for example measured in seconds, while **KS** is a set of skill indices - we again process the features using the pipeline illustrated in Fig. 8.1. All

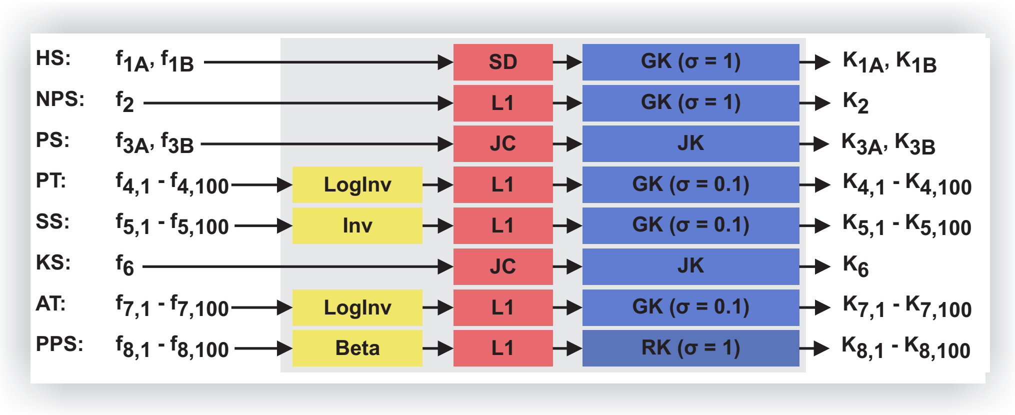

HS: f_{1A}, f_{1B} → SD → GK (σ = 1) → K_{1A}, K_{1B}
NPS: f_2 → L1 → GK (σ = 1) → K_2
PS: f_{3A}, f_{3B} → JC → JK → K_{3A}, K_{3B}
PT: $f_{4,1}$ - $f_{4,100}$ → LogInv → L1 → GK (σ = 0.1) → $K_{4,1}$ - $K_{4,100}$
SS: $f_{5,1}$ - $f_{5,100}$ → Inv → L1 → GK (σ = 0.1) → $K_{5,1}$ - $K_{5,100}$
KS: f_6 → JC → JK → K_6
AT: $f_{7,1}$ - $f_{7,100}$ → LogInv → L1 → GK (σ = 0.1) → $K_{7,1}$ - $K_{7,100}$
PPS: $f_{8,1}$ - $f_{8,100}$ → Beta → L1 → RK (σ = 1) → $K_{8,1}$ - $K_{8,100}$

Figure 8.2: Extracted features for online classification and according processing pipelines. The features use the processing pipeline and modules explained in Fig. 8.1: Depending on their nature, features are preprocessed (yellow) before computing pairwise similarities (red). The resulting similarity matrices are then transformed into Kernels (blue).

features and the processing modules applied to them are displayed in Fig. 8.2. We again use the Jaccard index (**JD**) and a set kernel (**JK**) for the so called set features (**PS**, **KS**). As mentioned before, the Beta cumulative distribution (**Beta**), followed by the L1-norm (**L1**) and an exponential kernel (**RK**) are used for performance features (**PPS**). The time features (**PT**, **AT**) use the pipeline explained in the previous section (Sec. 8.1.2). **SS** is pre-processed using the inverse, as the number of samples for unplayed skills is set to ∞. To measure the distance in the highest reached skills (**HS**), we compute the shortest path on the skill net (**SD**). The obtained similarity matrices $\mathbf{K_i}$ are transformed to distance matrices $\mathbf{D_i}$ through a constant shift ($\mathbf{D_i} = 1 - \mathbf{K_i}$).

Feature processing yields a set of more than 400 distance matrices. Feature selection is performed by ranking the features according to their degree of correlation to the correct labels (of the clustering). An optimal matrix $\mathbf{T}$ is computed, which is a square-matrix containing the pairwise hamming distances between the labels of the samples: $\mathbf{T}(i, j) = 0$, if the samples i and j belong to the same cluster, and $\mathbf{T}(i, j) = 1$ otherwise. For each matrix $\mathbf{D_i}$, we compute the distance dt to the optimal matrix with the Frobenius norm: $dt = ||(\mathbf{T} - \mathbf{D_i})||_F$. The features are then sorted in ascending order by their distance dt. For classification, the best combination b of the ten features with minimal distance to the optimal matrix $\mathbf{T}$ (2^{10} possibilities) is used. The distance matrix $\mathbf{D}$ is obtained by adding up the distance matrices $\mathbf{D_i}$ of the features $\mathbf{f_i}$ contained in b. Classification is performed by using a k-nearest neighbors scheme on $\mathbf{D}$. The best combination b and the optimal k are found using a 9-fold cross validation. The classification accuracy is computed on the same folds (not nested).

8.1.4 Performance Prediction

The online classification algorithm described in Sec. 8.1.3 enables us to assign children to a cluster after each played session of the training. We can therefore use cluster information to predict the student's future performance, training success and mathematical characteristics. We identified a set of interesting features (see Tab. 8.2) that we like to predict. These features can be attributed to four different areas:

1. *Long-term training performance* (**PAS**, **NR**, **HS**): End level reached within the tutoring system. We predict the passed skills **PS** and the passed number ranges **NR** (A_{0-10}, $B_{0-10}+$, $B_{0-10}+$, A_{0-100}, $B_{0-100}+$, $B_{0-100}-$, A_{0-1000}, $B_{0-1000}+$, $B_{0-1000}-$) during the training, and the level reached (highest skill **HS**, separately for *Part A* and *Part B*) at the end of the training. For *Part B*, we also distinguish between addition (+) and subtraction (-).
2. *Short-term training performance* (**NSS**, **NSR**): Prediction of student responses. The number of trials **NS** needed to pass a skill are predicted, as well as the number of trials **NRS** to pass a range. Both measures are only predicted for skills (ranges) that were passed by the cluster majority as well as by the test sample.
3. *Individual knowledge gaps* (**KS**, **KNR**): Identification of particular deficient areas of knowledge of the student. We identify key skills **KS** and individual problem areas **KA** (A_{0-10}, $B_{0-10}+$, $B_{0-10}+$, A_{0-100}, $B_{0-100}+$, $B_{0-100}-$, A_{0-1000}, $B_{0-1000}+$, $B_{0-1000}-$) of the children.
4. *External test results* (**EPT**): Prediction of external post-test scores. We predict the scores for the external addition and subtraction tests conducted in the user studies (see Chapter 4): The HRT and the AC (detailed in Sec. 4.2).

Prediction of features is performed using cluster information (as described in Tab. 8.2). The prediction of long-term training performance is interesting for analysis as the predicted features are correlated to the learning trajectories. The identification of knowledge gaps helps to find subtypes of mathematical learning patterns and can be used to increase the degree of individualization (e.g., putting more emphasis on the training of key number ranges). Prediction of external test results is especially important for model validation. The prediction of short-term performance can be used to improve adaptation (e.g., minimizing frustration).

8.2 Evaluation of proposed algorithm

To evaluate the accuracy of classification and prediction of our method, we used log files collected from participants from the *BMBF-study* (described in Chapter 4). The data set at hand contains 88 participants (68% females). 50 participants (72%

Table 8.2: Predicted features (abbreviations in bold) along with error measures. $\mathbf{f_p}$ denotes the predicted value, $\mathbf{f_t}$ the actual value of the feature, and CE the classification error: $\#(\mathbf{f_p} \neq \mathbf{f_t})/\#played$. **JC** (Jaccard Index), **SD** (skill distance) and **L1** (L1 norm) correspond to the distance measures described in Fig. 8.1 (red). Prediction for all the features is performed using cluster information.

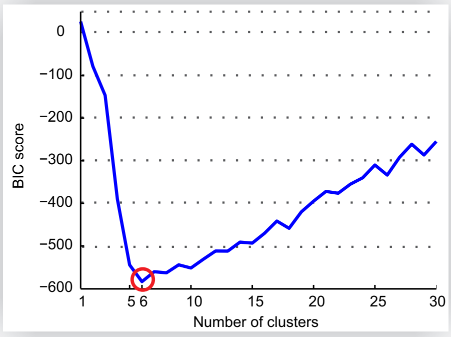
	Description	**Error measures**
PAS	Indices of passed skills during training. A skill is predicted as passed, if the cluster majority passed it.	**JC**
NR	Indices of passed number ranges during training. A range is predicted as passed, if the cluster majority passed it.	**JC**
HS	Indices of highest skills passed by cluster majority during training (separately for *Part A* and *Part B*).	**SD**
NSS	# samples needed to pass a skill (cluster mean). Predicted only for skills passed by cluster majority.	median($\mathbf{L1}/\lvert\mathbf{f_t}\rvert$)
NSR	# samples needed to pass a number range (cluster mean). Predicted only for ranges passed by cluster majority.	median($\mathbf{L1}/\lvert\mathbf{f_t}\rvert$)
EPT	Absolute and relative (#correct tasks/#tasks) post test score (cluster mean): **HRT+**, **HRT-**, **AC+**, **AC-**.	**L1**
KS	Indices of key skills (see Def. 5.1). A skill is classified as key skill, if the cluster majority has problems.	CE, Recall, Precision
KNR	Indices of key number ranges. A range is classified as key number range, if it contains at least one key skill.	CE, Recall, Precision

females) were diagnosed with developmental dyscalculia (DD), and 38 participants (63% females) were control children (CC). The log files stem from six weeks of training and contain 27 complete training sessions (of 20 minutes) per child. On average, each child solved 1430 tasks (SD $\sigma = 212$) during the six weeks. The number of solved tasks per session corresponded to 52.9 (SD $\sigma = 7.8$).

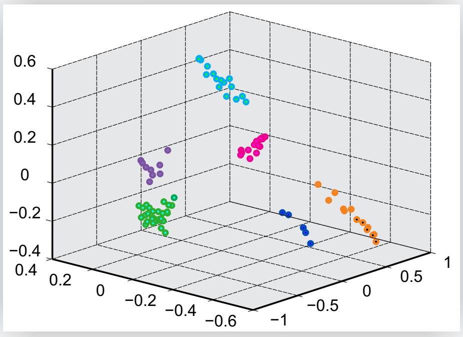
In the following, we first describe the resulting clusters and interpret them according to the mathematical characteristics of the children. Then, we analyze the classification accuracy over time as well as the predictive performance of our method.

8.2.1 Resulting clusters and interpretation

The optimal number of clusters was estimated using the BIC (see Sec. 8.1.2), the best BIC score was reached for $k^* = 6$ clusters. The BIC score for the different



(a) BIC score by cluster number.



(b) Resulting clusters in three dimensions.

Figure 8.3: The best BIC score is reached for $k^* = 6$ clusters (a). This result is supported by the clear separability of the transformed data in three dimensions (b).

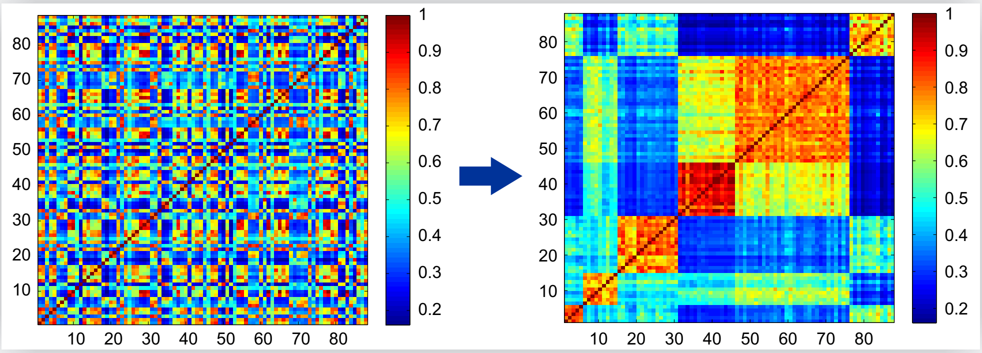


Figure 8.4: Similarity matrix used as an input for the offline clustering (left). Similarity matrix sorted by group label after clustering (right). The red color denotes high similarity, while blue denotes low similarity. The six resulting clusters are clearly visible on the diagonal of the sorted similarity matrix (right).

cluster numbers is illustrated in Fig. 8.3(a). This result is supported by the clear separability of the transformed data in three dimensions, displayed in Fig. 8.3(b).

The six clusters are also clearly visible on the diagonal of the sorted similarity matrix: Figure 8.4 (left) illustrates the similarity matrix **D** used for clustering, while Fig. 8.4 (right) shows the sorted similarity matrix. The x- and y-axes of the matrices denote the sample indices. High similarities are displayed in red, while blue denotes low similarity.

The six resulting clusters (denoted by *C1*,...,*C6*) can also be interpreted regarding

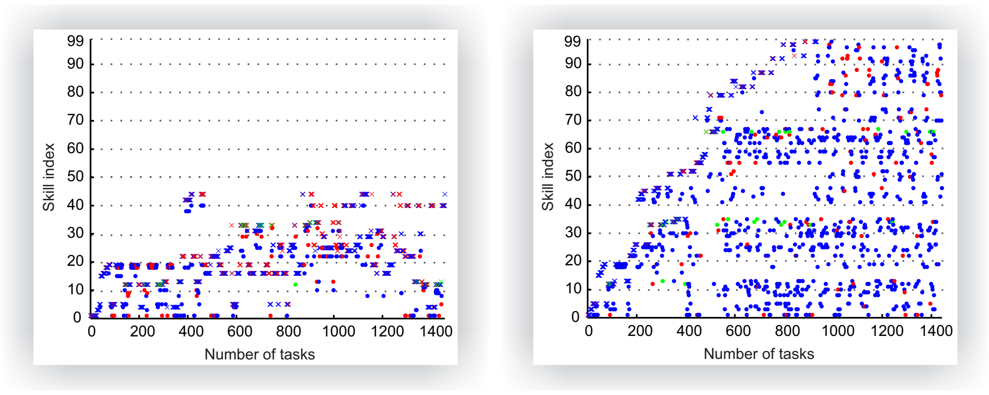


Figure 8.5: Example trajectories of two children from clusters C1 (left) and C6 (right). A cross denotes a task played at the actual difficulty level while a dot denotes a random re-test of an already mastered skill. Red stands for a wrong answer, blue for correct, green for neutral. The child from cluster C6 mastered all the skills of the training program after about 900 tasks, while the child from C1 showed difficulties in mastering the skills.

the characteristics and distinct learning patterns of the students, which are reflected in their training trajectories. Two example trajectories of children from clusters *C1* and *C6* are displayed in Fig. 8.5. While the student from cluster *C6* finished the training, *i.e.*, passed the most difficult skill of the program after about 900 tasks, the student from cluster *C1* mastered only about 50% of the skills over the course of the training.

A detailed description of the different clusters is given in Tab. 8.3. We list demographic information about the cluster members, such as the age. Furthermore, we look at the cluster size, with special attention to whether the cluster members are classified as having DD. And finally, we also investigate characteristics from the training with `Calcularis`: How many skills did the children master during the training and what level did they reach? In which areas of the training program did the cluster members exhibit problems?

The children assigned to *C1* have only passed the number range from 0-10. The difficulties with number representations ($\mathbf{PP}_{\mathbf{A}_{100}} = 1.00$) as well as procedural knowledge ($\mathbf{PP}_{\mathbf{B}_{100}} = 0.99$) imply an early disorder of numerical functions. Indeed, all children of this group were diagnosed with DD. Children in *C2* have passed the number range $0-100$ for *Part B* (*arithmetic operations*), but exhibit difficulties in *Part A* (*number representations*). This learning pattern ($\mathbf{PP}_{\mathbf{A}_{100}} = 1.00$) suggests problems with domain-specific functions such as quantity comparison and symbolic representation. In contrast to *C2*, children in *C3* passed the number range 0-100 for *Part A*, but not for *Part B*. This observation indicates intact number processing, but difficulties in understanding and executing procedures ($\mathbf{PP}_{\mathbf{B}_{100}} = 0.99$) . The

Table 8.3: Data per cluster (**C1,...,C6**): Number of children **NC** (%), mean age **AG** (SD σ), number of passed skills **NPS**, probability of having problems **PP** in different areas and number ranges of the training. **NC** and **AG** are given for all samples as well as only for CC and DD children. Number ranges mastered during training are marked bold. The proportion of children with DD decreases with increasing cluster performance.

		C1	**C2**	**C3**	**C4**	**C5**	**C6**
NC	all	13 (14.77)	5 (5.68)	16 (18.18)	9 (10.23)	30 (34.09)	15 (17.05)
	CC	0 (0.00)	2 (40.00)	5 (31.25)	4 (44.40)	16 (53.30)	11 (73.30)
	DD	13 (100.0)	3 (60.00)	11 (68.75)	5 (55.60)	14 (46.70)	4 (26.70)
AG	all	9.26 (0.87)	8.18 (0.42)	8.60 (0.67)	8.52 (1.29)	8.78 (0.93)	8.53 (0.87)
	CC	-	8.06 (0.03)	8.10 (0.49)	7.52 (0.27)	8.16 (0.53)	8.11 (0.44)
	DD	9.26 (0.87)	8.26 (0.58)	8.82 (0.64)	9.32 (1.21)	9.49 (0.78)	9.67 (0.71)
NPS	A, B	12, 9	12, 14	15, 12	19, 22	22, 25	22, 30
PP	A_{10}	**0.80**	**0.95**	**0.79**	**0.31**	**0.39**	**0.19**
	B_{10}	**0.68**	**0.20**	**0.57**	**0.11**	**0.14**	**0.14**
	A_{100}	1.00	1.00	**0.94**	**0.91**	**0.89**	**0.49**
	B_{100}	0.99	**0.98**	0.99	**0.96**	**0.87**	**0.30**
	A_{1000}	x	x	x	0.98	**0.72**	**0.56**
	B_{1000}	x	x	x	0.98	0.99	**1.00**

clusters *C4* and *C5* have passed the number range 0-100 for both parts and the number range 0-1000 for *Part A*, respectively. *C6* is the best performing cluster, with children having passed all number ranges and thus finished the training. The performance differences between clusters *C4*, *C5* and *C6* are probably due to differences in capacity and availability of domain-general functions (attention, working memory, processing speed). Notably, *C4*, *C5* and *C6* contain children with DD (26.7% in *C6*). This fact can be attributed to age differences: children with DD belonging to cluster *C6* attend the 4^{th} or 5^{th} grade of elementary school. The interpretation of learning patterns confirms the usefulness of trajectory information for clustering.

8.2.2 Classification Accuracy

Classification is performed online after each training session. We classify the children to a particular subgroup depending on their current training status and compute classification accuracy using cross validation. As expected, classification accuracy increases with the number of available training sessions (illustrated in Fig. 8.6). Five sessions are already sufficient for our algorithm (blue) to cluster 50% of the children

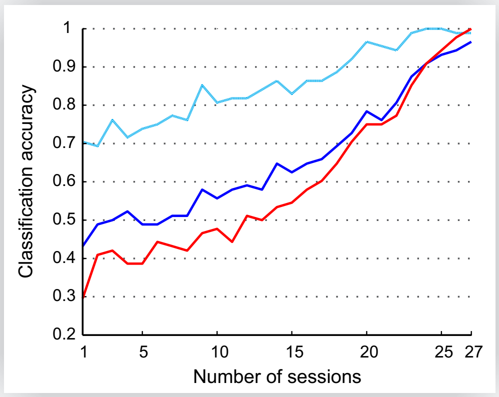


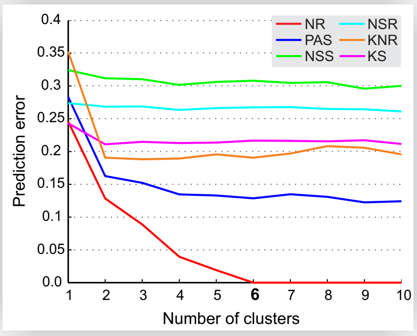
Figure 8.6: Classification accuracy over time. Accuracy using offline features (red), the algorithm described in Sec. 8.1.3 (blue) and portion of children classified correctly or to a direct neighbor cluster (light blue). Five sessions are enough for our method (blue) to classify 50% of the children correctly.

correctly. The accuracy is further increased to 60% after 14 sessions, and 70% is reached after 19 played sessions. A random assignment would result in $16.\overline{6}\%$ of correctly classified children. An accuracy of 34.09% could be reached by assigning all children to the largest cluster (C5, see Tab. 8.3).

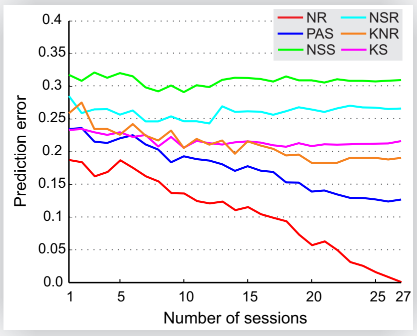
Considering that some neighboring clusters are close to each other (for instance, *C1* and *C2* are statistically distinguishable but similar), the assignment of a child to a direct neighbor of the correct cluster will not significantly deteriorate prediction quality. The estimation of the percentage of children assigned to the correct cluster or its direct neighbor (light blue) yields a success rate higher than 70% already after five sessions. The classification with the global features used for clustering (red) performs worse for small numbers of sessions, and equally well after 20 sessions. This behavior highlights the importance of using local features for classification at an early stage in the training.

8.2.3 Predictive Performance

The online classification after each training session allows for predicting students' performance in the four selected areas (detailed in Tab. 8.2) based on cluster information. In the following, we analyze prediction accuracy in two ways: We first investigate prediction accuracy at the end of the training (after 27 sessions) for different cluster numbers. In a second step, we look at online performance prediction and compute predictive performance after each training session using the optimal number of $k^* = 6$ clusters.



(a) Offline prediction errors for selected features by cluster number.



(b) Predictive performance for six clusters over the course of the training.

Figure 8.7: Offline prediction errors for selected features plotted by the number of clusters (a). Prediction errors tend to decrease with increasing cluster number and stagnate when reaching the optimal number of $k^* = 6$ clusters. Predictive performance increases over the course of the training (b). The abbreviations of the features correspond to those defined in Tab. 8.2.

Prediction errors after 27 sessions (offline prediction) were calculated for $k = 1$ to $k = 10$ clusters using a cross validation. Prediction for the specific features was computed as described in Tab. 8.2. Figure 8.7(a) shows the prediction errors for selected features. For most of the features, prediction errors decrease with an increasing number of clusters up to $k^* = 6$ clusters and stagnate afterward. **NSS** and **NSR** however, do not show a high cluster dependency. As these features are predicted for skills (number ranges) mastered by the cluster majority, the number of skills (number ranges) for which we can predict **NSS** (**NSR**) depends on **PAS** (**NR**). The exact errors for all features of the four selected areas (detailed in Tab. 8.2) for $k = 1$ and $k^* = 6$ clusters are listed in Tab. 8.4. Most errors were significantly reduced (indicated by a two-sided t-test corrected for multiple comparisons with Bonferroni-Holm) by using the cluster information. The high prediction accuracy of the long-term training performance (**PAS**, **NR**, **HS**) shows that clustering the children based on trajectory features is indeed meaningful. Furthermore, the accurate prediction of post-test results **EPT** demonstrates the correlation between achievement in external assessments and in-tutor performance and thus proves the validity of the student model. The promising results in the identification of knowledge gaps (**KS, KNR**) provide a valuable tool in the analysis of learning patterns and allow experts to elaborate individualized learning strategies.

The accurate predictions of knowledge gaps together with the good prediction of short-term training performance (**NSS, NSR**) enable a tutoring system to better adapt the training to individual children. This, however, requires online perfor-

Table 8.4: Offline prediction errors for $k = 1$ and $k^* = 6$ clusters. For **EPT** features, absolute and relative errors (in brackets) are given and the numbers for **KS** and **KNR** denote classification error, recall and precision. The **HS** error is given for *Part A* and *Part B*. The abbreviations of the features correspond to those defined in Tab. 8.2. Most errors are significantly reduced by using the cluster information.

Feature	**Prediction error (k = 1)**	**Prediction error ($\mathbf{k^*} = 6$)**
PAS	0.28	0.13*
NR	0.25	0.00*
HS	2.69, 5.72	0.34*, 1.34*
NSS	0.32	0.31
NSR	0.27	0.26
HRT+	4.70 (0.12)	3.69* (0.09*)
HRT-	5.67 (0.14)	4.50* (0.11*)
AT+	3.26 (0.16)	2.61* (0.13*)
AT-	2.98 (0.15)	2.33* (0.12*)
KS	0.24, 0.10, 0.95	0.22*, 0.33*, 0.73*
KNR	0.35, 0.90, 0.55	0.19*, 0.82*, 0.74*

* $p < .01$

mance prediction. Online prediction errors for the relevant features were computed after each session using cross validation and relying on cluster information from the classification. Predictive performance over time for selected features is illustrated in Fig. 8.7(b). As expected, the prediction errors depend on the classification accuracy (see also Fig. 8.6), *i.e.*, prediction errors decrease with increasing classification accuracy. As already observed in the offline prediction task, **NSS** and **NSR** are cluster independent and therefore, prediction errors of these features do not decrease with increasing classification accuracy. Performance in long-term prediction is again good: Already after five sessions, prediction errors for **NR** and **PAS** drop below 0.25. Results in identification of knowledge gaps (**KS, KNR**) are also promising: Prediction errors for these features are around 0.2 after five sessions. The good prediction accuracy reached already after few trainings allows to draw conclusions about short-term performance and knowledge gaps.

8.3 Discussion

Training individualization is essential in an intelligent tutoring system (ITS). Most research has focused on *short-time performance prediction*. The method proposed in this chapter is geared to *long-time performance prediction*. Prior knowledge about training outcome and mathematical characteristics of the children is potentially interesting for teachers and therapeutics.

Students training with a computer-based program often show a high diversity. This is especially true for our data sets, as they contain log files from children with and without DD. Clustering children into subgroups with similar learning patterns seems therefore promising. And indeed, our results demonstrate that the prediction accuracy can be improved when taking clustering information into account. Our findings are in line with previous studies (Baker et al., 2011; Pardos et al., 2012a,b; Trivedi et al., 2011; Gong et al., 2012; Trivedi et al., 2012) employing clustering algorithms for improving short-time performance prediction.

The clusters obtained using our algorithm can be interpreted according to mathematical characteristics of the children. The interpretations of the weakest clusters (*C1*, *C2* and *C3*) are particularly interesting as they can be mapped to the different subtypes of DD proposed by von Aster (2000).

Online classification to different subgroups has the potential to gain knowledge about the children early in the training and to make predictions of future performance. The online classification of the children to a particular subgroup has shown to be an inherent problem in the beginning of the training, but by using local features the classification accuracy was notably improved, enabling accurate prediction of students' future performance. Therefore, our approach of offline clustering followed by an online classification seems suitable for making long-term predictions and contributes to a better understanding of a child's learning characteristics and thus to a better support for children with learning difficulties.

CHAPTER 9

Affective modeling

The learning outcome of a student working with a (computer-based) training program is strongly influenced by the affective states of the user. A distracted or bored student will not learn efficiently. Also negative emotions such as frustration or fear might slow down learning. Affective modeling provides the possibility of detecting states which are obstructive for learning and to intervene accordingly.

In general, affective models can be inferred from several sources, such as sensor data (Cooper et al., 2010; Heraz and Frasson, 2009) or user input data (Baker et al., 2004; Johns and Woolf, 2006; Arroyo and Woolf, 2005; Baschera et al., 2011). Existing affective models, however, focus mostly on one specific learning domain or are designed for a specific training program. In this work, we (theoretically) explore the possibility of a general framework for engagement learning, focusing on learning disabilities.

We will start our investigations based on an engagement model for spelling learning in children with dyslexia presented by Baschera et al. (2011). The model can adapt the training to individual students based on a data-driven identification of engagement states from student input. We argue that that the assumption of similar engagement patterns in children with developmental dyscalculia (DD) or dyslexia is justified and, thus, that a similar engagement model would be beneficial.

In the following, we will first introduce characteristics of dyslexia and the computer-based spelling training `Dybuster` (Gross and Vögeli, 2007; Kast et al., 2007), on which the model presented by Baschera et al. (2011) is based. In a second step, we will assess comorbidities and similarities in engagement between the two learning disabilities. We will then extend the introduced framework to the more general case of engagement modeling. Finally, we will analyze the re-usability of the engagement model for spelling learning and define desirable properties of a general model of engagement dynamics for software tutoring.

9.1 Learning disabilities and engagement

Developmental dyslexia and DD are both specific learning disabilities inferring a lack of success in language processing and mathematics, respectively. In this section, we discuss the case of dyslexia along with existing intervention programs. Furthermore, we introduce the domain of spelling learning as well as the computer-based training program `Dybuster` (Gross and Vögeli, 2007; Kast et al., 2007). And finally, we highlight the similarities between the two learning disabilities which indicate the presence of similar engagement patterns. Note that a detailed introduction to DD and the training program `Calcularis` can be found in Chapter 2 and Chapter 3, respectively.

9.1.1 Dyslexia

Developmental dyslexia is a specific learning disability which affects the acquisition of reading and writing skills (World Health Organization, 1993). Children with developmental dyslexia tend to exhibit inconsistent orthography speed and accuracy problems, as well as difficulty in segmenting and manipulating phonemes in words. In addition to poor writing and reading skills, poor speech production and spelling are other symptoms of developmental dyslexia (Goswami, 2003). Currently, developmental dyslexia is thought to originate from a neurological disorder with genetic origin (Galaburda et al., 1985, 2006; Schulte-Korne et al., 2004; Demonet et al., 2004; Ziegler et al., 2005). The prevalence of this disability is estimated to range from 5% to 17.5% in English speaking countries (Shaywitz, 1998), and to about 10% in German speaking countries (Russeler et al., 2006).

There exist a lot of intervention programs to remediate developmental dyslexia that have been scientifically evaluated in children (and adults). These programs predominantly aim at training auditory and visual functions using approaches such as low-level auditory perceptual learning (Tallal, 2004; Robichon et al., 2002; Santos et al., 2007; Besson et al., 2007; Gaab et al., 2007; Uther et al., 2006), practice of speech-like auditory stimuli (O'Shaughnessy and Swanson, 2000; Hatcher et al., 2006), practice of specific manipulations of speech-like signals (Tallal, 2004), improvement of high- and low-level visual functions (Bacon et al., 2007; Lorusso et al., 2006) and combined training of auditory and visual functions (Kujala et al., 2001). Other intervention techniques combine the training of reading and writing skills (Vadasy et al., 2000; Edwards, 2003; Shaywitz et al., 2004). Lately, a few multi-modal training programs have been proposed as well (Kujala et al., 2001; Gross and Vögeli, 2007; Kast et al., 2007).

9.1.2 Spelling learning

Spelling a word can be seen as translating from spoken language to written language. In an alphabetic language, like for example English or German, the spoken phonemes need to be matched to graphemes. This matching is not unique because some phonemes can be matched to several graphemes (for instance, the phoneme /f/ can be matched to the graphemes ‘f’ and ‘v’ in German). For spelling learning, different models have been proposed so far. One model for instance suggests that spelling is learnt through the identification of implicit and explicit rules (Hilte and Reitsma, 2011; Ehri, 2000; Cassar and Treiman, 1997; Landerl and Reitsma, 2005; Pacton et al., 2001). Children build up a mental print lexicon, but also abstract regularities from print and are taught rules that underlie their spelling system. It has been shown that children already use phonological and morphological rules from an early age. Another model suggests that spelling of new words is learnt by analogy to known words called reference words (Bosse et al., 2003; Campbell, 1985; Marsh et al., 1980; Martinet et al., 2004; Nation and Hulme, 1996, 1998). Both of these presented models imply that spelling learning is a rather *non-hierarchical* process. Rather than learning and understanding concepts and strategies that build up on each other, the process consists of memorizing the phoneme-grapheme matching and its irregularities or of building analogies to existing words.

9.1.3 Dybuster

`Dybuster` (Gross and Vögeli, 2007; Kast et al., 2007) is a multi-modal training program for spelling learning. The central idea of the training software is to recode a sequential textual input string into a multi-modal representation using a set of codes. These codes reroute textual information through multiple undistorted visual and auditory cues. This training strategy builds up the memory strength of graphemes and phonemes. Visual cues include colors, shapes and topology. Based on the information theoretical model of `Dybuster`, eight different colors are used in the software. The mapping of letters to colors is the result of a multi-objective optimization. For example, letters easily confused by dyslexics, e.g., ‘m’ and ‘n’, map to visually distinct colors. The idea is to associate colors with letters to eliminate mistakes due to letter confusion. The shapes are: spheres for small letters, cylinders for capital letters, and pyramids for the umlauts. The graph structure finally shows the decomposition of a word into syllables and graphemes. An additional auditory code computes a word-specific melody that is played to the user when entering a word.

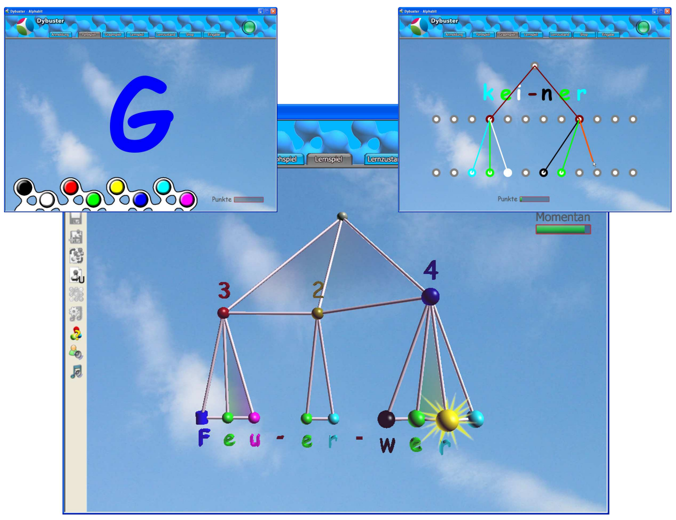


Figure 9.1: The three learning games of Dybuster (illustration by Baschera and Gross (2010a)): Color game to train the associations between colors and letters (top left), Graph game for the training of the syllable structure (top right) and Word Learning game with visual presentation of the different cues (bottom center).

The different codes not only transfer information, but also stimulate different senses. This multi-sensory stimulation enhances perception and facilitates the retrieval of memory (Lehmann and Murray, 2005; Shams and Seitz, 2008).

The tree different games of Dybuster are illustrated in Fig. 9.1 (Baschera and Gross, 2010a). In the Color game (Fig. 9.1 (top left)), children learn the associations between colors and letters. Children need to remember the colors of the different letters: The color fades out over time and children need to pick the right one. In the Graph game, children graphically segment a word into its syllables and letters (Fig. 9.1 (top right)). These first two games are played at the beginning of the training to learn the codes that are integrated in Dybuster. In the third game Word Learning, representing the actual learning game, the program presents the alternative representations (graph, colors, shapes) of a word (Fig. 9.1 (center)). A voice dictates a word and the children hear a melody computed from the involved letters and the lengths of the syllables. Children then need to type the word on the keyboard. To avoid displaying completely misspelled words, the training program provides immediate visual and auditory feedback to errors. The sequence of words presented to the child is adapted to the skill level and the error profile of the children.

In Dybuster, the selection of words to be prompted is adapted to the skill level of the children. The word selected to be trained next is the word with the highest progress potential with respect to training time. The knowledge representation is an estimate of individual mal-rule difficulties. Mal-rules define different error types which a child can commit. Possible error categories are, e.g., capitalization errors, typing errors (depending on key distance or for technical reasons), letter confusion (visual or auditory similarity) or erroneous phoneme-grapheme matching. As immediate feedback is presented after an erroneous letter, error classification is ambiguous, i.e., different deficits can lead to the same final error. To deal with this ambiguity, `Dybuster` uses an inference algorithm for perturbation models based on Poisson regression (Baschera and Gross, 2010b). The algorithm is designed to handle unclassified input with multiple errors described by independent mal-rules. During the training, the representation of the student's mastery of the domain is continuously updated after each entered word. Based on these estimates, a prediction of further spelling performance and a classification of committed errors for each individual student can be estimated. In addition to this spelling knowledge representation, the word selection controller accounts for the optimal time to repetition (time until a previously misspelled word is repeated).

9.1.4 Comorbidities and similarities in engagement

Developmental dyslexia and DD, both brain-based disorders, often exhibit comorbidity, which is the co-occurrence of two or more disorders in the same individual. Studies show that individuals with DD do often show language difficulties as well, and vice versa, that dyslexic individuals often suffer from difficulties in arithmetic (von Aster and Shalev, 2007; Ostad, 1998; Lewis et al., 1994; Badian, 1999; Barbaresi et al., 2005; Dirks et al., 2008; Ackerman and Dykman, 1995). More importantly, children with these learning disabilities often exhibit comorbidities with ADHD (Shaywitz et al., 1994; Germanò et al., 2010; Fletcher, 2005; Barbaresi et al., 2005). In addition, children with learning disabilities often show anxiety or aversion against the subject (Rubinsten and Tannock, 2010). Furthermore, they tend to underperform in school and later in profession (Bynner, 1997). These facts suggest that children with learning disabilities will exhibit low intrinsic motivation and attentional problems and thus, monitoring of engagement dynamics becomes even more important. Since similar implications are relevant for the two learning disabilities, we assume the appearance of similar engagement states for developmental dyslexia and DD.

9.2 General engagement dynamics modeling framework

To define a framework for building a general engagement dynamics model, we extract and analyze the main steps of the model for engagement dynamics in spelling learning (Baschera et al., 2011). The resulting model is a dynamic Bayesian network (DBN) (Murphy, 2002) representing different affective states as well as their relationships. In brief, we can define the following framework:

1. **Indicator definition**: An indicator variable, giving an indication of the engagement state of the children needs to be determined to label the data. This variable can be measured using sensor data (Cooper et al., 2010; Heraz and Frasson, 2009) or by relying entirely on input data as in the engagement model for spelling learning. Entirely data-driven indicators are usually noisy and highly dependent on the learning domain.
2. **Feature extraction**: A set of recorded features needs to be extracted. This set contains measures of input and error behavior, timing, and variations of the learning setting induced by the system controller. Possible features were proposed in previous work (Baker et al., 2004; Johns and Woolf, 2006; Arroyo and Woolf, 2005; Baschera et al., 2011). The set of meaningful features is strongly influenced by the learning environment.
3. **Feature selection**: To select the features, the relation between the extracted features and the indicator variable needs to be estimated, for example by using a LASSO logistic regression.
4. **Model building**: In a final step, the graphical model needs to be inferred from data. The parameters of the DBN can be estimated using expectation maximization (EM) (Dempster et al., 1977). The quality of different graphical models can be assessed by computing the Bayesian Information Criterion (BIC). Model validation can also be performed with Approximation Set Coding (Haghir Chehreghani et al., 2012).

This framework gives an overview of the steps to be taken in order to build a model for engagement dynamics in any domain. Steps 1 and 2 are essential when trying to find a valid model. These two initial steps, however, are also highly dependent on the particular learning domain and the learning environment. The indicator function and the set of features that we applied for the engagement model in spelling learning are therefore not directly applicable to other domains (such as learning mathematics) for the purpose of modeling engagement dynamics.

9.3 Engagement model for mathematics learning

Constructing a model of engagement dynamics requires a generic framework to support generalization of engagement behavior. We start by referring to the previously developed model for engagement dynamics in spelling learning (Baschera et al., 2011) and explore its re-usability. Furthermore, we assess the limitations of the existing model and provide suggestions on how to overcome them.

As discussed above, steps 1 and 2 of the general framework are essential. They highlight the dependence on the learning domain and on the specific environment and can therefore not directly be applied to another learning domain or training environment. Steps 3 and 4 on the other hand are independent of the learning domain or the training environment. In the following, we therefore first assess the first two steps of the engagement model for spelling learning to identify the parts of the model that can be reused. Furthermore, we define desirable properties of an indicator function (step 1) and a feature set (step 2) applicable to learning in general and make a first draft of a possible general feature set.

9.3.1 Indicator Function

The model for engagement dynamics in spelling learning uses the error repetition probability (ERP) as a noisy indicator. If the student is in a distracted state, more careless errors will occur which are unlikely to be repeated (low ERP). If the student is in a non-receptive state (inhibits learning), committed errors will probably be repeated (high ERP). This indicator function is meaningful under the following (strict) assumptions:

- Stationary learning environment: The learning environment consists of only one type of task (here the typing of words).
- Non-hierarchical learning domain: The learning works in a non-hierarchical way, for example through memorization. This assumption means that a word is learned through memorizing the spelling in the case of `Dybuster`.

The learning environment for mathematics learning (`Calcularis`) as well as the learning domain do not fulfill these properties. `Calcularis` consists of a number of skills at varying difficulty levels, each of them depending on each other. Performance or error measures can thus not easily be compared across the different skills. Furthermore, mathematics learning is very hierarchical. Besides knowing rules or building procedural knowledge, conceptual knowledge (understanding the 'why') needs to be built. If a child commits an error such as '12-5=3', it makes no sense to repeat exactly the same task after a certain amount of time. The child needs to learn how a task including a carry is handled. Having learned this concept, the child can solve all tasks involving carrying.

How should an appropriate indicator function for a hierarchical learning domain and a learning environment employing different skills look like? Why do we need an indicator function in the first place? As we rely on input data only, no ground truth about the emotional state of the user is available. The indicator function represents the emotional state (for example engagement) over time and thus provides us with a labeling of the data and therefore we deal with supervised learning instead of unsupervised learning. Assuming an interplay between human learning and affective dynamics (Kort et al., 2001), an indicator based on performance measures in the learning environment can be selected. However, being in an engaged state is a necessary but not sufficient condition for learning. The indicator function therefore needs to differentiate between different reasons for low progress in the environment. Besides not being engaged, the tasks posed can be too easy or too difficult (not matching the skill level of the user) or there can be task comprehension problems. All these cases need to be taken into account. Furthermore, the indicator function needs to consider the hierarchical structure of the skills and the dependencies among them and thus also account for previous knowledge. Still, an indicator function relying purely on input data will always be an approximation of ground-truth. The input data can, however, be enhanced to increase the reliability of the indicator. `Calcularis`, for example, also records careless input of the children such as random key strokes or mouse clicks. These inputs give an additional indication of the engagement state.

9.3.2 Feature Set

The set of features used for the engagement dynamics model in spelling learning is very specific and in particular also very much adapted to the learning domain and the learning environment used. Table 9.1 (Baschera et al., 2011) lists and describes all the extracted features of the model.

The features used can be divided into three categories. Features in the *Timing* category are useful to indicate attention, but also particularly specific to the learning environment. Features such as the input rate **IR** and its variance **IRV** assume an environment where the results are entered via the keyboard and where the typing velocity is meaningful, which is not the case for `Calcularis`. Also features such as **TfE** and **TtNE** assume an immediate feedback on the error (before the child has typed the whole result). On the other hand, think time **TT** and off time **OT** indicate the child's performance also in the mathematics learning environment. Also in the second category focusing on *Input & error behavior*, only few features can be re-used. Help calls (**HC**) are for example not possible in every environment. The **FC** feature is only meaningful if feedback on errors is given already while the child enters the result. And the **SPE** feature is specific to the learning domain. In contrast, features such as repetition error (**RE**) or error frequency (**EF**) describe general error behavior (Does the user repeat errors? How many errors does the user make?) and thus are meaningful for any learning domain. The third category (*Controller Induced*) is completely dependent on the learning environment, as these features are induced by the controller of the particular environment. Table 9.2 discusses which features are specific to the learning domain and the environment of spelling learning, and which features could be reused for the mathematics learning environment.

As is evident from Tab. 9.2, the given feature set is specifically designed for the spelling learning environment, yielding very good results. For this reason, most of

Table 9.1: Extracted features and abbreviations (bold) used in the following (Baschera et al., 2011). The feature set includes timing and behavioral as well as controller induced features.

Feature	**Description**
Timing	
Input **R**ate	Number of keystrokes per second.
Input **R**ate **V**ariance	Variance of the **IR**.
Think **T**ime	Time from dictation of word to first input letter of student.
Time **f**or **E**rror	Time from last correct input letter to erroneous input letter.
Time **t**o **N**otice **E**rror	Time from error input letter to first corrective action.
Off **T**ime	Longest time period between two subsequent letter inputs.
Input & Error Behavior	
Help **C**alls	Number of help calls (repeating the dictation).
Finished **C**orrectly	True if all errors are corrected when enter key is pressed.
Same **P**osition **E**rror	True if multiple errors occur at one letter position of a word.
Repetition **E**rror	State of previous input of the same word (three states: *Correct* / *Erroneous* / *Not Observed*).
Error **F**requency	Relative entropy (Kullback and Leibler, 1951) from observed to expected error distribution (given by the student model (Baschera and Gross, 2010b)) over last five inputs. Positive values are obtained from larger errors numbers, negative values from smaller ones.
Controller Induced	
Time **t**o **R**epetition	Time from erroneous input to respective word repetition.
Letters **t**o **R**epetition	Number of entered letters from erroneous input to respective word repetition.

the features cannot be directly applied to a different learning domain or a different learning environment such as mathematics learning. However, we can divide the features designed for the spelling learning environment into different feature categories and derive a general feature set from those. We use the categories *input behavior, problem statement, problem-solving behavior, performance* and *environment*. Table 9.3 shows the categories as well as our suggestion for a general feature set associated with these categories for engagement dynamics modeling.

The features of the comprehensive feature set can be used for different learning domains and environments and are particularly suitable for hierarchical learning domains such as mathematics learning. Table 9.4 demonstrates that most features

Table 9.2: Assessment of feature set for the engagement dynamics model in spelling learning. Most features cannot be directly reused for the mathematics learning environment as they are specific to spelling learning and the `Dybuster` environment.

Feature	**Assessment**	**Reason**
Timing		
Input **R**ate	No	Input rate not meaningful for mathematics learning.
Input **R**ate **V**ariance	No	Same reason as for the **IR**.
Think **T**ime	Yes	Can be replaced by answer time, i.e., the time the child needs to answer the task.
Time **f**or **E**rror	No	Only meaningful in an environment with immediate feedback on errors.
Time **t**o **N**otice **E**rror	No	Feedback is only given after the whole result has been entered.
Off **T**ime	Yes	Could be redefined to be the time until the child starts answering the task.
Input & error behaviour		
Help **C**alls	No	No help calls possible in the environment.
Finished **C**orrectly	No	Feedback is only given after the whole result has been entered.
Same **P**osition **E**rror	No	Only meaningful for spelling learning.
Repetition **E**rror	Yes	Might be replaced by assessing the previous opportunity the child had to apply a certain skill.
Error **F**requency	Yes	Student model needs to compute expected error distribution.
Controller Induced		
Time **t**o **R**epetition	No	Repetition of exactly same task is not done.
Letters **t**o **R**epetition	No	Repetition of exactly same task is not done.

could be directly applied to the mathematics learning environment, such as the one provided by `Calcularis`.

9.4 Discussion

In this chapter, we explored the idea of transferring existing results in the context of engagement modeling in spelling learning to general applications for learning disabilities. Our assumptions are scientifically justified by the significant co-occurrence of dyslexia and DD with ADHD and the similar implications such as anxiety and low intrinsic motivation of the two learning disabilities. This observation constitutes a clear indicator of the existence of similar engagement dynamics, thereby suggesting general measures and models of engagement.

We performed a detailed analysis of similarities and differences of the two disabilities as well as the according learning environments. Our analysis of the learning domain and the learning environments, of their corresponding student models, as well as of the experimental data, suggests that the model for spelling learning can be extended to the case of mathematics learning. Our findings show, however, that indicator functions and features are specific to the learning domain. Table 9.5 summarizes the similarities and dissimilarities of the two investigated cases.

From this comparison we conclude that there are substantial differences in the learning domain, which in turn directly influence the learning environment and the student model. Furthermore, these differences indirectly affect the experimental data as well. Therefore, the application of the indicator function and of the feature set specified for the model of engagement dynamics in spelling learning is fairly sophisticated. Rather, a more general indicator function and a comprehensive feature set need to be defined. At present, this is an area of active research.

Table 9.3: Sketch of a general feature set (abbreviations in bold) for engagement modeling derived from the spelling learning environment. Extracted features are consistent with previous work (Baker et al., 2004).

Generalized feature	**Description**
Input behaviour	
Input **T**ype	The type of the input, e.g., mouse, keyboard, pull-down menu, etc.
Valid **I**nput	True if the input is valid, e.g., input string only contains numbers.
Input **S**tatistics	Statistics of the input as for example mean input rate or input rate variance.
Problem-oriented **I**nput	True if the input is related to the problem, e.g., user enters text into the answer.
Problem statement	
Problem **D**ifficulty	Ideally an overall measure of the problem difficulty.
Problem **T**ype	The kind of problem at hand.
Problem **F**amiliarity	True if the user is familiar with the kind of problem.
Problem-solving behavior	
Time to **S**olution	Total time spent on this problem until solution.
Time **L**ast **S**olutions	Total time spent on the last *n* problems.
Time **D**eviation	Standard deviation from mean time to solution for this kind of problem.
Answer **T**ime	Time until user starts answering the problem after she sees the problem statement.
Problem **A**pproach	The user's approach to the problem, e.g., trial and error, systematic, etc.
Help **U**sage	If a help system is available how is it used, e.g., frequency of use.
Performance	
Correctness of **A**nswer	Assessment of user answer: correct, wrong or misconception.
Answer **A**ssessment	User performance meets model expectations (e.g., posterior probability).
Error **I**nformation	Information about the committed error, e.g., spelling error.
Error **R**epetition	Number of errors in the past for the same kind of problem.
Error **F**requency	Frequency of certain error types.
Error **C**ount	Number of errors similar to the current error in the last *n* problems.
Environment	
Time **B**etween **P**roblems	Time from last similar problem to this one.
Similar **P**roblems **C**ount	Number of problems similar to the current one in the last *n* problems.
Work **B**etween **P**roblems	Amount of work between the current and the last similar problem.
Session **D**uration	Duration of the training session.
Time of the **Day**	Time of the day the training session takes place.

Table 9.4: Assessment of the general feature set (introduced in Tab. 9.3) for the case of mathematics learning. Most features of this general set could be directly applied to the `Calcularis` environment.

Generalized feature	**Assessment**	**Reason**
Input behaviour		
Input **T**ype	Yes	Mouse, keyboard, joystick.
Valid **I**nput	Yes	Input is a valid number.
Input **S**tatistics	No	Input statistics not meaningful for the specific environment.
Problem-oriented **I**nput	Yes	Click at the right place (where you should click).
Problem statement		
Problem **D**ifficulty	Yes	Can directly be derived from the student model.
Problem **T**ype	Yes	Trained skill.
Problem **F**amiliarity	Yes	True if the user has trained the same skill before.
Problem-solving		
Time to **S**olution	Yes	Directly applicable.
Time **L**ast **S**olutions	Yes	Directly applicable.
Time **D**eviation	Yes	Directly applicable.
Answer **T**ime	Yes	Directly applicable.
Problem **A**pproach	Yes	Problem omission can be detected.
Help **U**sage	No	No help system available.
Performance		
Correctness of **A**nswer	Yes	Directly applicable.
Answer **A**ssessment	Yes	Comparison of student's performance against estimated model performance.
Error **I**nformation	Yes	Directly applicable using the bug library (Sec. 3.4).
Error **R**epetition	No	Repetition of exactly same task is not done.
Error **F**requency	Yes	Directly applicable using the bug library (Sec. 3.4).
Error **C**ount	Yes	Directly applicable using the bug library (Sec. 3.4).
Environment		
Time **B**etween **P**roblems	Yes	Time from last problem that trained the same skill to this one.
Similar **P**roblems **C**ount	Yes	Number of problems that trained the same skill in the last *n* problems.
Work **B**etween **P**roblems	Yes	Amount of work between last problem that trained the same skill and this one.
Session **D**uration	Yes	Directly applicable.
Time of the **Day**	Yes	Directly applicable.

Table 9.5: Comparison of the two cases of DD and dyslexia in terms of the learning domain and environment as well as the student model and the experimental data available from `Dybuster` and `Calcularis`.

Category	**Dyslexia**	**Dyscalculia (DD)**
Learning disability	Brain-based disorder	Brain-based disorder
	Comorbidities (DD, ADHD)	Comorbidities (Dyslexia, ADHD)
	Aversion & anxiety against the subject	Aversion & anxiety against the subject
Learning domain	Static (non-hierarchical)	Hierarchical
	Learning through memorization & analogies	Conceptual knowledge important
Learning environment	One main learning game	Range of games ordered hierarchically
	Multi-modal cues recode textual input string	Visual cues encode properties of number
	Difficulty of word adapted to user	Selection of games and tasks adapted to user
Student model	Poisson-based perturbation model	Dynamic Bayesian network
	Selection of word with highest progress potential	Non-linear, rule-based task selection
Experimental data	Input logs with inputs, errors and timestamps	Input logs with inputs, errors and timestamps
	Input from keyboard	Input from keyboard, mouse and joystick
	No additional information	Recording of invalid inputs

CHAPTER 10

Classification of children with developmental dyscalculia

Developmental dyscalculia (DD) has an estimated prevalence of 3% – 6% in German speaking countries (Shalev and von Aster, 2008). It has been shown that basic number processing and understanding capabilities acquired in preschool are essential for later mathematical performance (Landerl et al., 2004; Hannula and Lehtinen, 2005; Mazzocco and Thompson, 2005; Krajewski and Schneider, 2009). Therefore, early intervention and detection are important.

However, the diagnosis of DD involves the assessment of the child's numerical as well as domain general abilities by standardized tests. These standardized tests are time consuming and need to be conducted by an expert. A computer-based diagnosis tool would allow for an inexpensive and mostly unsupervised screening in the classroom and indicate children at risk for DD.

So far, few computer-based tests for DD exist. `Dyscalculium` (Beacham and Trott, 2005) is a screening tool for students in higher education. The tool was evaluated with 19 students and demonstrated a sensitivity of 83.3% and a specificity of 92.3%. `Dyscalculia Screener Digital` (Butterworth, 2003) is a computer-based test that assesses basic numerical capacities of children from 6 to 14 years. This test has been standardized in the UK.

In this chapter, we describe the development of a computer-based screening tool for DD. The tool is purely data-driven and is based on the log file data from `Calcularis`. Test duration is adaptive, with a maximum duration of 30 minutes. In the following, we describe the features extracted from the available data as well as the feature selection. We then introduce the algorithms for static and adaptive classification and finally evaluate the accuracy of the test. The work presented in this chapter was developed in the context of a master thesis (Klingler, 2013), which contains further details about the applied approach.

10.1 Feature extraction

The screener tool is based on log files from 89 participants, collected in the *BMBF-study* (see Chapter 4). 49 children (73% females) were diagnosed with DD, while the other 40 children (65% females) were control children (CC). All children completed a minimum of 26 training sessions. On average, children completed 29.5 sessions (SD $\sigma = 1.8$). Over the course of the training, participants solved on average 1541 tasks (SD $\sigma = 231$), the number of solved tasks per session corresponded to 52.3 (SD $\sigma = 7.3$). There were 28 participants in 2^{nd} grade, 46 children from the $3rd$ grade and 15 children visiting the 4^{th} grade of elementary school. As the children visiting the 4^{th} grade were all diagnosed with DD, they were excluded from the analyses unless noted otherwise.

For the classification task, we extracted three different feature groups from the data: *Skill dependent*, *game dependent* and *path dependent* features. The extracted features for all groups are listed in Tab. 10.1 (Klingler, 2013).

Skill dependent features provide information about tasks associated with a specific skill. We for example extract the average performance (**P**) of the children (ratio of correctly solved tasks) per skill. Furthermore, we also assess the average answer time (**AT**) of the children at each skill. We expect children with DD to show a lower average performance and longer answer times, particularly for skills from the area of arithmetic operations. It has been shown that children with DD exhibit the same mathematical problems as control children, but to a larger extent (Murphy et al., 2007). Furthermore, children with DD tend to suffer from difficulties in acquiring simple arithmetic procedures and exhibit a deficit in fact retrieval (Ostad, 1997, 1999). In addition, we also count the number of typical mistakes (**TM**) the child commits at each skill (see Sec. 3.4 for the list of typical mistakes contained in the bug library of Calcularis).

Path dependent features provide important information about a child's training performance. In `Calcularis`, each child pursues a different path through the skill network (illustrated in Fig. 3.4), *i.e.*, the learning path is individually adapted to the children. Figure 3.6 displays the skill sequences of three different users in addition between 0-100. We therefore extract the skill sequences (**NSN**) of the children over a given time period. Furthermore, we also collect the transitions between skills (**ESN**) during this time period. The time period is limited by the maximum test duration.

Table 10.1: Extracted features and abbreviations (bold) used in the following (Klingler, 2013). Some features are specific to a certain game, while others can be used for all skills. Path dependent features look at the training sequence of the children.

Feature	**Description**
Skill dependent features	
Performance	Ratio of correctly solved tasks at a specific skill.
Answer **T**ime	Average answer time at a specific skill.
Typical **M**istakes	Number of typical mistakes committed at a specific skill.
Path dependent features	
Nodes **S**kill **N**et	Set of skills trained during a given time period.
Edges **S**kill **N**et	Set of transitions (between skills) made during a given time period.
Game dependent features	
Estimation	Ratio between number of overestimates and task count.
Secret **N**umber	Ratio by which the remaining interval is reduced.
Ordering	Ratio of false positive and incorrectly solved tasks.
Landing	Distance to correct position of the given number.

As opposed to general features such as for example the answer time, which can be extracted for each skill, *game dependent* features make only sense for skills associated with a specific game. For the `Estimation` game (see Sec. 3.2.3), we for example extract the ratio between the number of overestimates and the number of solved tasks (**E**), *i.e.* we assess how often the children overestimated the presented point sets. In the `Secret Number` game (see Sec. 3.2.3), children have to guess a number in a given interval. After each guess, they are told, if the secret number is smaller or larger than the guessed number. Our initial analyses have demonstrated that control children tend to apply a more elaborate strategy when guessing. We therefore measure the average ratio by which the remaining interval is reduced after each guess (**SN**). In the `Ordering` game (see Sec. 3.2.3) children have to decide if a given sequence of numbers is sorted in ascending order. Here, we assess the ratio bet-ween the number of false positives (*i.e.*, examples incorrectly indicated as correctly sorted) and the total number of incorrectly solved tasks (**O**). For the `Landing` game (see Sec. 3.2.3), we extract the distance between the position indicated by the child and the correct position of the given number (**L**).

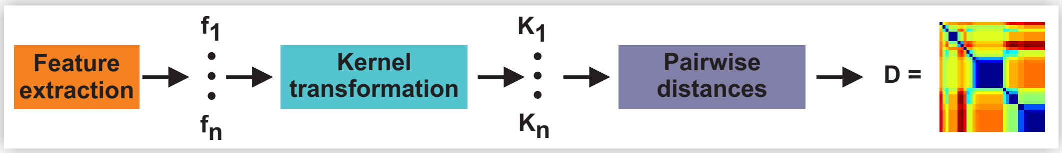


Figure 10.1: Processing pipeline yielding the pairwise feature distance matrix **D** serving as an input for the clustering algorithm. Feature values **f** are processed (turquoise) before computing pairwise distances (purple).

10.2 Feature selection

Extracting the feature types described in Tab. 10.1 yields a few 100 features. We expect that some of these features are correlated. Furthermore, each feature corresponds to a set of tasks the children have to solve. Thus, the number of features directly influences the test duration, which is limited. We therefore only select the best subset of the features for classification. To find this representative subset, we cluster the different features into groups of similar features and choose only one representative feature per cluster for classification.

To be able to cluster the features, we need a pairwise similarity measure between each pair of features F_i and F_j. However, the different feature types have very different ranges and a direct comparison between the features is therefore not meaningful. For example, **AT** measures the average answer time in seconds, while **P** denotes a performance value between 0 and 1. We therefore need to process the features to obtain the pairwise distance matrix **D** for the clustering. The according processing pipeline is displayed in Fig. 10.1.

In a first step, we apply a *Kernel transformation* to the different features to make them comparable: We compute a similarity matrix $\mathbf{K_i} \in [0,1]^{N\times N}$ for every feature F_i, where N denotes the number of children. Therefore, $\mathbf{K_i}$ contains the pairwise similarities between each pair of children j and k regarding feature F_i. We use different kernels for the different feature types. For the answer time **AT**, we combine a Gaussian Kernel with a log transform to obtain

$$\kappa_{AT}(\mathbf{x},\mathbf{z}) = exp\left(\frac{\|log(\mathbf{x}) - log(\mathbf{z})\|^2}{2\sigma^2}\right), \tag{10.1}$$

where **x** and **z** denote the respective feature values. The log transform is useful for removing outliers and was successfully used in previous applications (see Chapter 8). For the performance features **P**, we use a Beta cumulative distribution combined with a Gaussian kernel. We assume that learning can be modeled by a logistic function: This function exhibits a small gradient near the interval boundaries and a larger gradient in between. We approximate this property by using the kernel function

$$\kappa_P(\mathbf{x},\mathbf{z}) = exp\left(\frac{\|betacdf(\mathbf{x}) - betacdf(\mathbf{z})\|^2}{2\sigma^2}\right), \tag{10.2}$$

with **x** and **z** representing the respective feature values. As the *path dependent* features **NSN** and **ESN** are set features, we apply a Jaccard kernel transformation to them:

$$\kappa_{NSN,ESN}(\mathbf{X},\mathbf{Z}) = 2^{\frac{\mathbf{X}\cap\mathbf{Z}}{\mathbf{X}\cup\mathbf{Z}}} - 1, \tag{10.3}$$

where **X** and **Z** denote the values of the respective *path dependent* features. For the **SN** features, values can be put into different categories: Valid guesses reducing the remaining interval and invalid guesses (outside the given interval). For the valid guesses, the kernel should be sensitive to small differences. For the invalid guesses, the differences between the actual values are not important. We combine a Gaussian Kernel with an exponential transformation to model this property and obtain

$$\kappa_P(\mathbf{x},\mathbf{z}) = exp\left(\frac{\|exp(-\mathbf{x}) - exp(-\mathbf{z}) - 2\|^2}{2\sigma^2}\right), \tag{10.4}$$

with **x** and **z** again representing the respective feature values. For all other features (**TM**,**E**,**O**,**L**), we apply a standard Gaussian kernel.

In a second step, we compute *pairwise distances* between the obtained similarity matrices $\mathbf{K_i}$. We compute the distance between each pair of features F_i and F_j by calculating the Frobenius norm between their similarity matrices $\mathbf{K_i}$ and $\mathbf{K_j}$:

$$\mathbf{D_{ij}} = \left\|\mathbf{K_i} - \mathbf{K_j}\right\|_F. \tag{10.5}$$

The resulting matrix **D** can be directly used for clustering. As the measurements are characterized by relations, *i.e.*, they represent dissimilarities between each pair of features F_i and F_j, we perform pairwise-clustering (PC) (Hofmann and Buhmann, 1997) on **D**.

The optimal number of clusters k^* can be determined by the Bayesian Information Criterion (BIC) (Pelleg and Moore, 2000), calculating the effective number of parameters as the normalized trace of the kernel transformation matrix (Haghir Chehreghani et al., 2012). A second possibility to determine k^* is the use of the maximum test duration T: We choose k^* as the maximum number of clusters yielding an expected test duration smaller than T.

10.3 Static classification - Support Vector Machines

In this section, we present a first *static classification* approach for classifying the children into a group with DD and a control group. We call the approach static, as the resulting model will be based on a fixed set of features, *i.e.*, the selected features will be the same for all children. We are dealing with a supervised classification, *i.e.*, the training data provided to the algorithm are all labeled.

Support vector machines (SVM) are among the best performing algorithms for classifying data into two groups (Caruana and Niculescu-Mizil, 2006; Statnikov et al., 2008). Furthermore, SVMs have a convex loss function and therefore convergence to the global minimum is guaranteed. Moreover, the quadratic programming problems posed by SVMs can be solved iteratively, which allows training on large data sets (Shalev-Shwartz et al., 2007). We therefore train a SVM for our static classification problem. A detailed introduction to SVMs can be found in (Bishop, 2006).

SVMs are linear classifiers. For data not linearly separable in the original space, we can apply kernel transformations to find a separating hyperplane in a (possibly) higher dimensional space, constructed in a nonlinear way from the original space. An illustration of this so called kernel trick is given in Fig. 10.2 (Klingler, 2013).

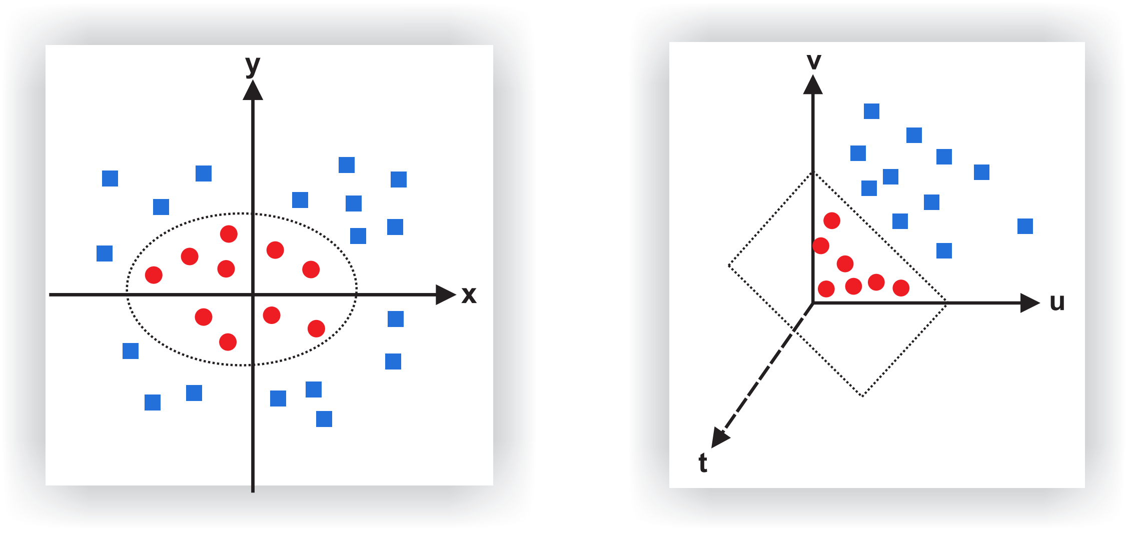
From the feature selection (described in Sec. 10.2), we have appropriate kernels $\mathbf{K_i}$ available for each feature type. To allow the SVM to use all the available data and incorporate the information provided by each feature type, we need to combine these kernels. The kernel $\mathbf{K_{SVM}}$ that combines all the F feature kernels $\mathbf{K_i}$ is given by

$$\mathbf{K_{SVM}} = \sum_{i=1}^{F} \mathbf{K_i}. \tag{10.6}$$

Note that we need to normalize the kernels $\mathbf{K_i}$ before combining them to ensure that the ranges of the matrix values are the same for each kernel $\mathbf{K_i}$, *i.e.*, each kernel equally contributes to $\mathbf{K_{SVM}}$.

10.4 Adaptive classification

The model presented in Sec. 10.3 is based on a given number of features and therefore tasks. This model is only able to make a classification based on full observability, *i.e.*, all feature values have to be known for classification. Furthermore, the



(a) Original space $\mathcal{X}$. (b) Transformed space $\mathcal{F}$.

Figure 10.2: Illustration of the kernel trick (illustration adapted from Klingler (2013)): In the original space $\mathcal{X}$ (a) the two groups are not separable with a linear classifier. After transformation (b), they can be linearly separated using a hyperplane.

output of the static model is binary. It would, however, be good to have not just a binary answer to the classification task, but a certainty measure for the predicted group label.

In this section, we develop a model that adapts the test time to the individual child, *i.e.*, the features used for classification as well as the length of the test vary over the children. Furthermore, this model will output a probability instead of a binary answer.

10.4.1 Probabilistic classifier

Using a probabilistic classifier, we obtain an uncertainty measure for the group label and can stop the test early. We therefore employ a Bayesian network to solve the classification task. The resulting Bayesian network is illustrated in Fig. 10.3 (Klingler, 2013). We denote the group label by Y, *i.e.*, $Y = 1$ for control children and $Y = 0$ for children with DD. Furthermore, let f_i denote the value of feature F_i for a child m. We assume that the available features F_i, $i \in \{1, \ldots, F\}$ are conditionally independent given Y. Note that this assumption is valid because of our feature selection step (described in Sec. 10.2): We cluster features into groups of similar features and choose only one representative feature per cluster. Correlation between the clusters therefore tends to be low.

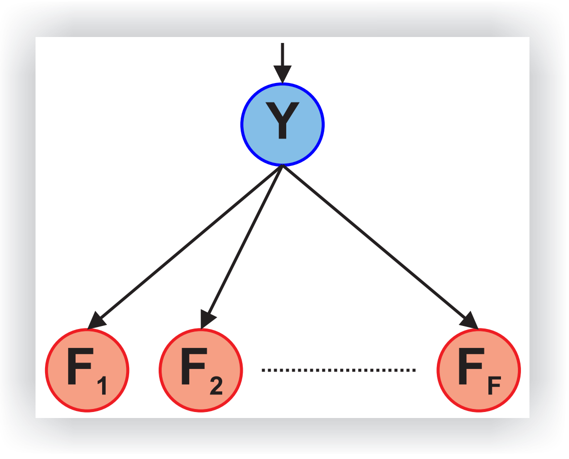


Figure 10.3: Structure of the graphical model for the probabilistic classifier (Klingler, 2013). The group label is denoted by Y (blue), while the F_i (red) represent the features. We assume the different features to be independent.

Given our assumptions, the probability of the network (for a child m) illustrated in Fig. 10.3 can be written as

$$p(f_1, \ldots, f_F, y) = \left(\prod_{i=1}^{F} p(f_i|y)\right) \cdot p(y), \tag{10.7}$$

where the probabilities $p(f_i|y)$ are drawn from a continuous or discrete probability distribution (depending on the feature type). We assume a normal distribution for the features **NSN**, **ESN**, **E**, **SN**, **O** and **L**. A Beta distribution is used for the performance feature **P** and a Gamma distribution for the answer time feature **AT**. Furthermore, feature **TM** employs a Poisson distribution. For each combination of feature type and probability distribution, we computed a maximum likelihood optimization on the data as well as the BIC. For each feature type, we then chose the probability distribution that best models the data, based on the BIC score. Parameters are given by the maximum likelihood optimization. The prior probability $p(y)$ can be determined by the estimated prevalence of DD (Shalev and von Aster, 2008). To classify a child m, we need to compute the posterior probability of the group label Y given N observed features f_i:

$$p(Y=1|f_1, \ldots, f_N) = \frac{\left(\prod_{i=1}^{N} p(f_i|Y=1)\right) \cdot p(Y=1)}{p(f_1, \ldots, f_N)} \tag{10.8}$$

Note that due to the independence assumption, we can deal with partial observability, *i.e.*, $N \leq F$. Furthermore, N grows with the number of observed features. After observing the first feature F_1, we can compute $p(Y=1|f_1)$. Having observed F_2, we infer $p(Y=1|f_1,f_2)$ and so on. The predicted group label $\hat{Y}$ can then be computed as

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

10.4.2 Online feature selection

We have already pre-selected a set of F features via clustering (see Sec. 10.2). Dealing with an adaptive classification, the online feature selection answers the following two questions:

1. In what order should we acquire the features? An optimal ordering would select the most informative features for the test first.
2. When should the model stop acquiring new features, *i.e.*, at which point in time should the model output the final answer?

Order of features

To determine the optimal ordering of the tasks, we have to compute the amount of group information contained in each feature. We prefer features, where the feature values differ a lot across the groups (DD and CC) and are similar within the group. To assess the quality of each feature F_i, we use a statistical test. Let $\mathbf{f_i^0}$ be the vector containing the feature values of the i^{th} feature for the children with DD, while $\mathbf{f_i^1}$ contains the values of the CC children. For each feature F_i, we perform a *two-sample t-test* to test if the mean values of the according normal distributions are equal. We therefore obtain a p-value p_i for each feature. We then order the features by sorting their p-values in ascending order, *i.e.*, the feature with the lowest p-value is asked first. We denote this criterion as the *significance decision*.

Another option for ordering the features (denoted as *influence decision*), is to select the features based on their possible influence on the outcome. Let $F_{obs} = \{F_1,...,F_t\}$ be the set of observed features after time t and let $\mathbf{f_{obs}}$ denote the values of the features in F_{obs}. The next feature that will be observed is F_{t+1}. Furthermore, let F_{fut} denote the set of unobserved features and $\mathbf{f_{fut}}$ the respective feature values. We choose the feature F_{t+1} that contradicts the current belief of the model the most: If the current estimate $p(Y=1|\mathbf{f_{obs}}) > \tau$, *i.e.*, the model believes that the child belongs to the control group, we pick F_{t+1} such that $p(Y=1|\mathbf{f_{obs}}, f_{t+1})$ is minimized. However, the values of the features in F_{fut} are not known. We therefore perform a worst case approximation and compute the index of the next feature based on the already observed values f_i from the training set

$$t+1 = \underset{i,F_i \in F_{fut}}{\arg\max} \max_{n \in \{1,\dots,N\}} p_{f_i^{(n)}}(Y=0|\mathbf{f_{obs}}, f_i^{(n)}), \tag{10.9}$$

where $n \in \{1,\dots,N\}$ are the indices of the children in the training set and $f_i^{(n)}$ denotes the value of feature F_i for child n from the training set. If the current belief of the model was that the child exhibits DD, we would pick F_{t+1} such that $p(Y=1|\mathbf{f_{obs}}, f_{t+1})$ is maximized. The formula for this maximization can be directly derived from Eq. (10.9).

Stopping criterion

We assessed three different possibilities for selecting a stopping criterion. The first simplest approach is the use of upper and lower thresholds. If the current belief of the model is either above the upper or below the lower threshold, we finish the test. We call this approach the *threshold stop*.

In a second approach (denoted as *expected worst stop*), we compute a worst case approximation to the set of future features F_{fut}. For simplification of notation, let $p_{(1|\mathbf{f_{all}})}$ denote $p(Y=1|\mathbf{f_{obs}},\mathbf{f_{fut}})$ in the following. To derive the approximation, we compute $p_{(1|\mathbf{f_{all}})}$ by using

$$p_{(1|\mathbf{f_{all}})} = \frac{(\prod_{i,F_i \in F_{obs}} p(f_i|Y=1)) \cdot (\prod_{i,F_i \in F_{fut}} p(f_i|Y=1)) \cdot p(Y=1)}{p(\mathbf{f_{obs}},\mathbf{f_{fut}})}. \tag{10.10}$$

To simplify the notation even more, we define

$$\alpha_y := \left(\prod_{i,F_i \in F_{obs}} p(f_i|y) \right), \qquad \beta_y := \left(\prod_{i,F_i \in F_{fut}} p(f_i|y) \right)$$

and can therefore rewrite Equation 10.10 as

$$p_{(1|\mathbf{f_{all}})} = \frac{\alpha_1 \cdot \beta_1 \cdot p(Y=1)}{\alpha_0 \cdot \beta_0 \cdot p(Y=0) + \alpha_1 \cdot \beta_1 \cdot p(Y=1)} = \frac{1}{1 + \frac{\alpha_0 \cdot \beta_0 \cdot p(Y=0)}{\alpha_1 \cdot \beta_1 \cdot p(Y=1)}}. \tag{10.11}$$

To compute the posterior probability of $p_{(1|\mathbf{f_{all}})}$, we therefore just need to calculate β_0 and β_1, all the other variables are already known. Again, we do not know the values of the features in F_{fut}. To estimate β_0 and β_1, we therefore again make use of feature values from the training set that maximally contradict the current belief state of the model. Let $\mathcal{I}$ be the set of unobserved feature indices in F_{fut}. For every feature $F_i \in F_{fut}$, the feature value can be taken from a different child n_i. Thus, the estimate for $\hat{\beta}_0$ and $\hat{\beta}_1$ is given by

$$\frac{\hat{\beta}_0}{\hat{\beta}_1} = \frac{\beta_0}{\beta_1}(\{n_1, \ldots, n_{|I|}\}) = \frac{\prod_{i,F_i \in F_{fut}} p_{f^{(n_i)}}(f_i|Y=0)}{\prod_{i,F_i \in F_{fut}} p_{f^{(n_i)}}(f_i|Y=1)} = \prod_{i,F_i \in F_{fut}} \frac{p_{f^{(n_i)}}(f_i|Y=0)}{p_{f^{(n_i)}}(f_i|Y=1)}, \quad (10.12)$$

where $f^{(n_i)}$ denotes the value of feature F_i for child n_i from the training set.

Let us assume that the current estimate of the model is $p(Y = 1|\mathbf{f_{obs}}) > \tau$. We therefore want to minimize $p_{(1|\mathbf{f_{all}})}$. From Eq. (10.11), we know that this is equal to maximizing the estimate $\hat{\beta}_0/\hat{\beta}_1$. To maximize this estimator, we proceed as follows: For every feature $F_i \in \mathcal{F}_{fut}$, we pick the feature values of a child n_i in the training set such that $(p_{f^{(n_i)}}(f_i|Y = 0))/(p_{f^{(n_i)}}(f_i|Y = 1))$ is maximized. From Equation 10.12 it follows that this procedure maximizes $\hat{\beta}_0/\hat{\beta}_1$. Having estimated $\hat{\beta}_0/\hat{\beta}_1$, we can compute $p_{(1|\mathbf{f_{all}})}$. If $p_{(1|\mathbf{f_{all}})} > \tau$, *i.e.*, the future (worst case) belief is the same as the current belief of the model, we can stop the test.

If the current belief of the model was that the child exhibits DD, we would select the feature values of each feature $F_i \in F_{fut}$ such that $p_{(1|\mathbf{f_{all}})}$ is maximized. The procedure for this maximization can be directly derived from Eq. (10.11) and Eq. (10.12).

The third stopping criterion (denoted as *next worst stop*) is similar to the second one, however, instead of estimating the worst case using all future features $F_i \in F_{fut}$, we look only at the next feature F_{t+1}. We therefore want to select the next feature F_{t+1} such that it maximally contradicts the current belief of the model. Again, the values of the features in F_{fut} are not known. We therefore perform a worst case approximation and compute the index of the next feature based on the already observed values f_i from the training set using Eq. (10.9). If the estimate $p_{(1|\mathbf{f_{all}})}$ does not contradict the current belief of the model, we can stop the test.

10.5 Assessment of static and adaptive classification

We evaluated the classification accuracy of the static and the adaptive approach on log files from the *BMBF-study* (described in Sec. 10.1) by using a *nested* 10-fold cross validation: We randomly split the data into 10 folds of equal size. We then divided each fold into a training set and a test set. The model selection, *i.e.*, the selection of the hyperparameters of the model was then performed individually on each training set using bootstrapping. For the static classification approach, hyperparameters are for example the choice of kernels or kernel parameters (such as σ in the Gaussian kernel). A hyperparameter for the adaptive classification if for example the choice of the probability distributions for the features. The fitted models were then (again individually) tested on each test set. By employing the nested validation, we do not only cross validate the performance of the model, but also the model selection process. A detailed introduction to nested validation can be found in (Hastie et al., 2001).

To assess the quality of our models, we evaluated the classification accuracy as well as its sensitivity and specificity. The sensitivity is computed as the ratio between the number of true positives (children correctly classified as having DD) and the total number of children with DD. We therefore measured, how good the model is at finding the children with DD. The specificity is defined as the ratio between the true negatives (children correctly classified as being CC) and the total number of CC. The specificity therefore describes the test's ability to exclude a condition (in our case DD) correctly.

10.5.1 Static model evaluation

To select the features for the SVM, we performed a clustering and chose only one representative feature per cluster. The optimal number of clusters k^* can be determined using the BIC, calculating the effective number of parameters as the normalized trace of the kernel transformation matrix used for the PC. The resulting BIC for 1 to 100 clusters is displayed in Fig. 10.4 (Klingler, 2013).

According to the BIC score the optimal number of clusters is $k^* = 2$. This result is not satisfactory, *i.e.*, selecting only two features leads to a bad performing classifier. We therefore limited the number of clusters to $k^* = 16$ based on the maximum test duration. Per feature, we included five tasks, which leads to 80 tasks in the test. In the following we will refer to this as *semi-automated feature selection*. The features selected using this approach are listed in Tab. 10.2 (Klingler, 2013). We also assessed the performance of manually, expert-based selected features: We extracted **NSN** and **ESN** (described in Tab. 10.1) based on the first 40 tasks the child solved during the training and on 40 other tasks starting from skill *Addition 1,1 TC* (see Fig. 3.4). We call this *manual feature selection* in the following. With this approach, the test has the same number of tasks (80), but we can include more features, *i.e.*, we can include all the skill and game dependent features along the chosen path.

In Sec. 10.2, we designed custom kernels for the different feature types. To show the benefits of these custom kernels, we compared the classification with the custom kernels to a classification employing a standard Gaussian kernel for all feature types. For this analysis, we performed the feature selection employing the *semi-automated*

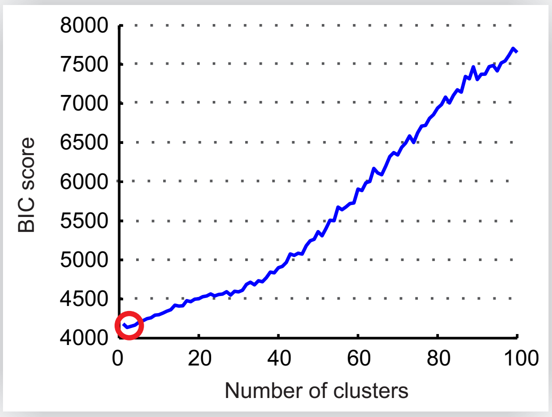


Figure 10.4: Model selection using the BIC score (Klingler, 2013). The optimal number of clusters is $k^* = 2$ (marked with a red circle). This result is not satisfactory as the selection of only two features leads to a bad classifier.

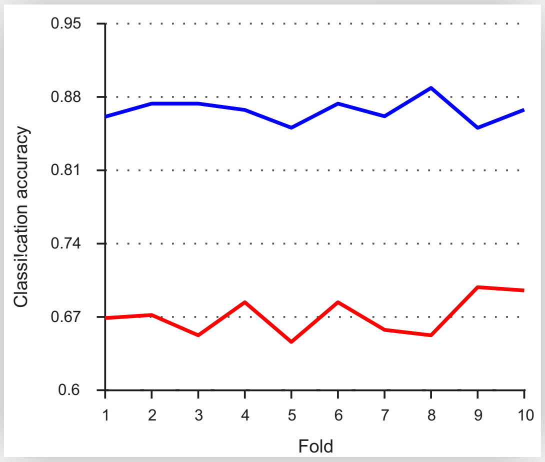


Figure 10.5: Classification accuracy over the different folds of the 10-fold cross validation employing custom kernels (blue) and standard kernels (red) (Klingler, 2013). Our custom designed kernels improve the classification accuracy.

feature selection approach. The resulting classification accuracy for the different kernels is displayed in Fig. 10.5 (Klingler, 2013). The SVM employing our custom kernels outperforms the model using standard kernels by a large margin.

Having selected the features as well as the kernels, we can evaluate the classification accuracy for the different combinations by employing the nested 10-fold cross

Table 10.2: List of features selected with the *semi-automated feature selection* approach (Klingler, 2013) along with corresponding games (see Sec. 3.2.3) and skills (see Fig. 3.4). The abbreviations of the feature types are consistent with Tab. 10.1.

Feature type	**Game**	**Skill**
AT	Calculator	*Addition 1,1 TC*
AT	Calculator	*Subtraction 1,1 TC*
AT	Calculator	*Addition 3,1 TC*
AT	Ordering	*Ordinal 2, 0-100*
AT	Calculator	*Subtraction 2,2*
AT	Calculator	*Subtraction 2,1*
P	Calculator	*Addition 1,1 TC*
P	Calculator	*Subtraction 1,1 TC*
P	Transfer	*Verbal/Concrete→Arabic, 0-1000*
P	Calculator	*Subtraction 1,1*
P	Transfer	*Verbal/Concrete→Arabic, 0-100*
P	Estimation	*Estimation*
P	Slide rule	*Support Subtraction 1,1*
TM	Landing	*Arabic→Numberline, 0-1000*
TM	Calculation	*Addition 2,1*
TM	Landing	*Verbal→Numberline, 0-100*
TM	Landing	*Verbal→Numberline, 0-10*
SN	Secret number	*Ordinal 3, 0-100*

validation. Applying a custom kernel combined with the *manual feature selection* exhibits a mean classification accuracy of 0.88 (SD $\sigma = 0.05$). When using the *semi-automated feature selection*, the resulting classification accuracy amounts to 0.86 (SD $\sigma = 0.04$). Employing standard Gaussian kernels for all feature types combined with a *manual feature selection* results in an average classification accuracy of 0.66 (SD $\sigma = 0.05$).

Table 10.3: Performance comparison of the Bayesian network classifier regarding different combinations of task ordering approaches and stopping criteria (Klingler, 2013). The best Bayesian network classifier reaches an accuracy of 0.92 after on average 35 tasks.

Feature ordering	**Stopping criterion**	**Accuracy (SD σ)**	**Sensitivity**	**Specificity**	**#tasks**
significance decision	next worst stop	0.92(0.05)	0.92	0.91	35
significance decision	threshold	0.91(0.04)	0.91	0.88	30
influence decision	expected worst stop	0.89(0.04)	0.92	0.88	75
significance decision	expected worst stop	0.89(0.05)	0.90	0.88	70
influence decision	threshold	0.87(0.06)	0.89	0.85	35
influence decision	next worst stop	0.87(0.89)	0.85	0.91	35

10.5.2 Evaluation of adaptive model

In order to select the features of the adaptive model, we presented different approaches regarding the ordering of the tasks as well as the stopping criterion (see Sec. 10.4.2). We evaluated the classification accuracy of the Bayesian network classifier by employing different combinations of feature ordering and stopping criteria. From Tab. 10.3 (Klingler, 2013) it becomes obvious that the best accuracy is reached by applying the *significance decision* to determine the ordering of the features and the *next worst stop* criterion to decide when the test can be finished. The last column of Tab. 10.3 displays the average number of tasks needed to come to a decision. The accuracy of the Bayesian network classifier (best case) is better than the accuracy reached with the static SVM classifier (0.88 (SD $\sigma = 0.05$)). Furthermore, the adaptive model needs on average 35 tasks to classify a child, while the static SVM classifier needs 80 tasks.

For the Bayesian network classifier, we used different distributions to model the data (described in Sec. 10.4.1). We compared the performance of a Bayesian network classifier using these fitted distributions for each feature type to a Bayesian network classifier employing only normal distributions. The ROC (receiver operating characteristics) curve of these two models is displayed in Fig. 10.6 (Klingler, 2013). For both models, we used all the features selected by the *semi-automated feature selection* approach. Again, modeling the data using different distributions proves to be beneficial.

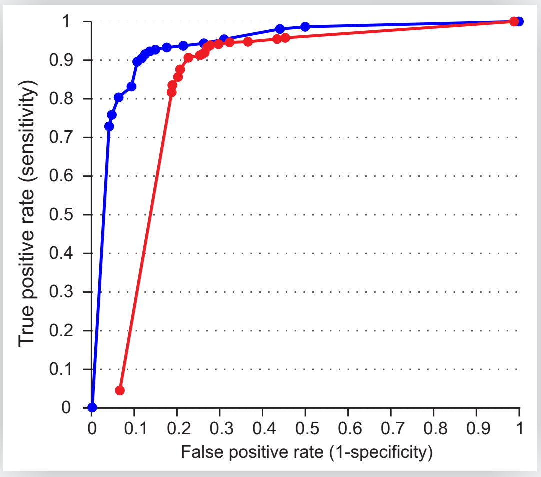


Figure 10.6: ROC curves for the Bayesian network classifier using normal distributions only to model the data (red) and using our fitted distributions (blue). Each point on the curves corresponds to a different probability threshold τ at which the model decides that a child is a control child (Klingler, 2013). Modeling the data with fitted distributions is beneficial for classification accuracy.

10.6 Discussion

In this chapter, we have introduced a static and an adaptive method for classifying children into two groups: One group with DD and one control group. The adaptive Bayesian network classifier outperforms the static SVM classifier regarding classification accuracy (SVM: 0.88 (SD $\sigma = 0.05$), Bayesian network: 0.92 (SD $\sigma = 0.05$)) as well as test length (SVM: 80 tasks, Bayesian network: 35 tasks). A reason for this might be that it is not always beneficial to pick as many features as possible for classification: Some features not differentiating well between the group labels might actually deteriorate classification accuracy. The Bayesian network classifier has a further advantage: By adapting the threshold τ used for deciding if a child belongs to the control group, we can tune the sensitivity and specificity values, *i.e.*, an expert can decide if the sensitivity or the specificity of the classifier is more important.

The performance of our classifier is comparable to that of existing computer-based screener tools regarding sensitivity and specificity. `Dyscalculium` (Beacham and Trott, 2005) reaches a sensitivity of 0.82 and a specificity of 0.92, while our best Bayesian network classifier exhibits a higher sensitivity (0.92) and a comparable specificity (0.91). Furthermore, the average test duration for our best Bayesian network classifier is low with 14 minutes.

Regarding the selected features, our test compares well to standard tests for DD. Our screener includes many of the skills (such as number comparison or spatial number representation) described in previous work (Butterworth et al., 2011). Furthermore, feature types such as performance, answer times and typical mistakes were employed in other screening tools as well (Beacham and Trott, 2005; Butterworth, 2003).

Our presented screening tool has two main limitations. First, the classifiers were trained and evaluated based on data from a user study. Log files of children contain six weeks of training data. Therefore, the data set at hand contains learning effects as well as adaptation effects (to the training environment). Second, the distribution between the group labels was close to a uniform distribution: 49 children were diagnosed with DD, while the other 40 children were CC children. The actual prevalence of DD is, however, estimated to be 3 – 6%. In a next step, the screening tool therefore needs to be evaluated in school classes, comparing its results to those of standardized tests.

CHAPTER 11

Conclusion

In this thesis, we presented a complete loop in the data-driven development of an intelligent tutoring system. We started by designing and implementing `Calcularis`, a computer-based training program for children with difficulties in learning mathematics. This included the complete development of all components of the system, *i.e.*, a curriculum along with appropriate games and a special design for numerical stimuli, a dynamic Bayesian network student model with a non-linear control algorithm, and a bug library containing typical errors and misconceptions of the domain. In a second step, we evaluated the first version of `Calcularis` in two user studies to prove its effectiveness. Based on the collected data from these studies, we assessed the quality of the student model and the control algorithm in a third step. By applying logistic regression, we validated the structure of the Bayesian network. Furthermore, we improved the existing student model on the basis of the collected log files: We suggested a constrained latent structured learning algorithm to improve prediction accuracy as well as a cluster-based prediction method for long-term predictions of student characteristics. Moreover, we also investigated possible extensions in engagement modeling. Finally, we extended `Calcularis` by a screening tool for developmental dyscalculia.

In the following, we will review the principle contributions of the thesis and discuss the limitations and further work.

11.1 Review of principal contributions

We developed the intelligent tutoring system `Calcularis` (described in Chapter 3) aimed at children with developmental dyscalculia or difficulties in learning mathematics. The curriculum of the software is based on actual research regarding neuro-cognitive models of numerical development (von Aster and Shalev, 2007). Furthermore, we developed a special design for numerical stimuli using colors, forms and topologies to represent properties of numbers. This design is intended to enhance the different number modalities and to strengthen the links between them. Moreover, the transfer of information through different channels stimulates perception and facilitates the retrieval of memory (Lehmann and Murray, 2005; Shams and Seitz, 2008). `Calcularis` features a dynamic Bayesian network student model (Murphy, 2002). Compared to previous work using Bayesian Knowledge Tracing (Corbett and Anderson, 1994; Koedinger et al., 1997), we are able to model the hierarchy and dependencies between the different skills of a learning domain. This graph structure also supports the non-linear control algorithm: In contrast to other systems (Conati et al., 2002; Koedinger et al., 1997; Gross and Vögeli, 2007), we allow backward movements in the hierarchy (going back to easier skills). The attached bug library also adds to this strategy: If the child commits typical errors, remediation skills, which are not necessarily direct precursors of the actual skill, are trained.

The evaluation of `Calcularis` in two user studies (detailed in Chapter 4) demonstrated the effectiveness of the program. Children improved significantly in addition and subtraction over a training period of six weeks regarding correctness and problem solution times. This decrease in problem solution times can be seen as a shift to increased fact retrieval (Geary et al., 1991; Lemaire and Siegler, 1995; Barrouillet and Fayol, 1998; Jordan et al., 2003). After three months of training, children also demonstrated a refined spatial number representation, confirming the results of previous studies (Siegler and Booth, 2004; Booth and Siegler, 2006, 2008; Halberda et al., 2008) which demonstrated significant correlations between arithmetical learning and the quality of numerical magnitude representation. Last but not least, children liked to play with the training program and reported that the training improved their mathematical abilities.

In Chapter 5, we validated the skill model of `Calcularis` using learning curves. We explored the potential parameter estimate biases that may result when fitting learning curves to data from tutoring systems that employ a mastery-learning mechanism. To analyze these biases, we investigated a wide set of modeling techniques and used the re-tests of previously mastered skills in `Calcularis` to check whether judged mastery is retained. We investigated variations of logistic regression models including the Additive Factors Model (Cen et al., 2007, 2008) and others that were explicitly designed to adjust for mastery-based data. We extensively analyzed properties and prediction accuracy of the different models and discussed implications for use and interpretation.

A further validation, namely the data-driven assessment of the quality of the student model and the controller, was conducted in Chapter 6. Our analyses demonstrated that the developed model adjusts rapidly to the knowledge state of the user. Furthermore, the non-linear design of the control algorithm is beneficial for learning.

In Chapter 7, we improved the accuracy of the dynamic Bayesian network student model (Murphy, 2002). We introduced a method called *constrained structured prediction with latent variables* for efficiently learning the parameters of a probabilistic graphical model. Our regularization of the parameter space via constraints improves prediction accuracy while ensuring an interpretable model. We demonstrated that our method outperforms the original model using expert parameters as well as parameters fitted with an unconstrained optimization. Furthermore, we conducted experiments on large-scale data sets from different learning domains and proved that our fitted Bayesian network models also significantly outperform previous work applying Bayesian Knowledge Tracing (Yudelson et al., 2013) in prediction accuracy.

We also introduced a *cluster-based prediction method* for the long-term prediction of mathematical characteristics of the students (see Chapter 8). By using a clustering and classification approach, we are able to predict the overall training outcome, external post-test results as well as specific mathematical problems of the children based on cluster information. Our results are in line with previous work (Baker et al., 2011; Pardos et al., 2012a,b; Trivedi et al., 2011; Gong et al., 2012; Trivedi et al., 2012) demonstrating that the use of cluster information improves prediction accuracy.

Furthermore, we also investigated the possibility of extending `Calcularis` to not only adapt to the knowledge, but also to engagement states of the users. In Chapter 9, we explored the possibility of a general framework for engagement learning, focusing on learning disabilities. We started our explorations based on an engagement model for spelling learning (Baschera et al., 2011). Our analyses demonstrated that the model can be generalized and thus applied to different learning domains. Some parts of the model, however, remain specific to the domain.

Finally, we also extended `Calcularis` with a diagnosis tool for computer-based screening of developmental dyscalculia based on input data. The static and adaptive classification approaches introduced in Chapter 10 exhibit a high accuracy (best adaptive classifier: 0.92) along with high sensitivity and specificity. The values are comparable to the performance of previous screening tools (Beacham and Trott, 2005). Furthermore, the average test duration when employing the adaptive classification algorithm is with 14 minutes very short.

11.2 Future work

The presented work covers different areas of interest, ranging from psychology to student modeling. In this section, we will investigate the limitations of this thesis and discuss potential future work in the different areas.

The user studies have demonstrated larger improvements in arithmetic operations (addition and subtraction) than in number representation. However, the different number representations as well as number understanding in general are very important for later mathematical performance (Landerl et al., 2004; Hannula and Lehtinen, 2005; Mazzocco and Thompson, 2005; Krajewski and Schneider, 2009). Furthermore, the observations of the study supervisors as well as our data analyses have shown that children do not acquire enough conceptual knowledge in the domain of *arithmetic operations*. We therefore plan the introduction of new games (and skills) in both areas of the training program. In the *number representations* area, games training number comparisons and structured sets will be designed. For the area of *arithmetic operations*, games introducing the concepts of addition and subtractions (such as a balance) as well as games for learning number facts could be added.

By applying our *latent structured prediction method*, we have already improved the prediction accuracy of the existing student model. However, our Bayesian network student model can handle only discrete binary variables, *i.e.*, all the skills of the model are assumed to be binary (mastered or not) and also task outcomes are either correct or wrong.

A possible extension would be the introduction of multidimensional observations nodes. This would allow us to model the task outcome more accurately. For the `Landing` game, we could for example apply three states: *correct*, *close*, *far away*. For addition and subtraction tasks, the answer time of the children could be captured by additional states such as *correct, but too slow*. Answer times in arithmetic operations are very important as they give an indication of the strategy used to solve the task. Children with developmental dyscalculia tend to apply immature strategies and exhibit a deficit in fact retrieval (Ostad, 1997, 1999).

A further step towards a more accurate model of student knowledge is the introduction of continuous task outcomes. In the `Landing` game, children have to indicate the position of a given number on a number line. Instead of using a binary variable to represent the task outcome, the distance from the correct position could be directly modeled using for example a normal distribution. Continuous task outcomes, however, represent a challenge for learning and inference as well as parameter interpretability. Instead of guess and slip probabilities, the mean and standard deviation of the distribution would be fitted. Despite these challenges, continuous (or multidimensional) task outcomes present an interesting direction for future work, as the increase of the representational power of the student model has the potential to improve prediction accuracy.

Prediction accuracy could, however, also be improved on the existing student model by adapting learning and prediction. Given the high diversity of students using a tutoring system, a clustering approach could be used to identify subgroups of children with similar mathematical learning patterns. Parameters could then be fit by subgroup. Our work on long-term prediction of mathematical characteristics (see Chapter 8) has already demonstrated the potential of a clustering and classification approach. Furthermore, clustering approaches have been used to improve the accuracy of Bayesian Knowledge Tracing models (Pardos et al., 2012b).

Another (similar) possibility for improving prediction accuracy of the existing model is the use of individualization techniques. Individualization techniques have been used to increase prediction accuracy of Bayesian Knowledge Tracing models (Pardos and Heffernan, 2010a; Wang and Heffernan, 2012; Yudelson et al., 2013; Wang and Beck, 2013). The resulting improvements were, however, only marginal.

Affective modeling is gaining importance in computer-based education due to the recognized influence of affective states on the learning outcome. Our work presented in engagement modeling was a pure theoretical exploration. We have shown that the engagement model for spelling learning (Baschera et al., 2011) could be extended to a general framework for modeling engagement dynamics. An obvious next step would therefore be to apply the suggested general framework to the `Calcularis` data set. However, a pure data-driven modeling approach requires an indicator function, *i.e.*, a function providing the affective state of the children. Therefore, a combination of sensor measurements and input data seems promising. Camera data can for example easily be collected by using integrated laptop cameras or webcams. Labeled camera data provide a direct measurement of the affective state and could replace the indicator function used in the engagement model for spelling learning (Baschera et al., 2011).

A further exciting possibility for measuring affective states would be an eye tracking system. Such a system would not only provide information about the student's attention, but could also be used to assess graphical features of `Calcularis`. With such a system, we could for example validate the special design for numerical stimuli used in `Calcularis`.

It would also be interesting to explore the possibility of a game version of `Calcularis`. Games could be embedded in a complex world. Furthermore, characters and a story could be added to the environment. A playful environment along with a storyline could increase motivation of children. Ideas for motivational elements could be found in literature on game design (Fullerton et al., 2004).

The latest addition to `Calcularis` is the diagnosis tool for developmental dyscalculia. However, this tool was trained and evaluated based on data from a user study and therefore the data set at hand contains learning effects as well as adaptation effects (to the training environment). Furthermore the distribution between the group labels was close to a uniform distribution, while the actual prevalence of developmental dyscalculia is estimated to be 3 – 6% (Shalev and von Aster, 2008). This tool therefore needs to be assessed in real life, *i.e.*, with students from school classes. This assessment will allow for a comparison of the computer-based screener with standardized tests. Depending on the number of participating children, the tool could be standardized as well. Currently, the tool is only available for children in 2^{nd} and 3^{rd} grade. An extension to younger children would be highly desirable: It has been shown that basic number processing and understanding capabilities acquired in preschool are essential for later mathematical performance (Landerl et al., 2004; Hannula and Lehtinen, 2005; Mazzocco and Thompson, 2005; Krajewski and Schneider, 2009). Therefore, early detection and intervention are highly important and would enable optimal support for children with developmental dyscalculia.

Bibliography

Ackerman, P. T. and Dykman, R. A. (1995). Reading-disabled students with and without comorbid arithmetic disability. *Developmental Neuropsychology*, 11:351–371.

Amershi, S. and Conati, C. (2009). Combining Unsupervised and Supervised Classification to Build User Models for Exploratory Learning Environments. *Journal of Educational Data Mining*, pages 18–71.

Ansari, D. and Dhital, B. (2006). Age-related Changes in the Activation of the Intraparietal Sulcus during Nonsymbolic Magnitude Processing: An Event-related Functional Magnetic Resonance Imaging Study. *Journal of Cognitive Neuroscience*, 18(11):1820–1828.

Arnold, D., Fisher, P., Doctoroff, G., and Dobbs, J. (2002). Accelerating math development in Head Start classrooms. *Journal of Educational Psychology*, 94:762–770.

Arroyo, I., Beal, C. R., Murray, T., Walles, R., and Woolf, B. P. (2004). Web-Based Intelligent Multimedia Tutoring for High Stakes Achievement Tests. In *Proceedings of Intelligent Tutoring Systems*, pages 468–477.

Arroyo, I. and Woolf, B. P. (2005). Inferring Learning and Attitudes from a Bayesian Network of Log File Data. In *Proceedings of Artificial Intelligence in Education*, pages 33–40.

Arsalidou, M. and Taylor, M. J. (2011). Is 2+2=4? Meta-analyses of brain areas needed for numbers and calculations. *NeuroImage*, 54(3):2382 – 2393.

Ashcraft, M. H. and Faust, M. W. (1994). Mathematics anxiety and mental arithmetic performance: An exploratory investigation. *Cognition & Emotion*, 8(2):97–125.

Bacon, A. M., Handley, S. J., and McDonald, E. L. (2007). Reasoning and dyslexia: a spatial strategy may impede reasoning with visually rich information. *British Journal of Psychology*, 98:79–92.

Badian, N. A. (1983). Arithmetic and nonverbal learning. In Myklebust, H. R., editor, *Progress in Learning Disabilities*, pages 235–264. Grune an Statton, New York.

Badian, N. A. (1999). Persistent arithmetic, reading, or arithmetic and reading disability. *Annals of Dyslexia*, 49:45–70.

Baker, C., Tenenbaum, J. B., and Saxe, R. (2005). Bayesian models of human action understanding. In *Proceedings of Neural Information Processing Systems*, pages 99–106.

Baker, F. B. (2001). *The basics of item response theory*. ERIC.

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

Baker, R. S., Corbett, A. T., and Koedinger, K. R. (2004). Detecting Student Misuse of Intelligent Tutoring Systems. In *Proceedings of Intelligent Tutoring Systems*, pages 531–540.

Baker, R. S. J. D., Pardos, Z. A., Gowda, S. M., Nooraei, B. B., and Heffernan, N. T. (2011). Ensembling predictions of student knowledge within intelligent tutoring systems. In *Proceedings of User Modelling, Adaptation and Personalization*, pages 13–24.

Barbaresi, M. J., Katusic, S. K., Colligan, R. C., Weaver, A. L., and Jacobsen, S. J. (2005). Math learning disorder: Incidence in a population-based birth cohort 1976-1982. *Ambulatory Pediatrics*, 5:281–289.

Barr, A., Beard, M., and Atkinson, R. C. (1976). The computer as a tutorial laboratory: the stanford {BIP} project. *International Journal of Man-Machine Studies*, 8(5):567 – 596.

Barrouillet, P. and Fayol, M. (1998). From Algorithmic Computing to Direct Retrieval: Evidence From Number and Alphabetic Arithmetic in Children and Adults. *Memory & Cognition*, 26:355–368.

Baschera, G.-M. (2011). *Modeling and Evaluation of Computer-Assisted Spelling Learning in Dyslexic Children*. PhD thesis, ETH Zurich.

Baschera, G. M., Busetto, A. G., Klingler, S., Buhmann, J., and Gross, M. (2011). Modeling Engagement Dynamics in Spelling Learning. In *Proceedings of Artificial Intelligence in Education*, pages 31–38.

Baschera, G.-M. and Gross, M. (2010a). Dybuster - Ein adaptives, multi-modales Therapiespiel für Legastheniker. In *Spielend Lernen*, pages 85–93. Fraunhofer Verlag.

Baschera, G. M. and Gross, M. (2010b). Poisson-Based Inference for Perturbation Models in Adaptive Spelling Training. *International Journal of Artificial Intelligence in Education*, 20(4):333–360.

Beacham, N. and Trott, C. (2005). Screening for dyscalculia within HE. *MSOR Connections*, 5:1–4.

Beck, J. E. (2005). Engagement tracing: Using response times to model student disengagement. In *Proceedings of Artificial Intelligence in Education*, pages 88–95.

Beck, J. E. and Sison, J. (2006). Using Knowledge Tracing in a Noisy Environment to Measure Student Reading Proficiencies. *International Journal of Artificial Intelligence in Education*, 16(2):129–143.

Besson, M., Schön, D., Moreno, S., Santos, A., and Magne, C. (2007). Influence of musical expertise and musical training on pitch processing in music and language. *Restorative Neurology and Neuroscience*, 25(3-4):399–410.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York, USA.

Boeck, P. (2008). Random Item IRT Models. *Psychometrika*, 73(4):533–559.

Booth, J. and Siegler, R. (2006). Developmental and individual differences in pure numerical estimation. *Developmental Psychology*, 42:189–201.

Booth, J. L. and Siegler, R. S. (2008). Numerical Magnitude Representations Influence Arithmetic Learning. *Child Development*, 79(4):1016–1031.

Bosse, M. L., Valdois, S., and Tainturier, M. J. (2003). Analogy without priming in early spelling development. *Reading and Writing*, 16:693–716.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breslow, N. E. and Clayton, D. G. (1993). Approximate Inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association*, 88(421):9–25.

Brunskill, E. and Russell, S. (2011). Partially Observable Sequential Decision Making for Problem Selection in an Intelligent Tutoring System. In *Proceedings of Educational Data Mining*, pages 327–328.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In Sleeman, D. H. and Brown, J. S., editors, *Intelligent Tutoring Systems*, pages 157–184.

Butterworth, B. (2003). *Dyscalculia screener*. NFER Nelson Publishing Company Ltd., London, UK.

Butterworth, B. (2005a). The development of arithmetical abilities. *Journal of Child Psycholgy and Psychiatry*, 46:3–18.

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

Butterworth, B., Varma, S., and Laurillard, D. (2011). Dyscalculia: From brain to education. *Science*, 332(6033):1049–1053.

Bynner, J. (1997). Basic skills in adolescents' occupational preparation. *Career Development Quarterly*, 27:777–786.

Campbell, R. (1985). When children write pseudowords to dictation. *Journal of Experimental Child Psychology*, 57:26–41.

Carey, S. (2001). Cognitive Foundations of Arithmetic: Evolution and Ontogenesis. *Mind and Language*, 16(1):35–55.

Carey, S. (2004). Bootstrapping & the origin of concepts. *Daedalus*, 133:59–68.

Caruana, R. and Niculescu-Mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 161–168.

Cassar, M. and Treiman, R. (1997). The beginnings of orthographic knowledge: children's knowledge of double letters in words. *Journal of Educational Psychology*, 89:631–644.

Cattell, R., Weiss, R., and Osterland, J. (1997). *Grundintelligenztest Skala 1 (CFT)*. Hogrefe Verlag, Göttingen.

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

Cen, H., Koedinger, K. R., and Junker, B. (2008). Comparing Two IRT Models for Conjunctive Skills. In *Proceedings of Intelligent Tutoring Systems*, pages 796–798.

Cohen Kadosh, R., Cohen Kadosh, K., Kaas, A., Henik, A., and Goebel, R. (2007a). Notation-dependent and -independent representations of numbers in the parietal lobes. *Neuron53*, pages 307–314.

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

Conati, C. (2002). Probabilistic assessment of users emotions in educational games. *Applied Artificial Intelligence*, 16:555–575.

Conati, C., Gertner, A., and VanLehn, K. (2002). Using Bayesian Networks to Manage Uncertainty in Student Modeling. *User Modeling and User-Adapted Interaction*, 12:371–417.

Cooper, D. G., Muldner, K., Arroyo, I., Woolf, B. P., and Burleson, W. (2010). Ranking Feature Sets for Emotion Models Used in Classroom Based Intelligent Tutoring Systems. In *Proceedings of User Modelling, Adaptation and Personalization*, pages 135–146.

Corbett, A. T. and Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278.

Dehaene, S. (2011). *The Number Sense: How the Mind Creates Mathematics*. Oxford University Press.

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

Demonet, J. F., Taylor, M. J., and Chaix, Y. (2004). Developmental dyslexia. *Lancet*, 363(9419):1451–1460.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Desmarais, M. C., Maluf, A., and Liu, J. (1995). User-expertise modeling with empirically derived probabilistic implication networks. *User Modeling and User-Adapted Interaction*, 5(3-4):283–315.

Dirks, E., Spyer, G., van Lieshout, E., and de Sonneville, L. (2008). Prevalence of combined reading and arithmetic disabilities. *Journal of Learning Disabilities*, 41:460–473.

Dowker, A. (2001). Intervention in numeracy: the development of a Numeracy Recovery project for young children with arithmetical difficulties. *Support for Learning*, 16:6–10.

Dowker, A. D. (2003). Interventions in numeracy. In Thompson, I., editor, *New directions in numeracy education*, pages 127–135. Open University Press, Buckingham (UK).

Edwards, L. (2003). Writing instruction in kindergarten: examining an emerging area of research for children with writing and reading difficulties. *Journal of Learning Disabilities*, 36(2):136–148.

Ehri, L. C. (2000). Learning to read and learning to spell: two sides of a coin. *Topics in Language Disorders*, 20:19–36.

Falmagne, J.-C., Koppen, M., Villano, M., Doignon, J.-P., and Johannesen, L. (1990). Introduction to Knowledge Spaces: How to Build, Test, and Search Them. *Psycholgical Review*, 197(2):201–224.

Field, A. (2009). *Discovering statistics using SPSS*. Sage.

Fletcher, J. M. (2005). Predicting math outcomes: Reading predictors and comorbidity. *Journal of Learning Disabilities*, 38:308–312.

Fletcher-Flinn, C. M. and Gravatt, B. (1995). The Efficacy of Computer Assisted Instruction (CAI): A Meta-Analysis. *Journal of Research on Computing in Education*, 27:48–61.

Frank, M. C. and Tenenbaum, J. B. (2010). Three ideal observer models for rule learning in simple languages. *Cognition*, 120(3):360–371.

Fuchs, L. S., Fuchs, D., Hamlet, C. L., Powell, S. R., Capizzi, A. M., and Seethaler, P. M. (2006). The Effects of Computer-Assisted Instruction on Number Combination Skill in At-Risk First Graders. *Journal of Learning Disabilities*, 39(5):467–475.

Fullerton, T., Swain, C., and Hoffman, S. (2004). *Game Design Workshop: Designing, Prototyping, and Playtesting Games*. CMP Books.

Gaab, N., Gabrieli, J. D. E., Deutsch, G. K., Tallal, P., and Temple, E. (2007). Neural correlates of rapid auditory processing are disrupted in children with developmental dyslexia and ameliorated with training: an fMRI study. *Restorative Neurology and Neuroscience*, 25(3-4):295–310.

Galaburda, A. M., LoTurco, J., Ramus, F., Fitch, R. H., and Rosen, G. D. (2006). From genes to behavior in developmental dyslexia. *Nature Neuroscience*, 9(10):1213–1217.

Galaburda, A. M., Sherman, G. F., Rosen, G. D., Aboitiz, F., and Geschwind, N. (1985). Developmental dyslexia: four consecutive patients with cortical anomalies. *Annals of Neurology*, 18(2):222–233.

Geary, D. C. (2004). Mathematics and learning disabilities. *Journal of Learning Disabilities*, 37(1):4–15.

Geary, D. C., Brown, S. C., and Samaranayake, V. A. (1991). 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.

Germanò, E., Gagliano, A., and Curatolo, P. (2010). Comorbidity of ADHD and dyslexia. *Developmental Neuropsychology*, 35(5):475–493.

Gerster, H. (1982). *Schülerfehler bei schriftlichen Rechenverfahren - Diagnose und Therapie*. Herder.

Gong, Y., Beck, J. E., and Ruiz, C. (2012). Modeling multiple distributions of student performances to improve predictive accuracy. In *Proceedings of User Modelling, Adaptation and Personalization,* pages 102–113.

González-Brenes, J. P. and Mostow, J. (2012a). Dynamic Cognitive Tracing: Towards Unified Discovery of Student and Cognitive Models. In *Proceedings of Educational Data Mining*, pages 49–56.

González-Brenes, J. P. and Mostow, J. (2012b). Topical Hidden Markov Models for Skill Discovery in Tutorial Data. *NIPS - Workshop on Personalizing Education With Machine Learning*.

Goswami, U. (2003). Why theories about developmental dyslexia require developmental designs. *Trends in Cognitive Sciences*, 7(12):534–540.

Griffin, S., Case, R., and Siegler, R. (1994). Rightstart: Providing the central conceptual prerequisites for the first formal learning of arithmetic to students at risk for school failure. In McGilly, K., editor, *Classroom lessons: Integrating cognitive theory and classroom practice*. Cabridge, MA: MIT Press.

Gross, M. and Vögeli, C. (2007). A Multimedia Framework for Effective Language Training. *Computer & Graphics*, 31:761–777.

Haffner, J., Baro, K., Parzer, P., and Resch, F. (2005). Heidelberger Rechentest (HRT): Erfassung mathematischer Basiskomptenzen im Grundschulalter.

Haghir Chehreghani, M., Busetto, A. G., and Buhmann, J. M. (2012). Information theoretic model validation for spectral clustering. In *Proceedings of Artificial Intelligence and Statistics*, pages 495–503.

Halberda, J., Mazzocco, M. M. M., and Feigenson, L. (2008). Individual differences in non-verbal number acuity correlate with maths achievement. *Nature*, 455(7213):665–668.

Hannula, M. M. and Lehtinen, E. (2005). Spontaneous focusing on numerosity and mathematical skills of young children. *Learning and Instruction*, 15(3):237–256.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning: data mining, inference, and prediction.* Springer, New York, USA.

Hatcher, P. J., Hulme, C., Miles, J. N., Carroll, J. M., Hatcher, J., and Gibbs, S. (2006). Efficacy of small group reading intervention for beginning readers with reading-delay: a randomized controlled trial. *Journal of Child Psychology and Psychiatry*, 47(8):820–827.

Hazan, T. and Urtasun, R. (2010). A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction. In *Proceedings of Neural Information Processing Systems*, pages 838–846.

Heraz, A. and Frasson, C. (2009). Predicting Learner Answers Correctness Through Brainwaves Assesment and Emotional Dimensions. In *Proceedings of Artificial Intelligence in Education*, pages 49–56.

Hilte, M. and Reitsma, P. (2011). Effects of explicit rules in learning to spell open- and closed-syllable words. *Learning and Instruction*, 21(1):34–45.

Hofmann, T. and Buhmann, J. M. (1997). Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(1):1–14.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., and Rommelse, K. (1998). The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 256–265.

Jarušek, P. and Pelánek, R. (2012). Analysis of a Simple Model of Problem Solving Times. In *Proceedings of Intelligent Tutoring Systems*, pages 379–388.

Johns, J. and Woolf, B. (2006). A Dynamic Mixture Model to Detect Student Motivation and Proficiency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 163–168.

Jordan, N. C., Hanich, L., and Uberti, H. Z. (2003). The development of arithmetic concepts and skills. recent research and theory. In Baroody, A. and Dowker, A., editors, *Mathematical thinking and learning disabilities*, pages 359–389. Erlbaum, Mahwah (NJ).

Kardan, S. and Conati, C. (2011). A framework for capturing distinguishing user interaction behaviours in novel interfaces. In *Proceedings of Educational Data Mining*, pages 159–168.

Kardan, S. and Conati, C. (2013). Comparing and Combining Eye Gaze and Interface Actions for Determining User Learning with an Interactive Simulation. In *Proceedings of User Modelling, Adaptation and Personalization*, pages 215–227.

Karmiloff-Smith, A. (1992). *Beyond Modularity*. MIT Press.

Käser, T., Busetto, A. G., Baschera, G.-M., Kohn, J., Kucian, K., von Aster, M., and Gross, M. (2012). Modelling and optimizing the process of learning mathematics. In *Proceedings of Intelligent Tutoring Systems*, pages 389–398.

Kass, R. (1989). Student modeling in intelligent tutoring systemsimplications for user modeling. In *User models in dialog systems*, pages 386–410. Springer.

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

Kaufmann, L., Delazer, M., Pohl, R., Semenza, C., and Dowker, A. (2005). Effects of a specific numeracy educational program in kindergarten children: A pilot study. *Educational Research and Evaluation*, 11(5):405–431.

Kaufmann, L., Handl, P., and Thöny, B. (2003). Evaluation of a numeracy intervention program focusing on basic numerical knowledge and conceptual knowledge: A pilot study. *Journal of Learning Disabilities*, 35:564–573.

Kaufmann, L. and von Aster, M. (2012). The Diagnosis and Management of Dyscalculia. *Dtsch Arztebl Int*, 109:767–778.

Kaufmann, L., Wood, G., Rubinsten, O., and Henik, A. (2011). Meta-Analyses of Developmental fMRI Studies Investigating Typical and Atypical Trajectories of Number Processing and Calculation. *Developmental Neuropsychology*, 36(6):763–787.

Khalili, A. and Shashaani, L. (1994). The effectiveness of computer applications: A meta-analysis. *Journal of Research on Computing in Education*, 27:48–61.

Kim, E.-S., Noh, Y.-K., and Zhang, B.-T. (2012). Learning-style recognition from eye-hand movement using a dynamic Bayesian network. *NIPS Workshop on Personalizing Education on Machine Learning*.

King, W., Giess, S., and Lombardino, L. (2007). 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.

Klingberg, T., Fernell, E., Olesen, P. J., Johnson, M., Gustafsson, P., Dahlström, K., Gillberg, C. G., Forssberg, H., and Westerberg, H. (2005). Computerized Training of Working Memory in Children With ADHD - A randomized, controlled trial. *Journal of the American Academy of Child & Adolescent Psychiatry*, 44(2):177–186.

Klingler, S. (2013). A screening tool for children at risk of developmental dyscalculia. Master thesis, ETH Zurich.

Koedinger, K., Baker, R., Cunningham, K., Skogsholm, A., Leber, B., and Stamper, J. (2010). A Data Repository for the EDM community: The PSLC DataShop. In *Handbook of Educational Data Mining*. CRC Press, Boca Raton, FL.

Koedinger, K. and McLaughlin, E. (2010). Seeing language learning inside the math: Cognitive analysis yields transfer. In *Proc. of the 32nd Annual Conference of the Cognitive Science Society*, pages 471–476.

Koedinger, K., Stamper, J., McLaughlin, E., and Nixon, T. (2013). Using Data-Driven Discovery of Better Student Models to Improve Student Learning. In *Proceedings of Artificial Intelligence in Education*, pages 421–430.

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

Kort, B., Reilly, R., and Picard, R. W. (2001). An Affective Model of Interplay Between Emotions and Learning: Reengineering Educational Pedagogy - Building a Learning Companion. *Advanced Learning Technologies*, pages 43–46.

Krajewski, K. and Schneider, W. (2009). Early development of quantity to number-word linkage as a precursor of mathematical school achievement and mathematical difficulties: Findings from a four-year longitudinal study. *Learning and Instruction*, 19(6):513–526.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. (2006). Factor Graphs and the Sum-product Algorithm. *IEEE Trans. Inf. Theor.*, 47(2):498–519.

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

Kucian, K. and Kaufmann, L. (2009). A developmental model of number representation. *Behavioral and Brain Sciences*, 32:313–373.

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

Kucian, K., von Aster, M., Loenneker, T., Dietrich, T., and Martin, E. (2008). Development of Neural Networks for Exact and Approximate Calculation: A fMRI Study. *Developmental Neuropsychology*, 33(4):447–473.

Kujala, T., Karma, K., Ceponiene, R., Belitz, S., Turkkila, P., and Tervaniemi, M. (2001). Plastic neural changes and reading improvement caused by audiovisual training in readingimpaired children. *Proceedings of the National Academy of Sciences USA*, 98(18):10509–10514.

Kulik, C.-L. C. and Kulik, J. A. (1991). Effectiveness of computer-based instruction: An updated analysis. *Computers in Human Behavior*, 7:75–94.

Kulik, J. (1994). Meta-analytic studies of findings on computer-based instruction. In Baker, E. L. and O'Neil, H. F., editors, *Technology assessment in education and training*. LEA publishers, Hillsdale (N.J.).

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86.

Kullik, U. (2004). Computerunterstützte Rechentrainingsprogramme. In Lauth, G., Grünke, M., and Brunstein, J. C., editors, *Interventionen bei Lernstörungen*, pages 329–337. Hogrefe, Göttingen.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289.

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

Landerl, K. and Reitsma, P. (2005). Phonological and morphological consistency in the acquisition of vowel duration spelling in Dutch and German. *Journal of Experimental Child Psychology*, 92:322–344.

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

Lemaire, P. and Siegler, R. S. (1995). Four Aspects of Strategic Change: Contributions to Children's Learning Of Multiplication. *Journal of Experimental Psychology: General*, 124:83–97.

Lenhard, A., Lenhard, W., Schug, M., and Kowalski, A. (2011). Computerbasierte Mathematikförderung mit den Rechenspielen mit Elfe und Mathis I. Vorstellung und Evaluation eines Computerprogramms für Erst- bis Drittklässler. *Zeitschrift für Entwicklungspsychologie und Pädagogische Psychologie*, 43(2):79–88.

Lewis, C., Hitch, G. J., and Walker, P. (1994). 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.

Li, Q. and Ma, X. (2010). A Meta-analysis of the Effects of Computer Technology on School Students Mathematics Learning. *Educational Psychology Review*, 22(3):215–243.

Lorusso, M. L., Facoetti, A., Paganoni, P., Pezzani, M., and Molteni, M. (2006). Effects of visual hemisphere-specific stimulation versus reading-focused training in dyslexic children. *Neuropsychological Rehabilitation*, 16(2):194–212.

Marsh, G., Friedman, M., Welch, V., and Desberg, P. (1980). The development of strategies in spelling. In Frith, U., editor, *Cognitive processes in spelling*, pages 339–354. Academic, London.

Martinet, C., Valdois, S., and Fayol, S. (2004). Lexical knowledge develops from the beginning of literacy acquisition. *Cognition*, 91:11–22.

Mayo, M. and Mitrovic, A. (2001). Optimising ITS Behaviour with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education*, 12:124–153.

Mazzocco, M. M. and Thompson, R. E. (2005). Kindergarten Predictors of Math Learning Disability. *Learning Disabilities Research & Practice*, 20(3):142–155.

Minka, T. P. and Qi, Y. A. (2003). Tree-structured Approximations by Expectation Propagation. In *Proceedings of Neural Information Processing Systems*.

Mooij, J. M. (2010). libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173.

Mostow, J., Hauptmann, A. G., Chase, L., and Roth, S. F. (1993). Towards a Reading Coach that Listens: Automated Detection of Oral Reading Errors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 392 – 397.

Muir, M. and Conati, C. (2012). An Analysis of Attention to Student-Adaptive Hints in an Educational Game. In *Proceedings of Intelligent Tutoring Systems*, pages 112–122.

Murphy, K. P. (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. `http://www.cs.ubc.ca/~murphyk/Thesis/thesis.html`.

Murphy, M. M., Mazzocco, M. M. M., Hanich, L. B., and Early, M. C. (2007). Cognitive characteristics of children with mathematics learning disability (mld) vary as a function of the cutoff criterion used to define mld. *Journal of Learning Disabilities*, 40(5):458–478.

Murray, R., Ritter, S., Nixon, T., Schwiebert, R., Hausmann, R., Towle, B., Fancsali, S. E., and Vuong, A. (2013). Revealing the Learning in Learning Curves. In *Proceedings of Artificial Intelligence in Education*, pages 473–482.

Mussolin, C., De Volder, A., Grandin, C., Schlogel, X., Nassogne, M. C., and Noel, M. (2010). Neural correlates of symbolic number comparison in developmental dyscalculia. *Journal of Cognitive Neuroscience*, 22(5):860–874.

Naglieri, J. A. and Johnson, D. (2000). Effectiveness of a Cognitive Strategy Intervention in Improving Arithmetic Computation Based on the PASS Theory. *Journal of Learning Disabilities*, 33(6):591–597.

Nation, K. and Hulme, C. (1996). The automatic activation of sound-letter knowledge: an alternative interpretation of analogy and priming effects in early spelling development. *Journal of Experimental Child Psychology*, 63:416–435.

Nation, K. and Hulme, C. (1998). The role of analogy in early spelling development. In Hulme, C. and Joshi, R. M., editors, *Reading and spelling: Development and disorders*, pages 433–445. Erlbaum, Mahwah, NJ.

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

Nieder, A. (2012). Supramodal numerosity selectivity of neurons in primate prefrontal and posterior parietal cortices. *Proceedings of the National Academy of Sciences*, 109(29):11860–11865.

Noël, M.-P. and Rousselle, L. (2011). Developmental Changes in the Profiles of Dyscalculia: An Explanation Based on a Double Exact-and-Approximate Number Representation Model. *Frontiers in Human Neuroscience*, 5.

O'Shaughnessy, T. E. and Swanson, H. L. (2000). A comparison of two reading interventions for children with reading disabilities. *Journal of Learning Disabilities*, 33(3):257–277.

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

Ostad, S. A. (1998). Comorbidity between mathematics and spelling difficulties. *Logopedics Phoniatrics Vocology*, 23:145–154.

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

Pacton, S., Perruchet, P., Fayol, M., and Cleeremans, A. (2001). Implicit learning out of the lab: the case of orthographic regularities. *Journal of Experimental Psychology: General*, 130:401–426.

Pardos, Z. A., Gowda, S. M., Baker, R. S., and Heffernan, N. T. (2012a). The sum is greater than the parts: ensembling models of student knowledge in educational software. *SIGKDD Explor. Newsl.*, 13(2):37–44.

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

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

Pardos, Z. A., Trivedi, S., Heffernan, N. T., and Sárközy, G. N. (2012b). Clustered knowledge tracing. In *Proceedings of Intelligent Tutoring Systems*, pages 405–410.

Pavlik, P. I., Cen, H., and Koedinger, K. R. (2009). Performance Factors Analysis - A New Alternative to Knowledge Tracing. In *Proceedings of Artificial Intelligence in Education*, pages 531–538.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pelleg, D. and Moore, A. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *Proceedings of the International Conference on Machine Learning*, pages 727–734.

Piech, C., Sahami, M., Koller, D., Cooper, S., and Blikstein, P. (2012). Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*.

Price, G. R., Holloway, I., Räsänen, P., Vesterinen, M., and Ansari, D. (2007). Impaired parietal magnitude processing in developmental dyscalculia. *Current Biology*, 17.

Rafferty, A. N., Brunskill, E., Griffiths, T. L., and Shafto, P. (2011). Faster teaching by POMDP planning. In *Proceedings of Artificial Intelligence in Education*, pages 280–287.

Rafferty, A. N., Davenport, J., and Brunskill, E. (2013). Estimating Student Knowledge from Paired Interaction Data. In *Proceedings of Educational Data Mining*, pages 260–263.

Rafferty, A. N., Lamar, M., and Griffiths, T. L. (2012). Inferring learners' knowledge from observed actions. In *Proceedings of Educational Data Mining*, pages 226–227.

Rafferty, A. N. and Yudelson, M. (2007). Applying Learning Factors Analysis to Build Stereotypic Student Models. In *Proceedings of Artificial Intelligence in Education*, pages 697–698.

Räsänen, P., Salminen, J., Wilson, A. J., Aunio, P., and Dehaene, S. (2009). Computer-assisted intervention for children with low numeracy skills. *Cognitive Development*, 24(4):450–472.

Rau, M. A., Aleven, V., and Rummel, N. (2009). Intelligent Tutoring Systems with Multiple Representations and Self-Explanation Prompts Support Learning of Fractions. In *Proceedings of Artificial Intelligence in Education*, pages 441–448.

Resnick, L. B. (1984). A Developmental Theory of Number Understanding.

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

Rivera, S. M., Reiss, A. L., Eckert, M. A., and Menon, V. (2005). Developmental Changes in Mental Arithmetic: Evidence for Increased Functional Specialization in the Left Inferior Parietal Cortex. *Cerebral Cortex*, 15(11):1779–1790.

Robichon, F., Besson, M., and Habib, M. (2002). An electrophysiological study of dyslexic and control adults in a sentence reading task. *Biological Psychology*, 59(1):29–53.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Roth, V., Laub, J., Kawanabe, M., and Buhmann, J. M. (2003). Optimal Cluster Preserving Embedding of Non-metric Proximity Data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(12):1540–1551.

Rubinsten, O. and Henik, A. (2005). Automatic activation of internal magnitudes: a study of developmental dyscalculia. *Neuropsychology*, 19:641–648.

Rubinsten, O. and Sury, D. (2011). Processing Ordinality and Quantity: The Case of Developmental Dyscalculia. *PLoS ONE*, 6(9):e24079.

Rubinsten, O. and Tannock, R. (2010). Mathematics anxiety in children with developmental dyscalculia. *Behavioral and Brain functions*, 6:46.

Russeler, J., Gerth, I., and Munte, T. F. (2006). 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.

Santos, A., Joly-Pottuz, B., Moreno, S., Habib, M., and Besson, M. (2007). Behavioural and event-related potentials evidence for pitch discrimination deficits in dyslexic children: improvement after intensive phonic intervention. *Neuropsychologia*, 45(5):1080–1090.

Schoppek, W. and Tulis, M. (2010). Enhancing Arithmetic and Word-Problem Solving Skills Efficiently by Individualized Computer-Assisted Practice. *The Journal of Educational Research*, 103(4):239–252.

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

Schwing, A. G., Hazan, T., Pollefeys, M., and Urtasun, R. (2012). Efficient Structured Prediction with Latent Variables for General Graphical Models. In *Proceedings of the International Conference on Machine Learning*.

Shalev, R. and von Aster, M. G. (2008). Identification, classification, and prevalence of developmental dyscalculia. *Enc. of Language and Literacy Development*, pages 1–9.

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

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal Estimated Sub-Gradient Solver for SVM. In *Proceedings of the International Conference on Machine Learning*, pages 807–814.

Shams, L. and Seitz, A. R. (2008). Benefits of multisensory learning. *Trends in Cognitive Science*, 12:411–417.

Shaywitz, B. A., Fletcher, J. M., and Shaywitz, S. E. (1994). A Conceptual Framework for Learning Disabilities and Attention-Deficit/Hyperactivity Disorder. *Canadian Journal of Special Education*, 9(3):1–32.

Shaywitz, B. A., Shaywitz, S. E., Blachman, B. A., Pugh, K. R., Fulbright, R. K., and Skudlarski, P. (2004). Development of left occipitotemporal systems for skilled reading in children after a phonologically-based intervention. *Biological Psychiatry*, 55(9):926–933.

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

Shute, V. and Psotka, J. (1994). Intelligent Tutoring Systems: Past, Present, and Future. Technical report, Human Resources Directorate, Manpower and Personnel Research Division, Air Force Material Command.

Siegler, R. S. and Booth, J. L. (2004). Development of numerical estimation in young children. *Child Development*, 75(2):428–444.

Spitzer, M. (2009). Ja, ich kann! Selbstbild, Selbstbejahung und nachhaltige Leistungsfähigkeit. *Nervenheilkunde*, 28:425–430.

Sriperumbudur, B. K. and Lanckriet, G. R. G. (2009). On the Convergence of the Concave-Convex Procedure. In *Proceedings of Neural Information Processing Systems*, pages 1759–1767.

Stamper, J. C. and Koedinger, K. R. (2011). Human-machine Student Model Discovery and Improvement Using DataShop. In *Proceedings of Artificial Intelligence in Education*, pages 353–360.

Statnikov, A., Wang, L., and Aliferis, C. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9(1):1–10.

Tallal, P. (2004). Improving language and literacy is a matter of time. *Nature Reviews Neuroscience*, 5(9):721–728.

Trivedi, S., Pardos, Z. A., and Heffernan, N. T. (2011). Clustering students to generate an ensemble to improve standard test score predictions. In *Proceedings of Artificial Intelligence in Education*, pages 377–384.

Trivedi, S., Pardos, Z. A., Sárközy, G. N., and Heffernan, N. T. (2012). Co-clustering by bipartite spectral graph partitioning for out-of-tutor prediction. In *Proceedings of Educational Data Mining*, pages 33–40.

Tseng, P. (1993). Dual coordinate ascent methods for non-strictly convex minimization. *Journal of Mathematical Programming*, 59(1-3):231–247.

Uther, M., Kujala, A., Huotilainen, M., Shtyrov, Y., and Naatanen, R. (2006). Training in Morse code enhances involuntary attentional switching to acoustic frequency: Evidence from ERPs. *Brain Research*, 1073-1074:417–424.

Vadasy, P. F., Jenkins, J. R., and Pool, K. (2000). Effects of tutoring in phonological and early reading skills on students at risk for reading disabilities. *Journal of Learning Disabilities*, 33(6):579–590.

Van De Rijt, B. A. M. and Van Luit, J. E. H. (1998). Effectiveness of the Additional Early Mathematics program for teaching children early mathematics. *Instructional Science*, 26(5):337–358.

van der Linden, W. J. (2006). A Lognormal Model for Response Times on Test Items. *Journal of Educational and Behavioral Statistics*, 31(2):181–204.

Van Luit, J. E. H. and Naglieri, J. A. (1999). Effectiveness of the MASTER Program for Teaching Special Children Multiplication and Division. *Journal of Learning Disabilities*, 32(2):98–107.

Vogel, S. and Ansari, D. (2012). Neurokognitive Grundlagen der typischen und atypischen Zahlenverarbeitung. *Lernen und Lernstörungen*, 1:135–149.

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

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

Wainwright, M. J. and Jordan, M. I. (2008). *Graphical models, exponential families, and variational inference*. Foundations and Trends in ML.

Wang, Y. and Beck, J. (2013). Class vs. Student in a Bayesian Network Student Model. In *Proceedings of Artificial Intelligence in Education*, pages 151–160.

Wang, Y. and Heffernan, N. T. (2012). The student skill model. In *Proceedings of Intelligent Tutoring Systems*, pages 399–404.

Weinert, F. and Schneider, W., editors (1999). *Individual development from 3 to 12: Findings from the Munich Longitudinal Study*. Cambridge University Press, New York, NY.

Weiss, R. (2006). *Grundintelligenztest Skala 2 - Revision - (CFT 20-R)*. Hogrefe Verlag, Göttingen.

Wiegerinck, W. and Heskes, T. (2003). Fractional Belief Propagation. In *Proceedings of Neural Information Processing Systems*, pages 438–445.

Wilson, A. and Dehaene, S. (2007). Number sense and developmental dyscalculia. In Coch, D., Dawson, G., and Fischer, K., editors, *Human behavior, learning, and the developing brain: Atypical Development*, pages 212–238. Guilford Press, New York.

Wilson, A. J., Dehaene, S., Dubois, O., and Fayol, M. (2009). Effects of an Adaptive Game Intervention on Accessing Number Sense in Low-Socioeconomic-Status Kindergarten Children. *Mind, Brain and Education*, 3(4):224–234.

Wilson, A. J., Dehaene, S., Pinel, P., Revkin, S. K., Cohen, L., and Cohen, D. (2006a). Principles underlying the design of ”The Number Race”, an adaptive computer game for remediation of dyscalculia. *Behavioral and Brain Functions*, 2:19.

Wilson, A. J., Revkin, S. K., Cohen, D., Cohen, L., and Dehaene, S. (2006b). An open trial assessment of ”The Number Race”, an adaptive computer game for remediation of dyscalculia. *Behavioral and Brain Functions*, 2:20.

Wilson, M. and De Boeck, P. (2004). Descriptive and explanatory item response models. In De Boeck, P. and Wilson, M., editors, *Explanatory item response models: A generalized linear and nonlinear approach*. New York: Springer.

World Health Organization (1993). *ICD-10: The international classification of diseases*. Geneva: World Health Organization, Vol. 10: Classification of mental and behavioural disorders edition.

Wright, R. J. (2003). A mathematics recovery: Program of intervention in early number learning. *Australian Journal of Learning Disabilities*, 8(4):6–11.

Yudelson, M. and Brunskill, E. (2012). Policy Building - An Extension To User Modeling. In *Proceedings of Educational Data Mining*, pages 181–191.

Yudelson, M. V., Koedinger, K. R., and Gordon, G. J. (2013). Individualized Bayesian Knowledge Tracing Models. In *Proceedings of Artificial Intelligence in Education*, pages 171–180.

Yuille, A. L. and Rangarajan, A. (2003). The Concave-Convex Procedure. *Neural Comptutation*, 15(4):915–936.

Ziegler, A., Konig, I. R., Deimel, W., Plume, E., Nothen, M. M., and Propping, P. (2005). Developmental dyslexia-recurrence risk estimates from a german bi-center study using the single proband sib pair design. *Human Heredity*, 59(3):136–143.