Applying Probabilistic Models for Knowledge Diagnosis and Educational Game Design

Anna Rafferty

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-61
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-61.html

May 12, 2014

Applying Probabilistic Models for Knowledge Diagnosis and Educational Game Design

by

Anna Noonan Rafferty

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Thomas L. Griffiths, Co-chair
Professor Dan Klein, Co-chair
Professor Pieter Abbeel
Professor Marcia C. Linn

Spring 2014

Applying Probabilistic Models for Knowledge Diagnosis and Educational Game Design

Abstract

Applying Probabilistic Models for Knowledge Diagnosis and Educational Game Design

by

Anna Noonan Rafferty

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Thomas L. Griffiths, Co-chair

Professor Dan Klein, Co-chair

Computer-based learning environments offer the potential for innovative assessments of student knowledge and personalized instruction for learners. However, there are a number of challenges to realizing this potential. Many psychological models are not specific enough to directly deploy in instructional systems, and computational challenges can arise when considering the implications of a particular theory of learning. While learners' interactions with virtual environments encode significant information about their understanding, existing statistical tools are insufficient for interpreting these interactions. This research develops computational models of teaching and learning and combines these models with machine learning algorithms to interpret learners' actions and customize instruction based on these interpretations. This approach results in frameworks that can be adapted to a variety of educational domains, with the frameworks clearly separating components that can be shared across tasks and components that are customized based on the educational content. Using this approach, this dissertation addresses three major questions: (1) How can one diagnose learners' knowledge from their behavior in games and virtual laboratories? (2) How can one predict whether a game will be diagnostic of learners' knowledge? and (3) How can one customize instruction in a computer-based tutor based on a model of learning in a domain?

The first question involves automatically assessing student knowledge via observed behavior in complex interactive environments, such as virtual laboratories and games. These environments require students to plan their behavior and take multiple actions to achieve their goals. Unlike in many traditional assessments, students' actions in these environments are not independent given their knowledge and each individual action cannot be classified as correct or incorrect. To address this issue, I develop a Bayesian inverse planning framework for inferring learners' knowledge from observing their actions. The framework is a variation of inverse reinforcement learning and uses Markov decision processes to model how people choose actions given their knowledge. Through behavioral experiments, I show that this framework can infer learners' stated beliefs, with accuracy similar to human observers, and that feedback based on the framework improves learning efficiency. To extend this framework to educational applications outside of the laboratory, I extended the inverse planning framework to diagnose students' algebra skills from worked solutions

to linear equations, separating different sources of mathematical errors. I tested the framework by developing an online algebra tutor that provides students with the opportunity to practice solving equations and automatically diagnoses their understanding after they have solved sufficient equations. Preliminary experiments demonstrate that Bayesian inverse planning provides a good fit for the majority of participants' behaviors, and that its diagnoses are consistent with results of a more conventional assessment.

The results of the previous studies showed that not all tasks result in learner behavior that can be used to perfectly diagnose knowledge. In many cases, actions may be ambiguous, resulting in a diagnosis that places some probability on one possible knowledge state and some probability on another. I developed an optimal game design framework to predict how much information will be gained by observing a player or players' actions if they were to play a particular game: gaining more information from a game means that the diagnosis is less ambiguous. This framework extends optimal experiment design methods in statistics. It can limit the trial and error necessary to create games for education and behavioral research by suggesting game design choices while still leveraging the skills of a human designer to create the initial design. Behavioral results from a concept learning game demonstrate that the predicted information gain is correlated with the actual information gain and that the best designs can result in twice as much information as an uninformed design.

The final part of this dissertation considers how to personalize instruction in a computer tutor, relying on knowledge about the domain and an estimate of the students' knowledge. This builds on the idea of assessing learners' knowledge from their actions and considers more broadly how to sequence assessment and personalized instruction. In a computer-based tutor, there may be a cost to time spent on assessment, as the time could alternatively have been spent allowing the learner to work through new material; however, this time spent on assessment may also be beneficial by providing information to allow the computer to choose material more effectively. I show that partially observable Markov decision processes can be used to model the tutoring process and decide what pedagogical action to choose based on a model of the domain and the learner. The resulting automated instructional policies result in faster learning of numeric concepts than baseline policies.

My research demonstrates that applying a computational modeling approach to a diverse set of problems in computer-assisted learning results in new machine learning algorithms for interpreting and responding to complex behavioral data. The frameworks developed in this research provide a systematic and scalable way to create personalized responses to learners. These frameworks show the potential of interactive educational technologies to not only provide content to learners but to infer their understanding from innovative assessments and provide personalized guidance and instruction.

To my parents

Contents

List of Figures

Acknowledgments

Graduate school has been an intense journey, challenging and strengthening me both academically and personally. The excitement about interdisciplinary work that I brought into graduate school has been enriched by developing a much greater depth of understanding in each of computer science, psychology, and education, as well as an appreciation for the difficulties and rewards of combining these topics. The opportunities to teach and to mentor undergraduate researchers as a graduate student have been extremely rewarding. The students and research assistants I have worked with have influenced both this dissertation, assisting in conducting many of the behavioral experiments, and my broader career goals, helping me to discover the joys of teaching.

My advisor, Tom Griffiths, has been invaluable in helping me to learn and grow as a scholar. His support of my decision to change research topics and focus on the projects I was most passionate about made graduate school immensely more enjoyable and meaningful. His advising has helped me concretely in learning to conduct and communicate research, as well as more broadly by providing an example for how to support students as they experience success, failure, and long periods of working things out. I am also grateful to Dan Klein for his mentoring of my teaching skills. He is an exemplary teacher, and as a teaching assistant in his class, I saw how motivating assignments could lead students to engage with materials in intense and exciting ways.

My graduate school work would have been much harder without the support of both wonderful collaborators and external funding. All of the work in this dissertation is the result of collaboration with Tom Griffiths; additionally, Michelle LaMar, Matei Zaharia, Patrick Shafto, and Emma Brunskill collaborated on parts of the work in Chapters 3-6. Fellow members of the CoCoSci Lab have provide constructive suggestions as well as a supportive atmosphere throughout graduate school. I appreciate the help of my dissertation committee in preparing and framing my work. I have been fortunate to receive funding from a National Science Foundation Graduate Fellowship, a National Defense Science and Engineering Graduate Fellowship, and a National Academy of Education/Spencer Dissertation Fellowship. These awards have granted me significant academic freedom, and the dissertation fellowship has provided access to a community of researchers.

My work on research projects outside of but related to my dissertation has been vital for developing my understanding of education research and its challenges. In my work with WestEd, I have been able to assist with classroom studies from inception to completion. Jodi Davenport has provided multitudes of support and advice for transitioning to a new role upon completion of graduate school. At Berkeley, my work in a course on designing education technologies led to a fruitful partnership with Marcia Linn. She helped me to recognize ways that I might contribute to technologies currently used in the classroom, leading to several years of collaboration with her group. I am grateful for her mentoring in bridging computer science and education, and for the wonderful community I have found in her lab, especially in my work with Libby Gerard.

Throughout graduate school, my family and friends have provided much needed support and encouragement. I thank my parents for always celebrating my achievements and teaching me to cope with struggle. Rob Bayer has provided a sounding board and an outside perspective when I have been most immersed in the details of my work. These close personal relationships help to revitalize me and bring meaning to my work.

Chapter 1

Introduction

In Neal Stephenson's novel *The Diamond Age* (Stephenson, 1995), a young girl is educated about the world around her through an illustrated primer. This primer is an interactive educational technology that customizes its lessons based on observing the girl's actions and approaches to problems, and provides information at appropriate times based on the girl's interactions. In addition to providing a curriculum of information to the reader, the primer allows readers to ask questions and provides responses based on the reader's current level of understanding. The lessons that the primer teaches the novel's protagonist are not limited to particular subjects, such as reading or math, but help her learn to make choices and solve problems; the ways in which it assesses these skills are in turn realistic scenarios that provide opportunities to apply her skills.

While the primer is an invention of science fiction, its personalization of educational content, interactive features, and ability to interpret and respond to learners' behaviors provide examples of how a truly intelligent educational technology might function. Creating responsive educational technologies has generated immense amounts of interest and attention. There is evidence that personalized interactions and curriculum can lead to more effective learning. For example, research on human tutoring has found that one-on-one instruction and mastery learning can result in performance two-standard deviations beyond group instruction (Bloom, 1984). Proponents of computer-based learning environments suggest that the same learning effectiveness can be achieved through intelligent, automated tutors (Corbett, 2001); these tutors aim to provide material at the right level for an individual learner and to continue providing the learner with practice until she has mastered the material, just as the primer customizes its instruction and activities based on its user. However, creating effective educational technologies that can respond to learners and customize instruction requires addressing a number of challenges. Educational content must be developed that facilitates interaction, and there must be a way to automatically evaluate students' interactions with this content to provide guidance or make decisions about what activities a student should complete next. For this approach to scale to a wide variety of domains and activities, resulting in a primer that can teach learners about any topic of interest, it must be possible to systematically adapt the same algorithms to data and behavior with very different surface properties.

Recent work on educational technologies has begun to address some of these challenges. A number of large scale efforts have focused on creating content and distributing it to learners. Massive open online courses (MOOCs) enroll tens or hundreds of thousands of students in a single online class, with students watching lectures, completing homework and exams, and interacting with course staff and one another on discussion forums (Kolowich, 2013). Analysis of students' behavior in one MOOC showed that more total time was spent on lab questions and homework problems than watching lectures, demonstrating that MOOCs provide opportunities for problem solving and can allow students to construct understanding through active engagement (Breslow et al., 2013). While there is considerable disagreement about the eventual impact of MOOCs on traditional education, they have been successful at reaching large numbers of students, addressing aspects of the content delivery challenge (Martin, 2012; Meisenhelder, 2013; Pappano, 2012). However, MOOCs differ in many ways from the vision of the illustrated primer. They typically focus assessment on easily evaluated activities, such as multiple choice questions or questions with numeric answers. These activities make it possible for a single teacher to evaluate the multitude of students in a single course, but are not necessarily the most effective for helping learners. More interactive activities can encourage student motivation (Duschl, Schweingruberm, & Shouse, 2007), and lead to increased student learning (Mintzes, Wandersee, & Novak, 2005). Additionally, MOOCs typically do not personalize the course of instruction, requiring all students to complete the same sequence of activities and providing limited personalized guidance to students.

A number of online resources besides MOOCs have focused on creation of educational content and problem solving practice activities. For example, the Web-based Inquiry Science Environment (WISE; Linn, Lee, Tinker, Husic, & Chiu, 2006) provides online science inquiry activities that are designed to promote student learning and have been refined through iterative investigations. WISE provides some opportunities for automatically assessing students' responses outside of multiple choice questions, such as through the use of natural language technologies to score short answers (Leacock & Chodorow, 2003), but does not generally interpret and provide guidance on students' actions in freeform interactive activities. The website Khan Academy (Khan, 2006-2014) is another source of educational content, providing videos on a diverse array of topics and problem solving practice for selected domains. Khan Academy provides some customization of instruction by continuing to have students solve the same type of problem until the system predicts that they have mastered the relevant skill (Thompson, 2011).

Intelligent tutoring systems also frequently support mastery learning through adaptive instruction (Polson & Richardson, 2013). These systems create models of student learning, typically tracking mastery of several skills over the course of a student's engagement with the system (see (Desmarais & Baker, 2012) for an overview of modeling approaches). Groen and Atkinson (1966) provide an early example of creating a mathematical model for sequencing instruction, focused on learning flashcards. Later models have expanded on this work, but most modeling has continued to focus on structured domains and activities. In these domains, modeling is simplified because knowledge can be broken into independent skills, and structured activities permit linking a single response to a small number of these skills. One of the most common structured models is Bayesian knowledge tracing (Corbett & Anderson, 1995). Knowledge tracing is a Markov model of procedural skill acquisition that has been particularly successful, with applications including mathematics and reading skills (e.g., Beck & Sison, 2006; Corbett, 2001). While intelligent tutoring systems have been successful at improving students' skills, they still offer a very different model of engagement and learning than the illustrated primer. Activities where students make freeform choices are rarely modeled and used to diagnose understanding, and developing systems that do interpret these choices typically requires significant time and domain expertise.

In this dissertation, I address some of the challenges of creating an engaging, multi-domain resource like the primer. By using general-purpose algorithms and incorporating computational models of learning from psychology and cognitive science, I develop new frameworks for diagnosing people's understanding and balancing assessment and instruction. Computational models of learning provide a way to formalize domain-specific knowledge, such as the organization of topics in a domain, and to represent principles of human learning that may be less domain-specific, such as how people tend to generalize from limited information. I focus specifically on Bayesian probabilistic models of learning, which have been increasingly successful in cognitive science and psychology for modeling cognition (see Chater, Tenenbaum, & Yuille, 2006; Tenenbaum, Griffiths, & Kemp, 2006, for a partial overview). These models are generative and can incorporate new information incrementally, leading to flexible models for education. I apply the models to understanding more freeform behavior than is typically modeled in existing computer-based learning activities. I also reason about these models using decision-theoretic algorithms, providing a domain-independent way of customizing instruction given a particular model of learning. Interpreting learners' behaviors and making determinations about how and when to respond to these behaviors are necessary steps for building adaptive educational technologies that support learners without curtailing their activities to fit a particular structured path.

I begin by developing a generative inverse planning framework that models students' action planning to diagnose their understanding, drawing inferences about specific misunderstandings that a student may have. This model can be applied to interpret behavior in virtual environments and educational games, and unlike previous efforts, it is not limited to a specific environment and does not need to be trained using data from each learning environment and task. The framework relies on Markov decision processes to model the virtual environment or game; the flexibility of Markov decision processes ensures that the model, which is a variation of inverse reinforcement learning, can be applied to data from a wide array of different activities. I provide an overview of Markov decision processes in Chapter 2. I then describe our Bayesian inverse planning model for knowledge diagnosis, and present three experiments demonstrating that the model's inferences are about as accurate as those of a human observer and that these inferences can be used to guide personalized feedback (Chapter 3).[1]

I demonstrate the flexibility of Bayesian inverse planning by extending the algorithm to model students' algebra abilities. While solving linear equations appears to be very different than choosing actions in a game, the same structure can be applied to model both types of behavior. In Chapter 4, I use the algorithm to diagnose specific components of algebra understanding, separating the ability to perform accurate arithmetic from applying the rules of algebra. The algorithm is applied to data collected by an online tutor that I developed. The tutor enables people to solve

[1]Much of the work in this dissertation was conducted in collaboration, especially with my advisor, Thomas L. Griffiths. To acknowledge this collaboration, I use second-person pronouns throughout this dissertation when referring to joint intellectual contributions.

linear equations and uses these solutions to diagnose their understanding and direct them to appropriate resources to correct their misunderstandings. This tutor provides a platform for future investigations of how best to use detailed diagnoses of understanding to personalize guidance for learners, and helps to connect learners to the existing algebra resources that are likely to be most helpful to them. Through an experiment, we demonstrate that the model's diagnosis of separate mathematical skills is consistent with a more conventional assessment. The Bayesian inverse planning model is a new way to diagnose people's understanding based on their behaviors, and our applications of the algorithm demonstrate that its success is not limited to only a single domain or to activities where there are a small number of misunderstandings. The extension to algebra provides a template for other applications of the model to complex domains, developing solutions for coping with large environments and action spaces.

One of the strengths of using Bayesian inverse planning to model people's action choices is that the model is generative: it can be used to simulate what actions people would be likely to take if they had a particular understanding. I take advantage of this feature when addressing my second question: how can we automatically design games to be more diagnostic of people's knowledge or cognitive processes? Creating more diagnostic assessments is a common goal in traditional assessment: computer adaptive testing, for instance, selects questions that will be most effective at reducing uncertainty about a person's understanding (Van Der Linden & Glas, 2000). A similar idea is common in the design of traditional experiments, where optimal experiment design focuses on choosing designs that are as informative as possible about a question under investigation (see Atkinson, Donev, & Tobias, 2007, for an overview). Such issues are perhaps even more important in games, which are often not originally envisioned as assessments and which have complex structures that may make it difficult to intuitively select a good design for assessment. In addition to their use in education, games are of increasing interest in the behavioral sciences. Collecting data to address questions in psychology and cognitive science via games, rather than traditional experiments, can lead to increased participant motivation and engagement.

In Chapter 5, I adapt ideas from optimal experiment design to identify the game design that will be most informative, making use of Bayesian inverse planning to estimate the informativeness of individual game designs. To test this optimal game design method, we investigate the difficulty of learning different types of Boolean concepts. This area has been well-studied in previous work (e.g., Feldman, 2000; Griffiths, Christian, & Kalish, 2008; Nosofsky, Gluck, Palmeri, McKinley, & Glauthier, 1994; Shepard, Hovland, & Jenkins, 1961), allowing us to a compare the results of a game-based investigation to the results of typical laboratory experiments. We demonstrate that the estimates of a game's informativeness correlate with the true information gain when the games are played by human participants, with the best games gaining twice as much information as random games. Our results also point to the complexity of interpreting people's actions in interactive environments. People may bring their own motivations to these environments, such as wanting to better understand the game or meet some goal that is not highly rewarded by the incentive structure created by the game designers; while such issues could occur in traditional experiments, they are less likely due to simpler tasks. This complicates the analysis of results, but we show that inverse reinforcement learning techniques can be used to infer people's motivations and that taking these motivations into account results in more accurate predictions of informativeness.

The previous contributions focused primarily on assessment, with some attention to how assessment results might be used to personalize feedback. However, making use of assessments to customize educational resources is a complex task: time spent assessing the student must be balanced with time providing the student with new information. For example, it may not be necessary for a resource like the primer to fully identify which misunderstanding characterizes a student if it has recognized that some misunderstanding is present. Further, automated instructional systems must consider what examples and other instructional activities are most appropriate for a given learner at any given time, sequencing materials to optimize learning. In Chapter 6, I formalize the problem of tutoring a student as a partially observable Markov decision process (POMDP) in which the automated tutor must select individual items to achieve some learning objective. This representation integrates the domain model, student model, and model of student responses, and allows us to explore what differences in teaching practice are dictated by different assumptions about student learning. In general, previous work has not focused on this interaction between models of student learning, one's model of the domain, and choices of pedagogical actions. We test this model by using it to teach people two different types of categories. These experiments are the first instance of using a POMDP formalization to teach human learners. By considering three different learner models, we demonstrate the characteristic differences in policies that emerge from different assumptions.

Overall, these investigations demonstrate the power of applying Bayesian cognitive models to educational questions. Because the algorithms we have developed are not tailored to specific domains, they facilitate customization and interpretation of a broad range of activities, bringing us closer to the goal of a resource that can support learners across domains through rich analyses of the learners' interactions. Each model we develop relates to one of the roles of the primer: Bayesian inverse planning could be used to interpret learners' behaviors, optimal game design could be used to design scenarios for the primer to provide to the learner, and POMDPs could allow the primer to decide how to sequence new information for the learner and when to prompt the learner to demonstrate her knowledge. My approach to developing these algorithms formalizes the problems under investigation to make it easier to take advantage of advances in computer science and statistics for improving computer-based educational systems. In Chapter 7, I discuss future directions for this approach and summarize my contributions.

Chapter 2

Markov decision processes

When people perform complex tasks or interact with games and virtual environments, they must engage in sequential action planning, making multiple decisions over time as they attempt to achieve their goals. These decisions reflect both the context in which they are acting and their estimates of short and long term consequences. Sequential action planning is also engaged in by computer-based tutors: the tutor chooses a series of pedagogical activities based on the student's knowledge as well as the likely long-term effects on learning. To interpret people's action choices, as described in Chapters 3, 4, and 5, I rely on Markov decision processes (MDPs); I use a variation on these processes, partially observable Markov decision processes (POMDPs) in Chapter 6.

MDPs provide a natural, decision-theoretic framework for sequential planning problems where a series of actions must be taken and these actions may have non-deterministic consequences (see Sutton & Barto, 1998, for an overview). MDPs model an agent's actions over time, in conjunction with the environment in which the agent is acting. They thus allow us to make detailed inferences about the reasons for people's actions and also allow us to simulate how people might act in a given situation. In this chapter, I provide a formal specification of MDPs and a brief overview of some of the ways that MDPs have been used that are most closely related to my focus of modeling people's action planning.

2.1 Formal specification

MDPs are formally defined as a tuple $\langle S, A, T, R, \gamma \rangle$. At each time step t, the agent and environment are in some state s; in this section, I will consider MDPs where the set S of possible states is discrete, although MDPs can be generalized to continuous state spaces and I consider such a space in Chapter 4. As shown in the graphical model depiction in Figure 2.1, states must be defined so that given the state at time t, states at times prior to t are independent of states at times after t; this property is known as the Markov property. At each time step the agent chooses some action $a \in A$. After the action is taken, the agent and environment transition into a new state based on the action that was chosen as well as the current state. The *transition model* provides conditional probability distributions $p(s'|s,a)$ describing the likelihood of each next state given the current

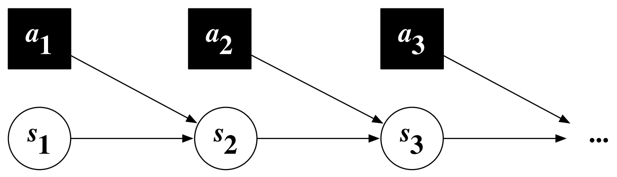


Figure 2.1: Graphical model depiction of an MDP. At each time step, the state is observed, and the agent chooses an action. The state transitions stochastically based on the current state and chosen action.

state and action. The transition model provides a flexible way of specifying how the state of the environment is affected by the agent's actions, allowing for the possibility that the environment or the consequences of an agent's actions may be probabilistic.

MDPs encode the reward or incentive structure of an environment in the reward model R: what is the agent trying to achieve (or avoid) through its actions? For any state s, action a, and next state s', $R(s,a,s')$ gives the immediate reward (or cost) of taking action a in state s and transitioning to state s'. The reward model only specifies immediate rewards or costs; long term consequences of actions are not incorporated into this model. However, long term consequences are frequently of interest to the agent when planning its actions. For instance, if an action inevitably leads to a high cost in several time steps, it is less desirable than one that does not have this later cost, even if both have the same immediate cost. The expected long term value of taking a particular action in a given state is defined as the expected sum of future discounted rewards. Known as the Q-value, it can be computed by combining the dynamics of the environment and the reward model:

$$Q(s,a) = \sum_{s' \in S} p(s'|a,s) \left(R(s,a,s') + \gamma \sum_{a' \in A} p(a'|s') Q(s',a') \right), \tag{2.1}$$

where $\gamma \in [0,1]$ is a discount factor that represents the relative value of immediate versus future rewards.

The Q-value calculation must take into account how an agent is likely to act in future time steps. For example, if an agent chooses actions completely randomly, she will likely achieve far smaller total rewards than an agent who always chooses the action with the highest expected reward in each state. The *policy* $p(a|s)$ gives the probability that an agent will choose action a while in s for all states $s \in S$ and $a \in A$. For a given state, an agent who acts randomly would have a uniform probability $p(a|s)$ of choosing each action. An optimal policy for an MDP is defined as a policy that maximizes the expected value of the Q-function over all possible states, meaning that for each state s, the policy places non-zero probability only on actions a that are in the set $\text{argmax}_a Q(s,a)$. Given a particular MDP and policy, the Q-function can be calculated using a dynamic program known as value iteration (Bellman, 1957). This procedure can be applied even in cases where the policy is dependent on the Q-values, as in the case of an optimal policy.

2.2 Modeling a game using Markov decision processes

Many environments in which agents make choices can be formalized as MDPs. For example, most board games can be represented as MDPs: the configuration of the pieces typically represents the state of the game, and the player has to choose an action that will affect the configuration, resulting in a transition to a new game state. While different games may have very different characteristics, the same flexible framework can be used to model them. In some games, for instance, the transition model is deterministic: given a state and action, there is only one possible next state. In other games, such as those where a die is rolled or a card is drawn, the transition model must be stochastic: the outcome is not determined completely by the choice of action and the current state, but the probability of each possibility can be calculated.

We now consider a specific example of a spaceship navigation game that can be modeled as an MDP. Figure 2.2a shows a screenshot of the game, which is a simplified version of the game used in Chapter 3. The player is trying to navigate the spaceship from its current position (s_9) to Earth (s_G); the spaceship cannot go past the edges of the grid, nor can it enter a square with a "hostile alien" (e.g., the upper left corner). Each labeled square in the grid represents a position that the spaceship can occupy. The state of the game can be represented by the location of the ship. Actions in the game correspond to presses of one of the colored buttons; at each time step, the player chooses one of the four buttons to press, or chooses to stop pressing buttons and "land" the ship.

The transition model describes how the state is likely to change based on individual actions. In this game, the buttons usually move the ship one square in the direction indicated by the arrows in Figure 2.2a, but due to small meteors, the ship sometimes moves in another direction instead. If the player tries to move the ship off the grid or into a hostile alien square, the ship remains in its current position. The transition model encodes this description as a collection of conditional probability distributions $p(s'|s,a)$. For example, if the player pressed the *teal* button with the ship in its current location of s_9, the distribution $p(\cdot|teal,s_9)$ would have three next states with non-zero probabilities: $p(s_4|teal,s_9)$, $p(s_{10}|teal,s_9)$, and $p(s_{12}|teal,s_9)$. s_{10} would be the highest probability next state since *teal* usually moves the ship right. Because it does not matter how the ship reached its current state, the transition process follows the Markov property: the next state is independent of previous states given the current state.

In the spaceship game, the player's goal is to land the ship on Earth in as few moves as possible. At each time step, the player chooses to either press one of the buttons or to land the ship. There is a small cost for each button press and a large cost for landing the ship anywhere other than Earth. After landing, the game terminates, so there are no future costs or rewards. In Figure 2.2b, part of the Q-function for the game is shown, assuming an optimal policy. In state s_9, for instance, the Q-values for buttons that usually move right, up, or down have similar values, while the button that usually moves the ship left has a more negative Q-value: moving left does not help the ship get closer to Earth, and thus has higher long term costs. Figure 2.2c shows the policy that results from the Q-function, with colored arrows representing the best direction to move from each square.

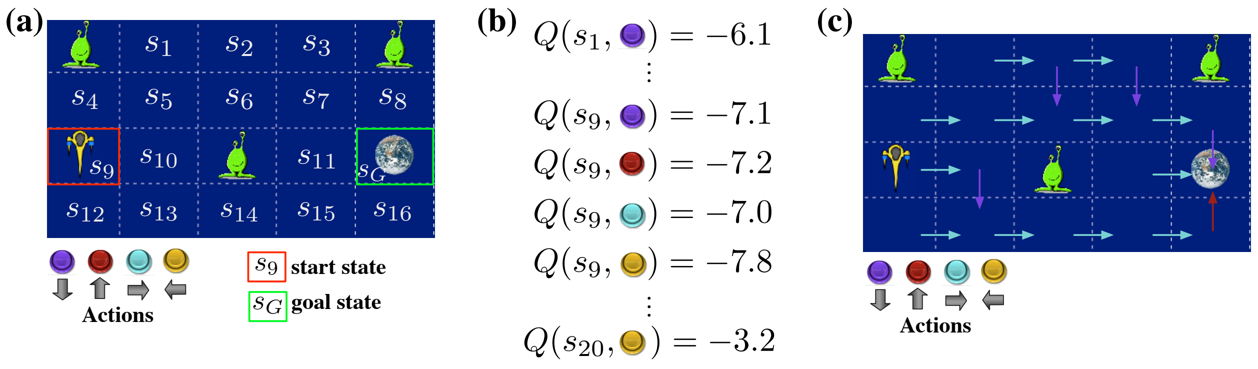


Figure 2.2: Modeling a spaceship game using a Markov decision process. (a) States and actions in the game. Each grid square corresponds to a state, and each colored button can be pressed to move the spaceship. Arrows represent the direction that each button usually moves the ship, although movement is noisy. (b) A portion of the Q-function for the game, assuming an optimal policy. (c) The policy for the game. Arrows are colored to indicate which button has highest Q-value in each state, with the direction of the arrow indicating the most likely place for the ship to move after that button press.

2.3 Applications of Markov decision processes

MDPs have traditionally been used in planning and decision making. By specifying the components of the MDP and solving for an optimal policy, one can calculate the best action to take in any given state. This approach has a diverse array of applications, from robotics to recommender systems (for an overview, see Feinberg, Shwartz, & Altman, 2002; Puterman, 2005). The broad success of MDPs for action planning has also translated into uses in education and intelligent tutoring systems. Barnes and Stamper (2008) used MDPs to choose what hint to give students in a logic tutor based on what solutions had been successful for other students who completed the same problem. In this case, the state, action, and transition models were calculated empirically based on previous solutions. This approach provided substantial coverage of new students' solution attempts. MDP policies have also been used to make instructional decisions about what action an automated tutor should take next given the previous interaction of the tutor and student (Chi, Jordan, VanLehn, & Hall, 2008). This work found that dialogue decisions in a physics tutor could be optimized based on transition and reward models learned from the data of previous participants; the state's represented learners' knowledge and the context of the tutor using observed features. The tutorial policy optimized to improve learning outcomes outperformed a policy optimized to decrease performance. Within the domain of games, MDPs and reinforcement learning have been used previously to predict player actions and adapt game difficulty (Erev & Roth, 1998; Andrade, Ramalho, Santana, & Corruble, 2005; Tan & Cheng, 2009).

While MDPs are most commonly used by agents to plan their actions, other applications have focused on inferring parts of an MDP given observations of another's actions. For example, inverse reinforcement learning infers an agent's policy or reward function, facilitating the training of robotic agents when it is easier to generate examples of the desired behavior than to explicitly state the MDP (e.g., Abbeel & Ng, 2004; Ng & Russell, 2000; Russell, 1998; Ziebart, Maas, Bagnell, & Dey, 2008). Typically, there may be multiple solutions that are consistent with observed actions, leading to the use of regularization or Bayesian methods in which prior distributions are placed over possible policies or reward functions (e.g., Ramachandran & Amir, 2007).

In cognitive science, MDPs have been applied to model human action planning and infer people's goals (C. L. Baker, Saxe, & Tenenbaum, 2009). A growing body of work relies on inverse reinforcement learning to model the inferences that people make about other's goals after observing their actions (C. L. Baker, Tenenbaum, & Saxe, 2006; C. L. Baker et al., 2009; C. L. Baker, Saxe, & Tenenbaum, 2011; Tauber & Steyvers, 2011; Ullman et al., 2010). This approach can help to explain people's intuitions about whether an agent is helping or harming another agent, and can incorporate changes in goals as well as asymmetry of information between the agent who is acting and the person who is interpreting that agent's actions. Because inverse reinforcement learning is inherently probabilistic, this framework can account for the fact that people are not deterministic rational agents but may act noisily. In the next three chapters, I will build on the idea of using inverse reinforcement learning to interpret people's actions by developing a framework in which people's actions may be used to draw inferences about their understanding or features of their cognitive processes.

Chapter 3

Knowledge diagnosis via Bayesian inverse planning

In Chapter 1, I highlighted the ability of the fictional illustrated primer to observe someone's actions and draw inferences about the person's understanding and skills based on these observations.[1] While such inferences are typically difficult for computers, they occur naturally for people. For instance, based on observing someone take a needlessly long route to get to a particular location, one might infer that the person does not know that road construction has been completed on a shortcut that would take her there more quickly. This inference is an example of recognizing someone's misunderstanding through observation alone: the person believes that the road cannot be traversed, but in fact, the road is passable.

Developing an algorithm to make such inferences automatically would allow computer-based educational programs to intervene and correct misconceptions exhibited in freeform tasks; such interventions have been found to improve student understanding in other types of tasks (Davis, Linn, & Clancy, 1995; Liu, Lin, & Kinshuk, 2010). For example, imagine a student playing a biology game. Her responses to specific situations in the game, such as what sequence of actions she takes to adapt an organism to a new environment, can indicate her knowledge about particular elements of cell biology. If she never makes particular adaptations or takes actions in a suboptimal order, this can indicate gaps in her knowledge, leading to targeted remediation; conversely, the student's actions might indicate that she has mastered the current topic and is ready for the next activity. Automating these assessments is beneficial because it does not require interrupting students to explicitly query their knowledge and can provide a detailed picture of students' misconceptions. The benefits of "stealth assessments" that occur within a student's normal activities have been noted by Shute (2011), and prior work has found that such embedded assessments can be useful in the classroom (Feng, Heffernan, & Koedinger, 2009; Razzaq et al., 2005).

The ability to use complex series of actions to automatically diagnose student knowledge is becoming more relevant with the increasing use of games and interactive virtual environments in

[1]This chapter is based on work conducted in collaboration with Michelle M. LaMar and Thomas L. Griffiths. Parts of this work are included in Rafferty, LaMar, and Griffiths (in press). Thanks go to Benjamin Shapiro, HyeYoung Shin, and Christina Vu for their assistance in conducting the behavioral experiments described in this chapter.

education. These environments bring us closer to the way people actually use their knowledge for real world problem solving. Within these environments, students often perform many individual actions to complete a task, resulting in fine-grained data about the choices that students make. These data contain much more information than simply whether the student completed the task successfully or not, and we would like to use these data to make fine grained inferences about a student's knowledge, including her misconceptions, just as a teacher could infer this information by observing the student. However, existing assessment models in education are generally not suited to interpreting such sequential process data. These models typically assume the data are conditionally independent given student ability, and consider only success or failure, rather than the way that these outcomes are achieved. In this chapter, we consider an alternative to existing models in which we focus on modeling how people choose their actions based on their beliefs or understanding. This detailed model then allows us to gain insight into a student's knowledge by observing her actions.

Specifically, we formalize action planning using the Markov decision process model described in the previous chapter. We characterize a person's knowledge as her beliefs about how her actions affect the state of the world and what states are most beneficial for achieving her goals. We then propose a framework for automatically inferring these beliefs. This model could be applicable to a variety of action understanding tasks, but our interest is centered on educational settings in which false beliefs (misconceptions) are likely to be common. Markov decision processes allow us to make inferences about students' beliefs by specifying how those beliefs combine with their goals to determine their actions. In the previous chapter, we highlighted several instances of using MDPs to model human action-planning. This existing work focuses on inferring people's goals, as represented by subjective reward models; in contrast, we use the transition model to represent people's understanding. We can then diagnose the person's understanding by inferring a distribution over possible transition models based on observing the person's actions.

A variety of work focuses on understanding the actions of others, ranging from neuroscience to cognitive science to computer science. Work in neuroscience supports the idea that people can recognize the false beliefs of others simply through observing their actions (Grèzes, Frith, & Passingham, 2004). These researchers found that human observers show different activation patterns when observing a person lift a box when that person has correct versus incorrect expectations of the weight of the box. Related work has examined what other inferences about people's mental states can be made through observing actions, finding that relatively accurate inferences about people's goals and confidence can be made even when differences in actions are minute (e.g., Becchio, Manera, Sartori, Cavallo, & Castiello, 2012; Patel, Fleming, & Kilner, 2012). Cognitive science has also approached the question of what inferences people can make about the beliefs of others. Most closely related to our work is that of Goodman, Baker, and Tenenbaum (2009), which examined people's inferences about the beliefs that another individual has about the consequences of her own actions. In both domains, research has generally focused on isolated actions, rather than the complex sequences of actions that might occur in an educational setting.

In computer science, work on plan recognition has also examined the problem of interpreting others' actions. A common problem in this domain is how to automatically identify someone's intended plan of action based on a set of observed actions (e.g., Gal, Yamangil, Shieber, Rubin, &

Grosz, 2008; Kautz & Allen, 1986; Lesh, Rich, & Sidner, 1999). This task has been recognized as potentially helpful in educational environments. Amir and Gal (2011) used a plan recognition framework to categorize sets of individual behaviors in a virtual chemistry lab, such as pouring one beaker into another, as part of larger semantic actions, such as a titration. Our work differs from plan recognition in that we assume that people may have misunderstandings about their actions, rather than assuming that people have full, accurate knowledge of how their actions affect the world.

In this chapter, I begin by introducing a Bayesian inverse planning framework to infer people's understanding from their actions, drawing on the background about Markov decision processes provided in the previous chapter. The Bayesian inverse planning framework is a novel modification of inverse reinforcement learning. I next introduce a simple environment that we use for three experiments exploring this framework. In Experiment 1, we show that the inverse planning model can recover learners' beliefs within this environment, and in Experiment 2, we show that the model's inferences are about as accurate as those of human observers. We then examine how to apply this framework to a common educational objective: customizing feedback based on inferences about the learner's understanding. In Experiment 3, we demonstrate that feedback informed by the model speeds learning in the planning environment relative to uninformed feedback, and we show that the model can easily be extended to handle a more complex space of possible beliefs that people might have.

3.1 Inferring learners' beliefs

By using Markov decision processes as a generative model of action planning, we can formally define how a person's beliefs connect to the actions that they choose. We propose a Bayesian inverse planning framework that allows us to infer these beliefs based only on observing a person's actions (see Figure 3.1 for an intuitive description applied to the spaceship game described in Chapter 2). This inverse planning framework relies on the insight that people are likely to choose actions that they think will help them achieve their goals. Thus, their beliefs are likely to be consistent with the chosen actions being better than other possible actions. The inverse planning framework simply formalizes this insight.

The model we develop uses an MDP to model people's actions. We assume that the reward function R, which encodes the person's goals, is known. We also assume the set S of possible configurations of the world is known. People's *hypotheses* about how their actions affect the world then formally correspond to transition models T: their understanding of how actions affect the current state can be encoded as probabilities $p(s'|a,s)$.

We now want to make inferences about how likely it is that someone has a particular hypothesis given that we have observed a series of actions $\mathbf{a} = (a_1, \ldots, a_n)$ that the person took to try to complete a goal. Given a fixed hypothesis space $\mathcal{T}$ of possible transition models and a given starting state s_1, we want to calculate the posterior distribution over possible hypotheses $T \in \mathcal{T}$:

$$p(T|\mathbf{a}, s_1, R, \gamma) \propto p(\mathbf{a}|s_1, T, R, \gamma) p(T). \tag{3.1}$$

By calculating a posterior distribution, we can determine both what hypothesis is most probable given the person's actions as well as how strongly the evidence supports this hypothesis over alternatives. Calculating this distribution requires knowing the prior distribution $p(T)$ and computing the likelihood for a particular series of actions given a hypothesis T. The prior distribution over hypotheses accounts for the fact that some beliefs about the effects of different actions may be more likely than others. This prior will vary based on the specific task, and provides a way for known information about likely misconceptions to be incorporated. For instance, educational research may indicate that certain misunderstandings are common in a particular domain or for a particular population, whereas others are less common. The prior can then be constructed to place higher probability on the common misunderstandings. In cases where no such information is available, a uniform prior can be used.

To compute the posterior, we must also calculate the likelihood, $p(\mathbf{a}|s_1,T,R,\gamma)$. This quantity corresponds to how likely it is that the person would choose the observed sequence actions given that they believe transitions occur as in model T. Note that having high likelihood does not imply that a sequence of actions is likely to result in the goal state, but only that this sequence is more likely than other sequences to be chosen. In the likelihood, we have conditioned the sequence of actions only on the first state, s_1; this corresponds to situations where the person knows the initial state but does not know later states with certainty, as occurs in the experiments we present. The Markov property states that s_{t+1} and s_{t-1} are conditionally independent given s_t, so we can calculate the likelihood using the following recursion:

$$p(\mathbf{a}|s_1,T,R,\gamma) = p(a_1|s_1,T,R,\gamma) \sum_{s' \in S} p(s'|a_1,s_1,T) p(a_2,\ldots,a_n|s',T,R,\gamma), \tag{3.2}$$

where $p(a_1|s_1,T,R,\gamma)$ is defined by the person's policy for choosing actions given the current state. Knowing this policy, or an approximation of it, is necessary to make any inferences about why a person chose a particular set of actions, and it is vital that this policy be dependent on the person's beliefs about T in order for the observations to give information about those beliefs. As in C. L. Baker et al. (2009), we assume that people can be modeled as following a noisily optimal policy. This policy, known as a Boltzmann policy, states that actions are chosen as follows:

$$p(a|s,T,R,\gamma) \propto \exp\left(\beta Q(s,a|T,R,\gamma)\right), \tag{3.3}$$

where $Q(s,a|T,R,\gamma)$ is the Q-function defined in the previous section and β is a parameter determining how close the policy is to an optimal policy. Intuitively, this policy corresponds to choosing actions that one believes have higher values than other possible actions, but retaining some probability on choosing suboptimal actions and being less sensitive to small differences in Q-values. As β becomes large, the Boltzmann policy converges to the optimal policy, while as β goes to 0, the policy converges to choosing actions uniformly at random.

We can now calculate the posterior distribution over the hypothesis space $\mathcal{T}$ by combining the prior and the likelihood. If this space is discrete, the posterior distribution can be calculated exactly by first calculating the $|\mathcal{T}|$ different Q-functions for the MDPs associated with each possible transition model, and then evaluating Equation 3.2 for each MDP. This is the approach we take for

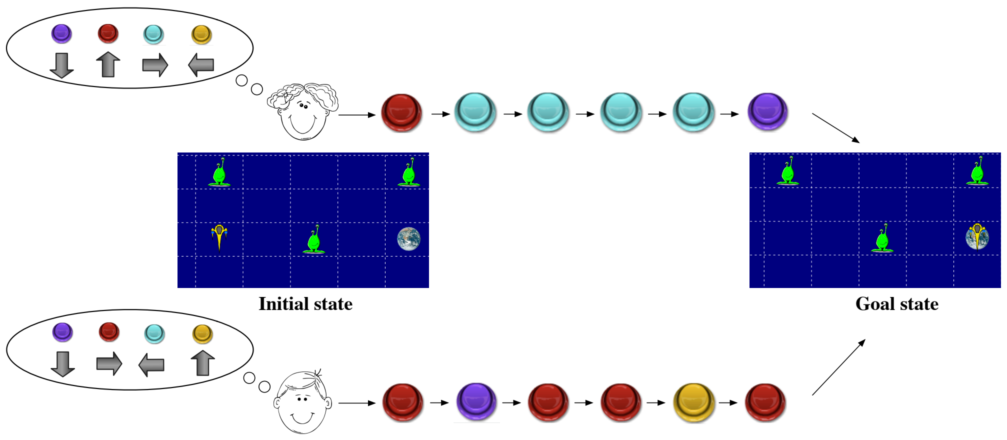


Figure 3.1: Players' beliefs about the buttons lead to different action sequences in the spaceship game. Both players use their understanding and their desire to move the ship from the start state to the goal state to choose their actions. In the model, people's beliefs correspond to different possible transition models T_i.

Experiments 1 and 2. In other cases, the hypothesis space may be continuous or discrete but very large, making it infeasible to calculate the posterior exactly. An approximate posterior distribution can then be calculated using Markov chain Monte Carlo techniques; we use this approach in Experiment 3.

In Chapter 2, we used a spaceship game as an example of an MDP. In the spaceship game, buttons control the spaceship's movement, and players are trying to move the ship from its current location back to Earth. We can use Bayesian inverse planning to model the situation where players may have incorrect beliefs about how the buttons work, and we wish to infer those beliefs based on the sequence of buttons that they press to move the ship to Earth. As shown in Figure 3.1, players' beliefs lead them to choose systematically different sequences of actions. Both players choose actions that they think are valuable, and thus their choices can be used to make inferences about their understanding.

Using Bayesian inverse planning to infer people's understanding has several potential advantages. First, it is extremely flexible and can be applied in a variety of situations. Given an appropriate definition of the state space, many tasks can be specified as MDPs, and the same general framework can be applied to make inferences about people's beliefs based on their actions. For instance, as described in the previous chapter, many board and video games can be formalized as MDPs, and stochastic factors can naturally be incorporated into MDPs. Additionally, inferences can be made after only a few actions, and multiple sets of observed actions can be used to refine inferences as further evidence accumulates. In an educational environment, this opens the possibility for the computer to intervene about a specific belief in a timely manner, and to accumulate evidence of misunderstanding over a series of different exercises. The model also provides a fine-grained way of evaluating student responses, rather than only focusing on whether the student was successful in the complete task. Thus, the model can diagnose specific gaps in understanding rather than merely labeling all unsuccessful students as "wrong."

3.2 Validating the method in the laboratory

We have now defined a general model for inferring people's beliefs about how their actions affect a particular environment. This model makes several assumptions about how people choose actions based on their knowledge, so we first validate the model by testing its accuracy in several lab experiments. For these experiments, participants played a more complicated version of the spaceship game described in the previous chapter. Participants used a computerized interface to pilot the spaceship to Earth using as short a path as possible (see Figure 3.2a). This version of the game had eight buttons, each of which either usually moved the ship in one direction or moved the ship in a direction at random. Participants were not told exactly how each button affected the ship's movement; instead, they learned this information by observing the effects of the buttons. Using the inverse planning framework, we attempt to infer participants' beliefs about how the buttons work. While this environment is relatively simple compared to many educational applications, it has the advantage of having many possible misconceptions that are easily articulated. We conduct three experiments in this environment. In Experiment 1, we validate the model by directly comparing its inferences to participants' stated beliefs, as collected at various points within the game. To determine how well the model performs compared to human observers, we ask new participants in Experiment 2 to infer the original participants' beliefs using the same information as the model; this allows us to see whether the model makes similar inferences to humans. Experiment 3 uses the model's inferences to guide feedback to participants, and investigates whether this informed feedback increases the speed of learning.

3.3 Experiment 1: Comparing the model's inferences to participants' beliefs

Methods

Participants

A total of 25 undergraduates at the University of California, Berkeley received course credit for their participation.

Stimuli

Participants interacted with the computerized interface shown in Figure 3.2a. The top portion of the interface was a 7×11 grid. The spaceship moved around this grid based on the button presses; all other objects remained stationary throughout the experiment. The bottom portion of interface contained the buttons to control the ship's movement, and a display field with information about the buttons that were pressed.

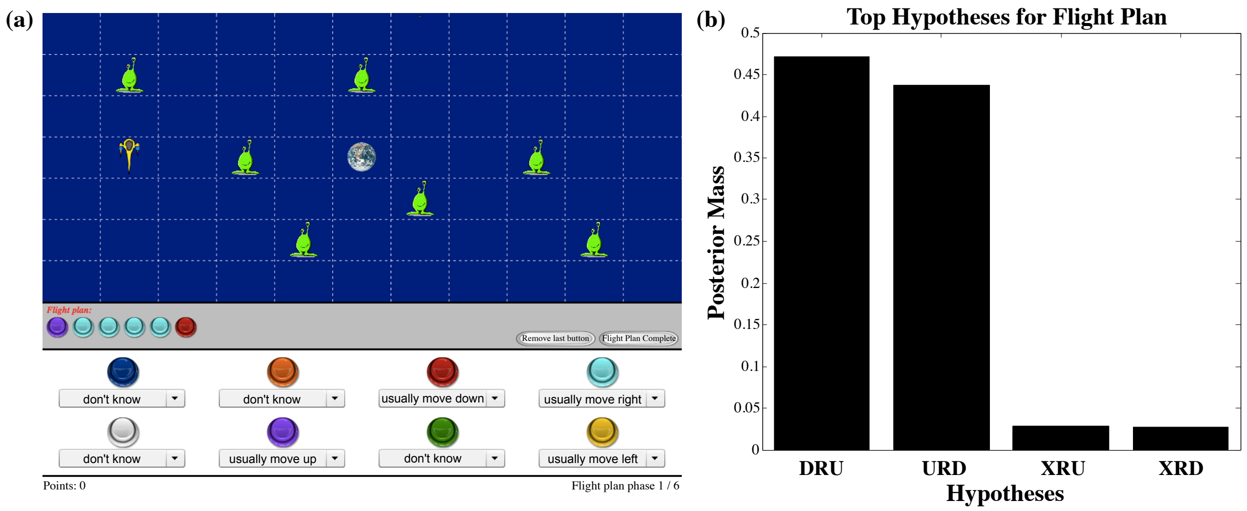


Figure 3.2: Diagnosing beliefs about spaceship controls from flight plans. (a) A screen shot from a participant entering a flight plan. The top portion shows the spaceship, Earth, and hostile aliens. The middle shows the entered flight plan, and the bottom portion shows the buttons as well as menus for participants to indicate how they think each button works. (b) The top hypotheses for the three buttons in the flight plan. The three letter codes indicate, in order, how *purple*, *teal*, and *red* work in the hypothesis: R means usually moves the ship right, D moves it down, U moves it up, and X moves it randomly.

Procedure

Participants were told that they would be learning how different buttons affected a spaceship's movements and using that knowledge to pilot the spaceship back to Earth. They were informed that each of the buttons either moved the ship one square in a single direction or in a random direction (due to being broken). Participants were also told that occasionally small meteors caused a button that should move the ship in one direction to move it in another direction, although they could not be sure when a movement was caused by meteors. Meteors caused the spaceship to move in an unexpected direction 15% of the time. Participants were informed that the ship would remain in the same place only if moving would cause it to go off the edge of the grid or to move into a square with a "hostile alien." Hostile aliens were placed on several squares in the grid to make the game more maze-like.

Participants alternated between *exploration* phases and *planning* phases. In exploration phases, participants pressed buttons and saw the result of each button press on the spaceship's location; eight button presses were allowed in each exploration phase. During these phases, the display field informed participants of their last action and its result (e.g., "You pressed the red button, and the spaceship moved up."). In planning phases, the spaceship was relocated to a random square, at least two squares away from Earth, and participants were told to enter a *flight plan*, consisting of a series of button presses, that would take the spaceship from its current square to Earth. The participants could not see the effect of each button immediately, but had to enter a complete sequence of buttons to guide the ship to Earth. This was to separate learning how the buttons worked from using that knowledge to complete a task. Participants indicated when they were finished, and then were told whether the flight plan had caused the spaceship to reach Earth. The result of the plan was determined by simulating the sequence of button presses using the true transition model for each button, which included the effects of meteors. Each participant completed six exploration and planning phases; the way that each button affected the ship's movement remained the same over these phases.

During all phases, there was a menu below each button containing the possible ways the button could make the ship move ("usually moves left," "usually moves right," "usually moves up," "usually moves down," "moves randomly," or "don't know"). All menus were originally set to "don't know," and participants were told to use the menus to record how they thought the buttons worked. After submitting a flight plan but before they were told the outcome of the flight plan, participants were asked to check that the menus reflected their current beliefs about how each button worked.

Modeling flight plans

To infer participants' beliefs about the buttons based on their actions in the flight planning phases, we model the flight planning task as an MDP. As in the simplified version of the game, the location of the ship corresponds to the state of the game, and each button is a possible action. There is also a landing action, corresponding to the participant submitting her flight plan: this action has zero reward if chosen when the spaceship is at Earth and a highly negative reward otherwise. All other actions have small negative reward, so shorter sequences of actions are favored over longer sequences. Assuming the magnitude of the negative reward for not reaching Earth is large enough that it is less costly to move to Earth than to stop immediately, the model is insensitive to the exact values used in the reward function, including whether the reward for reaching Earth is zero or a positive value.

For the inverse planning model, we define the hypothesis space $\mathcal{T}$ to match the instructions given to participants: each button either primarily moves the ship in one direction (left, right, up, or down) or it moves the ship in one of these four directions uniformly at random. In the former case, the button moves the ship in the primary direction 85% of the time and in another direction uniformly at random the other 15% of the time. Each hypothesis consists of the transition models for all eight buttons. We limit hypotheses to those which include buttons that move the ship in each of the four directions, resulting in $166,824$ hypotheses. We assume a uniform prior over the hypotheses as there is no reason to believe people will be biased towards particular hypotheses. If such data were available, the uniform prior could be replaced with a prior that incorporates this information.

Because we do not know *a priori* how optimal people's plans will be, we allow for uncertainty in the value of the noise parameter β. We marginalize over a discretized set of possible values for

β to infer the posterior over $\mathcal{T}$ for each plan, with:

$$p(T|\mathbf{a}, s_1, R) = \sum_{\beta} p(T, \beta|\mathbf{a}, s_1, R), \tag{3.4}$$

where $p(T, \beta|\mathbf{a}, s_1, R) \propto p(\mathbf{a}|T, \beta, s_1, R)p(T)p(\beta)$. We consider values of β from $0.5 - 5$ in increments of 0.5, and place a uniform prior $p(\beta)$ over these values. This procedure allows inference about participants' beliefs to be made without fixing β to any particular value.

Baseline model

In addition to using the inverse planning model, we evaluated plans using a simple baseline model. In this model, the horizontal and vertical displacements from the spaceship to Earth are calculated and compared to the number of button presses of each color. A button is then matched to a direction if it had the same number of presses as the displacement in that direction. For example, if the flight plan *(blue, blue, red, red, red)* was entered and the spaceship began two squares to the right and three squares down from Earth, the model would predict that the blue button moved the spaceship left and the red button moved it up. This model does not account for obstacles between the spaceship and Earth, and cannot make predictions about buttons that were not pressed.

Results and discussion

We ran the model on each flight plan that a participant entered to guide the ship to Earth. Our goal was to infer participant's beliefs at a given phase using data from the flight plan they created at that phase. Multiple phases were completed by each participant, but since we want to determine how well the model can infer beliefs from limited data, we do not seek to model learning or to use information from other flight plans created by the same participant. Initial inspection of the flight plans suggested that some participants may not have understood that each button press could move the ship only one square. For example, a plan with only one button press might be entered when the ship needed to travel a minimum of five squares to reach Earth. We eliminated these plans by having uninformed evaluators examine each of the original flight plans and determine whether there was any way the flight plan could bring the ship to Earth; more detail about the task that these evaluators were completing will be given in Experiment 2. To make clear when we are referring to participants in this experiment and when we are referring to the new evaluators (the participants in Experiment 2), we will refer to participants in this experiment as "planners."

We evaluated the model on the 101 flight plans that at least three of the four evaluators thought were valid plans. Within the model, each hypothesis specifies a full transition model with beliefs about all buttons. However, flight plans included an average of only 2.4 unique buttons, which is insufficient to fully specify the transition model. For example, if the ship started one square up and to the left of Earth, one might infer that buttons not used in the flight plan are less likely to move the ship down or to the right, but there is no information to distinguish among other possible transition models for these buttons. The inverse planning model places the same probability on all such hypotheses. We thus primarily evaluate the model based only on predictions about buttons that were used in the flight plan.

We first examine the *maximum a posteriori* (MAP) estimates of the inverse planning model for each flight plan. This is the mode of the probability distribution: the hypothesis on which the model placed the greatest posterior probability. For example, if a particular hypothesis T_1 had a posterior probability of 0.4 and all other hypotheses had smaller probabilities, the MAP hypothesis would be T_1. For 73% flight plans, the MAP hypothesis matched the planner's stated beliefs about all buttons pressed, and for 93% of plans, the MAP hypothesis matched the stated beliefs for at least some of the buttons in the plan.[2] As shown in Figure 3.3a, these results are significantly better than those of the baseline model, which gave a hypothesis that matched the planner's beliefs for only 60% of flight plans (exact McNemar's test, $\chi^2(1) = 8.89$, $p < .01$). These rates are also much higher than a chance baseline, which chooses one of the five options randomly for each button in the plan. This baseline would result in a complete match an average of 5.8% of the time (averaged across plans), with at least one button correct 36% of the time.

The MAP estimates provide one way of evaluating model performance, but the full posterior distribution can give more information about the strength of the model's estimates. Because this distribution gives the probability of each individual hypothesis, we can use it to explore whether the MAP estimate is strongly favored or whether several hypotheses have similar probabilities. This is especially important since some flight plans are inherently ambiguous. Consider the plan *(purple, teal, teal, teal, teal, red)* shown in Figure 3.2a, which was entered by a planner during one flight planning phase. Based on the ship's position, this is relatively unambiguous evidence that *teal* usually moves the ship right, but seems equally supportive of *purple* moving the ship up and *red* moving it down as of *red* moving the ship up and *purple* moving it down. Figure 3.2b demonstrates that inverse planning model is sensitive to this distinction by showing the probability mass on the four most probable hypotheses hypotheses, denoted as beliefs about *purple*, *teal*, and *red* moving the ship *right* (*R*), *down* (*D*), *up* (*U*), or *randomly* (*X*). Almost all of the posterior mass is on hypotheses in which *teal* moves the ship right, which is the belief for all four of the most probable hypotheses, but about half of the mass is on each of intuitively plausible possibilities for *red* and *purple*.

To assess how well the posterior distribution matches the planner's self-reported beliefs, we calculate the total posterior mass on hypotheses that are consistent with those beliefs. As shown in Figure 3.3b, the model places most of this mass on the correct hypothesis for the majority of the flight plans. The flight plans where the model places about half of the posterior mass on the correct hypothesis tend to be those that were ambiguous. Overall, the model placed an average of 0.74 of the posterior mass on transition models that matched the participant's description of how the buttons worked.

The shape of the posterior distribution can also provide indicators of participant misconceptions. If little posterior mass is placed on the transition model that governed the actual effects of the buttons, this suggests the participant likely has some misunderstanding. This is shown in the

[2] We could only compare model results to the participant's beliefs in 88 of the 101 plans because in 13 of the plans, the planner marked all buttons used in the plan as "don't know".

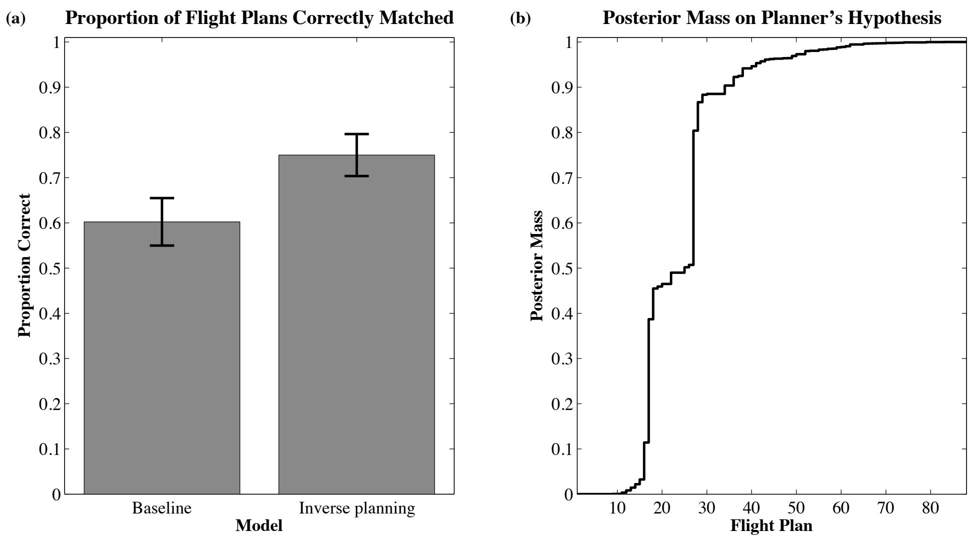


Figure 3.3: Model performance on matching the original planner's hypothesis in Experiment 1. (a) Comparison between the baseline model and the inverse planning model. In 60% of flight plans, at least one of the hypotheses given by the baseline model matched the planner's beliefs. In 73% of flight plans, the MAP estimate of the inverse planning model matched the planner's beliefs. Each flight plan is one instance of a participant entering a sequence of buttons to guide the ship to Earth. Error bars are equal to one standard error. (b) Posterior mass by flight plan on hypotheses that matched the planner's hypothesis. Flight plans are ordered by the model's performance.

data: in cases where the participant's stated beliefs reflect an incorrect understanding, an average of only 0.039 of the posterior mass is placed on hypotheses that correspond to the actual transition model for the buttons. In contrast, 0.82 of the posterior mass is on these hypotheses in cases where the participant is correct.

We can combine the model's diagnosis of correctness with its predictions about participants' beliefs in order to detect misunderstandings. For example, one participant began with the ship in the upper right corner, and entered a flight plan with two *teal* buttons, followed by five *red* buttons, and finally a *green* button. The participant correctly understood that *teal* usually moved the ship down, and the model placed 0.97 of the posterior mass on such hypotheses. The participant had misconceptions about the other two buttons, believing that *red* moved the ship left and *green* moved it down. In fact, both buttons moved the ship up. The model's predictions show this misconception: It places 0.99 of the posterior mass on *red* moving the ship left, and 0.95 of the mass on *green* moving it down. In contrast, less than 0.01 of the posterior mass is placed on either button moving the ship up. This information could be used to provide customized feedback to the learner, a possibility we explore in Experiment 3.

Overall, the results of Experiment 1 suggest that the inverse planning model is reasonably accurate at inferring planners' beliefs. It outperforms a simple baseline model, and because it outputs a posterior distribution, it provides information about how strongly the evidence supports its inferences and pinpoints participant misunderstandings.

3.4 Experiment 2: Comparing the model to human inferences

Experiment 1 demonstrated the accuracy of the inverse planning model. However, it is challenging to evaluate how good this accuracy is without a measure of the difficulty of matching planners' beliefs. In general, we would not expect the model to outperform human abilities to infer the beliefs of others, and we know that humans are not always able to make completely accurate inferences from observing someone else's actions. This is clearly the case in the flight planning task: as observed in the plan shown in Figure 3.2, there are cases in which multiple hypotheses are equally plausible given the observed actions, and because of the limited number of observations, accurately inferring a full transition model for all buttons is likely to be impossible for either human observers or any given model. In Experiment 2, we asked new participants to evaluate the plans made by the original planners. These observers provide a gold standard for how accurately it is possible to match the planners' beliefs. This experiment also allowed us to identify those plans that could not take the ship to Earth regardless of how the buttons worked.

Methods

Participants

A total of 24 undergraduates at the University of California, Berkeley received course credit for their participation.

Procedure

Each participant evaluated 25 flight plans, one from each of the planners in the original experiment; all flight plans were thus evaluated by four different participants. The initial instructions about the possible ways the buttons could work were the same as in Experiment 1. Rather than being told that they would be piloting the spaceship, however, participants were told that they would watch the aliens try to fly different spaceships to Earth. For each plan, participants were shown the same display of the spaceship and Earth as in Experiment 1, as well as the series of buttons that was pushed for the plan. Participants were told that this plan had been generated by aliens. Participants were asked to choose one of five options for how they thought that the alien who made the plan had believed the button worked: usually moves left, usually moves right, usually moves up, usually moves down, or moves randomly. Participants were only asked to evaluate the buttons that were actually used in each plan, and they were told that the way each button worked could change from one plan to another, due to being generated by different aliens in different ships. Three additional questions were asked about each plan. First, whether the plan is likely to take the spaceship back to Earth assuming the buttons work as the participant indicated. Second, whether the plan contains

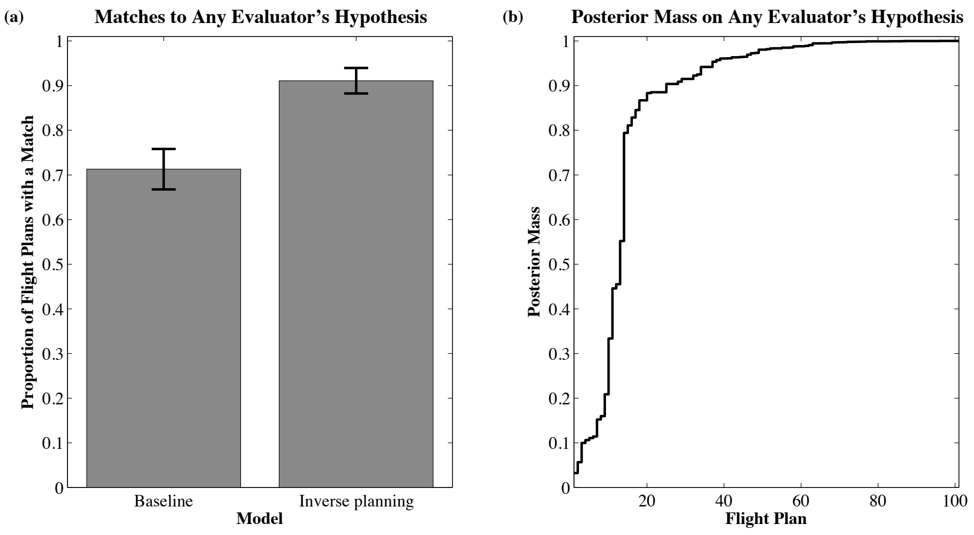


Figure 3.4: Model performance on matching evaluators' hypotheses in Experiment 2. (a) The proportion of flight plans for which the baseline model gave a hypothesis matching any of the evaluators' hypotheses versus the proportion of flight plans for which the hypothesis with highest posterior under the inverse planning model matched any of the evaluators' hypotheses. Error bars are equal to one standard error. (b) Posterior mass by flight plan on a hypothesis that matched any evaluator's hypothesis.

enough button presses to get the spaceship back to Earth. Finally, whether the plan was longer than the shortest plan that could plausibly take the spaceship back to Earth.

Results and discussion

As in Experiment 1, plans that fewer than three of the four evaluators (participants in Experiment 2) thought were likely to successfully bring the ship to Earth were eliminated. We first examined how well the human evaluators' responses matched the original planners' stated beliefs compared to how well the model matched the planners' beliefs. For each plan, we computed an evaluation accuracy by calculating the proportion of evaluators who gave the same hypothesis as the original planner. The mean evaluation accuracy over all plans was 0.75. The model's accuracy was thus comparable to human performance: it placed an average of 0.74 of the posterior mass on the hypothesis of the planner.

We next evaluated the inverse planning model's ability to capture the evaluators' inferences. For each flight plan, we computed the total posterior mass assigned to all hypotheses that matched at least one hypothesis produced by an evaluator. As shown in Figure 3.4a, the model placed an average of 0.87 of the posterior mass on such hypotheses, significantly outperforming the baseline model which gave the same hypothesis as one of the evaluators for 0.71 of the plans (exact McNemar's test, $\chi^2(1) = 20$, $p < .001$). Figure 3.4b shows that in most cases, the inverse planning model put almost all of the posterior mass on the evaluators' hypotheses, demonstrating that the model's inferences were similar to those of human evaluators. The inverse planning model was more successful at matching evaluators' hypotheses than the hypotheses of the original planners primarily due to the fact that many plans are ambiguous. For example, given that the spaceship is one square up and to the left of Earth, the plan *(red, yellow)* is equally good evidence that *red* usually moves the ship down and *yellow* usually moves it right as vice versa. With multiple evaluators, each of these hypotheses is likely to be given, but the original planner could only have a single hypothesis. Neither human observers nor the inverse planning model have sufficient information to infer which of the possible hypotheses was actually believed by the original planner.

One discrepancy between the inverse planning model's inferences and human inferences was in the strength of the model's predictions about the buttons that were not used in the plan. People often use one button repeatedly for multiple moves in the same direction, even if there is another button that they believe also moves the ship in the same direction. The model takes each press of the same button as evidence that no other button works the same way, since if there is another such button, it is likely that the person would choose that button at some point. People might instead be using a model that is Markov based on the current state of the ship and their last action: choosing to push the same button again is less effort and thus lower cost. While the current inverse planning model does not use such a reward model, it could easily be modified to have a larger state space and more complex reward function to better match the task demands. This flexibility is one of the advantages of the inverse planning model.

3.5 Experiment 3: Using the model to guide feedback

The previous experiments demonstrate that in the flight planning task, the inverse planning model can diagnose people's beliefs with about the same accuracy as human observers. Given this evidence about the validity of the model's inferences, we now consider how this model might be used in an educational setting. One area where it might be helpful is in guiding automated feedback. Based on the detailed diagnostic information that the model provides, the feedback that is most relevant to a learner can be selected. Previous work has found that more specific feedback can be more helpful than general feedback (Shute, 2008), and that feedback which directly supports learning can have large positive effects (Kluger & DeNisi, 1996).

To test whether feedback provided via the inverse planning model was an improvement over generic feedback, we conducted a new experiment using a modified version of the flight planning environment. After each flight planning phase, participants were provided with information about how one of the buttons worked. The button about which feedback was given was either selected randomly or via the inverse planning model. If model-based feedback results in participants being able to correctly label how more buttons work after fewer phases of learning, it suggests that the inferences made by the model are sufficiently accurate to guide feedback and that at least in some circumstances, specific diagnostic information can enable practical measures for improving learning.

We also used this experiment to test whether the model could be applied to more complex hypothesis spaces. In the previous experiments, we assumed that for each button, the participant was certain about her belief that it functioned in a particular way. However, this may not always be the case. For example, a participant might believe that the *blue* button either moves the ship randomly or to the left, but be uncertain about which of these two movement patterns is correct. Such hypotheses may be likely to occur when a person has only limited information, and showing that the inverse planning can still diagnose uncertain beliefs is important for real world applications. In this experiment, we extend the hypothesis space to include such uncertain beliefs and modified the flight planning phases to include multiple flight plans, which provide the opportunity for a participant to show evidence of such uncertainty.

Methods

Participants

A total of 60 participants were recruited from the University of California, Berkeley and received course credit for their participation.

Stimuli

Participants used a modified version of the interactive flight planning environment described in Experiment 1. Each participant was randomly assigned to receive feedback based on the inverse planning model or feedback chosen randomly. Thirty participants received each type of feedback.

Procedure

Participants were told that they would be learning how to fly a particular type of alien spaceship. They were informed that they were part of the human resistance movement, and that as a pilot in this movement, they had two tasks: (1) learn how each button affected the movement of the spaceship, and (2) use this knowledge to create flight plans that would take ships back to Earth. The interface gave participants instructions about exploration phases and flight planning phases, just as in Experiment 1, and participants completed each type of phase six times. Exploration phases were identical to Experiment 1, except that participants were allowed six button presses rather than eight. The number of button presses per exploration phase was reduced in order to increase the number of phases required for participants to learn the meaning of all buttons. By increasing the number of phases over which learning is likely to occur, the potential impact of feedback is increased for both conditions.

The flight planning phases were changed somewhat from Experiment 1. The first change was that there were four flight plans per phase, rather than only one. This was to allow participants the opportunity to show uncertainty about how a particular button worked, consistent with the extension of the model to a more complex hypothesis space.

The second change to the experiment was that some flight plans were for ships that had buttons that could not be used. This modification was to encourage participants to use a variety of different buttons and to learn how all of the eight buttons worked. Since one of our outcome measures was what proportion of buttons a participant had correct beliefs about, we wanted to lessen the probability that participants would learn about only four buttons, one for each direction, and then use only those buttons. The existence of ships with buttons that could not be used by participants was explained within the context of the cover story: Each ship in the flight planning phase contained human refugees who had managed to steal the ship from the aliens; in some cases, the aliens sabotaged some of the buttons as they were leaving. Participants entered one flight plan for each of the four ships in each flight planning phase. The starting locations of the ships were chosen as follows. The grid was broken into four quadrants (upper-left, upper-right, lower-left, and lower-right). Within each quadrant, one location was chosen uniformly at random from those squares that were at least two steps from Earth. These four starting locations were then placed in random order, and flight plans were entered sequentially. For each flight plan, participants were told that they were piloting a different ship, which worked the same way as the other ships. To implement the sabotaged buttons, 0–3 buttons were chosen uniformly at random for each ship; these buttons were marked as broken.

After each flight planning phase, participants were asked to check that the drop-down menu below each button showed their current beliefs. Then, they were told how many total points they had earned with their flight plans. In order to motivate participants, we modified the point structure from Experiment 1 such that participants received more points for getting closer to Earth, even if they did not actually reach Earth. Each plan was worth a maximum of 300 points. If the spaceship ended at Earth, the participant received all 300 points, while if the spaceship ended at a location that was further from Earth than it started, the participant received zero points. Otherwise, the participant received a fraction of the maximum points equal to the proportion of the final distance from Earth compared to the starting distance. More specifically, if the ship ended m squares from Earth (measured as the sum of the displacements in the x- and y-directions) and started out n squares from Earth, the participant received $300 \times \frac{n-m}{n}$ points. Participants were informed about how points were calculated, and were told the total number of points that they earned from all four flight plans in the phase. This point structure encourages participants to attempt a flight plan even if they are missing crucial knowledge to complete the plan or if all buttons that would give the plan high probability of success are broken. For example, the ship might be up and to the right of Earth. If a participant believes that *red* moves the ship left but that the only button which moves the ship down is broken, this point structure rewards her for at least beginning the plan by trying to move the ship left.

Once participants had been told how many points they had earned, they were given feedback about one of the buttons. Since the inverse planning model required some time for computation, we computed the model's choice of feedback for participants in both conditions in order to have similar timing of feedback across conditions (see below for details on the algorithm to compute feedback). This computation took 10–15 seconds, and computation began after all flight plans for the phase were entered. Thus, participants generally had very little waiting time, since feedback computations mainly occurred while they were checking the drop-down menus and learning how many points they had earned. If additional computation time was required, participants were told that they were waiting for intelligence via an on-screen message. Once the button had been chosen, a message was displayed informing the participant that another member of the resistance had learned that the ⟨*color*⟩ button moved the ship ⟨*direction*⟩; the information that was given to participants was always accurate, regardless of feedback condition. The only difference was that participants in the random condition received feedback about a randomly chosen button and participants in the model-based feedback condition received feedback about a button chosen by the inverse planning model. Participants were asked to change the drop-down menu for that button to reflect the new information and could not continue until they had done so.

Inferring beliefs in a continuous hypothesis space

After each flight planning phase, we inferred the participant's beliefs about all eight of the buttons. As previously noted, we extended the hypothesis space in this experiment to a broader set of possible beliefs. The new hypothesis space included cases where a participant has uncertainty about exactly how a button works, but places greater confidence on some possibilities than others. This hypothesis space was realized by assuming that participants might act as if they had a belief distribution over possible movement patterns for a button. The hypothesis T is now collection of distributions $\{\theta^{(1)},\ldots,\theta^{(8)}\}$. Each $\theta^{(i)}$ is a probability distribution over possible patterns: usually moves the ship left, usually moves the ship right, usually moves the ship up, usually moves the ship down, or moves the ship randomly. For instance, $\theta^{(i)} = [0.5, 0, 0, 0, 0.5]$ would correspond to uncertainty about whether the ith button usually moves the ship left or moves the ship randomly.

Since participants know that each button does operate consistently, although they may not be certain which consistent pattern it follows, we use the following generative model of action planning: At the start of a flight plan f, participants choose one pattern for each button; let $b_f^{(i)}$ be the participant's belief about how button i is working in plan f. The probability of choosing pattern j for button i is $\theta_j^{(i)}$. The participant then chooses the sequence of buttons to guide the ship from its current location to Earth; planning occurs in the same manner as in Experiment 1. This process is repeated for each of the four flight plans in the phase.

Due to the discrete nature of the hypothesis space, the posterior distribution over hypotheses could be calculated exactly for Experiment 1. For Experiment 3, we instead calculate an approximate posterior distribution over the continuous hypothesis space using Gibbs sampling (Geman & Geman, 1984). We sample the variables $b_f^{(i)}$ corresponding to how each button works in each flight plan; at each iteration, we choose an i and f randomly, and sample a new value for $b_f^{(i)}$ given the current values of each other latent variable. To sample this value, we compute:

$$\begin{aligned}
p(b_i^{(f)}|b_{-i}^{(1:F)}, b_i^{(-f)}, \mathbf{a}^{(1)},\ldots,\mathbf{a}^{(F)}) &\propto p(\mathbf{a}^{(1)},\ldots,\mathbf{a}^{(F)}|b_i^{(f)}, b_{-i}^{(1:F)}, b_i^{(-f)})p(b_i^{(f)}|b_{-i}^{(1:F)}, b_i^{(-f)}) \\
&= p(b_i^{(f)}|b_{-i}^{(1:F)}, b_i^{(-f)})\prod_{j=1}^{F} p(\mathbf{a}^{(j)}|b_i^{(f)}, b_i^{(-f)}, b_{-i}^{(1:F)}) \\
&= p(b_i^{(f)}|b_i^{(-f)})\prod_{j=1}^{F} p(\mathbf{a}^{(j)}|b_i^{(j)}, b_{-i}^{(j)}) \\
&\propto p(b_i^{(f)}|b_i^{(-f)})p(\mathbf{a}^{(f)}|b_i^{(f)}, b_{-i}^{(f)}), \qquad (3.5)
\end{aligned}$$

where $\mathbf{a}^{(f)}$ are the observed actions for flight plan n, $b_{-i}^{(f)}$ is the assignment of patterns to buttons other than i for plan f, and $b_i^{(-f)})$ is the assignment of patterns for button i in plans other than f. We now consider how to sample each of the two terms in the final equation.

To calculate $p\left(b_i^{(f)}|b_i^{(-f)}\right)$, we place a Dirichlet prior on each θ_i. This then corresponds to a Dirichlet-multinomial model:

$$\begin{aligned} p(b_i^{(f)} = k|b_i^{(-f)}) &= \int_\theta p(b_i^{(f)}|\theta)p(\theta|b_i^{(-f)})d\theta \\ &= \frac{\text{count}(b_i^{(-f)} = k) + \alpha_k}{F - 1 + \sum_j \alpha_j} \end{aligned} \tag{3.6}$$

where α are the parameters of the prior on θ_i and F is the number of flight plans.

The second term required to sample $b_i^{(f)}$ is the probability $p\left(\mathbf{a}^{(f)}|b_i^{(f)}, b_{-i}^{(f)}\right)$. To compute the probability, we follow a similar recursive pattern to that in Experiment 1. However, sometimes ships in Experiment 3 had broken buttons. In those cases, we restricted the space of possible actions to those that were available, and calculated the policy given that restricted space.

The above Gibbs sampling procedure allows us to compute a series of samples for how the buttons work in each flight plan. For the feedback, we choose the button b^* that the participant is least likely to know the right pattern for:

$$b^* = \underset{i}{\text{argmin}}\, \bar{\theta}_{h_i^*}^{(i)} \tag{3.7}$$

where h_i^* is the true movement pattern of button i and $\bar{\theta}$ is our empirical estimate of θ. Buttons about which feedback has already been given are excluded. We calculate the empirical estimate $\bar{\theta}$ from the samples:

$$\bar{\theta}_k = \frac{\sum_{f=1}^{F} \text{count}(b_i^{(f)} = k) + \alpha_k}{\text{num flight plans} + \sum_j \alpha_j}, \tag{3.8}$$

again relying on the fact that this is a Dirichlet-multinomial model. To compute this estimate in the experiment, we set all $\alpha_j = 1$, corresponding to a uniform prior. We generated 10,100 samples, removing the first 100 samples for burn-in. This produced similar results to using more samples or a longer burn-in period.

Computing feedback

Feedback was provided six times during the experiment, once after each flight planning phase. Feedback was constrained to be chosen without replacement so that in each phase, participants received feedback for a button about which they had not previously been given feedback. For participants in the random condition, the button was chosen uniformly at random from the remaining buttons.

For participants in the model-based condition, we used the posterior distribution computed by the inverse planning algorithm to select the feedback. Based on the model's inferences, we chose to give feedback about button b^*, the button with lowest probability that the participant had the correct understanding:

$$b^* = \underset{i}{\operatorname{argmin}}\, \theta^{(i)}_{h^*_i} \tag{3.9}$$

where h^*_i is the index of the true pattern for button i and $\theta^{(i)}_j$ is the jth element of the vector $\theta^{(i)}$. The minimum is constrained to be over only those buttons that have not previously been chosen, and if multiple buttons have the same minimal $\theta^{(i)}_{h^*_i}$, one of these buttons is chosen uniformly at random. Thus, this computation corresponds to choosing the button about which we believe the participant is most likely to have incorrect beliefs.

Results and discussion

As shown in Figure 3.5, participants who received feedback based on the model were able to correctly identify more buttons at earlier phases than participants who received random feedback. The data were analyzed using a two-way repeated measures analysis of variance in which the percentage of correctly identified buttons was predicted by feedback condition (between subjects) and phase number (within subjects), including an interaction term. There was a main effect of condition on the percentage of buttons correct ($F(1,290) = 5.53$, $MSE = 0.516$, $p < .025$). There was also a significant main effect of phase ($F(5,290) = 184.5$, $MSE = 2.32$, $p < .001$), and the interaction between these two effects was significant ($F(5,290) = 3.31$, $MSE = 0.0416$, $p < .01$). Note that as expected, the effect is largest in the beginning and middle phases; if the number of phases was increased to eight, all participants in both groups would necessarily have all buttons correct as they receive information about one new button at each phase.

We also examined whether participants in the two conditions varied in how likely they were to correctly identify all buttons in a phase. To analyze this, we used GEEQBox, a Matlab toolbox, to perform a logistic regression analysis using generalized estimating equations to correct for repeated measures of the same participants (Ratcliffe & Shults, 2008). We regressed whether all buttons were correct on condition and phase, with no interaction term, and included a constant predictor in the model. The predictor for condition had value zero for participants in the model-based feedback condition and one for participants in the random feedback condition. An unstructured correlation matrix was assumed for the repeated measures correction. Both coefficients for condition (β_1) and phase (β_2) were significant ($\beta_1 = -0.98$, $p < 0.01$; $\beta_2 = 0.64$, $p < .001$); these values show that participants in the model-based feedback condition were more likely to have all buttons correct, and participants were more likely to have all buttons correct in later phases.

We hypothesized that the main reason that model-based feedback is more effective is that it is more likely to select buttons about which the participant has an incorrect belief. To test this, we compared how often each method selected such a button. We performed a similar logistic regression analysis as above, but with an outcome variable corresponding to whether the button selected for feedback had already been correctly identified by the participant. We measured participants'

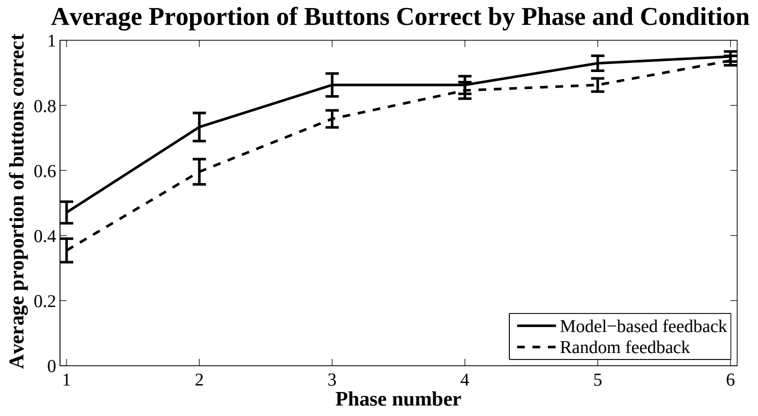


Figure 3.5: Average proportion of buttons correct, by phase and condition. Participants who receive model-based feedback have more buttons correct at earlier phases. Error bars are equal to one standard error.

choices immediately before giving feedback. For this analysis, we exclude cases where the participant had all buttons correct, as there is no chance of any feedback method choosing a button about which the participant is incorrect. Both coefficients were again significant (condition: $\beta_1 = -1.17$, $p < 0.001$; phase: $\beta_2 = -0.52$, $p < 0.001$): buttons about which the participant was incorrect were more likely to be selected for feedback in the model-based condition, and less likely to be selected in later phases. This supports our hypothesis that the reason model-based feedback results in faster learning is that it more frequently chooses buttons about which the participant has a misconception.

Overall, these results suggest that model-based feedback can speed learning. The model's inferences are accurate enough at recovering participants' beliefs to be informative, even though we know from the first two experiments that these inferences are not always correct. While the technique does require computational resources, the time to choose feedback was not prohibitive and generally took no more than 15 seconds, with participants rarely waiting for more than a few seconds. In this task, we gave relatively simple feedback. However, in more complex tasks where simply telling the learner about her misunderstanding may be less effective, the model could be used to personalize the examples and exercises that are presented to the learner, focusing on items that highlight the learner's misunderstanding, or to trigger remedial instruction on a particular concept. As Kluger and DeNisi (1996) discuss, there are many factors influencing whether a particular feedback intervention will be helpful to learners; our model provides an additional source of information that can be incorporated to construct effective feedback.

3.6 General discussion

People can make inferences about other's beliefs by observing their actions, and this ability has the potential to be very helpful in educational contexts. We have developed an inverse planning model for automatically making these inferences and shown that it can draw similar conclusions to people when given the same data. Our model relies on formulating action planning as a Markov decision process, in which people may have misconceptions about what transition model is operating in the environment. We then use a variation of inverse reinforcement learning to infer these misconceptions. This model can be used to infer either continuous or discrete hypotheses, making it applicable in a wide variety of situations. We showed that this model can be applied to customize guidance to learners in our final experiment, a common task in educational contexts. Additionally, the model we developed has been applied in other work to diagnose students' understanding in a cell biology game (Rafferty et al., in press).

In the remainder of the chapter, I discuss two additional issues. First, I consider the possibility of applying the model to a student who is learning or whose knowledge is unstable. Second, I consider the problem of defining an appropriate hypothesis space for a new application.

Unstable knowledge states

This chapter focuses on situations in which student knowledge is fixed. While we could have used data from previous phases to infer people's knowledge in later flight planning phases, we used only data from a single flight plan in order to assess how well the model could make inferences from limited data. However, there are many situations where someone might learn while completing a task or where we might observe the same student over a long period of time, during which her knowledge is presumably changing. One way of handling this issue is to incorporate a probabilistic model of learning; we discuss this approach and demonstrate its effectiveness in Chapter 5. The probabilistic model of learning must specify how particular experiences (such as a particular state-action-next state tuple) are likely to affect the student's knowledge. Defining such models for a generic domain can be challenging, but student modeling has had many successes and is an active area of research in intelligent tutoring systems (e.g., Chang, Beck, Mostow, & Corbett, 2006; Conati & Muldner, 2007; Corbett & Anderson, 1995).

There has also been work modeling people's action choices in information seeking tasks, in which a person needs to learn some information in order to be successful. For example, Fu and Pirolli (2007) examined how people navigate the internet when trying to find some information or learn about some topic. The inverse planning framework can be used to calculate the value of information seeking actions, although further experiments are needed to determine whether the noisily optimal policy is a good model of these actions or if an explicitly approximate method as in Fu and Gray (2006) is a better fit to human behavior; we also consider an approach to including reasoning about the value of information in Chapter 5. Overall, exploring the use of the inverse planning model to interpret data from an interactive environment where a student model has been defined is an importance extension of the current project. If a detailed model of learning in a domain is not available, a second option would be to include a more generic model of learning such that with some probability the person's knowledge is the same as in the past and with some probability the knowledge has changed. Thus, while we have only considered using the model to assess static knowledge, it is possible to extend its application to cases where learning occurs over the course of the behaviors.

Even in the absence of learning, the knowledge states may appear to change from one task to another. Research into misconceptions in education has shown that some misconceptions are unstable: a student may exhibit the misconception at some times but not at others (Hatano, Amaiwa, & Inagaki, 1996; Hennessy, 1994; Taber, 2000). One reason might be that the situations in which the student exhibits the misconception are sufficiently different from those in which she does not as to make different predictions about what action the student will choose, even given the misconception. This would be represented in our model by having a representation of the state space that differentiated the two situations. However, it may also be that the student's knowledge appears to be truly probabilistic. This can be represented in our model by defining a hypothesis space over probabilistic beliefs, which will often be continuous. For example, in Experiment 3, we represented participants' knowledge using a distribution over possibilities for each button. Thus, half of the time someone might behave as if she believed the button moved the ship up, and half of the time the person might behave as if she believed the button moved the ship randomly. The potential for a continuous hypothesis space allows probabilistic or unstable knowledge states to be easily represented. We further develop a case in which there is clear evidence for probabilistic application of misconceptions in the following chapter, focusing on algebra understanding.

Defining the hypothesis space

The flight planning environment was relatively easy to represent as a Markov decision process, and due to the instructions, the possible hypotheses were also reasonably well-defined. However, there are many applications where this is not the case, and even within these experiments, we considered two different ways of defining the hypothesis space. There is not a generic way to take a task and output the possible transition structures that correspond to relevant misconceptions. Instead, one must consider what types of misconceptions are of interest and along what axes people are likely to vary in their knowledge in order to construct an appropriate space. Investigating whether this process can be simplified for applying the model to new domains is an important area for future research, and a theme that will recur throughout this dissertation. One way of simplifying the process could be to try to induce the space of possible hypotheses automatically. This would require access to a large collection of existing data for the domain of interest, with actions from many different people. Such data would provide evidence for what types of knowledge variations occur in the domain.

For some specific domains, existing research has categorized either the space of dimensions on which people's knowledge varies or the specific misconceptions that people exhibit. For example, cognitive tutors cover topics in mathematics and use learner models in which knowledge is divided into a set of possible skills (Corbett & Anderson, 1995; Koedinger, Anderson, Hadley, & Mark, 1997). Such models often require significant time to construct, but recent work has addressed how to discover underlying skills automatically from students' interactions with a tutor or performance on a test (González-Brenes & Mostow, 2013; Waters, Lan, Studer, & Baraniuk, 2012). While these models do not generally identify non-normative rules that a student may believe, hypothesis spaces for our model could posit varying levels of competence in applying each skill. This would make our model applicable to a wide variety of domains where domain models of this form exist.

There have also been previous efforts to characterize specific misconceptions that students may have in particular domains, such as subtraction or algebra (e.g., Brown & VanLehn, 1980; Hatano et al., 1996; Payne & Squibb, 1990; Sleeman, 1984; Stacey & Steinle, 1999). This research identifies the space of misconceptions in these domains, and thus could be used to create hypothesis spaces for applying the inverse planning framework to games or problem solving tasks involving these domains. We use this research in the following chapter to apply Bayesian inverse planning to algebra. One of the strengths of the inverse planning framework is that it can leverage previous research exploring both what misconceptions occur as well as how common these misconceptions are.

Conclusion

We have developed a model for making inferences about people's knowledge based on observing their actions. We take a generative approach in which Markov decisions processes are used to model human action planning, and we use a variation of inverse reinforcement learning to infer people's beliefs about the effects of their actions on the environment. Our model assumes that people are approximately rational actors who choose actions that they believe will help them to accomplish their goals; the model allows us to infer what beliefs would be necessary for the observed actions to be (approximately) rational. We validated this model through behavioral experiments and applied it to providing feedback within a virtual environment. The model has the advantage of being flexible enough to be applied to a data from a variety of domains and to accumulate evidence over multiple observations. This model also has a number of practical applications in education, where it has the potential to be used to interpret data from the growing number of interactive educational technologies. Having established the accuracy of the model in the laboratory, I turn in the next chapter to a more realistic education domain: understanding students' algebra understanding by interpreting their freeform solutions to algebra equations.

Chapter 4

Diagnosing algebra understanding

In the previous chapter, we developed the Bayesian inverse planning algorithm for knowledge diagnosis, and demonstrated its success in the spaceship game. An important test of the framework is whether it can be used to diagnose understanding from tasks in more educationally-relevant domains. Data from these tasks may be more complex than the action sequences in the spaceship game, and it may be less straight forward to define the possible space of understandings. We thus turn to an investigation of diagnosing algebra understanding from students' worked solutions.[1] These solutions provide a very different type of data than that investigated in the previous chapter, providing an opportunity to examine the model's flexibility, and algebra itself is a topic of increasing importance for students. The Common Core State Standards (Common Core State Standards Initiative, 2010) mandate algebra as a core skill for high schoolers, and algebraic reasoning is vital for higher level science and mathematics. However, many students struggle with algebra, leading to the need for remedial courses and preventing these students from succeeding in higher education. Developing a Bayesian inverse planning algorithm for algebra offers opportunities for further research and technical improvement as well as practical impacts on student learners.

There currently exist a number of computer-based systems for helping students learn algebra, from intelligent tutors to websites like Khan Academy (Khan, 2006-2014). These systems typically take one of two approaches to assessing and modeling students' algebra understanding. They may structure the problems such that students enter their work in discrete parts, with each part corresponding to a different algebra skill; students generally must enter the current part correctly prior to moving on. Alternatively, these systems may use only students' final answers to infer their understanding, with many systems solely using information about whether an answer was correct or incorrect. Intuitively, while it may be pedagogically useful in some cases to structure students' behavior or interrupt their work to point out errors, it should not be necessary to do these things to infer students' understanding from their worked solutions. Just as a classroom teacher might instruct a student to "show [her] work" on an exercise to gain insight into the student's difficulties, the computer should be capable of drawing the same interpretations by making sense of the student's problem solving process.

[1]This chapter is based on work conducted in collaboration with Thomas L. Griffiths. Thanks go to Christina Vu for her assistance with annotating participants' solutions.

Bayesian inverse planning provides a framework for making these types of inferences without structuring students' solutions. We focus on diagnosing a student's understanding from observing how she solves linear equations. The model represents understanding as the tendency to make particular types of mathematical errors. This representation allows one to determine what concepts should be retaught or practiced; for example, it can differentiate between students who frequently make arithmetic errors but follow the rules of algebra from those who violate these rules but use accurate arithmetic. This is consistent with our broader goal of facilitating personalized interventions based on fine-grained inferences about students' understanding. Because Bayesian inverse planning is a generative, probabilistic model, it easily allows data from multiple problems to be incorporated into the diagnosis, just as multiple flight plans could be used to diagnose understanding in the previous chapter.

In this chapter, I begin by providing an overview of existing work modeling students' algebra skills. I next describe how to extend the Bayesian inverse planning model to interpret linear equation solving. I first test the model via simulations, and then detail the interactive website we developed to collect people's algebra problem solving data. I show that people's solutions as entered on the website can be used to diagnose their understanding, and that these diagnoses are consistent with a more conventional assessment. I end by discussing challenges and future directions for Bayesian inverse planning.

4.1 Existing research on algebra understanding

There has been a great deal of interest in modeling students' algebra knowledge and categorizing student errors, both in education and in cognitive science. Cognitive Tutor Algebra improves students' algebra skills and standardized test scores (Koedinger et al., 1997). This system uses knowledge tracing to track discrete student skills, and typically structures problem solving into individual steps which students must complete correctly before continuing. Another tutor, Aplusix, allows students to solve equations in a more freeform manner, but does not naturally maintain a model of knowledge across problems (Nicaud, Chaachoua, & Bittar, 2006). ALEKS (Falmagne, Cosyn, Doignon, & Thiéry, 2006) is commonly used for remedial mathematics practice in college settings, and relies on Knowledge Space Theory (Falmagne & Doignon, 2011) for modeling what types of problems students will be able to solve. While relying on sophisticated testing methods to assess understanding, this system uses only students' final answers to infer their understanding. ASSISTments provides students with algebra practice and guidance, and employs a detailed cognitive model for tracking skills (Razzaq, Heffernan, Feng, & Pardos, 2007). Bayesian inverse planning provides a way to add to these existing models by using free-form problem solving behavior to diagnose what a student understands and in what ways she misunderstands without requiring that individual steps be correct before the student continues.

There has also been considerable cognitive science work describing students' algebra skills and modeling their problem solutions. Koedinger and MacLaren (2002) created a detailed process model for how students solve algebra problems that were defined either symbolically or as word problems, including "buggy" rules that produce particular errors. The model both provided significant coverage of actual student data and buggy rules with plausible methods of acquisition. For example, students might learn a buggy rule through overgeneralizing an example. Brown and VanLehn (1980) also propose a theory of how errors in problem solving might occur. One of the components of the theory is that students only know how to do part of a procedure, and they patch the procedure with an erroneous action. Thus, an error might occur because a student is attempting to perform the correct operation but is unable to execute that operation. Research has also focused on categorizing patterns of algebra errors and specifying the types of buggy rules that occur (e.g., Matz, 1982; Payne & Squibb, 1990; Sleeman, 1984, 1985). Some work identified a limited set of "mal-rules" that could explain most students' solutions, although these rules were not always applied consistently (Payne & Squibb, 1990; Sleeman, 1984). While later work suggested that explicitly countering the mal-rules was no better than simply reteaching material (Sleeman, Kelly, Martinak, Ward, & Moore, 1989), detection of mal-rules and separation of different types of errors might still be helpful for guiding what material to reteach. Existing work categorizing algebra understanding provides a starting point for constructing the hypothesis space of misconceptions for Bayesian inverse planning.

4.2 Modeling algebra skills using Bayesian inverse planning

To diagnose algebra understanding from observations of students' solutions to linear equations, we develop a variation of the Bayesian inverse planning model described in the previous chapter. We define the space of possible hypotheses to represent the misconceptions that people might have about solving linear equations. By detecting patterns in a person's action choices and equation transformations, the inverse planning model can determine which of these hypotheses most likely represents the person's knowledge. A new technical challenge arises when developing a version of inverse planning to use for algebra: the state and action spaces are infinite. We first describe how we define the states, actions, and hypothesis space, and then turn to solutions for addressing the infinite state and action spaces.

The states and actions correspond to equations and the way a student changes the equation during her solution. The state can be represented as the list of terms on each side of the equation, ignoring the ordering of these terms (e.g., $2+3$ is equivalent to $3+2$). The actions are the possible transformations that a student might apply to an equation. We consider six types of actions in this initial system:

1. Move term: Move a term from one side of the equation to another.
2. Divide by a coefficient: Divide both sides by the coefficient of a term.
3. Multiply by constant: Multiply both sides by a constant.
4. Combine terms: Combine two terms on the same side of the equation.

5. Distribute over a parenthesized term: Multiply out a complex term like $3(x+6)$ to get $3x+18$. This action may also include distributing only part of the coefficient, such as transforming $-3(x+6)$ to $-(3x+18)$.

6. Stop solving: Stop solving the current problem.

The first five actions correspond to typical algebra problem solving behavior, with the action of multiplying by a constant being a more general form of dividing by a coefficient; below, we describe how errors in executing these actions are modeled. The final action is taken by a student who believes she has finished solving the problem or is giving up.

The reward model represents the goals in solving linear equations. We include a positive reward for choosing *stop* when the equation is of the form $x = c$ or $c = x$, where x is the variable and c is a constant, and a negative reward for choosing *stop* in any other state. A small negative reward is also incurred for each action, reflecting the fact that solving an equation using fewer steps is generally preferable to solving an equation using more steps.

Defining the hypothesis space

Recall that in Bayesian inverse planning, the hypothesis space represents all possible knowledge states, including possible misconceptions. In this case, the space thus includes misunderstandings about specific parts of linear equation solving. Each knowledge state is represented as a vector θ of six parameter values. Four of these parameters relate to error tendencies in applying specific actions, while the other two affect equation solving more broadly.

The four parameters related to specific actions are based on mal-rules discovered in prior work (Payne & Squibb, 1990; Sleeman, 1984). The first parameter, *sign error*, refers to moving a term from one side of the equation to the other without changing the sign: $2x+3=6$ becomes $2x=6+3$. The second parameter, *reciprocal error*, refers to multiplying by a coefficient rather than dividing by it: $5x=1$ becomes $x=5$ rather than $x=\frac{1}{5}$. The third parameter, *distributive error*, refers to multiplying only the first term in parentheses by the coefficient, rather than all terms: $4(x+3)$ becomes $4x+3$. Each of these parameters is a value between zero and one that reflects the probability the student will produce that error when applying the relevant action. For example, if $\theta_{\text{sign-error}} = 0.2$, then the probability of not changing the sign of the moved term during a move term action is equal to 0.2. These dimensions of θ thus govern the transition model operating in the MDP. Representing the erroneous rule applications as probabilities allows us to account for prior data showing that students do not always consistently apply mal-rules. The final action error parameter relates to combining terms. This parameter is equal to one if only like terms may be combined (only constants or only variables), and zero otherwise. This is implemented in the MDP by including versions of the *combine terms* action applied to non-like terms only if the hypothesis has this parameter equal to zero. This hypothesis state does not mean that only non-like terms will be considered, but that there is a wider space of available actions. This space must be considered when modeling how a student plans her actions, as it may have consequences for the sequence of actions that the student chooses.

In addition to the four action error parameters, there are two additional parameters that affect each action choice and transformation. The *arithmetic error* parameter is the probability that a student will make an arithmetic error in each operation in a transformation. For instance, when distributing the coefficient in $3(x+2+3)$, there are three opportunities to make an arithmetic error. Like the first three dimensions of θ, the arithmetic error parameter affects the transition probabilities. Including an arithmetic error parameter that is relevant to all actions allows us to distinguish between students who get problems wrong due to misunderstandings about the rules of algebra from students who have difficulties with arithmetic. The final parameter is the inferred noise parameter β in the Boltzmann policy (Equation 3.3), which represents how efficiently the student plans her actions. Unlike the other parameters, this parameter does not affect the transition model or the set of actions considered but relates to the student's choices of actions. Very low inferred values of β might also be an indication that the student's behavior is not well-modeled by inverse planning; we return to this issue when modeling real algebra solutions.

Computing the diagnosis

Just as in the previous chapter, the diagnosis of understanding is a posterior distribution over possible understandings (hypotheses). In this case, that corresponds to a posterior distribution $p(\theta|d_1,\ldots,d_N)$ over the possible student parameters given N observed problem solutions. As before, we calculate the posterior using Bayes' rule:

$$\begin{aligned}p(\theta|d_1,\ldots,d_N) &\propto p(d_1,\ldots,d_N|\theta)p(\theta)\\ &= p(\theta)\prod_{i=1}^{N} p(d_i|\theta) \qquad (4.1)\end{aligned}$$

where each term is the likelihood of the observed data for a given problem. Each term $p(d_i|\theta)$ can then be calculated using the transition model and policy, letting each problem solution d correspond to the sequence of observed states $s_{1:M} = \{s_1,\ldots,s_M\}$ (derived from the equations). These terms differ somewhat from the likelihood in the previous chapter because while all states are observed, no actions are directly observed; instead, we sum over possible actions:

$$\begin{aligned}p(s_{1:M}|\theta) &= p(s_1|\theta)\prod_{i=1}^{M-1} p(s_{i+1}|\theta,s_i)\\ &\propto \prod_{i=1}^{M-1}\sum_{a_i\in A} p(s_{i+1},a_i|\theta,s_i)\\ &= \prod_{i=1}^{M-1}\sum_{a\in A} p(s_{i+1}|\theta,s_i,a)p(a|\theta,s_i) \qquad (4.2)\end{aligned}$$

where the final state s_M is always a null state, reached by taking the *stop* action. As in the previous chapter, we assume the student chooses actions noisily optimally: $p(a|\theta,s_i) \propto \exp(\theta_\beta Q(s_i,a))$, where θ_β is the planning parameter dimension of θ. The set of transition models and action policies we consider is defined by the range of the parameters that compose the θ vector. Since five of these six parameters are continuous, we compute an approximate posterior distribution using Metropolis Hastings, an MCMC method; this mirrors our approach in the case of modeling people as having uncertainty about how the buttons worked in the final flight planning experiment. We place a Beta$(1,3)$ prior on the four continuous probability parameters in θ, favoring parameter values closer to normative algebra solving. The parameters for this prior distribution were not tuned.

To calculate the student's policy, we must solve the Q-function for the MDP defined by a particular θ. We compute an approximate solution because the state and action spaces are infinite. First, we discretize the space of actions, a common strategy in MDPs with large or continuous action spaces (e.g., Busoniu, Babuska, De Schutter, & Ernst, 2010). We create generic parameterized versions of each action that incorporate information about what types of terms (constant, variable, or complex) are being acted on but ignore the specific coefficients of those terms. Q-values are then learned for each generic action. To discretize the state space, we aggregate states into a finite set of possibilities, and then solve $Q(s,a)$ using the aggregated states and discretized actions, such that all equations which map to the same aggregated state will have the Q-values (Sutton & Barto, 1998). While this approximation may not be as accurate as some newer algorithms, it allows the Q-function to be calculated relatively quickly. Speed is important as a new Q-function must be computed for each MCMC sample.

4.3 Diagnosis using simulated data

We have described how to use Bayesian inverse planning to infer a posterior distribution representing tendencies towards particular algebra errors. We first use simulated data to investigate how accurately the error parameters can be recovered, as several sources of noise could lead to inaccuracies. First, the algorithm cannot observe what actions were used to transform equations from one state to another. While we sum over all possible actions, there still is ambiguity in which action caused a particular transformation. Second, the number of problem solutions provided limits the confidence of our estimates. If only a few solutions are available, the posterior distribution may be close to uniform, reflecting insufficient evidence to estimate the parameters. This mirrors the issues in Experiment 1 in the spaceship game, where flight plans were consistent with multiple possible beliefs. Finally, both the inferred posterior and the solutions to the MDP for each sample of the posterior are approximate. Simulations thus provide a test of how well our model could perform if it perfectly represented people's equation solving.

Methods

For each simulation, we randomly sampled a vector of parameters from the prior to represent the true hypothesis of the learner. For the planning parameter, we restricted the range to $[0.5,5]$, reflecting reasonable values we would expect for learners. We then had the simulated student solve $N = 20$ or $N = 50$ randomly generated problems. Using Bayesian inverse planning, we

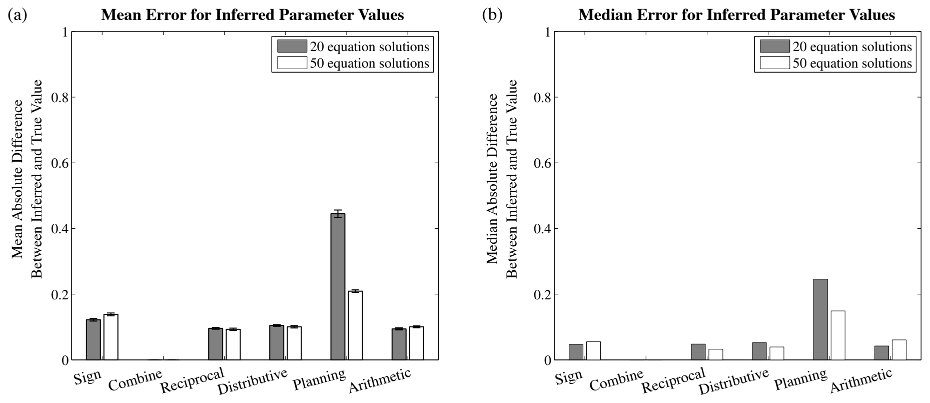


Figure 4.1: Simulation results showing the (a) mean absolute difference and (b) median absolute difference between the mean of the samples from the posterior distribution and the true parameter values. Error bars reflect one standard error.

approximated the posterior distributions given the generated data. We used $10,000$ samples with an additional $1,000$ samples for burn in. Fifty simulations were conducted for each value of N.

Results

As shown in Figure 4.1, the algorithm recovers the true parameters of the simulated learners relatively accurately. The highest error occurs for the planning parameter, which is expected given that this parameter has a larger range than the other parameters. Inspection of the simulation results reveals that large differences between the inferred and true parameters occur most often when the learner's error parameters are high, making it more difficult to distinguish the cause of incorrect transformations. For example, the simulated student might misapply an algebra rule by multiplying both sides of $3x = 6$ by the coefficient of x rather than its reciprocal, but also make an arithmetic error when multiplying 6 and 3. The end result of $x = 17$ would be consistent with either one error or two errors. When the error parameters are large, such multi-error transformations are more common. While the overall accuracy of the inferences as measured by the mean values of the posteriors is similar when $N = 20$ or 50 equations, the inferred posterior distributions are more concentrated with greater numbers of solutions (Figure 4.2).

4.4 Evaluation with human data

The results of the simulations suggest that we can recover the original parameters of the learner with reasonable accuracy. However, interpreting human equation solving is likely to be more difficult. People may exhibit more variety in their actions than our algorithm models, and they may not naturally provide problem solutions at the level of granularity that we expect. We thus turn to an experiment in which we designed an interface to collect equation solving data and use the algorithm

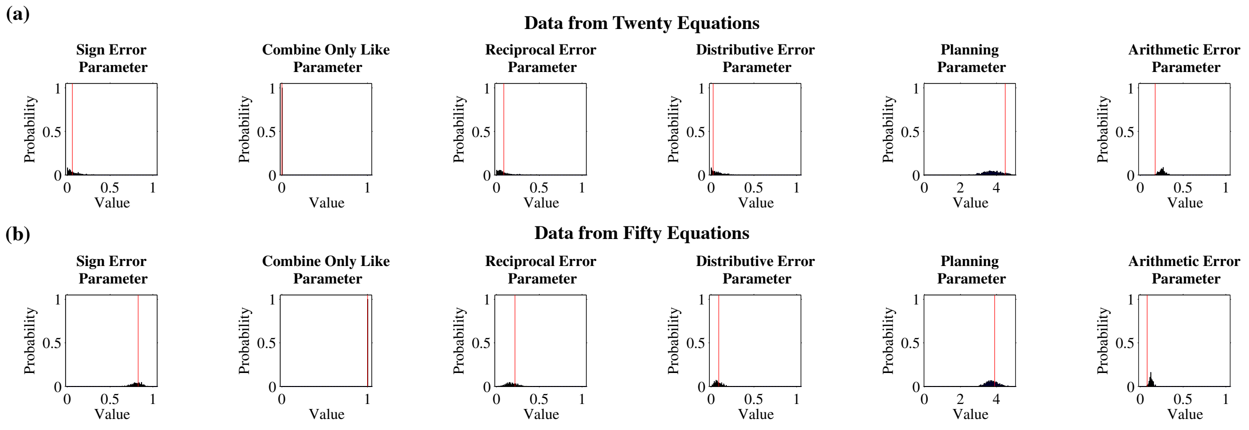


Figure 4.2: Posterior distributions from representative simulations given data from (a) twenty solutions and (b) fifty solutions. The true parameter values are shown via red lines. In both simulations, the red lines are close to the sampled values, but with fifty solutions, the posterior distributions are more concentrated, especially for the planning and arithmetic error parameters.

to interpret that data. The interface, the Berkeley Algebra Tutor (see Figure 4.3), encourages participants to enter each step of their equation solving, allowing us to collect detailed problem solving data. We evaluate what proportion of those steps can be interpreted by the Bayesian inverse planning model, and explore whether the model's inferences are consistent with other measures of their arithmetic and algebra understanding.

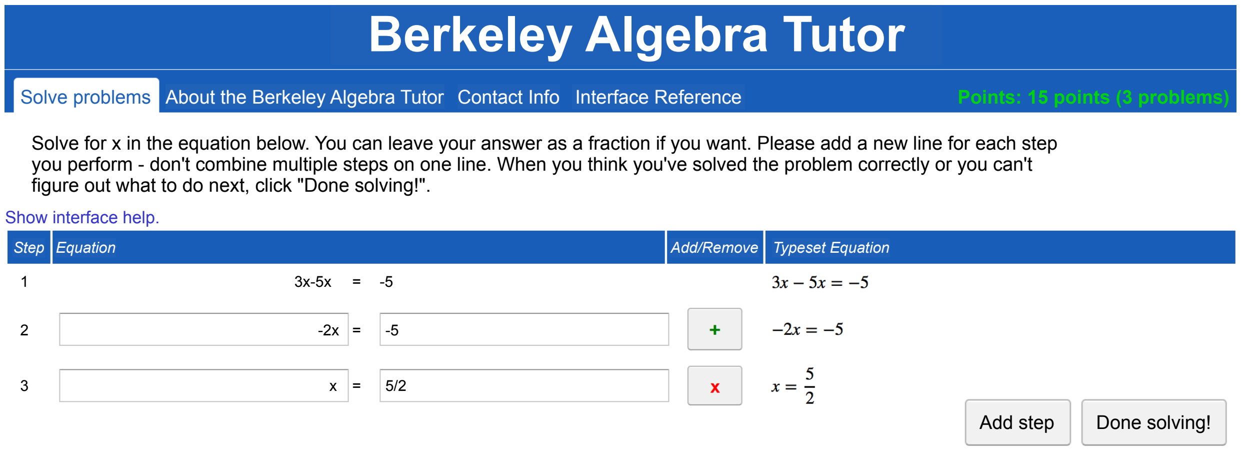
Methods

Participants

Seventy participants were recruited via Amazon Mechanical Turk. Participants were asked to only sign up for the experiment if they were at least eighteen and had taken some form of algebra but had not completed college math courses beyond algebra. They thus represent our target population of people who have had some algebra training, but may have misunderstandings or gaps in their knowledge.

Stimuli

Participants completed several activities: an online worksheet, a questionnaire, and problem solving on the algebra tutor website. The worksheet consisted of three sections: (1) ten arithmetic problems, (2) five simplification problems involving combining terms and the distributive property, and (3) twelve algebra equations. The arithmetic problems covered addition, subtraction, division, and multiplication of both positive and negative numbers. Four problems include fractions. The next section focused on combining like terms and distribution. It included items such as simplifying $2+3a+5a+9$ and simplifying $x+y-3(z+w)$; items were based on the algebra skills



Solve for x in the equation below. You can leave your answer as a fraction if you want. Please add a new line for each step you perform - don't combine multiple steps on one line. When you think you've solved the problem correctly or you can't figure out what to do next, click "Done solving!".

Step	Equation			Add/Remove	Typeset Equation
1	3x-5x	=	-5		$3x - 5x = -5$
2	-2x	=	-5	+	$-2x = -5$
3	x	=	5/2	x	$x = \frac{5}{2}$

Figure 4.3: User interface for the problem solving website. The first line in the interface shows the problem the user must solve, and the user adds lines to show her problem solving steps.

tests in Gough (2004) and Sleeman (1984). The final section of the test had linear equations to be solved; the types of these equations were based on those listed in Birenbaum, Kelly, and Tatsuoka (1993) and Sleeman (1984).

The questionnaire consisted of six questions. Four asked about math background and completion of post-secondary education, and two asked about age and gender.

The online tutor interface is shown in Figure 4.3. Participants are shown the equation that they must solve, and with each problem step, they add an additional line to show their work. The last column in the interface updates with the rendered version of the user's input in real time, with the intent of lowering barriers to interpreting typed mathematics.

Procedure

There were two parts to the experiment, which could be completed at different times: an online worksheet followed by a questionnaire with demographic questions (denoted *worksheet*), and use of the interactive equation solving interface (denoted *website*). Half of participants completed the *worksheet* activity first, and half completed the *website* section first. The instructions for the first activity informed participants of the second activity and instructed them to only complete the first part if they intended to also complete the second part. Participants were paid one dollar after completing the first activity, and three dollars after completing the second activity. Together, the two activities required about one hour. Access to the second activity was granted after the participant completed the first activity; most participants chose to provide their email addresses, and were emailed once access to the second activity had been granted. Participants had up to one week following their initial participation to complete the second activity.

In the *worksheet* activity, participants completed the online worksheet followed by the questionnaire. They were instructed that they could use scratch paper, but that they should not use a calculator.

In the *website* activity, participants solved twenty problems using the Berkeley Algebra Tutor website. Just as for the worksheet, participants were informed that they could use scratch paper if they wished but that they should not use a calculator or online resources. Prior to solving problems, they completed a short tutorial, which instructed them how to use the interface and included several interactive activities to show participants how to separate their problem solving into individual steps. The tutorial was intended to discourage participants from only giving final answers, rather than showing their work. After the tutorial, participants began solving problems. The numbers used in the problems were generated randomly, but each problem matched one of the types of problems used in the worksheet (e.g., $\textit{constant} + \textit{variable} = \textit{constant}$). Two of each of ten types of problems were included. When the participant indicated she was done solving a problem, she was told whether her solution was correct. To encourage motivation, participants received points for each problem: ten points for completing a problem correctly, and one point for attempting a problem but answering incorrectly.

Computing diagnoses

We used the MCMC sampling procedure described previously to compute a posterior distribution over the parameters for each participant. Each line in the website data was treated as one step in the participant's solution and was automatically parsed into an equation representation. Identical consecutive lines and blank lines were omitted. Each diagnosis was based on $10,000$ samples, with an additional $1,000$ samples for burn in.

Initial inspection of results showed that some participants performed multiple actions within one line in the online interface. For instance, one participant entered the step $-5x = -5-(2x-8)$ followed by $-5x = -2x+3$. This transformation cannot be realized with a single action, but seems likely to be the result of first multiplying out the complex term to get $-5x = -5-2x+8$ and then combining the terms -5 and 8. To address this issue, we augmented the sampling procedure to sum over sequences of actions. For computational simplicity, we approximate the probabilities by only considering a sequence of length n actions if no sequence $\mathbf{a}$ of $n-1$ actions has any $p(s_j|s_i,\mathbf{a}) > 0$. For a sequence of two actions, the transition probability between states s_i and s_j can be computed as follows:

$$\begin{aligned}
p(s_j|s_i) &= \sum_{a\in A} p(s_j,a|s_i)\\
&= \sum_{a\in A} p(a|s_i)\sum_{s_k} p(s_j,s_k|a,s_i)\\
&= \sum_{a\in A} p(a|s_i)\sum_{s_k} p(s_k|a,s_i)\sum_{a'\in A} p(s_j|a',s_k)p(a'|s_k),
\end{aligned}$$

where we make use of independence relations guaranteed by the Markov property. The same recursion can be repeated for multiple skipped steps, although tractability issues arise; we thus consider sequences with a maximum of two skipped steps (three total actions).

Results and discussion

Of the 70 participants who completed the first activity, 90% completed the second activity. We analyze only data from those 63 participants who completed both activities. All participants who chose to respond to our demographic questions indicated that they had taken algebra. The average score on the worksheet was 68.7%, with an average of 65.7% correct on the equation solving portion. On the website, participants correctly answered an average of 66.0% of equations correctly (13.2 equations, range: $[0, 20]$), demonstrating that overall success rate was very similar for the website and the worksheet. Total problems correct on the algebra portion of the worksheet and the website were correlated ($r(61) = 0.628$, $p < .001$).

Coverage of participant input

We first explored the model's coverage of the transformations when different numbers of skipped steps were allowed. There were a total of $5,539$ transformations, each consisting of a pair of consecutive lines that were entered into the website. Without any skipped steps, 865 of these transformations (15.6%) could not be interpreted as corresponding to any of the actions. Inspecting these transformations shows that many of them correspond to a skipped step, with some participants especially prone to combining multiple actions in a single step. When the model is permitted to consider skipped steps, its coverage improves. When one skipped step is considered, only 98 transformations cannot be interpreted (1.77%), and when two skipped steps are considered, 34 transformations cannot be interpreted (0.61%). While not all interpretations may truly reflect the intentions of the participant, these results demonstrate that allowing only two skipped steps results in over 99% coverage of the observed transformations using variations on a small set of actions.

Both accurate and inaccurate transformations were interpreted by the model, although inaccurate transformations had lower coverage. To test whether a transformation was accurate, we used a symbolic mathematics library to determine if the same value for the unknown was valid for both steps in the transformation (Kramer, 2010-2014). 87.5% of the proposed transformations were mathematically valid. When skipped actions were not allowed, the model interpreted 88% of the correct transformations and 52.5% of the incorrect transformations. Thus, the majority of erroneous transformations could be attributed to a single erroneously executed action, although some transformations likely reflected errors that were not in our model.

Classification of participant errors

To understand what types of errors participants made and how well the model's inferences about transformations matched human inferences, we annotated a subset of the problem solutions. One hundred of the 392 problems that had at least one inaccurate transformation (as annotated by the symbolic mathematics library) were randomly selected; these problems included 524 total transformations. Two annotators examined each transformation and recorded several features. First, each annotator recorded whether the transformation was mathematically accurate. This duplicates the inferences of the mathematics library, but provides a check in case the participant's work could not be interpreted accurately by the library and helps to orient annotators for the other tasks. Next, each annotator examined the actions that were given non-zero probability in one of the samples by Bayesian inverse planning. If one of these actions was judged to match the participant's intention, the transformation inference was recorded as "accurate"; otherwise, the inference was "inaccurate." This provides a measure of whether the high coverage numbers provided above reflect correct inferences about intentions, at least as compared to human inferences. If the transformation was mathematically incorrect, the annotator also recorded the type of error. Prior to annotation, ten specific error types were identified based on examination of participant solutions. An eleventh error type was added to account for "unclassified" errors not covered by the other types; see Table 4.4 for error types and descriptions. The two annotators met prior to the start of annotation to discuss the coding scheme and agree upon the procedure.

Overall, the annotators exhibited good inter-rater reliability. For the annotation of whether a transformation was an error, Cohen's $\kappa = 0.86$, with disagreement on 6% of transformations. Since neither annotator found examples where the mathematics library would have been unable to interpret a participant's input, disagreements were reconciled via comparison to the library's output and manually checked by one of the annotators. In the reconciled annotations, 34% of transformations were classified as mathematically incorrect. Forty-six of the one hundred problems had two or more incorrect transformations; the average number of incorrect transformations per problem was 1.71. Some of the problems with multiple errors appeared to be cases of participants using non-standard notation or misusing the interface to enter their work. For example, consider the following transformations entered by a participant:

$$\begin{aligned} -2-2y &= -10 \\ -2-2y &= -10+2 \\ -2y &= -8 \end{aligned}$$

In the second step, it seems likely that the participant is moving the -2 to the right hand side, as a 2 is added to that side. However, the participant retains the -2 on the left hand side. In the third step, this extra -2 is deleted and $-10+2$ is correctly combined. This sequence of steps cannot be interpreted by our algorithm; in fact, it seems to be non-Markovian as the deletion in the third step is due to the state of the first step. Some participants seem particularly prone to such behavior, suggesting there may be a reason other than carelessness for these patterns; for instance, participants may misunderstand what is being represented by equality and the series of transformations or have been taught to show their work in non-standard ways during previous instruction. This issue highlights the fact that inferring why someone acts in a particular way can be a difficult task even for people and that customizing Bayesian inverse planning for a particular domain may require iterative improvements to include all relevant misunderstandings. Despite the existence of some steps that could not be interpreted by the algorithm, the inferred actions were generally consistent with human inference. The algorithm was judged to have correctly inferred the action for 87% of transformations by the annotators; inter-annotator agreement for this feature was substantial, with Cohen's $\kappa = 0.75$, and disagreements were resolved via discussion.

Some types of errors were more common than others. As with the other annotations, inter-annotator agreement was high for this feature and disputes were addressed via discussion (Cohen's

Error number	Error name	Description
1	**Arithmetic error**	Makes an error multiplying, dividing, adding, or subtracting that does not fall into another error category.
2	**Sign error**	Moves a term from one side to another but does not change sign. For example, $2x+3=6 \rightarrow 2x=6+3$.
3	Move and change value error	Moves a term from one side to another and changes its value. For example, $2x+3=6 \rightarrow 2x=6+5$.
4	**Multiply only first error**	Multiplies a coefficient into a complex term, but only multiplies the first term in parentheses. For example, $2(3+x) \rightarrow 6+x$.
5	Partial multiply error	Multiplies a coefficient into a complex term, but only fully multiplies part of the coefficient, leaving off the sign or using only the sign in some cases. For example, $-2(x+3) \rightarrow -2x+6$.
6	**Reciprocal error**	Multiplies rather than divides by a coefficient. For example, $2x=3 \rightarrow x=6$ or $\frac{1}{2}x=4 \rightarrow x=2$.
7	Add/delete term	Adds or deletes a term while making no other changes. For example, $2x+3=6 \rightarrow 2x=6$.
8	Add/delete sign	Adds or deletes a sign from a term while making no other changes. For example, $2x-3=6 \rightarrow 2x+3=6$.
9	Modify term	Modifies the value of a single term while making no other changes. For example, $2x=18 \rightarrow 2x=47$.
10	**Combine unlike terms**	Combines a variable and a constant term. For example $2x+3 \rightarrow 5x$.
11	Unclassified	Error does not fit any of the above classifications.

Figure 4.4: Descriptions of each error included in the annotation of participants' transformations. Errors with bolded numbers are directly captured by the θ inferred via Bayesian inverse planning, although other errors may be specific versions of mistakes captured by the algorithm (e.g., *move and change value error* is treated as an arithmetic error).

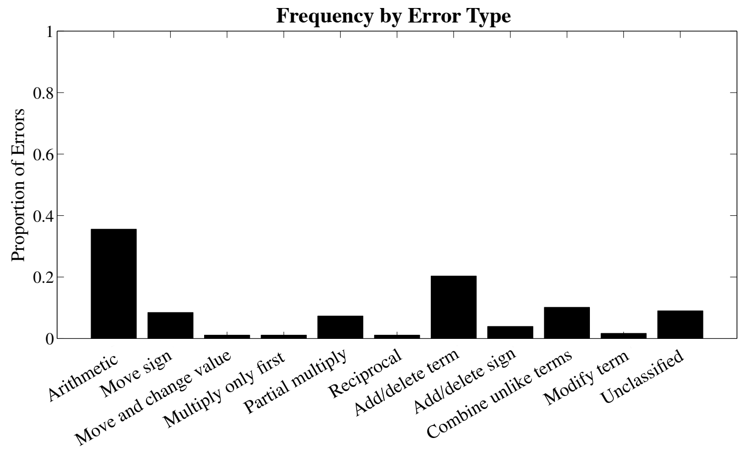


Figure 4.5: Errors by type, as classified by annotators. *Arithmetic* errors are the most common, followed by *add/delete term*.

$\kappa = 0.78$). As shown in Figure 4.5, *arithmetic errors* were the most common type of error. Errors that did not fit into the classification scheme made up 9% of annotated transformations, although some of these transformations were the result of two errors (e.g., *arithmetic error* and *combine unlike terms*) and would be interpreted correctly by the algorithm. While many of the error types are covered by Bayesian inverse planning, the large number of *add/delete term* errors suggests that an error of this type should be included in future versions of the model. One way to incorporate this error would be to add an *add/delete* action that has a high cost. Alternatively, one might consider adding a dimension to θ to reflect the propensity of individual participants to use this action. Given that this action could indicate carelessness or a much deeper misunderstanding of either algebra or the interface, recognizing that someone frequently adds and deletes terms could be a useful trigger for an intervention.

Diagnosis of mathematical skills

The inferred posterior distributions demonstrate the skills of individual participants. Figure 4.6 shows the posterior distributions for a single participant, with green lines showing the mean of each distribution. The arithmetic error parameter is relatively large (mean value 0.235), which is consistent with the participant's worksheet score of 5 of 10 arithmetic question correct. The distributive error parameter is relatively low (mean value 0.089), matching the fact that none of the participant's answers for the worksheet simplification questions were consistent with multiplying only the first item. The sign error parameter is somewhat higher (mean value 0.133). While none of the worksheet questions directly test this skill, inspection of the tutor results does show examples of this error. For instance, the participant transforms $-11x = 3x + 68$ into $-8x = 68$. This is likely due to a skipped step of $-11x + 3x = 68$, reflecting a sign error. The somewhat low planning parameter, indicating non-optimal planning, is likely due to the fact that the participant submitted two of the twenty problems on the website prior to giving answers of the form $x = c$ or $c = x$;

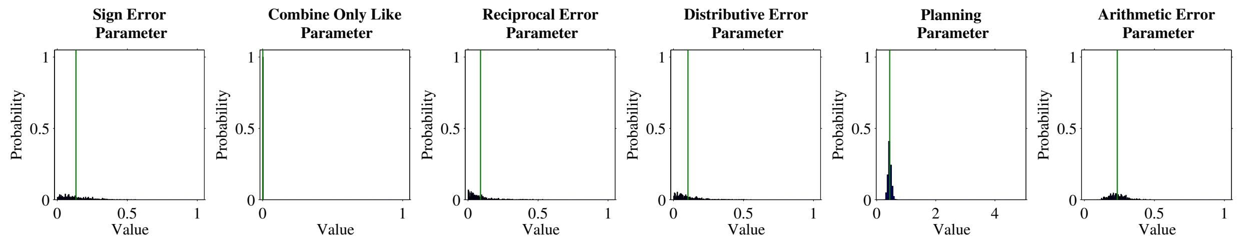


Figure 4.6: Inferred posterior distributions for a single participant, with green lines indicating mean values. The distributions provide a profile of this participant's skills.

on the worksheet, the participant was also unable to provide an answer for three of the equation solving questions.

To identify whether each participant's data was well fit by the model, we examined the inferred planning parameters. When this value is very small, the person is modeled as choosing actions nearly uniformly at random, generally indicating poor fit. 71% of participants had mean inferred planning parameters greater than 0.3 when the number of skipped steps permitted was set to zero, with 78% meeting this threshold when skipped steps were permitted. The cutoff for the planning parameter was chosen based on the distribution of the data: most excluded participants had inferred parameters below 0.2, and relatively few participants had inferred parameters between 0.2 and 0.4. Examining the data of those who were poorly fit by the model, we noticed that some participants truly did seem to be planning poorly while others made errors not covered by the model. In the first group are participants who frequently submitted problems without reaching a solution. In these cases, the low planning parameter may not actually reveal a poor fit, but is a reflection of the fact that this action is high cost: participants either do not in fact know how to sequence actions to reach a solution, or they are unwilling to make the effort to complete these actions. In many cases, however, the low planning parameter was due to errors that were not covered by our proposed actions. Some of these errors were systematic, suggesting that adding additional actions to the model would result in better fit. In other cases, we noticed the same pattern as when annotating individual problems: some participants repeatedly use the interface in ways that suggest misunderstandings not accounted for by the model. For example, one of the participants whose solutions frequently contained errors categorized as *add/delete term* had a mean inferred planning parameter of 0.012. Expanding the space of possible understandings would allow us to account for these participants.

While ideally the model would provide a good fit to all participants' data, it is encouraging that the planning parameter can be used to identify participants whose actions may be misinterpreted. Identifying participants with poorly fit data can mitigate problems due to incorrect action interpretations. Sometimes, transformations that are not part of our model will be interpreted as actions with errors, leading the algorithm to overestimate error parameters. For instance, one participant was given the problem $3x+7x+x=-7-3-9$, and added only one additional step, $3x+7x+x=\frac{-7}{3}-\frac{1}{7}-9$. Bayesian inverse planning can account for this transformation as multiplying by a constant, with several arithmetic errors. However, intuitively, the participant seems instead to have been trying to manipulate $-7-3$ in some way. I consider this issue further in the Section 4.5.

To assess the relation between the inferred posterior distributions and participants' responses to the worksheet, we focused on participants' arithmetic skills, which were measured in isolation by ten problems on the worksheet. Using only data from those participants well fit by the model, we looked at the correlation between the mean inferred value for the arithmetic error parameter and the number of correct responses on the arithmetic section of the worksheet. First using the inferences with two skipped steps, which provides the greatest coverage of the data, we found that there was a significant correlation between the mean inferred arithmetic error parameter and the scores on this section of the worksheet ($r(47) = -0.39$, $p < .01$): higher worksheet scores were associated with lower inferred arithmetic error rates on the website. This association was maintained even when fewer skipped steps were allowed: one skipped step permitted: $r(47) = -0.43$, $p < .01$; no skipped steps permitted: $r(43) = -0.49$, $p < .001$). Thus, despite the fact that some actions may be misinterpreted, the model's inferences are still generally consistent with an isolated assessment of students' arithmetic skills. This correlation is also not simply due to participants' overall competence being tied to their arithmetic skills. There was no significant correlation between the inferred arithmetic error parameter and the total score on the worksheet with the arithmetic section excluded (no skipped steps: $r(43) = -.24$, $p > .1$; one skipped step: $r(47) = .22$, $p > .1$; two skipped steps: $r(47) = -.18$, $p > .2$).

While we examined the relation between the inferred values for the other parameters and participants' scores on other sections of the worksheet, the connection between these variables was less clear. The worksheet section on simplification was relatively short (5 problems), and each problem may have involved arithmetic, combining terms, and the distributive property. The error associated with multiplying by a constant (*reciprocal error*) was tested only in the context of solving equations, where we would expect all parameters to have an effect on accuracy. Exploratory analyses using PCA and CCA to uncover more complex associations between the parameter values and worksheet performance were consistent with there being some signal present in the inferred parameter values to make inferences about performance on the worksheet, but were not readily interpretable for making strong connections between the diagnoses and the worksheet.

To better understand whether the diagnoses were consistent with problem-solving behavior in the tutor, we used data from our annotation of errors in individual problems. The 100 problems that were annotated were solved by 49 of the 63 total participants. Fourteen participants exhibited a *sign error* in the annotated data. These participants had a larger inferred move sign parameter than the other 35 participants whose annotated problem solutions did not have this error, regardless of the number of skipped steps allowed (no skipped steps: 0.24 versus 0.22; one skipped step: 0.26 versus 0.20; two skipped steps: 0.24 versus 0.19). Given the relatively small magnitude of the difference, further annotation efforts might be useful to better differentiate those who never exhibited this error in their problem solving from those who did make this error. Another error identified in the annotated data was *combine unlike terms*, which can be mapped to the combine only like terms parameter in the diagnosis. Only two participants exhibited this error in the annotated data. One of these participants frequently combined multiple actions in one step, resulting in little data for diagnosis when skipped steps were not allowed; this participant had a mid-range value for the combine only like terms parameter with no skipped steps, and a very low value when one or two skipped steps were permitted. The other participant always had a very low inferred value for this parameter, corresponding to a diagnosis of combining unlike terms. In our annotated data, we had few instances of participants making a distributive property error or of the reciprocal error; across all participants, the two corresponding parameters had smaller inferred values than the move sign error parameter. The consistency between some of our inferred parameters and our annotations provides suggestive evidence of the validity of the diagnoses; supplementing this evidence with a broader set of annotations, covering all data from a subset of participants, as well as with data from participants who exhibit evidence of the less common errors will be a next step for better understanding individual diagnoses.

4.5 General discussion

In this chapter, we have demonstrated how to extend the Bayesian inverse planning framework developed in the previous chapter to a real educational domain, algebraic equation solving. The framework separates distinct causes of errors, inferring a diagnosis representing the person's skills in several dimensions of mathematical ability. In simulations, we have shown that the algorithm can recover the true parameters of a simulated learner relatively accurately, especially when the true error parameters are not too large. The algorithm successfully interpreted data from human problem solvers, and inferences about people's arithmetic skills were consistent with their performance on arithmetic problems in isolation. I now consider future directions and challenges for this technique, building on the themes in the previous chapter.

The Bayesian approach that we have taken infers posterior distributions over the inferred parameters, and the parameters we consider are primarily real-valued. The advantage of this approach is that it clearly indicates how much the evidence supports particular parameter values, but it can require significant data to acquire confident estimates; this issue came up in the previous chapter, but represents an even larger issue for algebra solving given the amount of evidence gathered in each equation solution and the many continuous parameters in the hypothesis space. In simulation, we saw that even with fifty equation solutions, some of the posterior distributions placed significant weight on a range of different values. This indicates that caution must be used when making comparisons between students or between the relative value of different parameters when only means of the posterior distribution are used. However, while small differences in parameter values may not be detectable given the number of problems likely to be solved by real students, these differences are also unlikely to be significant for informing interventions. Another tactic would be to assign students particular problems to complete based on which parts of the diagnosis are least certain. In the next chapter, we develop a general framework for choosing assessments that are likely to result in less uncertain diagnoses.

Unlike approaches that rely only on students' final answers, Bayesian inverse planning closely models students' action planning. This allows the algorithm to make detailed inferences based on problem solving data, but also requires assumptions about the types of transformations students execute and the errors that they make. In some cases, these assumptions make it impossible to interpret particular transformations, as they do not correspond to any of our actions or errors, and in other cases, they may lead to incorrect inferences, as a transformation may be interpreted as having many errors rather than correctly recognized as not corresponding to any interpretable action. At the root of these issues is how to define the Markov decision process corresponding to linear equation solving such that all possible student actions are represented and systematic errors are reflected in differences in the underlying knowledge state. One way to address this issue is to consider how to expand the space of actions and erroneous versions of actions. Transformations that could not be interpreted provide a potential source for additional actions. In principle, this process could be automated by searching through a simple grammar of possible operations defined on individual terms (e.g., dividing a term by a coefficient) to identify what could account for each transformation; for instance, this could uncover errors like dividing only the first term on a side by a coefficient, rather than all terms. Common sequences of operations could then be added as actions or error types to the inverse planning algorithm. This procedure would require minimal human intervention to label errors for the purpose of identifying customized guidance to correct those errors.

Addressing the issue of erroneous interpretations of transformations based on existing actions is more difficult. There will always be ambiguity in terms of whether a student truly intended to take a particular action and made an error, or whether a different, perhaps unknown to the algorithm, action was intended. One sign of poor fit that we used in our analyses was an extremely low planning parameter; while this could accurately reflect a student's abilities, it is likely to occur more frequently for students who take actions that the algorithm does not model. There are several additional ways of identifying poor fit that we intend to explore. We currently allow transformations that are mathematically correct (in that the same value for the unknown is valid for both steps) to be attributed to actions with arithmetic or other errors. While in principle this is possible, as two arithmetic errors might cancel each other out, it may be more likely that this is an erroneous action interpretation. Not allowing the algorithm to consider erroneous versions of actions in cases of mathematically correct transformations might lead to poorer coverage but more accurate inferences. Poor fit might also be identified by the clustering of a student's arithmetic errors across transformations. If a student typically has few arithmetic errors but a few transformations with many of these errors, this could indicate that the transformations with many errors are in fact actions that the algorithm cannot interpret.

In general, the issue of interpreting student actions points to the tension inherent in seeking to draw precise inferences without overly strong assumptions. This theme is intrinsic to the problem of combining computational modeling and machine learning to improve and model learning in interpretable ways. In each of our investigations, we will see a version of this tension and in many cases, we will explore variations on our assumptions to determine the effect on learning outcomes. For example, in the previous chapter, we explored two different ways of defining the space of possible misunderstandings. In Chapter 7, I provide a more general overview of this theme.

In the previous chapter, we explored customized guidance to participants based on the model's diagnosis of their understanding. One key next step for this research is testing whether customized interventions of this kind are helpful to algebra learners. The algebra website that we developed to collect student data provides a platform for this future testing. We have implemented customized interventions for errors related to combining unlike terms, moving terms, multiplying by the reciprocal of a coefficient, the distributive property, and difficulty with arithmetic. Each of these errors is based on one parameter of the diagnosis of the participant's understanding, and most interventions include both an embedded video created by an existing educational website as well as scaffolded problem solving practice. These interventions thus have the potential to help learners find the resources that are most relevant to them based on continuing assessment of their skills. The interventions also provide a practical test of whether the diagnoses can improve learning.

Over the past two chapters, we have developed the Bayesian inverse planning framework and shown that it can be adapted to interpret behavior in several different domains, from navigation in a spaceship game to transformations of algebraic equations. Modeling students' action planning is a promising way to interpret freeform actions in a variety of interactive educational activities, and in this chapter, we have provided a template for how to address common issues that may arise, such as large or infinite state and action spaces. Our analysis of people's equation solving highlighted the fact that the majority of action sequences can be interpreted using only a small set of actions and suggested ways that the space of understandings and possible actions might be expanded to improve the model's inferences. While future work will include additional experiments aimed at demonstrating a tighter connection between the algorithm's inferences and people's performance outside of the website as well as testing with school-age students, this chapter provides a proof of concept for using Bayesian inverse planning in a complex, educationally-relevant domain.

Chapter 5

Designing more diagnostic interactive activities

The previous two chapters developed a Bayesian inverse planning framework to diagnose people's understanding based on observing their actions, and demonstrated that this framework is consistent with other assessments and can be applied to a variety of tasks and domains. However, our data also demonstrated that in some cases, there was insufficient information to confidently identify a single misunderstanding as corresponding to a learner's knowledge. Instead, several misunderstandings or a range of parameter values might have similar probabilities. This situation can make guiding interventions more difficult: more information must be collected to determine which intervention is appropriate, or the chosen intervention must be low cost or able to help learners regardless of their particular misunderstanding. If the diagnosis is meant to be used as an assessment, a diagnosis with multiple hypotheses with high probability makes the results difficult to interpret. To remedy this issue, I turn to the problem of designing more diagnostic interactive activities, including games, online assessments like our algebra website, and virtual laboratory tasks.[1]

Designing assessments is a challenging task. Ideally, assessments should provide as much information as possible in a limited amount of time, with the results of the assessment differentiating between learners who have unequal knowledge or skills. In assessment design, frameworks like evidence-centered design aim to ensure that assessments provide opportunities for learners to demonstrate differing levels of understanding and make explicit the connection between behavior and inferences about knowledge (Mislevy, Steinberg, & Almond, 2003). However, while these frameworks are applicable to technology-enhanced assessments, they can be difficult to use effectively to design interactive assessments. Designing these assessments often involves making choices about a large number of parameters, such as the points for each action or accomplishment in a game or the locations of items within an activity. These choices can have significant effects on the eventual diagnosis of the learner's understanding, but the human designer may not have strong intuitions about what settings will be best. Additionally, when versions of a task are generated au-

[1]This chapter is based on work conducted in collaboration with Matei Zaharia and Thomas L. Griffiths. Parts of this work were included in Rafferty, Zaharia, and Griffiths (2012).

tomatically, such as the random placement of the spaceship for the flight planning experiments in Chapter 3, different versions may be more or less diagnostic, with no input from a human designer. We would like to be able to present different, automatically-generated variations of an interactive task while restricting the variations to those that are likely to result in unambiguous inferences about a learner's understanding.

In this chapter, I focus on how to predict how much information can be gained from a particular interactive activity design and modify the design to result in gaining more information about the learner's knowledge or cognitive processes. The approach we take draws on ideas from optimal experiment design. Like designing interactive activities, designing traditional experiments also requires significant time and effort. While the number of parameters to adjust may be smaller, the experimenter must still set quantities such as what time delays to include in a memory task or what treatment dosages to compare in a medical experiment. The statistical theory of optimal experiment design aims to ease this problem by identifying the design that will give the most information about the dependent variable (Atkinson et al., 2007; Pukelsheim, 2006). In chemistry, this technique has been used to discover the value of various parameters relevant to a reaction, making laboratory syntheses more successful (e.g., Dantas, Orlande, & Cotta, 2002; Emery, Nenarokomov, & Fadale, 2000; Fujiwara, Nagy, Chew, & Braatz, 2005), and the approach was used to develop and validate a new method for synthesizing a compound that has now been used in industry (Shin et al., 2007). Optimal experiment design has also been used in pharmacology and clinical applications (e.g., Bruno et al., 1998; Derlinden, Bernaerts, & Impe, 2010; Haines, Perevozskaya, & Rosenberger, 2003; Simon, 1989), resulting in greater certainty about the effectiveness of new drug therapies while reducing trial costs. Across fields, the idea of setting experiment parameters to optimize the information gained about the phenomena under investigation has made it easier to obtain precise answers while minimizing resource use (e.g., Ajo-Franklin, 2009; Elvind, Asmund, & Rolf, 1992).

To address the challenge of designing interactive activities that will be informative about people's understanding, we introduce optimal game design, a new formal framework that extends optimal experiment design to identify designs for games or other activities that will diagnose people's knowledge more efficiently. While we call the framework "optimal game design" and the experiments in this chapter focus specifically on games, this framework can optimize any activity for which Bayesian inverse planning is applicable. The framework identifies the interactive activity design with maximal utility, where utility is defined as the expected information gain about a person's understanding or cognitive processes. Like optimal experiment design, our procedure takes an existing design and considers how to modify it to be most informative. For traditional experiments, these modifications might include parameters such as at what time intervals to test recall; for interactive activities, these modifications include parameters like the amount of points for different types of accomplishments or the location and frequency of particular objects in a game. Optimal game design makes use of Bayesian inverse planning to interpret people's actions, and to simulate how people might behave if they had a particular understanding. The framework leverages the skills of human designers for creating the initial design. By automating the process of refining that design, the framework limits the trial and error necessary to create an interactive activity that will provide useful data. This facilitates more efficient and interpretable assessments.

With this chapter, I also broaden our scope to consider the use of games in the social sciences as well as interactive activities in education. Games are increasingly popular behavioral research tools (e.g., Von Ahn, 2006; Siorpaes & Hepp, 2008; Puglisi, Baronchelli, & Loreto, 2008). They provide a way of recruiting large numbers of engaged participants, and offer a powerful method for increasing participant satisfaction and diminishing participant disinterest. They may also facilitate longer, more involved behavioral experiments. Yet, designing these games such that confident inferences about cognitive processes can be drawn from players' observed behaviors raises many of the same challenges as designing diagnostic interactive activities for education.

In this chapter, I first provide background on optimal experiment design. I then combine this idea with Bayesian inverse planning to create the framework for optimal game design. The remainder of the chapter applies this general framework to the specific case of learning Boolean concepts. I introduce a novel concept learning game and use our approach to optimize the game parameters. Through behavioral experiments, we show that optimized game designs can result in more efficient estimation of the difficulty of learning different kinds of Boolean concepts. These results demonstrate that this estimation can be complicated by people's own goals, which may not match incentives within the game, but can be accommodated within our framework. I end by summarizing the benefits of optimal game design as well as the limitations of this framework.

5.1 Bayesian experiment design

Bayesian experiment design, a subfield of optimal experiment design, seeks to choose the experiment that will maximize the expected information gain about a parameter θ (Atkinson et al., 2007; Chaloner & Verdinelli, 1995). In psychology, this procedure and its variations have been used to design experiments that allow for clearer discrimination between alternative models, where θ corresponds to an indicator function about which of the models under consideration is correct (Cavagnaro, Myung, Pitt, & Kujala, 2010; Myung & Pitt, 2009). Throughout this chapter, let ξ be an experiment (or game) design and y be the data collected in the experiment. The *expected utility (EU)* of a game is the expected information gain about the parameter θ:

$$\begin{aligned} EU(\xi) &= \int p(y|\xi)U(y,\xi)\mathrm{d}y \\ \text{where} \quad p(y|\xi) &= \int p(y|\xi,\theta)p(\theta)\mathrm{d}\theta \\ \text{and} \quad U(y,\xi) &= \int \left(H(p(\theta|y,\xi)) - H(p(\theta))\right)\mathrm{d}\theta, \end{aligned} \tag{5.1}$$

where the function $H(p)$ is the Shannon entropy of a probability distribution p, defined as $H(p) = \int p(x)\log(p(x))\mathrm{d}x$. The Bayesian experimental design procedure seeks to find the experiment ξ that has maximal EU. Intuitively, designs that are likely to result in more certainty about θ will have higher utility.

5.2 Optimal game design

We can now define a procedure for optimal game design, identifying the game or other interactive assessment with maximum expected information gain about some theoretical quantity of interest θ by using optimal experiment design and Bayesian inverse planning. Since the experiments in this chapter involve games specifically, we will use "game" to refer to any interactive assessment or game to which we can apply Bayesian inverse planning in order to simplify our presentation. The procedure we describe can be applied to diagnosing the understanding of an individual person, as is common in the educational assessments we discussed in previous chapters, or to drawing inferences about a cognitive model based on the actions of multiple individuals, as is common when using games for behavioral experiments. We focus on the case of using the actions of multiple people, while noting how to modify this procedure to optimize an assessment for an individual.

The optimal game design framework improves an existing game by adjusting its parameters to be more diagnostic; these parameters may correspond to point values, locations of items, or any other factor that can be varied. To apply Bayesian experiment design to the problem of choosing a game design, we define the expected utility of a game ξ as the expectation of information gain over the true value of θ and the actions chosen by the players:

$$EU(\xi) = E_{p(\theta,\mathbf{a})}[H(p(\theta)) - H(p(\theta|\mathbf{a},\xi))], \tag{5.2}$$

where $\mathbf{a}$ is the set of action vectors and associated state vectors for all players. The expectation is approximated by sampling θ from the prior $p(\theta)$, and then simulating players' actions given θ. Intuitively, players' actions in the game must be connected to parameter θ that we intend to infer; otherwise, we cannot hope to gain information about this parameter from the observed actions. We use MDPs to formalize the link between θ and actions by assuming that each θ implicitly corresponds to believing that a particular MDP governs the game; this is the same type of link we assume for Bayesian inverse planning. We can then simulate players' actions by calculating the Q-values for the MDP associated with a particular θ, and sampling from the noisily optimal Boltzmann policy used in previous chapters (Equation 3.3).

To compute $EU(\xi)$ once the players' actions have been simulated, the distribution $p(\theta|\mathbf{a},\xi)$ must be calculated. Using the assumption that each θ corresponds to a particular MDP, we can calculate a distribution over values of θ based on the observed sequences of actions $\mathbf{a}$ of all players in the game ξ:

$$\begin{aligned} p(\theta|\mathbf{a},\xi) &\propto p(\theta)p(\mathbf{a}|\theta,\xi) \\ &= p(\theta)p(\mathbf{a}|\mathrm{MDP}_\theta,\xi) \\ &= p(\theta)\prod_i p(\mathbf{a}_i|\mathrm{MDP}_\theta,\xi), \end{aligned} \tag{5.3}$$

where $\mathbf{a}_i$ is the vector of actions taken by player i and MDP_θ is the MDP derived for the game based on the parameter θ. This corresponds to using Bayesian inverse planning to interpret the set of all players' actions rather only an individual players' actions. Using the actions of all players to infer θ generally makes sense in behavioral experiments as we assume that θ corresponds to the parameters of a cognitive process that is shared, rather than to an individual's knowledge; the same type of procedure can be used to calculate the expected utility of a game design for an individual by using only a single vector of actions. In such cases, the prior distribution could also be adjusted to include previous information collected about the given individual. As we saw in our development of the Bayesian inverse planning model, the posterior distribution can be calculated exactly if there is a fixed set of possible θ or by using MCMC methods if the set of θ is large or infinite (see Gilks, Richardson, & Spiegelhalter, 1996).

Now that we have defined $p(\theta|\mathbf{a},\xi)$, we can use this to find the expected utility of a game. Equation 5.2 shows that this calculation follows simply if we can calculate the entropy of the inferred distribution. In the case of a fixed set of possible θ, $H(p(\theta|\mathbf{a},\xi))$ can be calculated directly. If MCMC is used, one must first infer a known distribution from the samples and then take the entropy of that distribution. For example, if θ is a multinomial and $p(\theta)$ is a Dirichlet distribution, one might infer the most likely Dirichlet distribution from the samples and find the entropy of that distribution.

The above procedure describes how to (approximately) calculate the expected utility of a particular game design ξ. To complete the procedure for optimal game design, any optimization algorithm that can search through the space of games is sufficient. Maximizing over possible games is unlikely to have a closed form solution, but stochastic search methods can be used to find an approximate solution to the game with maximum expected utility. For example, one might use simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983). This method allows optimization of discrete and continuous parameters, where neighboring states of the current game are formed by perturbations of the parameters to be optimized.

5.3 Optimal games for Boolean concept learning

We have described a general framework for automatically finding game designs that are likely to be highly informative about model parameters. This framework can be applied to optimizing games designed to infer parameters that are shared across individuals, as in many behavioral experiments, or to diagnose individual knowledge, as in the types of applications discussed in the previous chapters. To test how well this framework identifies informative designs, we applied it to investigating a particular psychological question: what is the relative difficulty of learning various Boolean concept structures? This question has been studied extensively in past work (e.g., Feldman, 2000; Griffiths et al., 2008; Nosofsky et al., 1994; Shepard et al., 1961), so we can compare our results to those produced using more traditional methods. We first describe Boolean concept learning, and then turn to the initial game we created and the application of optimal game design.

Boolean concepts

In Boolean concept learning, one must learn how to categorize objects that differ along several binary dimensions. We focus on the Boolean concepts explored in Shepard et al. (1961). In these concepts, there are three feature dimensions, resulting in 2^3 possible objects, and each concept

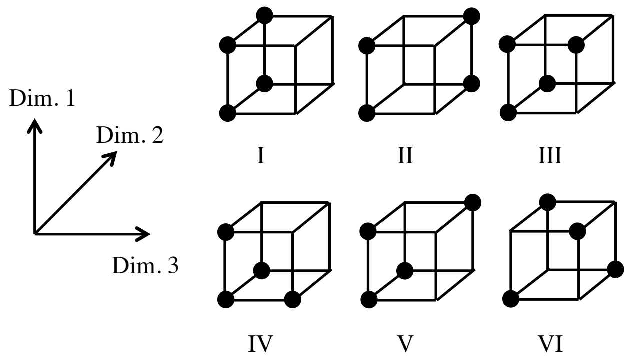


Figure 5.1: Boolean concept structures. In each structure, eight objects differing in three binary dimensions are grouped into two categories of four elements. Each object is represented as a corner of a cube based on its combination of features, and the objects chosen for one category in each problem type are represented by dots.

contains four objects. This results in a total of 70 concepts with six distinct structures, as shown in Figure 5.1. Shepard et al. (1961) found that the six concept structures differed in learning difficulty, with a partial ordering from easiest to most difficult of $\mathrm{I} > \mathrm{II} > \{\mathrm{III}, \mathrm{IV}, \mathrm{V}\} > \mathrm{VI}$. Similar results were observed in later work (Feldman, 2000; Nosofsky et al., 1994) although the position of Type VI in the ordering can vary based on how the stimuli are presented (Griffiths et al., 2008).

To model learning of Boolean concepts, we assume learners' beliefs about the correct concept h can be captured by Bayes' rule (Griffiths et al., 2008):

$$\begin{aligned} p(h|\mathbf{d}) &\propto p(h)p(\mathbf{d}|h) \\ &= p(h)\prod_{d\in\mathbf{d}} p(d|h), \end{aligned} \tag{5.4}$$

where each $d \in \mathbf{d}$ is an observed stimulus and its classification, and observations are independent given the category. The likelihood $p(d|h)$ is then a simple indicator function. If the stimulus classification represented by d matches the classification of that stimulus in hypothesis h, denoted $h \vdash d$, then $p(d|h) \propto 1$; otherwise, $p(d|h) = 0$. We seek to infer the prior $p(h)$, which represents the difficulty of learning different concepts and thus gives an implicit ordering on structure difficulty. In our earlier terminology, θ is a prior distribution on concepts $p(h)$. For simplicity, we assume all concepts with the same structure have the same prior probability, so θ is a 6-dimensional multinomial.

Corridor Challenge

To teach people Boolean concepts we created the game Corridor Challenge, which requires learning Boolean concepts to achieve a high score. Corridor Challenge places the player in a corridor of

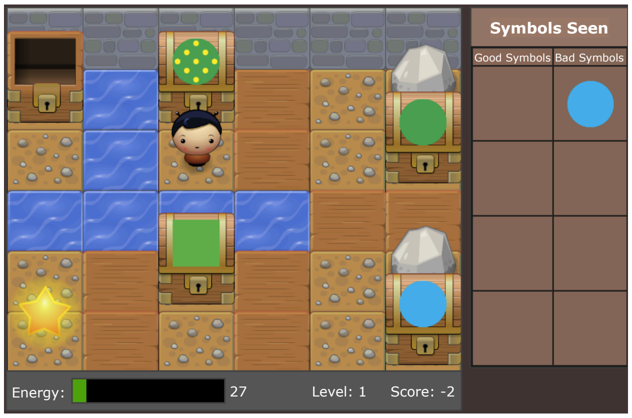


Figure 5.2: User interface for the Corridor Challenge game (Level 1 of the random game in Experiment 1). In this screenshot, the player has opened the first chest and moved to the second island.

islands, some of which contain a treasure chest, joined by bridges (Figure 5.2).[2] The islands form a linear chain and the bridges can be crossed only once, so players cannot return to previous chests. Some chests contain treasure, while others contain traps; opening a chest with treasure increases the player's score and energy, while opening a chest with a trap decreases these values. Each chest has a symbol indicating whether it is a trap; symbols differ along three binary dimensions and are categorized as a trap based on one of the Boolean concepts. Players are shown a record of the symbols from opened chests and their meanings (see the right hand side of Figure 5.2). Players are told to earn the highest score possible without running out of energy, which is depleted by moving to a new island or opening a trapped chest. When a player runs out of energy, the level is lost and she cannot explore the rest of the level; surviving a level earns the player 250 points. Corridor Challenge games may consist of several levels. Each level is a new corridor with different chests, but the same symbols are used and they retain the same meaning as on the previous level. At the start of each level, the player's energy is restored, but points are retained from level to level.

Optimizing Corridor Challenge

Applying optimal game design to Corridor Challenge requires specifying the parameters to optimize in the search for the optimal game, formulating the game as an MDP, and specifying the model for how the player's prior distribution over concept structures (θ) relates to the MDP parameters. The structure of Corridor Challenge allows for many variants that may differ in the expected information gain. To maximize expected information gain while keeping playing time relatively constant we limited the game to two levels, with five islands per level. We then used optimal game design to select the number of points gained for opening a treasure chest, points lost for opening a

[2]Corridor Challenge uses freely available graphics from `http://www.lostgarden.com/2007/05/dancs-miraculously-flexible-game.html`.

trap chest, the energy lost when moving, the symbols that appeared on the chests, and the Boolean concept used to categorize the chests. For simplicity, we assumed that the number of points gained (or lost) for a particular action is equal to the amount of energy gained (or lost) for that particular action.

Given particular specifications for these variants of the game, we can define an MDP. Note that we define the MDP based on a player's beliefs, since these govern the player's actions, and these beliefs do not include knowledge of the true concept that governs the classification of the symbols:
States: The state is represented by the player's energy, her current level and position in the level, and the symbols on all chests in the current level.
Actions: The player can open the current chest (if there is one) or move to the next island.
Transition model: The player transitions to a new state based on opening a chest or moving to a new island. In both cases, the symbols on the chests stay the same, with the current symbol removed if the player opens the chest. If a player chooses to move, she knows what state will result: her position will move forward one space and her energy will be depleted by a known constant. If the result is negative energy, then the game transitions to a loss state. However, if a player opens a chest, her beliefs about what state will occur next is dependent on $p(h|\mathbf{d})$, her beliefs about the true concept given the data $\mathbf{d}$ she has observed so far. The player will gain energy if the symbol x on the current chest is in the concept. Taking an expectation over possible concepts h, this probability is $p(x \text{ in concept}) = \sum_h I(h \vdash x)p(h|\mathbf{d})$, where $I(h \vdash x) = 1$ if x is in the concept h and 0 otherwise. The probability of decreased energy is $(1 - p(x \text{ in concept}))$. Based on the Bayesian model above, the player's current beliefs $p(h|\mathbf{d})$ are dependent on the prior probability distribution over concepts. Thus, the transition model assumed by the player is dependent on the parameter θ that we would like to estimate, which is this prior distribution.
Reward model: When the player moves from one island to another, the reward model specifies $R(s,a,s') = 0$, and when the player opens a chest, $R(s,a,s')$ is a fixed positive number of points with probability $p(x \text{ in concept})$ and a fixed negative number of points with probability $(1 - p(x \text{ in concept}))$.

By using the MDP framework and assuming that the player updates her beliefs after seeing information, we ignore the value of information in understanding people's decisions; that is, we assume people make decisions based on their current information and do not consider the effect that information gained now will have on their future decisions. We consider how we might tractably model this factor in Experiment 3.

5.4 Experiment 1: Inferring difficulty

To test our framework, we first used optimal game design to find a version of Corridor Challenge with high expected information gain, and then ran an experiment in which players played either the optimized game or a randomly chosen game with lower expected information gain.

Optimization of Corridor Challenge

We used simulated annealing (Kirkpatrick et al., 1983) to stochastically search over possible designs of Corridor Challenge. The expected information gain of a game was approximated by sampling 35 possible θ vectors uniformly at random (reflecting a uniform prior on θ), simulating the actions of $n = 25$ players in the game, and using the simulated data to infer $p(\theta|\xi, \mathbf{a})$. We approximated $p(\theta|\xi, \mathbf{a})$ using the Metropolis-Hastings MCMC algorithm (Gilks et al., 1996), with a Dirichlet proposal distribution centered at the current state. The noise parameter β controlling the degree of optimality of the Boltzmann policy was set to 1.

To execute the search, we parallelized simulated annealing by using several independent search threads. Every five iterations, the search threads pooled their current states, and each thread selected one of these states to continue searching from, with new starting states chosen probabilistically such that states with high information gain were more likely to be chosen. Each search state is a game, and the next state was found by selecting a parameter of the current game to perturb. If the selected parameter was real-valued, a new value was chosen by sampling from a Gaussian with small variance and mean equal to the current value; if the selected parameter was discrete, a new value was selected uniformly at random.

The stochastic search found games with significantly higher information gain than the initial games, regardless of starting point. This demonstrates that the evaluation and search procedure may be able to eliminate some trial and error in designing games for experiments. Qualitatively, games with high information gain tended to have a low risk of running out of energy, at least within the first few moves, and a diverse set of stimuli on the chests. These games also generally had positive rewards with larger magnitudes than the negative rewards. The game with the highest information gain used a true concept of Type II, although several games with similarly high information gain had true concepts with different structures. While the information gain found for any given game is approximate, since we estimated the expectation over only a sample of possible priors, this was sufficient to separate poor games from relatively good games; we explore this relationship further in Experiment 2.

Methods

After optimizing Corridor Challenge, we conducted a behavioral experiment to assess whether an optimized game resulted in greater information gain than a random game.

Participants

Fifty participants were recruited online and received a small amount of money for their time.

Stimuli

Participants played Corridor Challenge with parameters set based either on an optimized game (expected information gain of 3.4 bits) or on a random game (expected information gain of 0.6 bits). The symbols differed along the dimensions of shape, color, and pattern.

Procedure

Half of the participants were randomly assigned to each game design, and played the game in a web browser. Regardless of condition, the participants were shown text describing the structure of Corridor Challenge, and then played several practice games to familiarize them with interface. The first practice game simply had chests labeled "Good" and "Bad"; the next three games used Boolean concepts of increasing difficulty based on previous work. All practice games used different symbols from one another and from the final game. Practice games used the point and energy values from the game chosen for their condition (i.e., the random game or the game found by the search) in order to make players aware of these values, but the symbols in the practice games were identical across conditions. The fifth and final game was chosen based condition: either the optimized game or the random game. After completing the final game, participants were asked to rate how fun and how difficult the game was, both on 7-point Likert scales. Additionally, they were shown the stimuli and categorization information that they observed during the final game, and asked to classify the remaining stimuli from the game that were not observed.

Results

To assess the information gained from each game, we calculated posterior distributions over the prior probability of each Boolean concept based on the players' actions. These distributions were calculated using a Metropolis-Hastings algorithm on both the prior and the noise parameter β. Samples were generated from five chains with 100,000 samples each; the first 10% of samples from each chain were removed for burn-in. To infer the actual information gained for each game, we infer the maximum likelihood Dirichlet distribution based on these samples from the posterior. We then calculate the entropy of the inferred Dirichlet. The difference between this entropy and the entropy of the (uniform) prior distribution is the actual information gain of the game.

Figure 5.3 shows the inferred distribution over the prior probability of each concept (θ_i) based on participants' actions for the optimized game and the random game; if a concept has higher prior probability, it will be easier to learn. Qualitatively, the distributions inferred from the optimized game appear more tightly concentrated than those from the random game; this is confirmed by the actual information gain, which was 3.30 bits for the optimized game and 1.62 bits for the random game. This implies that we could halve the number of participants by running the optimized game rather than the random game, while achieving the same level of specificity.

For both games, the ordering of the mean prior probabilities of each type, shown by red lines in Figure 5.3, is the same as that found in previous work, except for Type VI. Our inferred distributions for Type VI placed significant probability on a broad range of values, suggesting that we simply did not gain much information about its actual difficulty. We do infer that Type VI is easier than Types III, IV, or V, consistent with some previous findings (Griffiths et al., 2008).

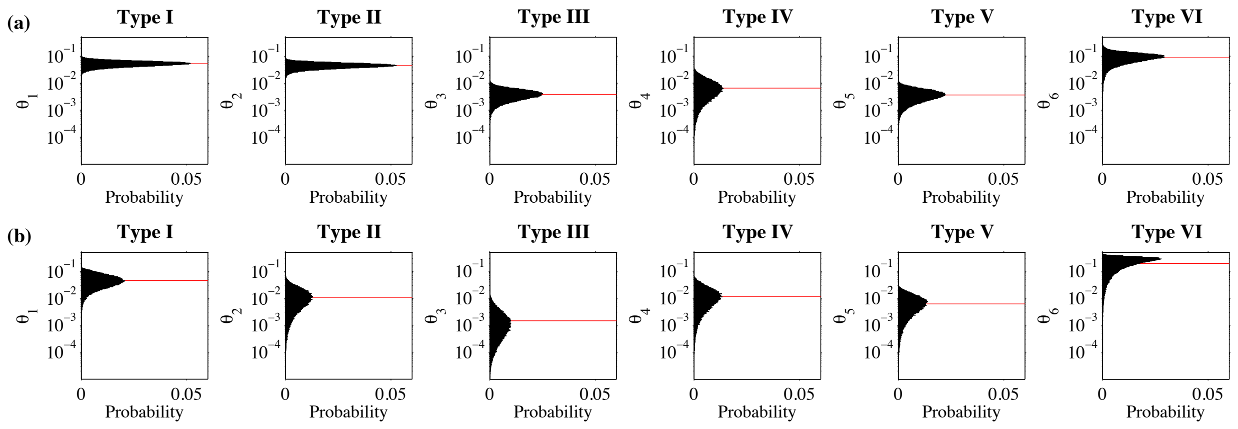


Figure 5.3: Results of Experiment 1, in the form of posterior distributions on concept difficulty from participants' responses in (a) the optimized game and (b) the random game; red lines indicate the mean of each distribution. Each panel shows the distribution over the inferred difficulty of a concept with the given structure (Types I-VI), as reflected by its prior probability in the Bayesian model. Concepts with higher prior probability are easier to learn. Note the logarithmic scale on the prior probability of each θ_i.

5.5 Experiment 2: Estimating information gain

To verify the relationship between actual and expected information gain, we conducted a second experiment in which players played games with a range of information gains. To isolate the impact of the symbols on the chests and the true concept we fixed the point structures to those found for the optimized game in Experiment 1 and conducted new searches over the remaining variables. We then selected games that had varying expected information gains, demonstrating that even without changing the incentive structure a range of information gains was possible.

Methods

Participants

A total of 475 participants were recruited online and received the same payment as in Experiment 1.

Stimuli

Participants played one of nineteen new games. The nineteen new game designs were selected by recording the game design and expected information gain for each iteration of simulated annealing. For this experiment, the points were fixed to be the same as in the optimized game in Experiment 1, so the search only varied the object on each chest and the true category. We then ran nineteen

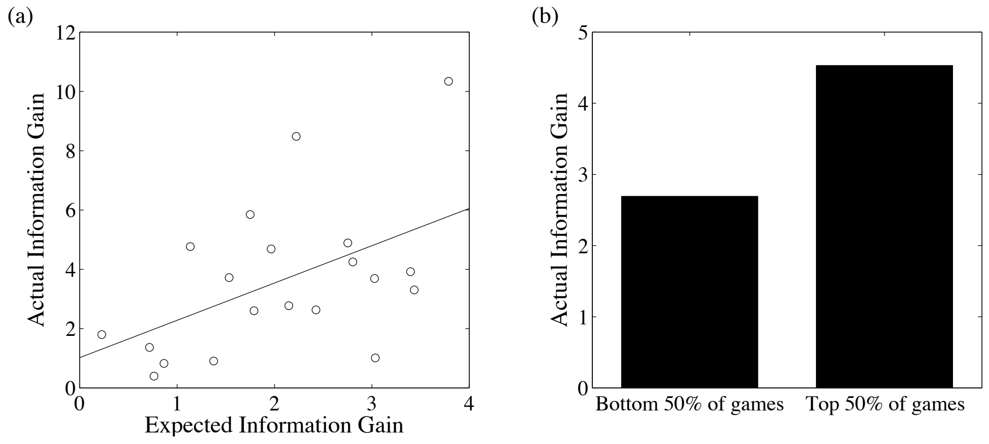


Figure 5.4: Results of Experiment 2. (a) Expected versus actual information gains ($r(18) = 0.51$, $p < 0.025$). Each circle represents a game, and the least-squares regression line is shown. (b) Actual information gain for the ten games with lowest expected information gain versus highest expected information gain.

independent search processes, each with a different initial game. From the resulting games, we hand-selected one game from each search thread such that the new collection of games had roughly evenly spaced information gains.

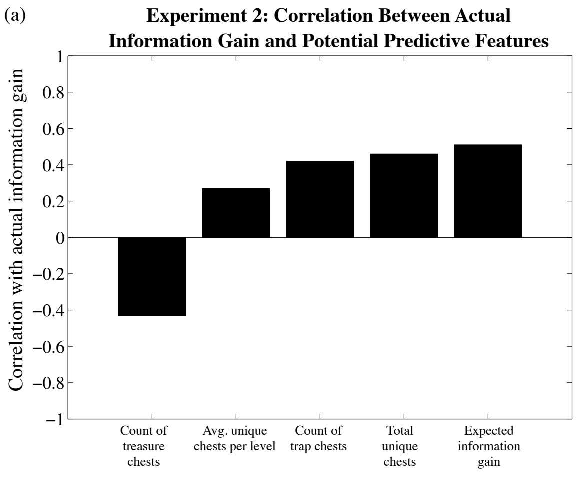
Procedure

Procedure matched Experiment 1.

Results

We compared the actual and expected information gains for the nineteen new games and the optimized game from Experiment 1, all of which used the same point structure. As shown in Figure 5.4a, expected and actual information gain were positively correlated ($r(18) = 0.51$, $p < 0.025$). This demonstrates that the design of the game does influence how much information we can infer from human players' actions, and that this gain is predicted by our estimates. While this correlation might seem modest, it has significant consequences for efficiency: on average, the ten games with highest expected utility resulted in a gain of 67% more bits of information than the ten games with lowest expected utility (Figure 5.4b).

One potential objection to the optimal game design framework, and computational modeling in general, is that the method is somewhat complex and considerable power is necessary to predict expected utility. We thus explored whether heuristics based on features of the game might be sufficient to predict information gain. As shown in Figure 5.5a, the expected utility of the games showed the highest correlation with actual information gain, although the total number of unique



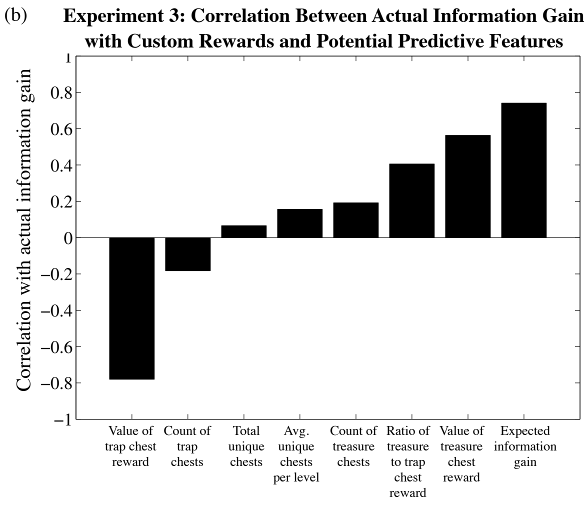


Figure 5.5: Correlations between proposed heuristics for predicting information gain and the actual information gain. (a) Correlations for Experiment 2. Expected utility (as calculated using optimal game design) has the highest correlation, and number of unique chests is the only heuristic with a significant correlation to actual information gain. (b) Correlations for Experiment 3. The correlations to information gain for the value of the trap chest and for expected utility are of similar magnitude, but only expected utility is consistently predictive across the two experiments.

symbols was also positively correlated with information gain ($r(18) = 0.46$, $p < 0.05$). While optimal game design and this heuristic have relatively similar correlations with information gain, we believe there is still an advantage in using the optimal game design framework, as this approach does not require generating appropriate heuristics for different games and it may not be intuitively obvious which heuristics will be good predictors of information gain. For example, the total number of treasure chests was negatively correlated with information gain, although this correlation was not significant. Additionally, as the number of features to optimize increases, the number of possible heuristics will also increase, making it difficult to choose a heuristic to rely on via intuition; we return to this issue in Experiment 3.

5.6 Experiment 3: Sensitivity to rewards

In Experiment 2, we showed that expected and actual information gain were correlated for a range of game designs. All of these game designs had the same incentive structure; the only differences were the categories being learned and the placement of items within the games. It is also possible to vary the rewards in the game designs, as was done in Experiment 1. This raises the question of how much people will internalize the reward structure and behave rationally with respect to it. People may incorporate their own goals into the game, such as wanting to learn about the true concept rather than maximize points, and thus exhibit unexpected behavioral changes based on

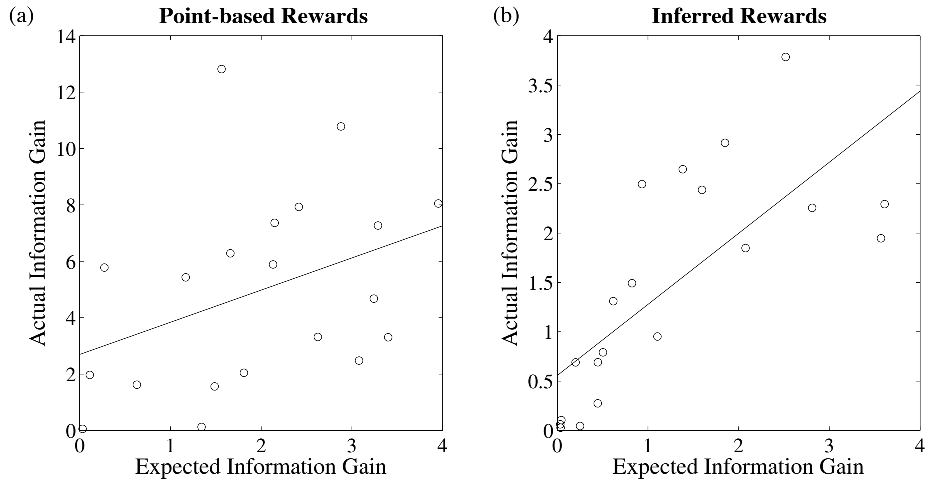


Figure 5.6: Results of Experiment 3. (a) Expected versus actual information gains with point-based reward ($r(18) = 0.38$, $p = .1$). Each circle represents a game, and the least-squares regression line is shown. (b) Expected versus actual information gains with inferred custom rewards ($r(18) = 0.74$, $p < .001$).

different reward structures. To investigate this possibility, we generated eighteen additional games with a range of expected information gains, allowing the incentive structure as well as the other parameters of the game design to change.

Methods

Participants

A total of 450 participants were recruited online and received the same payment as in Experiments 1 and 2.

Stimuli

Participants played one of eighteen new games. The search method for this experiment was the same as for Experiment 2 except that the point values for opening a treasure or trap chest and the energy lost for movement were allowed to vary. All games came from search threads with independent starting points, and games were hand-selected to span a range of expected information gains.

Procedure

Procedure matched Experiment 1.

Results

We analyzed the data from these new games combined with the data from the two games in Experiment 1. We first calculated the actual information gain about the prior distribution over concept types, assuming the participants' reward functions reflected the point structure. As shown in Figure 5.6a, the correlation between expected and actual information gain was not significant ($r(18) = 0.38$, $p = .1$). Inspection of participants' actions showed some choices that seemed unlikely to be rational with respect to the model. For instance, a participant might choose to open a chest even when she had low energy and little information about the concept, despite the fact that she could reach the end of the level without this action and earn the large level completion bonus. From the perspective of the model, this action is only predicted if the participant places very high probability on this being a treasure chest.

To test whether participants might be acting based on a different reward function than that given by the incentive structure, we modified the inference procedure to infer a reward function for each game based on the participants' actions. Previously, the inference procedure inferred a posterior distribution over the hypothesis space of six-dimensional multinomials; now, we changed the hypothesis space to be possible reward functions. These functions were specified by the value of opening a treasure chest, the value of opening a trap chest, and the value of completing the level. We constrained these values such that the value of opening a treasure chest and of completing the level were non-negative and the value of opening a trap chest was non-positive. We fixed the prior distribution to be equal to the mean of the posterior distribution from the optimized game in Experiment 1.[3] MCMC sampling then proceeded as in Experiment 1, resulting in a posterior distribution over the values for the parameters of the reward function.

The results showed that participants do seem to be acting based on a different reward function than that given by the point structure. While the reward function varies across games, as expected given that the point structure is likely to have some influence on behavior, it consistently places relatively low value on completing the level. This could reflect the fact that completing a level is not inherently rewarding to participants. Participant comments are consistent with people being more motivated by understanding the game than achieving maximal points. For instance, one of the most frequent comments by those who did not enjoy the game was that they were "confused" by the rule or that they did not understand the pattern. Thus, opening chests might be expected to have higher intrinsic reward than completing the level, despite the point structure.

One of the goals of inferring the participants' reward functions was to determine whether using the inferred functions would lead to a correlation between expected and actual information gain. If the confounding factor in the original analysis was the incorrect reward functions, then using these functions to re-calculate both the expected and actual information gains should lead to similar results as in Experiment 2. Thus, we fixed the reward function for each game to match the mean of

[3]In principle, one could jointly infer both an arbitrary reward function and the prior distribution, but in practice, this leads to identifiability issues wherein very different parameter configurations all have similar posterior probability. Since our interest here is whether there exists a custom reward configuration that would explain the participants' actions and we have a good estimate of the prior distribution from the previous game, fixing the prior distribution gives the best estimate of the reward functions.

the posterior distribution over reward functions for that game, and then used the original inference algorithm to infer a posterior distribution over the difficulty of learning the six different concept types. As shown in Figure 5.6b, expected and actual information gain are in fact correlated when the inferred reward functions are used ($r(18) = 0.74$, $p < .001$). This demonstrates the importance of knowing participants' goals when interpreting their actions. A participant's actions are only meaningful within the context of her goals and her understanding of how her actions affect her progress towards those goals. While participant actions can be used to make inferences about these factors, this may lead to incorrect conclusions if our assumptions about the relevant factors are wrong.

Returning to the issue raised in Experiment 2 of whether heuristics are as effective as expected utility, we calculated seven heuristics based on the characteristics of the games. Four were the same heuristics as in Experiment 2, while three were based on the reward functions, which were the same across all games in Experiment 2. We used the inferred custom rewards for these heuristics since the original rewards were inconsistent with participant behavior. As shown in Figure 5.5b, some of these heuristics are quite good at predicting expected utility. The value of a trap chest even has a slightly higher magnitude correlation with information gain than expected utility ($r(18) = -0.78$, $p < .001$). Heuristics thus can be effective at predicting information gain. However, their effectiveness seems to be less consistent than expected utility: the best heuristic for Experiment 2, the total number of unique symbols, has only a small correlation with information gain for the game in Experiment 3 ($r(18) = .066$, $p > .7$), and the best heuristic for Experiment 3 is based on the reward function, which would have no correlation at all with information gain for the games in Experiment 2. This suggests that the computational cost of optimal game design is balanced by its greater consistency.

5.7 Incorporating information gain into the reward function

The results of Experiment 3 showed that the reward functions provided in the games did not always match people's subjective reward functions. One possible reason for this was people's desire to learn more about the concept, making opening chests more valuable and finishing the level less valuable. However, these changes to the reward function do not directly address the fact that people may be sensitive to whether opening a particular chest is likely to give them more or less information about the concept. We can explore this issue by considering reward functions that directly incorporate information gain: how much is the person likely to learn by opening a given chest? We now infer new reward functions that include such a term for the games in Experiment 3, and compare whether this results in a better model fit than the original analysis.

Methods

To include information gain in the reward function, we set $R_{IG}(s,a,s') = R(s,a,s') + w \cdot \Delta H$, where $R(s,a,s')$ is the reward without including information gain and ΔH is the change in entropy in the player's estimated posterior distribution over concepts based on observing the results of taking

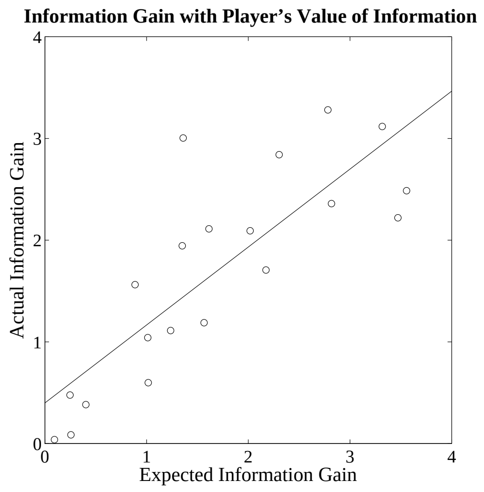


Figure 5.7: Results of Experiment 3 when an inferred reward function that includes information gain is used to estimate the prior distribution over Boolean concepts ($r(18) = 0.82$, $p < .001$). Each circle represents a game, and the least-squares regression line is shown.

action a in state s. The parameter w controls the weighting of information gain within the reward function. We then used the players' actions to infer a posterior distribution over the parameter w as well as the parameters in the original reward function, fixing the transition function to the mean of the inferred transition function from the optimized game in Experiment 1. The only difference between this procedure and that in Experiment 3 is that w can have non-zero weight.

Results

Using this procedure, we found the weight of information gain in each of the reward functions for the twenty games in Experiment 3. Information gain was always inferred to have a positive weight, except in one game where inspection of the samples showed that this parameter was covarying with the point value of opening a treasure chest. This indicates that people's actions are consistent with wanting to learn more about the concept.

To explore how the new reward functions affected the relationship between expected and actual information gain, we then fixed the reward functions and inferred the prior distribution over Boolean concepts, again as in Experiment 3. As shown in Figure 5.7, expected and actual information gain were correlated ($r(18) = 0.82$, $p < .001$). This correlation value is similar to that found in Experiment 3 ($r(18) = 0.74$). Using the Deviance Information Criterion (DIC; Spiegelhalter, Best, Carlin, & Van Der Linde, 2002), we compared the fit of the model with information gain and the model without information gain for each game. DIC is related to other information criterion measures and controls for differences between models in the effective number of parameters. This measure is easily computed from MCMC samples; lower DIC reflects a better model fit. The average DIC over the 20 games was 203 for the models with information gain (median: 204), compared to 212 for the models without information gain (median: 220). This difference suggests that the model which includes information gain is a somewhat better fit to the data than the model without this parameter. Thus, participants' behavior likely reflects a combination of incentives, some extrinsic and provided by the game and some intrinsic and reflecting a desire to learn and understand.

5.8 General Discussion

Refining a game or interactive activity to be diagnostic of psychologically or educationally relevant parameters can be a time-consuming process filled with trial and error. While the exact incentive structure or placement of objects in a game may have little impact on how enjoyable or engaging the game is, these factors can significantly impact how useful the activity is for diagnosing players' knowledge. I have presented a general framework for deciding how to set these factors by predicting which game designs will have the highest expected information gain. This framework adapts ideas from optimal experiment design and relies on Bayesian inverse planning to link players' actions to cognitively relevant model parameters. I now consider several possible challenges as well as future directions for this framework.

Our framework relies upon the idea that people behave in noisily optimal ways based on their current knowledge and goals. The previous two chapters provide evidence in support of this assumption in several domains. However, there may be inconsistencies between people's actual knowledge and goals and the model's assumptions about these factors. We saw evidence for this in Experiment 3, which demonstrated that invalid inferences can be drawn when these inconsistencies are present. Thus, care must be taken to monitor whether the MDP model is a good fit to people's behavior, perhaps using a similar strategy as in the previous chapter of examining the noisiness of participants' planning. It may also be necessary to modify the task, instructions, or model to more closely align the model's assumptions and the participant's beliefs. For instance, to more closely align model and participant reward functions in Experiment 3, one might give points for opening a chest with a previously unopened symbol, making the game's incentive structure closer to that which participants bring to the task. Alternatively, one might give monetary rewards based on the points the participant earned, making it more likely that participants will respond based on the game's reward structure. Results in behavioral economics suggest that aligning monetary incentives with participants' performance results in choices that are more consistent with behavior outside of an experimental setting, as choices in the experiment have real consequences (e.g., Oxoby, 2006); for games, this is likely to result in behavior that is more consistent with the game's incentive structure, since this structure will have an effect on participants' monetary rewards. While monetary incentives are likely to be appropriate in the context of behavioral experiments, an interesting area for future exploration is how to align incentives within educational contexts. This is particularly pertinent for the low-stakes, formative assessment contexts in which educational games might be most likely to appear. Within these contexts, it may be more useful to focus on the option of changing the game's incentives rather than students' motivations, as characteristics like a desire to learn should be fostered rather than discouraged. Care will still be needed, though, to determine what motivations are important to individual students.

I noted previously that for computational reasons, we ignore the value of information in understanding players' actions. This allows us to model the game as a static MDP given the player's current knowledge, rather than including a latent variable denoting the true category. However, our results suggest that people may consider the value of information when choosing actions. We showed that the player's expected information gain at a single step can be incorporated into the reward function in Section 5.7, resulting in a better fit to the observed data. The information gain at a single step is an approximation of the true value of information, but has the advantage of not significantly decreasing computational tractability. Modeling people's reasoning about the value of information may be especially important for games involving inquiry skills, where part of the challenge is in determining what information to gather to solve a problem. In some cases, the approximation of including the information gain at only a single step may be too limited, especially if players must use several actions to uncover information in the game. One way to address this limitation is by modeling the game as a partially observable MDP, where the state has certain unobserved variables. This significantly increases computational costs, but is in principle possible.

Over the previous three chapters, we have developed tools for designing games and other interactive activities and interpreting behavior in these environments. Interactive activities are increasingly popular tools in both behavioral research and education. They can address some of the issues in traditional experiments, such as flagging motivation or difficulty introducing participants to a complex task, and in education, they can engage learners and provide authentic opportunities for these learners to apply their understanding. The Bayesian inverse planning framework provides a way to interpret players' behaviors, and in this chapter, we have provided additional evidence that its inferences are consistent with other analyses. The optimal game design framework we have developed here provides a way for designers to make principled decisions and offers benefits for researchers and educators. More diagnostic games allow fewer participants to be used to gain the same information about a research question, providing the potential for drawing conclusions more quickly or for asking more complex questions that would otherwise require a prohibitive number of participants. This framework also offers opportunities for the types of educational games and activities we have considered in the previous chapters. By diagnosing knowledge more accurately or over a shorter period of time, instruction can be better targeted to individual learners and more time can be spent on learning rather than assessing. For example, adapting the algebra problems provided to users of our website could result in more concentrated posterior distributions, helping to alleviate low confidence due to too few problems solved. Optimal game design also provides an example of how our approach of combining computational modeling with machine learning leads to flexible approaches to educational problems: by using a probabilistic, generative model for interpreting actions, we were able to use the same model to interpret and simulate behavior. There are a number of areas for future exploration in this area, including developing methods to induce a possible space of misunderstandings from past data and creating models to infer both motivation and understanding during gameplay. While this chapter concludes my discussion of diagnosing people's understanding by observing their free-form actions and choices, the work I have presented provides a foundation for addressing these future questions.

Chapter 6

Faster teaching by POMDP planning

In the previous chapters, I have focused primarily on questions of assessment: how can we determine what people understand and misunderstand on their actions? However, using this information to customize the course of instruction is not necessarily trivial. Imagine an interactive technology like the illustrated primer that is attempting to teach a student and achieve particular learning objectives, with a diverse set of possible instructional materials and activities available. The technology must decide when to assess the learner and when to provide instruction that may change the student's understanding; while some activities may include both of these facets, many are likely to focus more on one or the other. There may be a variety of instructional activities that the technology can choose from, and these activities may be more effective in a particular sequence; for instance, some activities may build on information taught in a previous activity, and thus will only be useful if the student has mastered the prior material. The diagnosis of the student's understanding should clearly play a part in these decisions, but qualities of the materials and how they affect learning are also important. In this chapter, we develop and test a formal account of how a computer-based tutor should select pedagogical activities.[1]

To formalize the problem of how an automated tutor should select activities for a student, we consider what information the tutor should consider in making its selections. The tutor should take into account both immediate and potential long-term learning benefits of an action. For example, some activities may lead to learning that is more likely to transfer to other tasks, but that has less of an immediate performance benefit for the current task. The automated tutor should also be able to reason about the student's unknown knowledge, and to gain information about that knowledge based on observed student behavior; while human teachers may only be guided by coarse information about their students' knowledge (see VanLehn, 2011, for a literature review), finer-grained information could prove helpful to an automated tutor. Models like Bayesian inverse planning could be used to make inferences about a student's understanding based on her actions, although for simpler activities like multiple questions, less complex models will suffice. Finally, the auto-

[1]This chapter is based on work conducted in collaboration with Emma Brunskill, Thomas L. Griffiths, and Patrick Shafto. Parts of this work were included in Rafferty, Brunskill, Griffiths, and Shafto (2011). My thanks go to undergraduate research assistants Benjamin Shapiro, HyeYoung Shin, Christina Vu, and Julia Ying for their help in conducting the behavioral experiments described in this chapter.

mated teacher should incorporate a model of learning. This model should specify how a student's knowledge is likely to change based on her current knowledge and the pedagogical activity she is completing. These components of the tutor should be modular, such that we can consider different assumptions about learning without changing the entire architecture of the tutor and that we can easily adapt the tutor to different domains. This modularity echoes the goals described in Chapter 1: developing educational technologies that are systematic and scalable through the use of computational modeling. In this case, the modularity enables us to develop systems that can adapt to particular characteristics of a domain without relying on heuristics or requiring experts to define pedagogical strategies for individual domains.

Parts of the components for making such an adaptive tutor have been approached in previous work. For instance, there has been substantial interest in the cognitive science, education, and intelligent tutoring systems communities in modeling and tracking student learning. A number of results have demonstrated the benefit of taking a Bayesian probabilistic approach (see, e.g., Chang et al., 2006; Corbett & Anderson, 1995; Conati & Muldner, 2007; Villano, 1992). There has also been some previous work, such as the KLI framework (Koedinger, Corbett, & Perfetti, 2012), that has considered how to make pedagogical choices based on the types of student skills that are being targeted. This framework synthesizes a variety of work finding that the effects of different teaching strategies on learning are heavily modulated by the domain and task. However, there has in general been limited work on how to automatically compute teaching policy that leverages a probabilistic learner model in order to achieve a long-term teaching objective.

In this chapter, I propose using partially-observable Markov decision processes (POMDPs) for automatic tutoring, focused on cases where the automated tutor teaches a student individually. POMDPs allow one to compute a contingent policy for selecting sequential actions in situations where important information may be unobserved (Sondik, 1971). By using this model, we take a decision-theoretic approach that allows us to incorporate the sequential nature of the task, customizing choices based on the learner's observed behaviors as well as the previous pedagogical actions that have occurred; this model naturally allows consideration of both immediate and long-term gains. The specification of the POMDP also makes it relatively easy to consider different models of student learning and different domain models, meeting our goal of modularity. Here, we assume that the model of learning is known, and demonstrate how to select teaching actions given a student model. However, For automatic tutoring to be widely applicable in new domains, such models must eventually be learned from data; the POMDP framework is one way of conceptualizing what quantities must be learned and how these quantities relate to one another. Within the POMDP model, the automated teacher's beliefs about the learner's unknown knowledge is represented as a distribution, preserving the teacher's uncertainty about the student's true knowledge. Given a learning objective and a set of models describing the learning process, POMDPs provide a framework for computing a teaching policy that optimizes the objective. The objective function to be optimized can encompass multiple goals, such as attaining specific skills quickly and maintaining motivation, but these functions can be challenging to optimize; we address simpler learning objectives focused on the student's knowledge state and do not consider motivation or other affective issues.

Though POMDPs are related to other decision-theoretic approaches used in previous education research, they are more powerful in two key respects. First, POMDPs can use sophisticated models of learning, rather than assuming learners' understanding can be directly observed or approximated by a large number of features (as in Barnes & Stamper, 2008; Chi et al., 2008), and these models are likely to be more interpretable than feature-based approximations. As opposed to Chi et al. (2008), we focus our investigation on how POMDPs can be used to reason about the consequences for teaching of particular cognitive models; in their work, they focused on empirical investigation of using reinforcement learning to optimize a policy for a model based on observed features. Both approaches make valuable contributions, but differ somewhat in their aims. The POMDP approach is likely to be helpful when considering many existing cognitive models from psychology, as these typically include information about the learner's mental state. Second, in contrast to approaches that only maximize the immediate benefit of the next action (Conati & Muldner, 2007; Kujala, Richardson, & Lyytinen, 2008; Murray, VanLehn, & Mostow, 2004; Tang, Young, Myung, Pitt, & Opfer, 2010), POMDPs reason about both the immediate learning gain and the long-term benefit to the learner after a particular activity. While there are cases where considering only the immediate learning gain is optimal (e.g., Karush & Dear, 1967), this strategy is suboptimal in the general case. For instance, reasoning about long-term effects on learning provides a way to naturally incorporate diagnostic actions. Diagnostic actions, such as giving a quiz, may result in less immediate learning, but allow more effective pedagogical activities to be selected later. Incorporation of information about the effect of particular actions on learning is automatic in the POMDP framework, allowing one to avoid manually specifying heuristics about which teaching actions will be most effective.

POMDPs offer an appealing framework for selecting teaching actions, but there are often significant obstacles to practical implementation. Specifically, planning teaching requires modeling learning, and richer, more realistic models of learning lead to computational challenges for planning. We instead compute approximate POMDP policies, which make it feasible to use these more complex, realistic models of human learning. As a demonstration of the modular nature of POMDPs, we examine three different models of concept learning, and illustrate how, given the same learning objective, these lead to qualitatively different teaching policies. The use of several learner models allows us to examine whether effective policies can be computed even when the assumed model and true human learning differ. Additionally, these models vary in complexity, providing a test of how well the POMDP framework can scale to more complicated learner models. We explore the impact of these varying models in two simple concept-learning tasks, both through simulations and by teaching human learners. In order to focus primarily on the problem of pedagogical action selection, these tasks involve simpler assessments than those in previous chapters. While there exist a few recent papers exploring the use of POMDPs to compute teaching policies (Brunskill, Garg, Tseng, Pal, & Findlater, 2010; Brunskill & Russell, 2010; Folsom-Kovarik, Sukthankar, Schatz, & Nicholson, 2010; Theocharous, Beckwith, Butko, & Philipose, 2009), to our knowledge our work is the first to demonstrate with human learners that POMDP planning results in more efficient learning than baseline performance and the first to explore the impact of different models of learning on the computed policies.

In this chapter, I begin by providing an overview of POMDPs, demonstrating how they differ from the MDPs used in previous chapters. I then show how teaching can be formulated as a POMDP. I next explain specifically how to express concept-learning problems as a POMDP and describe three possible models of concept learning. Given the now fully specified model, I provide an algorithm for computing a policy for choosing pedagogical actions from the POMDP. To empirically evaluate the effectiveness of POMDP policies for increasing learning efficiency, we conduct several concept-learning experiments. In the first concept-learning task we consider, alphabet arithmetic, we use simulations and a behavioral experiment to evaluate this question. After finding that the policies are successful for this simple task, we use a second behavioral experiment to investigate this technique's success for teaching a more complex concept-learning task involving learning numerical concepts. I conclude by discussing the implications, limitations, and future directions of this work.

6.1 Partially observable Markov decision processes

POMDP planning is used to compute an optimal conditional policy for selecting actions to achieve a goal (Kaelbling, Littman, & Cassandra, 1998; Monahan, 1982). POMDPs differ from MDPs based on the agent's information about the state of the world: in a POMDP, the state at each time step is unobserved. For example, imagine a robot that needs to find a charging station and knows that it is somewhere in a maze but does not know where. By exploring the environment and making observations of the walls and intersections in the maze, the robot can better localize its location to find the charging station more quickly. However, the robot should only explore to the extent that this will help it achieve its goal: if it knows that it is in one of two possible locations in the maze and that in both locations a charging station can be reached by turning left and moving ten meters, it should simply proceed in that direction without further diagnosing its location. POMDP planning provides a way to choose actions that takes into account how uncovering unknown information (e.g., further diagnosing the robot's location) is likely to impact the agent's ability to achieve its goals. This planning model has been used for a wide variety of control tasks, including robotics (e.g., Kurniawati, Hsu, & Lee, 2008; Pineau, Montemerlo, Pollack, Roy, & Thrun, 2003), healthcare (e.g., Hoey, Poupart, Boutilier, & Mihailidis, 2005; Hu, Lovejoy, & Shafer, 1996), and dialogue systems (e.g., Atrash & Pineau, 2006; Roy, Pineau, & Thrun, 2000; Young et al., 2010). POMDP control policies indicate which actions to take, conditioned on the actions taken so far and observations of the environment, such that the expected cost is minimized (or the expected reward is maximized). These policies are thus updated as an agent gains more information about the environment, allowing it to choose more effective actions with less uncertainty about the current state.

Formally, a POMDP consists of a tuple $\langle S, A, Z, p(s'|s,a), p(z|s,a), R(s,a), \gamma \rangle$, where S is a set of states s, A is a set of actions a, and Z is a set of observations z (Sondik, 1971). As shown in Figure 6.1, the structure of the POMDP is similar to an MDP. An action a is taken at each time step, which together with the current state s results in a transition to the next state s'. These transitions are specified by the transition model $p(s'|s,a)$. The state s at any point is unobserved. Instead, information about the state is indirectly available via the observations. Given that action a is taken in state s, the observation model $p(z|s,a)$ indicates the probability of observing z. For example, in the case of the robot moving from an unknown starting location, the state s is its true location in

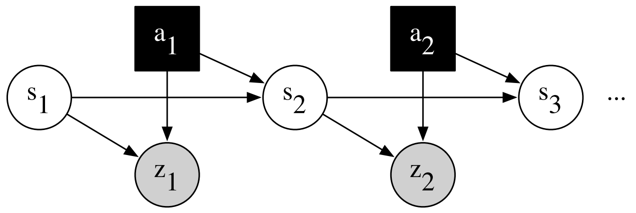


Figure 6.1: POMDP model. At each time step t, there is an unobserved state s_t and the planner chooses an action a_t. Based on the current state and chosen action, z_t is generated and observed by the planner, and the state transitions to s_{t+1}.

the maze, and it chooses whether to move or to turn in a particular direction. The next state s' is the robot's location after that action, which may still be unknown to the robot. However, the robot does know something about its location based on the walls and intersections it sees; this information is encoded by the observation model $p(z|s,a)$. Note that if all observation distributions $p(\cdot|s,a)$ place a probability of one on some z and a probability of zero on all other possible z, then the POMDP is equivalent to an MDP: the state is effectively observed at each time point.

Taking one action versus another in a particular state may be more or less costly. Just as in an MDP, agents may experience costs, rewards, or both during their interactions with an environment; because we will be mainly referring to costs in the experiments that we describe, we describe the POMDP planning framework in terms of costs, letting a reward be simply a negative cost. The costs experienced by an agent may vary based on its objectives as well as characteristics of the environment. For instance, in the case of the robot trying to find the charging station, it might wish to accomplish its task in minimal time; the cost structure would then specify that actions that are likely to take more time (e.g., moving longer distances) will incur higher costs. Alternatively, the robot might simply wish to find a charging station before it completely runs out of energy. In that cases, all actions might have cost zero, but entering a state where the robot has no energy would incur a very large cost. The cost model $R(s,a)$ is used to encode the cost structure; for every state s and action a, this model specifies a real-valued cost. POMDP planning seeks to choose actions that minimize the expected sum of discounted future costs. If the state were known at each time step, this quantity could be calculated in the same way as in an MDP: $\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)$, where γ is the discount factor. This factor represents the relative harm of immediate costs versus delayed costs. The discount factor is set by the planner.[2]

Given a POMDP, we want to compute a policy for how to select an action at each time step. An optimal policy should map the prior history of actions and received observations to the action that will minimize the expected sum of discounted future costs. For an MDP, the relevant information about the the prior history of actions and received observations is encompassed by the observed state. However, in a POMDP, the state is unobserved and the prior history grows at each step, making it difficult to directly compute a policy. A common alternative is to maintain a sufficient

[2]See Appendix A.1 for a list of free parameters related to our use of POMDP planning and information about how these parameters were set.

statistic, known as a *belief state* b, that represents the planner's distribution over potential states given the past actions and observations (Åström, 1965). In the case of the robot, this would be a distribution over where it currently is in the maze; some locations might have been ruled out by the observations, giving them probability zero, while other locations might be equally probable given the observations and actions that have occurred so far. Bayesian updating can be used to compute a new belief b^{az} after taking action a and receiving observation z from belief b:

$$b^{az}(s') = \frac{p(z|s',a)\int_s p(s'|s,a)b(s)ds}{\int_{s'} p(z|s',a)\int_s p(s'|s,a)b(s)dsds'}. \tag{6.1}$$

Intuitively, this update corresponds to taking the expectation over the next state given the distribution over states at the current time step, adjusted by the probability of receiving the actual observation in that next state. Thus, at each time step, the planner chooses an action to minimize the future cost, where information about past actions and observations is encoded by the current belief state. This process is equivalent to planning based on maintaining the entire history of actions and observations.

6.2 Modeling teaching as a POMDP

We now seek to formalize the problem of selecting individual teaching actions within the POMDP framework. Using POMDPs for education was first mentioned by Cassandra (1998), as part of a proposal for many different applications for POMDPs. This proposal was followed by several different areas of research on applying POMDPs to education. POMDP planning has been considered as a way of sequencing units of instruction (Brunskill et al., 2010; Brunskill & Russell, 2010). Additionally, simulation-based work has considered how to approximate the student state in order to use POMDP planning for domains where a "soft" prerequisite model is known (Folsom-Kovarik et al., 2010); in that work, the authors constrained the order in which a tutor tried to teach particular concepts. Finally, Theocharous et al. (2009) considered the problem of constructing the component models of a POMDP to teach a specific concept. In our work, we consider the problem of selecting individual pedagogical actions using a POMDP policy, where we do not have explicit information about which actions should precede other actions and where the student's knowledge state does not necessarily decompose into independent components. While many common models of student learning assume this decomposition (e.g., ACT-R, Anderson, 1993), other psychological models of learning do not, and the POMDP framework can be applied to either type of model. Our work provides a concrete demonstration of using POMDPs to teach human learners and considers the effects of mismatches between human learners and the assumptions of the model of learning. In this section, we demonstrate how teaching can be modeled within the POMDP framework by mapping each part of the model to a particular part of the teaching process, providing a roadmap for the applications that we consider as well as future applications of POMDPs to teaching.

Figure 6.2a shows our general formulation of the teaching process. The automated teacher must make a sequence of pedagogical choices. These pedagogical choices map to the actions taken at each time step in the POMDP (see Figure 6.2b). For example, the automated teacher might first have a student complete a short quiz (a pre-test), and then have the student complete a chemistry lab that has concepts that it believes the student has not yet mastered. Note that unlike in previous chapters, the actions correspond not to the *learner's* choices but to those of the automated tutor. The learner's state at any given time step t is unobserved and corresponds to the state s_t of the POMDP at time t. We consider the state to be a knowledge state, corresponding to what a learner currently understands about what she is being taught; the state here is analogous to the diagnosis of the inverse planning model. However, the state could in general be richer and include other information about the learner, such as her current level of motivation or her current affect. The learner's state may change based on the activity that the teacher chooses to give her (i.e., a_t); as shown in the graphical model in Figure 6.2b, the next knowledge state is dependent only on the current knowledge state and the pedagogical activity. Intuitively, changes in the learner's state correspond to learning. Such changes may reflect mastery of a new skill, forgetting of a previously learned skill, or some other change in understanding, such as making a new generalization that brings the student closer to correct understanding.

Modeling learner knowledge and changes in knowledge within the POMDP framework requires specifying a state space S of possible knowledge states and transition model $p(s'|a,s)$ for how knowledge changes. Different learner models may make different assumptions about how knowledge is encoded. At the simplest possible level, the state might only represent whether a particular topic, such as addition, has been mastered. Alternatively, states might represent different possible understandings a student might have about addition, one of which is normative and others which represent incomplete or non-normative understandings. The state space is thus similar to the space of possible hypotheses considered by the Bayesian inverse planning model developed in previous chapters. While a particular representation must be specified to compute a POMDP policy, one of the advantages of the POMDP framework is that it can work with a variety of possible representations, allowing one to determine what effects different assumptions would have on the optimal policy and how quickly one would expect a particular type of learner to master a topic. We consider several learner models in the experiments that follow.

The likelihood that a learner will give a particular response z to an item given her current knowledge state s is also a part of the learner model. This corresponds to the observation model $p(z|s,a)$, and intuitively, provides noisy information about the learner's understanding. For example, imagine the student has been asked for the answer to $3+8$ and responds 10. This response is less likely than 11 if the learner has a correct understanding, but could have occurred due to misreading or a "slip": it is not definitive proof that the student has not mastered addition. A more complex observation model is the likelihood in Bayesian inverse planning, where the observation corresponds to the person's sequence of action choices.

We assume that the automated teacher has a learner model that represents how the learner's knowledge changes and how the learner's responses are affected by her knowledge. The teacher can then use this model to update its beliefs about the student's current knowledge state based on new observations. This update can be done just as in Equation 6.1. The teacher begins with a belief state equal to the prior distribution $p(s)$ over possible knowledge states; this distribution might be used to encode known biases in student knowledge for a particular task, just as the prior in Bayesian inverse planning was used. After each action, the belief state is updated using the observation model to incorporate the learner's responses and the transition model to incorporate the effects of learning.

The final portion of the POMDP framework that must be adapted to the teaching domain is the cost model. The content of this model is dependent on the teacher's learning objectives. Since POMDP planning is used to find a policy that minimizes expected costs (or maximizes rewards), the cost model should specify the teacher's desired outcome as well as particular incentives for individual actions or states. For example, one simple learning objective would be to have the learner reach some particular knowledge state s, reflecting mastery, in as little time as possible. This could be encoded by having actions in state s cost zero, and other actions cost the amount of time that it is expected the student will take to complete them. Then, for instance, if there are only two actions and the student will almost certainly enter the desired state s after completing either action, the planning framework will favor the action with shorter expected time.

While we will use this simple time cost model, there are many other possible learning objectives that could be encoded in a cost model. For instance, one might have a domain where variable amount of material could be learned, and the objective is for the student to learn as much of the material as possible. This learning objective might more naturally be represented using rewards (rather than costs, which are negative), with larger rewards for knowledge states where more of the material has been learned. The resulting policies would still attempt to teach the learner all of the material, but if only a fixed amount of time was available, these policies might attempt to ensure that at least some material was learned rather than trying to teach all material with little probability of success.

Another class of possible learning objectives focuses on robust learning. For example, one might want to maximize the probability that a student will retain her knowledge for at least some period of time or that she will be able to transfer her knowledge to new problems. Such objectives imply that one has particular beliefs about the learner model, and in particular, the knowledge state representation and the observation model. If it is possible for a student to have learned more or less robustly, then there must be sets of states in which the learner will show similar responses to certain types of items, such as items of the same type she has already been exposed to. However, when asked about a transfer item, students whose knowledge state reflects deeper learning will give different responses than students who knowledge state reflects more shallow learning. This could be reflected in the cost model by having the desired knowledge state be that which reflects deep learning, with some cost for being in other knowledge states. The cost of being in the shallow knowledge state might be less than that of an arbitrary knowledge state but still non-zero; this corresponds to some mastery being seen as more valuable than no mastery, but still less desirable than robust mastery. In this case, it is clear that for the teacher to achieve the desired objectives, there must be some actions that are likely to help the student learn the knowledge in the robust manner and ideally, there will be actions available to allow the teacher to check the student's generalization abilities. In this example, one could also imagine combining the robust learning objective with costs for increased time to mastery, leading to policies that aim to quickly achieve robust learning. In general, the cost model provides a way to specify a diverse array of objectives, and POMDP planning can then be used to find a policy that will minimize the cost of achieving the objectives.

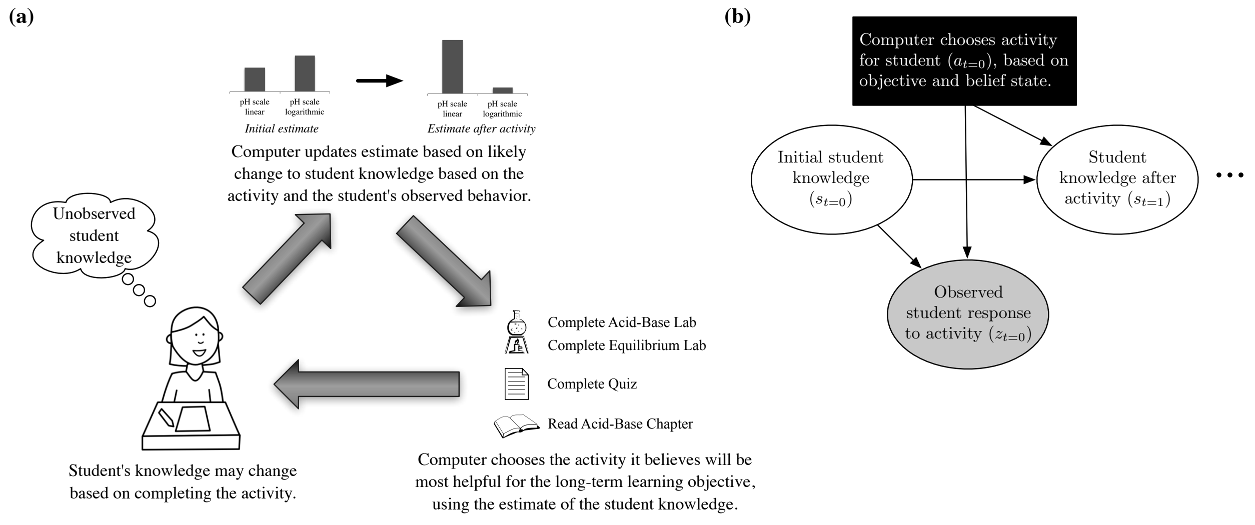

(a)
pH scale linear
pH scale logarithmic
Initial estimate
pH scale linear
pH scale logarithmic
Estimate after activity
Computer updates estimate based on likely change to student knowledge based on the activity and the student's observed behavior.
Unobserved student knowledge
Complete Acid-Base Lab
Complete Equilibrium Lab
Complete Quiz
Read Acid-Base Chapter
Student's knowledge may change based on completing the activity.
Computer chooses the activity it believes will be most helpful for the long-term learning objective, using the estimate of the student knowledge.
(b)
Computer chooses activity for student ($a_{t=0}$), based on objective and belief state.
Initial student knowledge ($s_{t=0}$)
Student knowledge after activity ($s_{t=1}$)
...
Observed student response to activity ($z_{t=0}$)

Figure 6.2: Mapping the POMDP model to teaching. (a) The teaching process consists of the computer choosing actions, which may be dependent upon its knowledge of the student. The student knowledge evolves based on the activities that she completes. By observing the student's behavior, such as how the student completes an interactive assessment, the computer can gain information about what the student understands and what misunderstandings she may have. (b) A graphical model representation showing how teaching corresponds to the components of the POMDP. Actions are pedagogical choices, states correspond to student knowledge, and observations are determined by student behavior. The computer teacher can make pedagogical choices to achieve some long-term objective, such as minimizing the time for the student to reach mastery.

Formulating teaching as a POMDP has several advantages. It provides a way of deriving an optimal policy for any teaching task and any learner model. This allows one to explicitly determine the expected consequences of making different assumptions about the learner or changing the learning objective. This can be helpful for evaluating the learner model and for determining whether a particular distinction actually has implications for teaching. The general framework is naturally modular, separating the parts of the teaching task and the assumptions made in each part of the model. This may be helpful for comparing or making improvements to automatic tutoring systems. Specifying a general framework also allows one to consider how particular methods for problem selection may be approximations to the optimal policy. By defining a problem selection method with respect to how it approximates the POMDP policy, one can make use of the existing POMDP literature and evaluate in what circumstances such an approximation is likely to perform well.

In the remainder of the chapter, we consider how this framework can be applied in concept-learning tasks. In such tasks, we use the time-based cost model that we described above: the cost of each action is the expected amount of time for the learner to complete the activity, and when the learner knows the correct concept, the action cost drops to zero. As a consequence, the computed policies select actions to minimize the expected time for the learner to understand the concept.

The space of tutorial actions may vary widely based on the domain being taught. We follow the tradition of concept learning via examples that has been explored in psychology, just as in Chapter 5, although we no longer focus on Boolean concepts. Within this type of concept learning, it is natural to consider three types of actions: *examples*, *quizzes*, and *questions with feedback*. *Examples* give the learner some information about the concept, but do not result in any observed behavior from the learner. *Quizzes* ask the learner a specific question related to the concept, but do not directly give the learner any new information about the concept. *Questions with feedback* combine these two action types by first asking the learner a question and then responding by telling the student the correct answer. *Example* and *quiz* actions are equivalent to the *tell* and *elicit* pedagogical actions that have been used previously in optimizations of intelligent tutoring systems (Chi et al., 2008). The POMDP can be used to find the optimal policy for teaching the learner the concept, taking into account the learner's responses to questions and balancing actions aimed primarily at diagnosis with those that provide information to the learner.

6.3 Learner models for concept learning

We consider three learner models, inspired by the cognitive science literature, that correspond to restrictions of Bayesian learning. Each learner model describes the state space of the POMDP as well as the transition and observation models. While the models we describe are only rough approximations of human concept learning, we will show that they are still sufficient to enable us to compute better teaching policies and that they can be applied to several different concept learning tasks. The three models we consider vary in complexity as well as in how closely they approximate human learning; this allows us to examine how well the POMDP approach can scale to more complex models. By using several different models, some of which we know are better approximations of human learning than others, we can also examine how closely the learner model must match human learning in order to lead to effective policies.

All of the models we consider share several assumptions about the concept-learning task. They each assume a discrete hypothesis space C of possible concepts. Such an assumption is reasonable in many contexts. For instance, the hypothesis space corresponding to possible meanings of a word might include binary vectors assigning each potential object as part of the concept or not, and the Boolean concepts in Chapter 5 were defined by a vector of which symbols were a part of the concept. The models we consider assume that size of the hypothesis space is finite, although it may be large. Since our models correspond to restrictions of Bayesian learning, they also assume that there is a prior distribution over the hypothesis space of concepts. Intuitively, this distribution represents learners' biases before they are exposed to any data about the concept; the difficulty of learning different concepts was represented by this prior distribution in Chapter 5. Finally, we assume that the domain is such that for any question the tutor might ask, each concept implies a single possible right answer. For instance, questions might ask whether particular objects are in the concept, and each possible concept in the hypothesis space specifies whether each object is in that concept. This assumption simplifies the problem somewhat, but could easily be modified to assume that concepts specify a probability distribution over possible answers to a question.

Memoryless model

We first consider a model in which the learner's knowledge state is the single concept she currently believes is correct, similar to a classic model of concept learning proposed by Restle (1962). In this model, the learner does not explicitly store any information previously seen. If an action is a *quiz* action, or if the provided evidence in an *example* or *question with feedback* action is consistent with the learner's current concept, then her state stays the same. If the action contradicts the current concept, the learner transitions to a state consistent with that action, with probability proportional to the prior probability of that concept:

$$p(s_{t+1} = c_i | s_t = c_j, a_t) \propto \begin{cases} p_0(c_i) & \text{if } c_i \text{ is consistent with } a_t \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

where $p_0(c_i)$ represents the prior distribution on concepts.

The observation model is deterministic: When asked to provide an answer to a question, the learner provides the answer z_n that is consistent with her current beliefs. This model underestimates human learning capabilities, and thus provides a useful measure of whether POMDP planning can still accelerate learning when a pessimistic learner model is used. This model is also attractive because it is less computationally complex than the other models we consider: The size of the state space is equal to the number of possible concepts $|C|$. Given this state space, the automated teacher's belief state b over the hidden learner state is a probability distribution over the $|C|$. Belief updating is performed using Equation 6.1, which will be an order $|C|^2$ operation.

Discrete model with memory

The key limitation of our first model is its lack of memory of past evidence. In general, this assumption is not accurate for human learning, although it is sometimes applicable to children (Levine, 1970). A more psychologically plausible state space is one in which learners maintain a finite memory of the past M actions in addition to their current guess of the true concept. This results in factored states that consist of s_h, the past M number-observation history, which is fully observed, and s_c, the hidden guess at the true concept. Like the memoryless model, this model assumes that the learner stores her current guess at the true concept, and this guess is updated only when information is shown that contradicts the guess. In this case, the learner shifts to a concept that is consistent with the current evidence and all evidence in the M-step history. The transition probability is again proportional to the initial concept probability. The transition model for s_h, representing the history, is deterministic. The observation model is also the same as in the memoryless case: The learner responds deterministically based on her current guess. Belief updating for the automated teacher can be updated in the same manner as for the previous model, with the size of the state space now equal to the number of possible concepts multiplied by the number of possible memory states.

Continuous model

A more complex view of learning is the type of Bayesian model applied in Chapter 5, in which a learner maintains a probability distribution over multiple concepts. Such an account allows one to model cases where a learner is unsure exactly which concept is correct but has ruled out some of the possibilities. In this case the state is a $|C|$-dimensional, continuous-valued vector that sums to 1, where C is the set of possible concepts. The ith position corresponds to the probability mass that the learner places on the ith concep. The state space S is an infinite set of all such vectors, the simplex $\Delta_{|C|}$.

The transition function assumes that for *quiz* actions, each state transitions deterministically to itself, as in the previous two models. For *example* and *question with feedback* actions, state dimensions for concepts that are inconsistent with the provided information are set to zero. Letting $p(s_{(t+1)i})$ be the ith entry in the distribution at time $t+1$, corresponding to the probability of the ith concept, then:

$$p(s_{(t+1)i}|s_t, a_t) \propto \begin{cases} p(s_{ti}) & \text{if } c_i \text{ is consistent with } a_t \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

The full joint transition probability is then re-normalized. This corresponds to a Bayesian generalization model with weak sampling (Tenenbaum & Griffiths, 2001). The observation model assumes the learner gives answer z_n to a question with probability equal to the amount of probability she places on concepts that have z_n as the correct answer for this question. For this model, the automated teacher must maintain a belief state over the infinite number of possible knowledge states. This requires approximating the belief state; we discuss the details of our approach to this issue in the next section.

Capturing deviations from the model

To improve the robustness of our policies to the coarse learner models we employ, all models include two extra parameters, a transition noise parameter ε_t and a production noise parameter ε_p. The transition noise parameter ε_t corresponds to the probability that the learner ignores a given teaching action, resulting in the learner not transitioning to a new concept. The production noise parameter ε_p corresponds to the probability that the learner produces an answer inconsistent with her current guess; this parameter plays a similar role to the guess and slip parameters common in some models of student knowledge (Corbett & Anderson, 1995). These parameters give the models extra flexibility to account for learners that do not behave precisely as the model predicts.

6.4 Selecting teaching actions using POMDP planning

Our goal is to compute a policy that selects the best action given a distribution over the learner's current knowledge state, the belief state. Offline POMDP planners compute such policies in advance. This approach requires pre-computing policies over the continuous space of possible belief

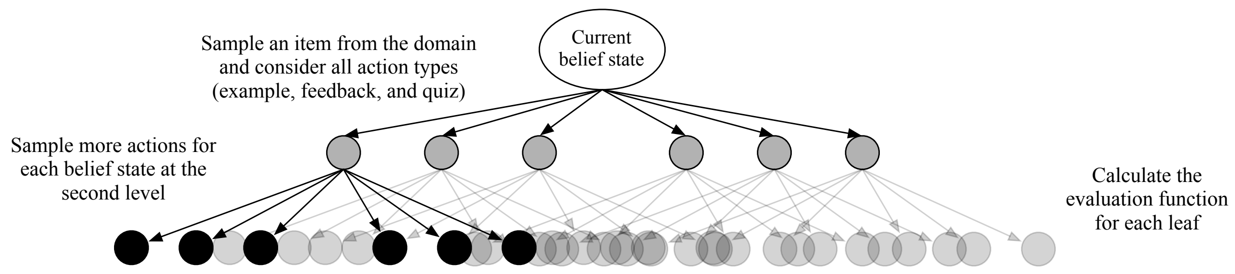

Current belief state
Sample an item from the domain and consider all action types (example, feedback, and quiz)
Sample more actions for each belief state at the second level
Calculate the evaluation function for each leaf

Figure 6.3: An example forward search tree for a two step horizon. The search considers the effect of each type of action applied with several different items from the domain; for example, to teach the learner the concept "odd numbers," the search would consider several different numbers that could be used for examples, quizzes, or questions with feedback. The number of actions sampled at each time step in the figure is only an example; in our actual experiments we sampled varying numbers of actions.

states.[3] The space of possible belief states is a simplex, as each belief is a probability distribution over the possible knowledge states. As the number of knowledge states grows, the dimensionality of this simplex is increased. Thus, as the size of the state space increases, offline approaches become infeasible. Since many teaching domains are likely to have large state spaces, we instead turn to online POMDP forward search techniques, which have proven promising in other large domains (see Ross, Pineau, Paquet, & Chaib-draa, 2008, for a survey).

We compute the future expected cost associated with taking different actions from the current belief state by constructing a forward search tree of potential future outcomes (see Figure 6.3). This tree is constructed by interleaving branching on actions and observations. To compute the values of actions next to the root belief state, the values of the leaf nodes are estimated using the evaluation function, and then their values are propagated up the tree, taking the max over actions and expectation over observations. After the tree is used to estimate the value of each action for the current belief, the best pedagogical action is chosen. The learner then responds to the action, and this response, plus the action chosen, is used to update the belief representing the new distribution over the learner's knowledge state. We then construct a new forward search tree to select a new action for the updated belief.

While forward search solves some of the computational issues in finding a policy, the cost of searching the full tree is $O((|A||Z|)^H)$, where H is the task horizon (i.e., the number of sequential actions considered), and requires an $O(|S|^2)$ operation at each node. This is particularly problematic as the size of the state space may scale with complexity of the learner model: the memoryless model has a state space of size $|C|$, while the discrete model with memory has state space of size $|C||A|^M$ and the continuous model has an infinite state space. To reduce the number of nodes we must search through, we take a similar approach to Ross, Chaib-draa, and Pineau (2008) and restrict the tree by sampling only a few actions. While this approach resolves some of the compu-

[3]Most state-of-the-art offline algorithms try to compute a policy over a subset of the reachable subspace, but this is still typically a very large region.

tational issues mentioned above, the cost of forward search is exponential in the depth of the tree, making it infeasible to plan to the true task horizon. We thus limit H to control the depth of the tree and use an evaluation function at the leaves. This evaluation function is based on the estimated probability that the student knows the correct concept.

Since the belief state in the continuous model is a distribution over an infinite set of states, we approximate the belief state for this model to make inference tractable. We represent the belief state as a weighted set of probabilistic particles and update these particles based on the transition and observation models (see Appendix A.2 for details). Particles inconsistent with the observations are eliminated. If no particles are consistent with the current observation, we reinitialize the belief state with two particles: one with a distribution induced by rationally updating the prior using all previous evidence and one with a uniform distribution. Depending on the number of particles used, this technique may be less computationally complex than the calculations for the other two models.

6.5 Empirical evaluation of optimal policies

We have now created a framework for approaching pedagogical action selection as a POMDP planning problem. This framework allows one to consider the costs and benefits of particular teaching actions with respect to their long-term effects on achieving a pedagogical objective. We have illustrated how to apply this to concept learning tasks, including several possible models of concept-learning that vary in terms of their assumptions about learners' knowledge. In the previous section, I described how to approximate an optimal policy from the POMDP, with relatively similar strategies for each of three concept learning models. We next test whether using a POMDP policy to choose pedagogical actions is effective at teaching human learners and feasible to use in real-time. Just as in previous chapters, we expect that our models do not precisely match characteristics of human learners in all ways. This means that the computed policies may not be optimal with respect to real learners; behavioral experiments allow us to explore whether they are still effective.

In the remainder of this chapter, we test the framework on two concept-learning tasks, alphabet arithmetic and learning numerical concepts. In alphabet arithmetic, learners infer a mapping from letters to numbers based on exposure to equations like $A+B=1$ that impose constraints on the possible mappings. While this task is artificial, it provides a preliminary evaluation of POMDP planning for problem selection and shares several important characteristics with real teaching domains: it is rich enough that learners may have misunderstandings, such as erroneous beliefs about which letter maps to which number, and that we expect some teaching policies to be more effective than others. The second concept learning task, the Number Game, involves a more complex space of numerical concepts and a larger number of possible teaching actions. For both tasks, we use simulations and behavioral experiments to test the effectiveness of POMDP planning for pedagogical action selection. These tasks differ from traditional domains covered in tutoring systems used in the classroom, but they share important characteristics with instruction in these domains: teaching occurs over a period of time, requiring sequencing information and incorporating the learners' behavior.

6.6 Simulation 1: Teaching simulated learners alphabet arithmetic

We first explore the performance of POMDP planning for teaching simulated learners alphabet arithmetic. These simulations address two questions: (1) how effective is POMDP planning when the assumed learner and the actual learner match, and (2) is POMDP planning still effective when the actual learner differs from that assumed by the POMDP? If POMDP policies that do not match the learner are still able to make learning more efficient, then this provides some evidence that the mismatch between human learners and the simple models of learning that we consider may not be insurmountable.

As described above, alphabet arithmetic involves learning a mapping between letters and numbers; in this case, we teach a mapping from the letters A–F to the digits 0–6. We assume learners have a uniform prior over mappings. For *example* actions, learners are shown an equation where two distinct letters sum to a numerical answer. For instance, A could be mapped to 0 and B to 1, and one might show the learner the equation $A+B=1$. *Quiz* actions leave out the numerical answer and ask the learner to give the correct sum. *Questions with feedback* combine these two actions. The planning goal for alphabet arithmetic is to minimize the amount of time for learners to correctly identify the mapping. Given this space of actions, we expect that modeling and tracking the learner's state may speed learning since teaching actions can explicitly address or diagnose the learner's misunderstandings.

Methods

We conducted four simulations for each type of learner (memoryless, discrete with memory, and continuous). One simulation used a *random* policy, and the other three simulations used a POMDP policy driven by each of the learner models. This allowed us to determine how quickly, for instance, a memoryless learner could be taught using a *memoryless* policy versus a *continuous* policy.[4] Fifty simulations were run for each of these twelve combinations of learner and policy.

To match the experiment presented in the next section, we alternated between *teaching phases*, in which we selected pedagogical actions for the simulated learner, and *assessment phases*, in which we checked whether the simulated learner had identified the mapping. Each teaching phase consisted of a sequence of three pedagogical actions. After the teaching phase, there was an assessment phase that varied slightly based on the type of simulated learner. For the memoryless learners and the discrete representation with memory learners, the current guess of the learner was compared to the true mapping. If they were identical, then the learner had mastered the concept and teaching was terminated. For the continuous learner, a mapping was sampled from the learner's current distribution over possible mappings, and if this mapping matched the true mapping, then teaching was terminated. If a learner failed to achieve mastery after 40 teaching phases, then

[4]For simplicity, we distinguish the POMDP policies based on the learner model they assume. For example, the *memoryless* policy is the POMDP policy that assumes a memoryless learner model.

teaching was also terminated; this was to match the experimental procedure in which teaching of human learners would be terminated after 40 phases regardless of performance.

Finding the POMDP policies requires setting the parameters of the cost function as well as the parameters of each learner model. To make the simulations as realistic as possible, we set these parameters based on data from teaching 20 human participants using a *random* policy; these were the participants in the control condition of Experiment 1, described below. The cost of each action that was used by the POMDP planner was the median time to complete each action type from the participants in the control condition: example actions took 7.0s, quiz actions took 6.6s, and question with feedback actions took 12s. When computing the action values within the forward search tree, we set the cost for a leaf node to be the probability of not passing the assessment phase multiplied by $10 \cdot \min_a R(a)$, a scaling of the minimum future cost.

We set the probability ε_t of ignoring a teaching action and the probability ε_p of making a production error when answering a question by finding the values that maximized the log likelihood under a given model of the data from the participants taught using a random policy; details of this procedure and the resulting values can be found in Appendix A.3.

For forward planning, we set the parameters of the algorithm to sample as many actions as possible given the constraints of planning in real-time (see Appendix A.1 for more details about the number of free parameters in the algorithm and how these parameters were set). In particular, we limited all computations to three seconds. Given this constraint, we set the lookahead horizon to two actions. Policies for the first nine actions were precomputed with ten actions sampled at each level. Caching the first nine actions allows us to consider more actions at each horizon, while still using a constrained number of actions to speed computations. Later actions were computed by sampling the following number of actions at each level: seven and six actions for the *memoryless* model; eight and eight actions for the *discrete model with memory*; and four and three actions for the *continuous* model. Sixteen particles were used for the *continuous* model, and $M = 2$ for the *discrete model with memory* (both for the simulated learner and the POMDP policy). The effects of varying these parameters are not extreme: sampling more actions at each level results in less variance, but does not tend to change the outcome across many simulations. The results are also not very sensitive to changes in the number of particles, although using very small numbers of particles performs poorly. The simulations for the discrete learner with memory are sensitive to changes in the assumed memory capacity of the learner; for instance, if memory capacity is set to zero, this learner is identical to the memoryless learner.

Results and discussion

For each type of simulated learner, we compared how the number of teaching phases required and the expected time to mastery varied based on the teaching policy. Expected time to mastery was computed by assuming that each action took the amount of time assumed by the POMDP planner. For instance, if a simulated learner identified the mapping after three actions, two of which were example actions and one of which was a quiz action, the expected time to mastery would be $2 \cdot 7.0 + 1 \cdot 6.6 = 20.6$ seconds. Initial inspection showed that the distribution of learning times exhibited a long right tail, so we analyzed results using medians, which are more robust than means to outliers and non-symmetric distributions.

As shown in Figure 6.4, the simulation results demonstrate that teaching using a POMDP policy can decrease both the number of teaching actions required and the expected time to mastery relative to the random policy, even if the learner does not match the learner model in the POMDP. For all three learner types, there was a significant main effect of teaching policy on the number of teaching actions (Kruskal-Wallis: memoryless learner, $\chi^2(3) = 26.5$, $p < .001$; discrete representation with memory, $\chi^2(3) = 14.8$, $p < .001$; continuous learner, $\chi^2(3) = 20.6$, $p < .001$) and on the expected time to mastery (Kruskal-Wallis: memoryless learner, $\chi^2(3) = 21.4$, $p < .001$; discrete representation with memory, $\chi^2(3) = 26.7$, $p < .001$; continuous learner, $\chi^2(3) = 46.5$, $p < .001$). We performed planned, pairwise comparisons between the POMDP policies and the *random* policy. For the learner with the discrete representation and memory as well as the learner with a continuous representations, all three POMDP policies significantly improved learning efficiency.[5] For the memoryless learner, only the *memoryless* policy resulted in significantly fewer teaching actions than the *random* policy (Kruskal-Wallis: *memoryless* versus *random*, $\chi^2(1) = 6.5$, $p < .025$); all three POMDP policies, however, resulted in lower expected time to mastery than the *random* policy (Kruskal-Wallis: *memoryless* versus *random*, $\chi^2(1) = 18.8$, $p < .001$; *discrete model with memory* versus *random*, $\chi^2(1) = 11.7$, $p < .001$; *continuous* versus *random*, $\chi^2(1) = 7.1$, $p < .01$). The lack of difference in the number of actions required is probably due to the large number of simulations in which the memoryless learner never identified the concept.

We also examined the simulations to see what types of policies emerged from different assumed learner models. Figure 6.5 shows part of the *discrete model with memory* policy. Here, the model shows an example, and then follows this with a quiz question. The question allows the model to check whether the learner has mastered part of the concept, and is less costly than showing another example, since quiz actions are generally completed more quickly than example actions. After the example, the next action is dependent on the answer that is given by the student, as the POMDP updates its belief state using the observation model. Some answers may result in more quiz questions, either because the student answers correctly or because the model is attempting further diagnosis, or in an example. Questions with feedback were rarely used by any of the POMDP policies because of their high cost.

There were some differences based on which learner model was assumed, and which type of learner was being taught. Overall, the policies based on the *discrete model with memory* and the *continuous model* used an average of at least 90% example actions, with the remaining actions be-

[5]Planned pairwise comparisons using Kruskal-Wallis tests were conducted. For number of teaching phases for the learner with the discrete representation and memory: *memoryless* versus *random*, $\chi^2(1) = 8.7$, $p < .005$; *discrete model with memory* versus *random*, $\chi^2(1) = 7.0$, $p < .01$; *continuous* versus *random*, $\chi^2(1) = 11.8$, $p < .001$. For expected time to mastery for the learner with the discrete representation and memory: *memoryless* versus *random*, $\chi^2(1) = 17.4$, $p < .001$; *discrete model with memory* versus *random*, $\chi^2(1) = 15.2$, $p < .001$; *continuous* versus *random*, $\chi^2(1) = 18.1$, $p < .001$. For number of teaching phases for the continuous learner: *memoryless* versus *random*, $\chi^2(1) = 12.9$, $p < .001$; *discrete model with memory* versus *random*, $\chi^2(1) = 5.4$, $p < .025$; *continuous* versus *random*, $\chi^2 = 17.6$, $p < .001$. For expected time to mastery for the learner with the discrete representation and memory: *memoryless* versus *random*, $\chi^2(1) = 31.9$, $p < .001$; *discrete model with memory* versus *random*, $\chi^2(1) = 18.5$, $p < .001$; *continuous* versus *random*, $\chi^2(1) = 35.2$, $p < .001$.

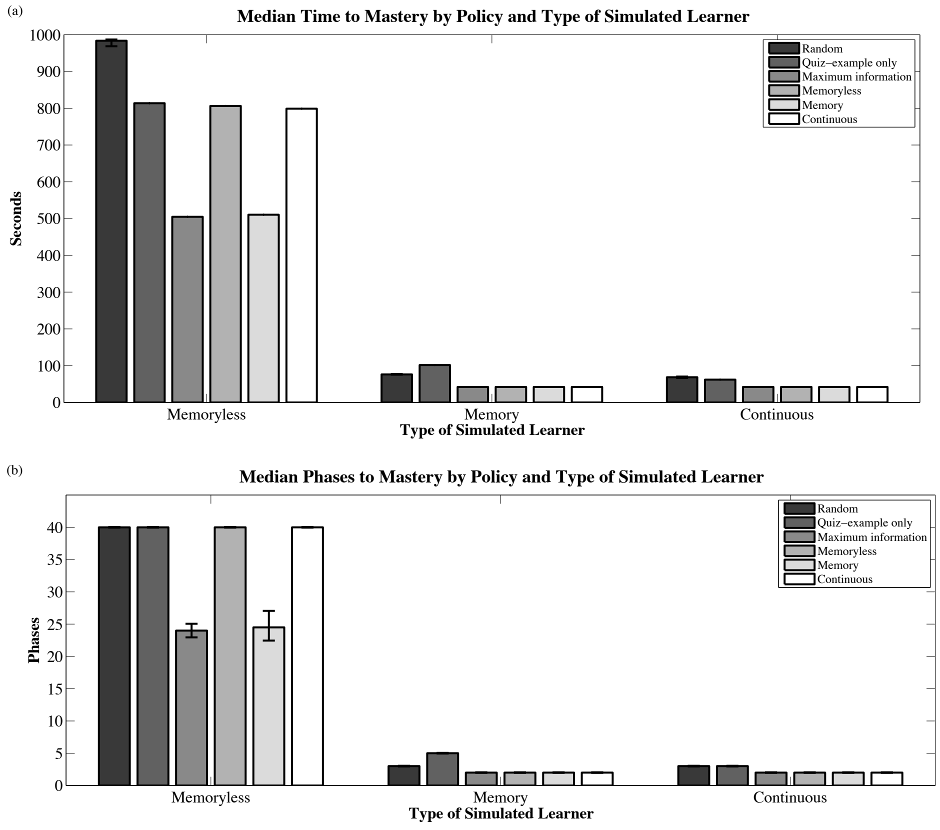


Figure 6.4: Results for Simulations 1 and 2: Median time (a) and phases (b) for simulated learners to reach mastery, by policy type. Error bars correspond to bootstrapped 68% confidence intervals (equivalent to one standard error). Results based on expected time to mastery are similar to those based on phases, except in the case of the memoryless learner. In that case, expected time to mastery is lower for all POMDP policies than for the *random* policy, although not lower than the *quiz-example only* policy.

ing almost exclusively quiz questions. One exception was the case of the *discrete memory model* for teaching and a continuous learner, where one simulation never reached mastery and the policy devolved into asking primarily quiz questions. Given a longer period to teach, this policy would presumably diagnose the learner's misunderstanding, and then use an example to correct that misunderstanding. The proportion of quiz versus example actions was more variable for the policy based on the *memoryless* model, probably because this model assumes a much more stochastic learner than the other two policies. This means that the model has less certainty about the learner's knowledge state, and thus less guidance for choosing what questions to ask the learner.

In summary, POMDP policies were in general effective at improving learning efficiency even when the assumed policy was incorrect. While different types of simulated learners varied dramatically in the time required to master a concept, these differences were generally not greatly mediated by which type of learner was assumed by the POMDP planner. Based on the type of learner as-

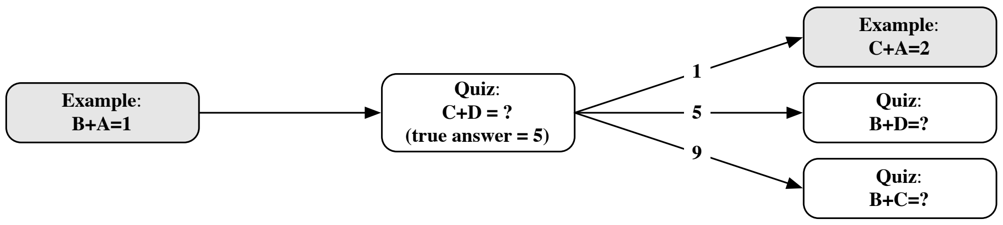


Figure 6.5: Part of a policy from the *discrete model with memory*. Possible student answers to the quiz are indicated on the arrows; some are omitted. Based on the student's response, the action after the quiz may correct a misunderstanding, try to better diagnose the cause of an incorrect answer, or continue quizzing to try to detect a misunderstanding. Actions after the quiz are contingent upon the student's response, reflecting the fact that action choices are based on the computer's beliefs about the student's knowledge, which are updated given the student's behavior.

sumed by the POMDP planner, however, the policies do have qualitatively different characteristics.

6.7 Experiment 1: Teaching human learners alphabet arithmetic

The simulation results above show that POMDP policies for selecting teaching actions can have a measurable effect on the rate at which the mapping is identified. They also suggest that mismatches between the learner and the assumed learner model do not make the POMDP policy ineffective, and that even a simple learner model can be used to derive effective POMDP policies. We next turn to a behavioral experiment to explore whether these findings also hold when teaching human learners, who may vary more dramatically and in different ways than the simulated learners.

Methods

Participants

A total of 40 participants were recruited online and received a small amount of monetary compensation for their participation.

Stimuli

All participants were randomly assigned three mappings between the letters *A–F* and the numbers 0–5. These mappings were learned in succession.

Procedure

Participants were assigned to either the *control condition*, in which teaching actions for all mappings were chosen randomly, or to the *experimental condition*. Assignment to condition was based

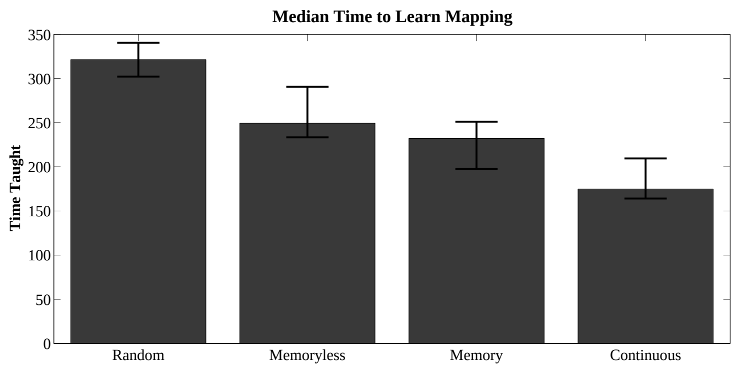


Figure 6.6: Median time to learn each mapping in Experiment 1, by policy type; error bars correspond to bootstrapped 68% confidence intervals (equivalent to one standard error).

on the time of participation, with all participants in the *control condition* assigned prior to participants in the *experimental condition*. This allowed us to use the results from the random control condition to set the parameters of the POMDP models, as described above. Each participant in the experimental condition experienced all three of the teaching policies in random order, one for each mapping learned. The experiment consisted of a sequence of teaching and assessment phases. In each teaching phase, a series of three teaching actions was chosen based on condition. After each teaching phase, participants completed an assessment phase in which they were asked to give the number to which each letter corresponded. Answers in the assessment phase were not used to update the beliefs of the POMDP models to allow for fair comparisons across conditions. Teaching of a given mapping terminated when the participant completed two consecutive assessment phases correctly or when 40 teaching phases had been completed. Within all phases, the equations the participant had seen were displayed on-screen, and participants could optionally record their current guesses about which letter corresponded to which number.

Computing policies

The cost for each action type and the setting of the ε parameters was the same as that in the simulations above. In all conditions, we also inserted a three second delay between actions in order to allow time for planning.

Results and discussion

We compared the number of phases as well as the time participants took to learn each mapping. Just as in the simulations, we analyzed results using medians. There was no significant within-subjects difference in the amount of time or number of phases to learn the first, second, or third mapping (Kruskal-Wallis $p > 0.8$), demonstrating that the results are not due to participants becoming more adept at the task.

Overall, participants taught by POMDP planning took significantly fewer phases to learn each mapping than participants in the control condition (three phases versus four; Kruskal-Wallis: $\chi^2(3) = 24.9$, $p < 0.001$) and also took significantly less time per mapping (232 seconds versus 321 seconds; Kruskal-Wallis: $\chi^2(3) = 16.5$, $p < 0.001$); see Figure 6.6. Planned pairwise comparisons show that all of the POMDP policies resulted in fewer phases to completion than the *random* policy (Kruskal-Wallis: *memoryless* versus *random*, $\chi^2(1) = 8.5$, $p < .005$; *discrete model with memory* versus *random*, $\chi^2(1) = 10.5$, $p < .005$; *continuous* versus *random*, $\chi^2(1) = 16.3$, $p < .001$), and all POMDP policies but the *memoryless* policy resulted in significantly faster learning (Kruskal-Wallis: *memoryless* versus *random*, $\chi^2(1) = 2.9$, *n.s.*, $p = .087$; *discrete model with memory* versus *random*, $\chi^2(1) = 7.4$, $p < .01$; continuous versus random, $\chi^2(1) = 12.5$, $p < .001$).

As in the simulations, differences in policies occurred based on the learner model used. Policies for both the *discrete model with memory* and the *continuous* model began with six independent equations that fully specify the mapping. This is the policy one might have hand-crafted to teach this task, demonstrating that despite approximations in planning, the POMDP planner finds reasonable teaching policies. Each of the policies for these two models gives examples until there is a high probability the learner is in the correct state, and then asks quiz questions, which are less costly than examples, to detect errors in the learned mapping.

The *memoryless* policy repeats specific example actions more often than the other policies since it assumes that the learner does not store previous actions in memory. This is clearly a pessimistic assumption, especially given that previously seen equations were displayed on-screen during the experiment. The fact that this model did not significantly decrease time to learn suggests such an unrealistic assumption may be detrimental for problem selection. However, the actions that are chosen do seem to be those that most limit the number of consistent hypotheses given both the structure of the mapping and the immediate preceding action, suggesting the we are finding a relatively good policy given the constraints of this learner model.

In the simulations, the types of actions chosen varied based on the assumed learner model. These variations persisted in the experiment. Figure 6.7 shows number of actions of each type at each point in time in the experiment where at least three participants remained. Overall, the *continuous* policy asked the fewest quiz questions, while the *memoryless* policy asked the most (39% of actions). The *memoryless* tended to ask quiz questions later than the *continuous* policy, though, resulting in fewer participants receiving any quiz questions. The *memoryless* policy likely asked questions more frequently than the other policies because the state of a memoryless learner after an example is known with less certainty than in the other two models: in those models, the new state is constrained to be consistent with multiple pieces of past evidence, whereas the memoryless learner's state is constrained only to be consistent with the current example. None of the policies used many feedback actions due to the fact that these actions are considerably more expensive than other actions.

While the actions did vary between the policies, one might wonder why these different policies had little variance in their teaching efficacy. Specifically, while the two more complex learner models significantly reduced the time to learn, they did not result in significantly different outcomes from one another or from the *memoryless* policy. However, the simulation predicts this result if our learners are similar to the discrete learner with memory or the continuous learner. In both of

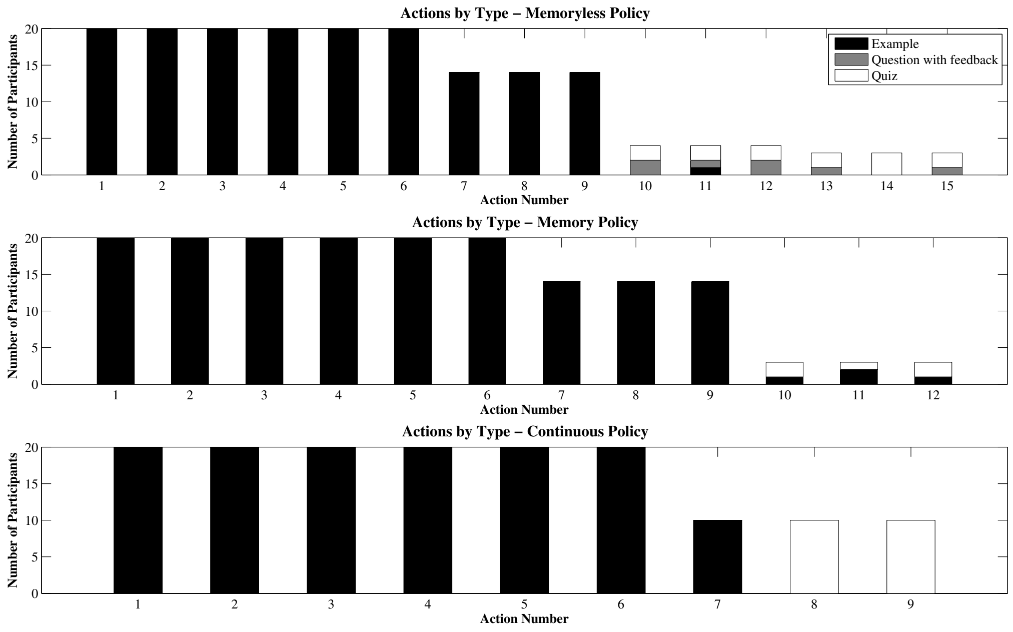


Figure 6.7: Action types at each time in the experiment, by condition. Each graph shows the number of participants for whom the *n*th action was an example, question with feedback, or quiz. Data is shown only for time points where at least three participants had not mastered the concept.

these cases, all of the teaching policies were equivalent to one another. These two learners are quite powerful, and able to master the concept within only a few trials; thus, we might expect that there are multiple optimal or near-optimal policies, such that using a policy from one model is still relatively effective for a different model. Since we perform only approximate POMDP planning, we expect that the computed policies are only near-optimal even with respect to the given learner model. Additionally, the lack of difference in outcomes from using different teaching policies may also be due to the fact that learning takes place within a relatively small number of actions, and we assess student knowledge only after every three actions. This is necessary to avoid continually querying the learner, but it means we cannot detect small changes in the number of actions required for mastery.

To summarize, the POMDP policies were effective for teaching human learners an alphabet arithmetic mapping, with policies based on the two more complex learner models significantly decreasing time on task as compared to a random control policy. These gains are similar to what we predicted based on the simulation results. This suggests that while the human learners may not exactly match any of our learner models, the differences are not so large as to prevent these models from being effective guides for pedagogical decisions.

6.8 Additional comparison policies for alphabet arithmetic

Experiment 1 showed that the POMDP planning framework could decrease the time to learn a mapping relative to a random policy. However, it explored only a single comparison policy, in which actions were chosen uniformly at random. We now consider two additional comparison policies: a random policy with only quizzes and examples (*quiz-example only*), and a policy that chooses the example that would result in the maximum information gain for the learner (*maximum information*). Each of these policies could be expected to perform better than the *random* policy, and understanding how they compare in simulation and behavioral results to the POMDP policies can further illustrate when and how these policies can be effective. We first briefly describe the two new policies and their effectiveness in simulation, and then turn to a second behavioral experiment.

New control policies

The *quiz-example only* policy was included since the results of Experiment 1 indicated that *questions with feedback* took longer than the other actions for learners to complete. Since these actions will make up roughly one third of all actions in the random policy, we thought this might lead to slower learning with the *random* policy based on the mix of actions rather than the intelligent sequencing of actions. While part of the power of the POMDP policies is to decide what mix of actions is appropriate, we believe that the *quiz-example only* policy provides a comparison that might add value over a *random* policy while not including a model of the learner.

The *maximum information* policy is similar to the POMDP policies in that it includes a model of the learner, but it does not plan over multiple time steps and is not as flexible in the types of learner models that it can be paired with. The *maximum information* policy calculates which action will produce the maximum information gain for the learner, where information gain is defined as the difference between the Shannon entropy of the learner's state before the action and the entropy of the new state after the action has been taken. The entropy of the state is calculated as $-\sum_{i=1}^{n} p(c_i)\log(c_i)$, where c_i is the ith concept (Shannon & Weaver, 1948); when much of the probability is on only a few concepts, the entropy of the state will be low, while a uniform distribution corresponds to the highest possible entropy. This quantity has been used in other work for selecting data for human and computer learners (e.g., MacKay, 1992; Tang et al., 2010). This policy only considers examples, since quizzes are assumed to not change the learner's state and as mentioned above, *questions with feedback* are more time consuming for learners without increasing the information gain. Because entropy will always be zero when the learner only has a single hypothesis, this *maximum information* policy requires a learner model where the hypothesis can be represented as a distribution over multiple concepts. Of our three learner models, only the continuous model represents the learner's beliefs as a distribution over concepts. Thus, the *maximum information* policy assumes that this is the correct learner model. In general, we would expect this model to perform relatively well, although it could perform poorly in cases where learners drastically diverge from the model's assumptions.

Simulation 2: Performance of additional controls for alphabet arithmetic

We simulated the two new control policies and compared the results to those described in the initial simulations. As shown in Figure 6.4, the *quiz-example* only policy generally performs similarly to the *random* policy. Both result in slower learning than the POMDP policies. In contrast, the simulated learners learn quite quickly from the *maximum information* policy. None of the POMDP policies are significantly faster than this policy. The *maximum information* policy is effective because this domain allows information to be progressively incorporated, such that the amount of information gained from a single action is a good heuristic for the overall progress in learning the concept. However, it is promising for the POMDP model that it does not in most cases fare worse than the *maximum information* policy, despite the approximations necessary to carry out planning. The POMDP model still maintains the advantage of being able to work with a broader array of learner models and to consider the benefits of diagnosing the learner's knowledge in addition to the benefits of trying to change that knowledge. Overall, these results suggest that while POMDP planning is not the only way to effectively select pedagogical actions, this method generally performs as well or better than the comparison methods.

6.9 Experiment 2: Effectiveness of new control policies for alphabet arithmetic

We conducted a second behavioral experiment to replicate the effective performance of the three POMDP policies and to examine the performance of the two new policies.

Methods

Participants

A total of 100 participants were recruited online and received a small amount of monetary compensation for their participation.

Stimuli

All participants were randomly assigned three mappings between the letters *A*–*F* and the numbers 0–5. These mappings were learned in succession.

Procedure

Participants were assigned to be taught by one of the five policies. Unlike in Experiment 1, participants taught by a POMDP policy were taught by the same type of policy for all three mappings, rather than one mapping being taught by each of the policies. Each policy was used to teach twenty participants. The remainder of the experimental procedure was the same as in Experiment 1.

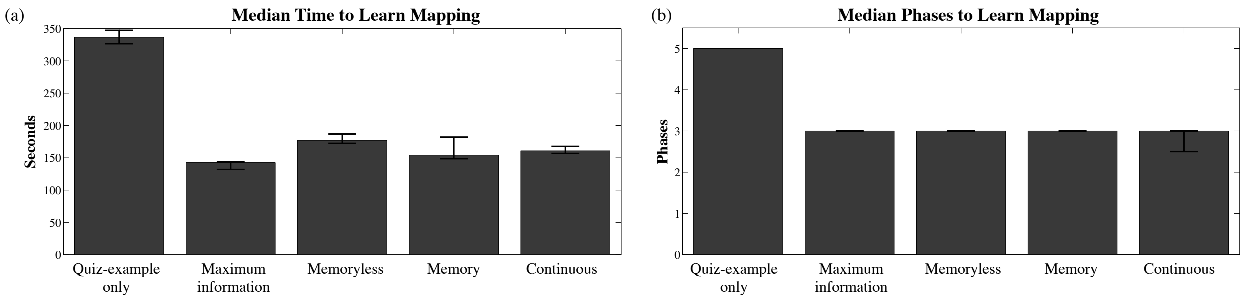


Figure 6.8: Median time to learn each mapping in Experiment 2, by policy type; error bars correspond to bootstrapped 68% confidence intervals (equivalent to one standard error).

Results

As shown in Figure 6.8, the POMDP policies were more effective than the *quiz-example only* policy in both time and phases to mastery and about as effective as the *maximum information* policy, mirroring the simulation results. As in Experiment 1, there was a significant effect of policy on the number of phases and the time to learn each mapping (Kruskal-Wallis: Phases: $\chi^2 = 91.2$, $p < .0001$; Time: $\chi^2 = 55.6$, $p < .0001$). Planned-pairwise comparisons showed that the POMDP policies and the *maximum information* policy were all significantly more effective than the *quiz-example only* policy (Kruskal-Wallis: Time: *memoryless* versus *quiz-example only*, $\chi^2(1) = 23.0$, $p < .0001$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 26.2$, $p < .0001$; *continuous* versus *quiz-example only*, $\chi^2(1) = 41.7$, $p < .0001$, *maximum information* versus *quiz-example only*, $\chi^2(1) = 40.0$, $p < .0001$; Phases: *memoryless* versus *quiz-example only*, $\chi^2(1) = 41.4$, $p < .0001$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 47.6$, $p < .0001$; *continuous* versus *quiz-example only*, $\chi^2(1) = 56.7$, $p < .0001$, *maximum information* versus *quiz-example only*, $\chi^2(1) = 52.7$, $p < .0001$). With correction for multiple comparisons, the *maximum information* policy and the POMDP policies were not significantly different from one another.[6]

In Experiment 2, participants taught by the POMDP policies tended to learn the mappings more quickly than when taught by the POMDP policies in Experiment 1, and all three POMDP policies had very similar median times to mastery. One reason for this discrepancy may be because participants in Experiment 2 were always taught by the same policy. This may result in more familiarity with how a concept is taught, resulting in faster learning. This is reflected in the data: unlike in Experiment 1, there was a significant effect on the time to mastery based on whether the mapping was the first, second, or third mapping learned (Kruskal-Wallis: $\chi^2 = 59.6$, $p < .0001$). A follow up multiple comparison test showed that the second two mappings were learned significantly more quickly than the first mapping.

[6]Kruskal-Wallis: Time: *memoryless* versus *maximum information*, $\chi^2(1) = 3.98$, $p = .0461$; *discrete model with memory* versus *maximum information*, $\chi^2(1) = 1.11$, $p > .2$; *continuous* versus *quiz-example only*, $\chi^2(1) = 1{,}29$, $p > .2$. Phases: *memoryless* versus *quiz-example only*, $\chi^2(1) = 2.62$, $p > .1$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 1.51$, $p > .2$; *continuous* versus *quiz-example only*, $\chi^2(1) = 0.68$, $p > .4$.

Overall, Experiment 2 replicates the main results of Experiment 1, demonstrating that the POMDP policies are more effective than a simple control policy. The trend of increasing complexity in the learner model leading to faster time to mastery was not replicated in this experiment, suggesting that simple models can be effective even if they are known to not match human leaners exactly. This experiment was unable to determine whether the POMDP policies were more effect than the *maximum information* policy, which represents a relatively sophisticated way of automatically choosing pedagogical actions. Showing that POMDP policies can result in an advantage over maximum information policies will require using a more complex concept space; we consider such a space in the remainder of this chapter.

6.10 Evaluating effectiveness in a larger state space: The Number Game

The first two experiments demonstrate that POMDP policies can accelerate learning relative to baseline policies in alphabetic arithmetic, with our approximations sufficient to conduct online planning to choose activities for learners. We now explore whether the POMDP framework can accelerate learning in a larger and more complex concept space, the space of numerical concepts used in the Number Game (Tenenbaum, 2000). In the Number Game a participant is trying to infer a number concept, which consists of a subset of numbers between 1 and 100. For example, both “even numbers” and “numbers that end in three” are possible concepts. In past Number Game research, information about the concept is typically given as a static set of one or more examples of numbers that are in the target concept, although other variations exist (Nelson & Movellan, 2001). Good performance generally requires multiple examples, which suggests that a sequential teaching strategy has the potential to accelerate this process. We modify the Number Game so that learning occurs over a sequence of steps. Each step consists of one teacher action, just as in alphabet arithmetic. The number game provides a larger space of possible concepts and actions in a domain that people have more experience with than alphabet arithmetic, making it an interesting domain in which to test the effectiveness of POMDP planning. As in our exploration of alphabet arithmetic, we first conduct simulations of teaching in this domain, and then turn to an experimental evaluation with human learners.

6.11 Simulation 3: Teaching simulated learners the Number Game

We initially explore teaching the number game using POMDP policies via simulation. These simulations can provide evidence for how differences between the simulated learner and the learner assumed by the POMDP policy affect the speed with which concepts are learned.

Methods

The methods for this simulation closely mirror those in Simulations 1 and 2: simulations are conducted for each combination of learner (memoryless, discrete with memory, and continuous) and teaching policy (*random*, *quiz-example only*, *maximum information*, and the three POMDP policies). There are many different number concepts that could be taught or that a learner could learn. Following previous work, we use a hypothesis space consisting of the 6412 most psychologically salient of the 2^{100} possible concepts, and a hierarchical prior p_0 over these concepts that was developed in prior work (Tenenbaum, 2000). We taught three concepts from the hypothesis space in both this simulation and Experiment 3: multiples of seven; multiples of four minus one; and numbers between 64 and 83 (inclusive). Fifty simulations were run for each combination of simulated learning, teaching policy, and target concept. To match the experiment described below, the *random* and *quiz-example* only policies were modified to sample half of the numbers from within the concept and half from outside the concept; further explanation for this change is provided in Experiment 3.

As in Experiment 1, the simulations alternate between teaching and assessment phases. In each teaching phase, a series of five teaching actions was chosen based on condition. Since there are 100 numbers, and three action types, there were 300 different possible teaching actions. After each teaching phase, participants completed an assessment phase in which they were shown a sequence of ten numbers, five randomly chosen from within the concept and five from outside of the concept, and asked whether each number was in the concept. As in the previous experiment, answers in the assessment phase were not used to update the POMDP models. Teaching was terminated when the simulated learner correctly responded to all numbers within a single assessment phase, or when 40 teaching phases had been completed.

To set the parameters of the POMDP policies, we followed the same procedure in Experiment 1, conducting the random condition prior to any other conditions and using these data to set action costs and ε parameters. The cost of each action type was the median time for participants in the random policy condition to complete these actions: 2.4s for example actions, 2.8s for quiz actions, and 4.8s for question with feedback actions. The two additional parameters for each learner model, ε_p and ε_t, were again set to maximize the log-likelihood of the data in the control condition (see Appendix A.3 for details). Actions for the first four teaching phases (20 total actions) were precomputed. For later actions, we again set the planning parameters such that all models would take about three seconds to compute an action. As before, these parameters were not optimized, and small changes in their values did not have large impacts on the approximate policies. The lookahead horizon was set to three for the *continuous* model, and two for the other models. At each level, the following number of actions were sampled: six and eight actions for the *memoryless* model; six and six actions for the *discrete model with memory*; and six, six, and eight actions for the *continuous* model. Sixteen particles were used for the *continuous* model, and $M = 2$ for the *discrete model with memory*.

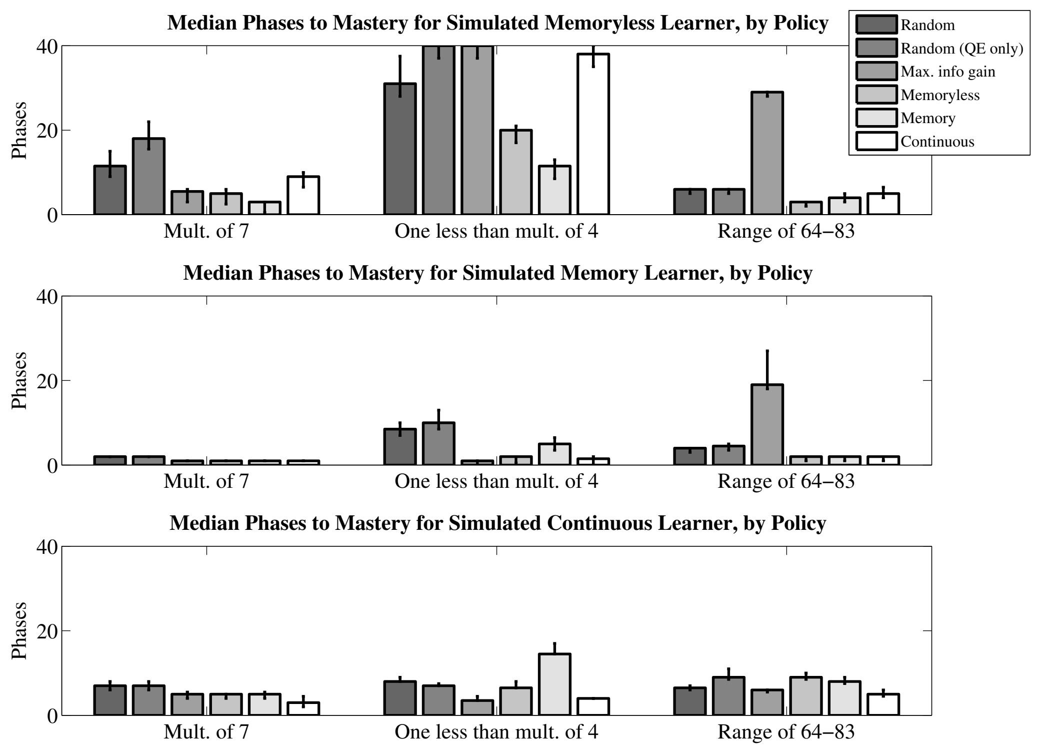


Figure 6.9: Median time for simulated learners to reach mastery, by policy type; error bars correspond to bootstrapped 68% confidence intervals (equivalent to one standard error).

Results and discussion

For each type of simulated learner, we compared how the number of teaching phases required and the expected time to mastery varied based on the teaching policy. Expected time to mastery was computed by assuming that each action took the amount of time assumed by the POMDP planner, as in Simulation 1. Following our analysis of alphabet arithmetic, we analyze results using medians. Ideally we would perform a nonparametric analogue of an analysis of variance to examine if there is a main effect of the teaching policy on learning time, controlling for concept. The nonparametric Friedman test for two-way experiments most closely resembles such a test, but it assumes equal variance across groups. Since this assumption is violated by our data, the p-values calculated in the Friedman test are not appropriate. Consequently, we calculated an empirical null distribution for the Friedman statistic using a bootstrap. We constructed $100,000$ alternate datasets by sampling with replacement from our empirical distribution of learning times across policies for the same concept and randomly assigned each sampled data point to one of the six policies. Computing Friedman's statistic for each alternate dataset then gives an appropriate null distribution for comparison with the experimental results.

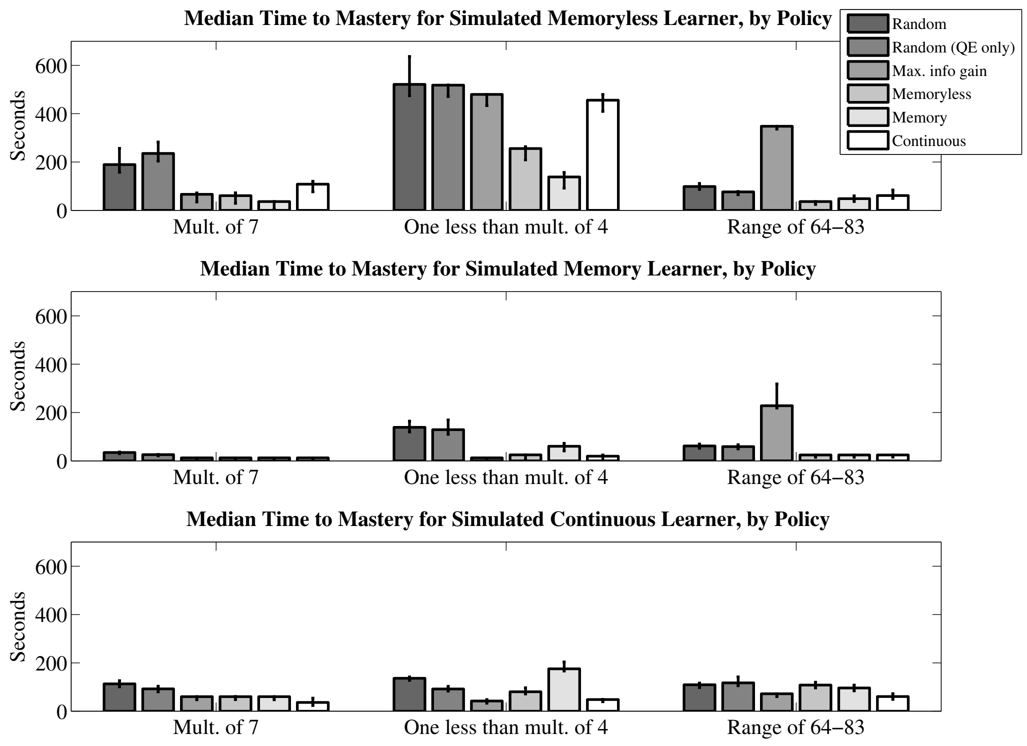


Figure 6.10: Median phases for simulated learners to reach mastery, by policy type; error bars correspond to bootstrapped 68% confidence intervals (equivalent to one standard error).

As shown in Figures 6.9 and 6.10, there is considerable variation in effectiveness for each policy based on what type of learner is assumed as well as what type of concept is being taught. For the simulated continuous learner, there was a significant effect of teaching condition on both time (Friedman: $\chi^2(5) = 111.5$, $p < .0001$) and phases (Friedman: $\chi^2(5) = 67.0$, $p < .0001$) to mastery. Both the *continuous* POMDP policy and the *maximum information* policy performed well, with neither outperforming the other. Planned pairwise comparisons between the POMDP policies and the control policies showed that the *continuous* policy outperformed the two random policies, but that the *memoryless* and *discrete model with memory* policies performed less well, especially when only phases to mastery was considered.[7]

[7]Planned pairwise comparisons using Friedman tests were conducted. For teaching phases to mastery: *memoryless* versus *random*, $\chi^2(1) = 14.5$, $p = .07$, *n.s.*; *discrete model with memory* versus *random*, $\chi^2(1) = -9.0$, $p > .1$, *n.s.*; *continuous* versus *random*, $\chi^2(1) = 42.9$, $p < .0001$; *memoryless* versus *quiz-example only*, $\chi^2(1) = 27.7$, $p < .005$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 4.2$, $p > .3$, *n.s.*; *continuous* versus *quiz-example only,* $\chi^2(1) = 56.1$, $p < .0001$. The *maximum information* policy outperformed the POMDP policies, although not significantly in the case of the *continuous* policy: *memoryless* versus *maximum information*, $\chi^2(1) = -31.8$, $p < .005$; *discrete model with memory* versus *maximum information*, $\chi^2(1) = -55.5$, $p < .001$; *continuous* versus *maximum*

For the simulations with the memoryless and memory learners, mismatches between the learner assumed by the POMDP policies or the *maximum information* policy and the actual type of the simulated learner led to differences in results. While for the memory learner, the *maximum information* policy performed as well as the POMDP policies when aggregated across concepts, it performed relatively poorly for the range concept. This poor performance also occurred with the memoryless simulated learner, where the *maximum information* policy was significantly worse than the *memory* and *memoryless* POMDP policies (Friedman, comparison of time to mastery: *memoryless* versus *maximum information*, $\chi^2(1) = 55.4$, $p < .0001$; *discrete model with memory* versus *maximum information*, $\chi^2(1) = 65.0$, $p < .0001$). This discrepancy is due to the fact that the *maximum information* policy assumes that the learner remembers information, and narrows too quickly to only the endpoints of the range. We discuss this issue further in the results of Experiment 3. For both the simulated memory and memoryless learners, there was a significant effect of teaching policy on phases and time to mastery (Friedman: phases to mastery for memoryless learner, $\chi^2(5) = 108.6$, $p < .0001$; time to mastery for memoryless learner, $\chi^2(5) = 144.8$, $p < .0001$; phases to mastery for memory learner, $\chi^2(5) = 173.1$, $p < .0001$; time to mastery for memory learner, $\chi^2(5) = 296.9$, $p < .0001$). For the memory learner, the POMDP policies outperformed the two random control policies.[8] For the memoryless learner, the *memory* and *memoryless* POMDP policies outperformed the two random control policies for phases to mastery, and all POMDP policies outperformed the random control policies for time to mastery.[9] Overall, these simulations show that there is considerable variability in the best way to teach a concept depending on the characteristics of both the learner and the particular concept being taught. In this larger and more complex domain,

information, $\chi^2(1) = -3.3$, $p > .5$, *n.s.*. For time to mastery: *memoryless* versus *random*, $\chi^2(1) = 46.0$, $p < .0001$; *discrete model with memory* versus *random*, $\chi^2(1) = 25.3$, $p < .01$, *n.s.* given correction for multiple comparisons; *continuous* versus *random*, $\chi^2(1) = 75.7$, $p < .0001$; *memoryless* versus *quiz-example only*, $\chi^2(1) = 33.0$, $p < .0005$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 12.2$, $p > .1$, *n.s.*; *continuous* versus *quiz-example only*, $\chi^2(1) = 62.6$, $p < .0001$. The *maximum information* policy also outperformed the POMDP policies when measured by time, although again not significantly in the case of the *continuous* policy: *memoryless* versus *maximum information*, $\chi^2(1) = -35.6$, $p < .005$; *discrete model with memory* versus *maximum information*, $\chi^2(1) = -56.4$, $p < .0001$; *continuous* versus *maximum information*, $\chi^2(1) = -6.0$, $p > .1$, *n.s.*.

[8] As for the continuous simulated learner, planned pairwise comparisons were conducted. For teaching phases to mastery: *memoryless* versus *random*, $\chi^2(1) = 54.9$, $p < .0001$; *discrete model with memory* versus *random*, $\chi^2(1) = 49.9$, $p < .0001$; *continuous* versus *random*, $\chi^2(1) = 82.4$, $p < .0001$; *memoryless* versus *quiz-example only*, $\chi^2(1) = 68.9$, $p < .0001$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 64.9$, $p < .0001$; *continuous* versus *quiz-example only*, $\chi^2(1) = 96.4$, $p < .0001$. For time to mastery: *memoryless* versus *random*, $\chi^2(1) = 85.1$, $p < .0001$; *discrete model with memory* versus *random*, $\chi^2(1) = 88.2$, $p < .0001$; *continuous* versus *random*, $\chi^2(1) = 118.0$, $p < .0001$; *memoryless* versus *quiz-example only*, $\chi^2(1) = 85.2$, $p < .0001$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 88.3$, $p < .0001$; *continuous* versus *quiz-example only*, $\chi^2(1) = 118.1$, $p < .0001$.

[9] For teaching phases to mastery: *memoryless* versus *random*, $\chi^2(1) = 50.7$, $p < .0001$; *discrete model with memory* versus *random*, $\chi^2(1) = 57.1$, $p < .0001$; *continuous* versus *random*, $\chi^2(1) = 6.8$, $p > .2$, *n.s.*; *memoryless* versus *quiz-example only*, $\chi^2(1) = 62.5$, $p < .0001$; *discrete model with memory* versus *quiz-example only*, $\chi^2(1) = 68,9$, $p < .0001$; *continuous* versus *quiz-example only*, $\chi^2(1) = 18.6$, $p = .03$, *n.s.* given correction for multiple comparisons. For time to mastery: *memoryless* versus *random*, $\chi^2(1) = 81.9$, $p < .0001$; *discrete model with memory* versus *random*, $\chi^2(1) = 91.6$, $p < .0001$; *continuous* versus *random*, $\chi^2(1) = 47.6$, $p < .0001$; *memoryless* versus *quiz-example only*, $\chi^2(1) = 73.8$, $p < .0001$; *memory* versus *quiz-example only*, $\chi^2(1) = 83.5$, $p < .0001$; *continuous* versus *quiz-example only*, $\chi^2(1) = 39.6$, $p < .0001$.

mismatches between the computer teacher's assumptions and the learner can lead to suboptimal policies and slower learning.

6.12 Experiment 3: Teaching human learners the Number Game

We now turn to experimentally investigating how well the POMDP policies can teach number concepts to human learners.

Methods

Participants

A total of 360 participants were recruited from the University of California, Berkeley and received course credit for their participation.

Stimuli

Each participant learned one randomly chosen number concept. The possible number concepts were the same as in Simulation 3: multiples of seven; multiples of four minus one; and numbers between 64 and 83 (inclusive).

Procedure

The procedure was similar to that in Experiments 1 and 2. Participants in Experiment 3 learned only a single concept, and they were assigned to be taught teaching actions chosen based on one of the three POMDP policies or by one of the three control policies (*random*, *quiz-example only*, or *maximum information*). Pilot testing demonstrated that randomly chosen teaching actions were extremely frustrating for participants, making disengagement likely in these conditions, so we modified the *random* and *quiz-example only* policies to place higher probability on numbers within the concept. These policies first sampled whether to choose a number within the concept or a number outside of the concept, with equal probability on each of the two possibilities, and then sampled uniformly within the chosen class of numbers.

As in Experiment 1, participants alternated between teaching and assessment phases; these phases had the same structure as those used in Simulation 3. Within all phases, the numbers that the participant had seen, as well as any category information that had been shown, were displayed on-screen.

Computing policies

As described in Simulation 3, parameters to compute the POMDP policies were set based on data from the *random* condition. Mirroring Experiment 1, a three second delay was inserted between all actions in all conditions to allow time for planning; if a model did not return an action within three seconds, the search was interrupted and the best action found so far was returned.

Results and discussion

We analyzed the median number of phases and the median time on task for participants to learn the number concept, following the same methods as in Simulation 3. As shown in Figure 6.11, there was a main effect of teaching condition: Participants taught using one of the POMDP teaching policies spent less time on task (Friedman: $\chi^2(5) = 65.8$, $p < 0.001$) and required fewer teaching phases to learn the concept (Friedman: $\chi^2(5) = 50.7$, $p < 0.001$). Pairwise tests between each teaching policy and the *random* condition showed that each individual policy was more effective than the *random* policy, both in terms of time on task (Friedman: *memoryless* versus *random*, $\chi^2(1) = 27.5$, $p < 0.001$; *discrete model with memory* versus *random*, $\chi^2(1) = 18.7$, $p < 0.001$; *continuous* versus *random*, $\chi^2(1) = 23.5$, $p < 0.001$) and number of teaching phases (Friedman: *memoryless* versus *random*, $\chi^2(1) = 25.7$, $p < 0.001$; *discrete model with memory* versus *random*, $\chi^2(1) = 18.7$, $p < 0.001$; *continuous* versus *random*, $\chi^2(1) = 25.1$, $p < 0.001$). Unlike in Experiment 1, the *quiz-example only* policy was generally more effective than the *random* policy, and performed as well as the POMDP policies in some cases. However, this policy performed poorly compared to the POMDP policies for the range concept (numbers between 64 and 83), and aggregated across concepts, each POMDP policy in general performed better than the *quiz-example only* policy.

While the teaching policies taught learners more quickly overall than the two random policies, these POMDP policies were not significantly different from one another in terms of effectiveness and there was considerable variation in which policy was most effective for each concept, similar to the variation shown in Simulation 3. In Experiment 1, we also saw that the different models did not result in significantly different outcomes from one another. Experiment 3 introduces the additional difficulties of a more complex concept space in which we might expect learners to differ more from one another and from our assumed policies. For example, some learners might have little familiarity with modular arithmetic, and thus not consider concepts like "multiples of four minus one." This would make this concept more difficult to learn, and would likely cause the learner to exhibit different probabilities of particular knowledge state transitions than we assumed. These differences in assumptions about what concepts are possible would also lead to different learning behavior for other concepts, since our learner models assume that the space of possible concepts is known and fixed. Thus, the reason that a single policy is not consistently more effective than others may be due to mismatches between the true learner model and the model assumed by the policy.

This difference is likely also the reason that the *maximum information* policy performed so poorly. The continuous model underlying this policy was likely overconfident and estimated learners as learning more quickly than they actually did. While this may have hurt performance of the *continuous* POMDP policy, it was even more detrimental for the *maximum information* policy, an effect that also appeared when this policy was teaching the memoryless and memory learners in Simulation 3. The *continuous* POMDP policy rarely chose quiz actions, but these actions could never be chosen by the *maximum information* policy. It thus could not revise its beliefs about the learner's state. Additionally, once the model estimated that the participant had learned the concept, it would choose examples essentially at random, since no example was expected to change the learner's state, or focus on only a small subset of actions that were estimated as potentially state changing given the transition epsilons. Anecdotally, this was frustrating to participants, and may have lead to disengagement. This frustration was exacerbated in the *maximum information* condition compared to the two random conditions since those policies chose half of their numbers from within the concept and half from outside; the *maximum information* policy was more likely to choose numbers outside of the concept when sampling at random as fewer than half of the numbers were in the concept. For the range concept of numbers between 64 and 83, the policy only showed examples near the endpoints in later teaching phases (e.g., 63 and 64), since it estimated that learners would have ruled out all non-range concept based on the previous examples and there was still a small amount of probability mass on range concepts near the true concept (e.g., numbers between 63 and 83). However, these examples were frequently repeated, again leading to frustration and reflecting the overconfidence of the model.

Problems due to discrepancies between the assumed learner model and human learners are likely to be exacerbated by the cost structure in this experiment. As in Experiment 1, we set the costs of different action types based on the median time it took participants in the control condition to complete each of the action types. In Experiment 1, this resulted in quizzes being less costly than examples; conversely, in this experiment, examples were the least costly action. This resulted in the POMDP policies having relatively few non-example actions: As mentioned above, there were almost no quiz actions for participants taught using the *continuous* model, and the *discrete model with memory* had the most quiz actions at 6.6%. This means that the *continuous* policy may have been overly confident in its estimate of the learner's state, and that it did not gain information about when that estimate was inaccurate. This issue occurs because the policy assumes that the learner model is accurate; incorporating more uncertainty into the learner models might help to alleviate this problem. Additionally, one could modify the incentive structure to make quiz actions less costly when significant time has elapsed between information-seeking actions.

Examination of the teaching policies can shed light on how different learner models lead to different characteristic actions. All participants began with the same precomputed policy for the first four teaching phases (20 teaching actions). In cases where there was a quiz question, contingent policies were precomputed. A quiz question occurred only for the *memoryless* policy, as the fifteenth action for the concept "multiples of seven." For the multiples of seven concept, most of these beginning example actions showed examples of numbers in the concept (positive examples) for both the *memoryless* and the *discrete model with memory* policies (see Figure 6.12). The *continuous* policy showed half positive examples and half negative examples, with the first negative example appearing in the fourth teaching action. One reason for this may be that multiples of seven can be uniquely defined in the concept space using only a few positive examples; thus, for the *continuous* policy and the policy for the *discrete model with memory*, the learner is expected to learn the concept relatively quickly unless she does not update her state.

The concept "multiples of four minus one" was harder for participants to learn than the other concepts. The prior in our model, which was created in previous work (Tenenbaum, 2000), pre-

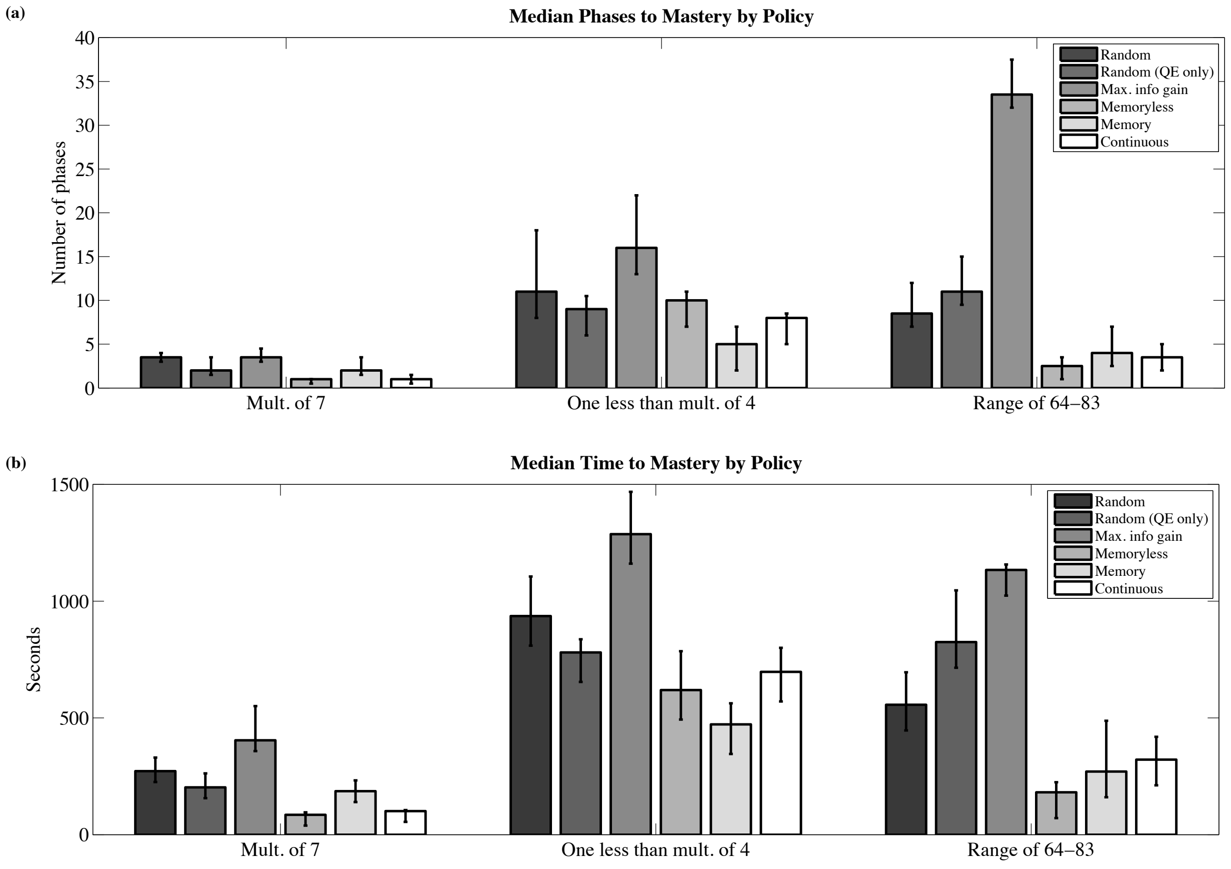


Figure 6.11: Median time (a) and phases (b) for participants to learn the concept in the Number Game. Participants taught using one of the POMDP policies are significantly faster than participants taught using a random policy.

dicts this result: this concept has substantially lower prior probability than the other concepts we considered. This concept may also be harder to learn due to the fact that it cannot be defined using only positive examples; all positive examples of this concept are also positive examples of "odd numbers." For all models, the examples shown are mainly odd numbers: in the first twenty examples, few even numbers are shown by both policies for the *continuous* model and *memoryless* model (three and four examples, respectively), and no even numbers are shown by the policy for the *discrete model with memory*. All three policies also show relatively few negative examples that are odd numbers: two for the *memoryless* model, one for the *discrete model with memory*, and three for the *continuous* model. While either of the policies based on discrete learners assume that the learner could learn the correct concept without negative examples of odd numbers, the continuous learner cannot converge on the correct concept unless some negative examples of odd numbers are shown.

Finally, for the range concept, all three policies again concentrate their actions on positive

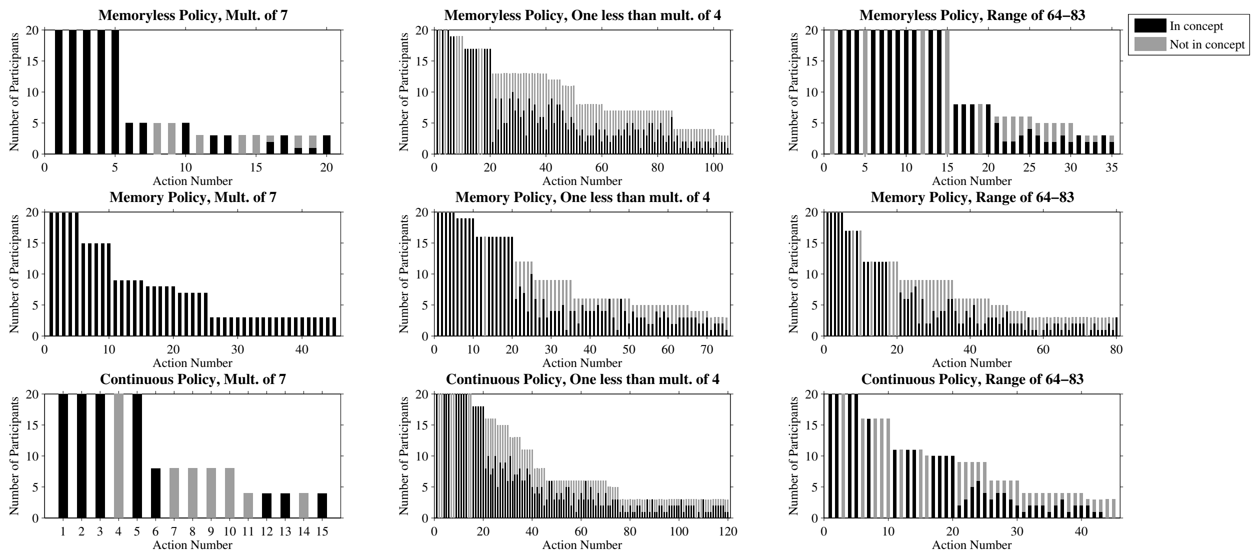


Figure 6.12: Number of participants where the action involved a number in or out of the concept at each time in the experiment, by concept and condition. Time points where there is data for at least three participants are shown. The graphs demonstrate that number choices differed both by what learner model was assumed and the concept that was taught.

examples and qualitatively, all three policies look relatively similar. The policies do not seem to focus as much on the endpoints of the range as one might intuitively expect. This may stem from the fact that the actions to choose from are sampled, so in most cases, the tutor may not have the option to show one of the endpoints (or a point just outside the endpoints). This points to the potential advantage of using a smarter sampling strategy for choosing which actions to consider, although the performance of the *maximum information* policy shows the potential downside of this approach when the tutor is too confident about the learner model.

Overall, the results from the Number Game demonstrate that POMDP policies can be effective for teaching a variety of concepts using the same computational framework. The policies that emerged for each model match what one might intuitively expect for teaching these number concepts, and also highlight how some characteristics of each model manifest in what patterns of choices are optimal. These results demonstrate that POMDPs can be more successful than even sophisticated strategies in the existing literature, such as policies using information gain.

6.13 General discussion

Automatically choosing pedagogical actions effectively is a complex task. It can be difficult to determine what choices will result in learning and to consider the tradeoffs of different decisions, especially given that one cannot directly observe the learner's knowledge. While previous chapters focused on how to make inferences about learners' knowledge, this chapter considers a key next step: what to do with information about a learners' knowledge, and how to decide when more information must be gathered. We have approached teaching as a decision problem in which a sequence of individual but interdependent pedagogical choices must be made. By framing teaching using the POMDP framework, we take into account the immediate and long term gains of each possible choice. This framework highlights the modular nature of the different components of the problem, such as the learner model and the pedagogical objective; the modularity of the framework allows the same general architecture to be applied to different educational domains. The POMDP framework allows one to determine an optimal teaching policy for a given domain and set of teaching materials with respect to a specified learning objective.

We have fleshed out how to apply the POMDP framework to teaching concepts and demonstrated the effectiveness of POMDP planning experimentally. This framework has not previously been fully explored for the general problem of teaching at the level of problem selection. We have developed this formulation such that it can be applied to a variety of teaching tasks by specifying the appropriate parameters. One of the potential advantages of this framework is that the produced policies automatically consider the utility of selecting instructional actions that aid learning versus diagnostic actions that result in a better estimate of the learner's state. Both of these types of actions occurred in the optimized policies, with their frequency varying based on the learner model and task. This suggests that at least in some cases, it is beneficial to monitor a learner's state and customize teaching based on that state estimate. This is consistent with results emphasizing the importance of adaptive, personalized guidance. When monitoring the learner's state, it is often useful to plan several steps into the future as monitoring actions are often not immediately useful for improving the learner's understanding; this planning is automatically incorporated with the POMDP formulation. In principle, monitoring of the student's current state could also be used to terminate teaching when the model has sufficient evidence that the student knows the mapping, rather than using assessment phases. The information necessary to decide whether to terminate can easily be retrieved from the belief state at any given time. However, the effectiveness of this criterion for terminating teaching can be very dependent on the accuracy of the underlying learner model. The experimental results showed that different learner models result in systematically different policies. This illustrates that optimal problem selection depends not only on knowledge of the domain but also on one's assumptions about the learner.

In the remainder of the chapter, I discuss how to incorporate other existing learner models into the POMDP framework. I then consider how learner models might be improved. Finally, I assess the computational limitations of our current work.

Incorporating existing models

One of the advantages of the POMDP framework is that it can be used to explore how assumptions about student learning affect optimal teaching policies. Many features of teaching, such as the need to sequence actions and the problem of diagnosing a learner's knowledge, are naturally integrated into POMDP planning, and the modular nature of this framework means that improvements in planning algorithms and improvements in learner models can be developed independently. While feature-based models that assume the student state is observable can also consider the implications of existing student models, additional work may be required to decide what features to use and it may be less natural to specify learner models without appealing to an unobserved student state. In future work, it would be useful to test the POMDP framework within an existing intelligent tutoring system and with student models that have been developed for particular educational domains. This would allow us to compare the differences between a POMDP policy and the current action selection policies within those systems.

There are many types of existing student models that one could use within the POMDP framework (e.g., Corbett & Anderson, 1995; Corbett & Bhatnagar, 1997; Li, Cohen, Koedinger, & Matsuda, 2011; Pardos, Heffernan, Anderson, & Heffernan, 2010). As mentioned in Chapter 3 (Section 3.6), there has been extensive work on designing student models in the intelligent tutoring systems community. POMDP planning is likely to work especially well with student models that assume students may make incorrect generalizations or that assume the effect of items on learning is contingent upon the current skill levels and with domains in which items may involve multiple skills. These types of models and domains predict that particular sequences of actions may be more beneficial than others, and that recognizing the students' current level of understanding, including particular misunderstandings, can lead to move effective pedagogical choices. For instance, Contextual Factors Analysis (Pavlik, Yudelson, & Koedinger, 2011) learns how different items contribute to particular skills, some of which may have transfer effects whereby practice on one type of item is beneficial to performance on other types of items due to the underlying skills that compose each type of item. One could learn a student model using this method, and then optimize problem selection for the learned model.

Additionally, the application of a formal framework like POMDPs can allow designers to determine the impact of changing the learner model on the optimal policy and on the expected time to mastery. Such an examination could allow one to determine which assumptions in the learner model are most crucial for effective instruction and which have few practical consequences. Lee and Brunskill (2012) examine this question with respect to whether individualized parameters in knowledge tracing would lead to substantially different numbers of practice opportunities for mastery. One can imagine similarly comparing the types of models discussed in Chapters 3 and 4, which attempt to model how students misunderstand, with learner models that model understanding as a binary; this could inform one's decision about the best type of model to use for a specific domain.

The POMDP framework can also be used as a way of comparing existing formulations for selecting teaching actions. Some action selection methods, such as choosing the action with the highest immediate expected utility, are approximations of the POMDP policy. By considering how these methods approximate the POMDP, one can use existing work to assess how close the approximation is to optimal (or whether it is optimal) and what simplifying assumptions are being made. This theoretical framework can thus help to unify existing methods, even in cases where using an optimal policy is impractical.

Improving learner models

Using POMDPs for teaching relies on having models of student learning for a given domain. These models are often hand-created and can be time consuming to construct, although recent research has made progress on constructing learner models from data (Barnes, 2005; Cen, Koedinger, & Junker, 2006; González-Brenes & Mostow, 2012). Even models learned from data, however, are often constrained to a particular structure that may not be appropriate for all types of tasks. Instead, we need a general approach for constructing probabilistic learner models that can be used within the POMDP framework; this issue echoes the problem of constructing the space of misunderstandings for the Bayesian inverse planning framework, consisting with understandings being equivalent to states in the learner models in this chapter.

The approach we have taken in this chapter is to build on work in cognitive science on probabilistic models of cognition. These probabilistic models have been successful in a variety of areas of cognition (Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010; Tenenbaum et al., 2006), including language (Chater & Manning, 2006; Seidenberg & MacDonald, 1999) and reasoning tasks (Hahn & Oaksford, 2007; Oaksford & Chater, 1994). We considered three learner models based on the literature, and fit the parameters of these models using human data. We then could use these models within the POMDP framework to derive optimal policies. Such an approach could be extended by further refining the models to better fit the data, reducing the number of assumptions that one must make about how learning occurs. The more closely that the learner model approximates human performance, the fewer issues that one is likely to have due to mismatches between the assumed model and human learners. Many of these probabilistic models have primarily been tested in laboratory settings, so further investigation to improve their fit to human performance should also explore how well these models generalize to more complex academic domains.

Previous work also suggests unique issues that one should consider when making learner models. For example, Walsh and Goschin (2012) prove a learning agent can learn more from less information if it is aware that it is being taught, rather than simply receiving information from the environment. Shafto, Goodman, and Griffiths (2014) found that human learners are similarly sensitive to whether examples are being provided by a teacher or being selected randomly. This has implications for what expectations to cultivate in the learner as well as for how one should model the learner's knowledge. Since Shafto et al. (2014) present a probabilistic model of learning, the implications of this theory of teaching can be directly determined by computing the optimal policy. Other research is likely to further expand what factors we consider in the learner model and to propose new theories about how people learn. By adopting a strategy in which the models in these theories are used as starting points that can be parameterized and fit to data, the time to create learner models can be reduced and we can benefit from increasing knowledge about how people learn.

Model limitations

In our use of POMDPs, we have focused on learner models that represent only the student's knowledge state, rather than considering all relevant factors, such as motivation and affect. In principle, POMDPs can use student models that account for these factors, which have received increasing attention in educational technologies (e.g., Conati & Maclaren, 2009; Robison, McQuiggan, & Lester, 2009). However, these factors are not automatically incorporated into the POMDP framework, and may necessitate a more complex objective function and cost structure. For example, one might want to maximize an objective function that incorporates both motivation and knowledge, while minimizing time; this is likely to make computation of the policy more difficult. One way to incorporate the learners' motivations and goals would be to assume that learners have their own reward functions; for instance, some learners may be more inclined to see what will happen if they try particular actions in a learning environment, while others simply want to complete the activity in minimal time. We saw evidence for such differences when interpreting players' actions in games in Chapter 5, and there has been extensive work on detecting "gaming" in intelligent tutoring systems, which occurs when a student is trying simply to get through material rather than to learn (R. S. Baker, Corbett, & Koedinger, 2004; Walonoski & Heffernan, 2006). In the POMDP framework, these differences could be incorporated as different observation models for relating the observed behaviors to the learners' knowledge; R. S. Baker et al. (2006) found that confronting students about gaming behavior was helpful for reducing gaming and increasing retention, suggesting that changing the pedagogical strategy in response to gaming can be effective. Again, increasing the space of observation models would increase the complexity of computing the POMDP policy. Regardless of how these additional factors are incorporated, they are likely to have important and interesting consequences for pedagogical action selection and to provide opportunities to explore more complex planning algorithms within the teaching domain.

Computational limitations

Computational challenges still exist for using POMDP planning: despite sampling only a fraction of possible actions and using very short horizons, planning took $2-3$ seconds per action. We suspect this challenge will apply to many other teaching tasks. In the simulations for alphabetic arithmetic, we did not find a great deal of improvement when increasing the number of actions sampled or the horizon beyond the limits we imposed in the experiments, but the effect of the computational approximation is likely to vary considerably based on the structure of the domain being taught. There are several possibilities for reducing the time to computer the POMDP policy as well as further improving the quality of the computed policy. Following ideas presented in prior POMDP planning algorithms (Ross, Pineau, et al., 2008), we believe that sampling actions based on the particular belief node in the tree would improve the search quality, as would using a more sophisticated evaluation function at the leaves. In particular, the evaluation function does not use the fact that failure on an assessment phase necessitates a complete additional teaching phase; this is relevant to any teaching problem in which assessments occur at fixed intervals. Finally, the relatively long horizons of this task and other teaching domains suggest that these problems may be better served by Monte Carlo Tree Search (MCTS) planning techniques, which have been very successful at producing good online policies long horizon planning problems such as the game Go (Gelly & Silver, 2007).

There are also cases where the computational challenges may be less severe due to the structure of the domain and the type of student model that is assumed. For example, many student models involve sets of independent skills that have either been mastered or not mastered (or more generally, where mastery is unidimensional; e.g., Corbett & Anderson, 1995). In models of this type where skills are not independent, additional skills can be added that cover the overlap between skills, resulting in a model where skills are independent. If each problem requires only a limited number of skills, then the change in student knowledge after each problem will be relatively localized, reducing the complexity of updating the belief state. Depending on the action and observation structure, it may even be the case that a greedy strategy is optimal, removing the need to plan ahead over multiple time steps (Karush & Dear, 1967). Yet, in many cases, there are a variety of actions with different tradeoffs in terms of the diagnostic benefit of an action versus the students' likely learning gain (and the type of learning gain; see Koedinger et al. (2012) for discussion of the interaction between types of pedagogical strategies and learning outcomes); in these cases, the myopic strategy of only looking one step ahead is generally suboptimal. Structured models are thus likely to reduce the computational challenges of planning with POMDPs, and in some cases may allow the use of offline methods to compute a policy that is close to optimal. However, for other domains and models, it is likely that computational limitations will have a continuing impact on the quality of POMDP policies for teaching. Even in these cases, framing pedagogical action selection within the POMDP framework allows one to benefit from continuing research on solving POMDPs as well as results concerning what approximations result in the best policies. Such an approach may thus be more scalable and easier to maintain than control policies that rely on heuristics or that are created by experts.

Conclusion

Deciding what pedagogical decisions to make involves reasoning about a number of different components, including the types of diagnoses developed in previous chapters, and balancing conflicting priorities. In this chapter, I addressed this issue by developing a framework for applying POMDP planning to teaching. This provided a way of conceptualizing how pedagogical action selection should depend on one's model of student learning and behavior, the structure of the domain, and one's pedagogical objective; this project thus extends our work on diagnosing learners' knowledge from their actions by incorporating the larger context in which this diagnosis might be used. The modularity of POMDPs is consistent with the goal introduced in Chapter 1 of developing approaches that can be applied to a variety of educational domains. We have shown that the POMDP framework can be used to select actions in real time in domains of moderate size, and demonstrated how despite mismatches between the assumed learner model and actual human learners, POMDP policies can lead to accelerated learning in a concept learning task. While engineering challenges remain, our results suggest that the formal specification of this framework can lead to both theoretical insights and practical improvements in selecting pedagogical actions.

Chapter 7

Discussion

In Chapter 1, I began with a discussion of the illustrated primer: a fictional educational "book" that provided personalized guidance to a learner and responded to her questions and behaviors intelligently. In the preceding chapters, I have described three new frameworks that seek to build the types of technologies that would be needed to create the primer. These frameworks combine probabilistic computational models and machine learning to address educational questions. I first focused on drawing inferences from observations of people's actions in games and interactive virtual environments. The Bayesian inverse planning model we developed allows one to diagnose people's understanding from their actions, using Markov decision processes to model the specific characteristics of the game or virtual environment in which the person is acting. Like the illustrated primer, the model does not need to artificially structure the educational tasks in order to use the learners' behavior to make fine-grained inferences. I then turned to how to design games and environments to be more diagnostic, attempting to maximize the information that is gained about people's understanding and cognitive processes. Both of these frameworks were primarily concerned with assessment. In the final chapter of this dissertation, I addressed the question of how a computer-based educational technology should choose pedagogical activities, using both the results of assessments and information about the typical course of learning in a domain.

In all of these investigations, I focused on developing and adapting machine learning algorithms and statistical models. Because the underlying models in these frameworks are areas of ongoing research, future work on the frameworks can take advantage of new advances in speed and scalability, increasing their potential impact. This approach also provides a way of better understanding existing work on educational technologies. For example, different approaches to pedagogical action selection can be viewed as approximations or heuristics for solving a POMDP. By formalizing the problems under investigation, the approach I have taken provides a principled way of determining the generalizability of findings from different systems, and predicting when a particular approach will be more or less effective.

I now turn to a broader discussion of these algorithms and the experiments we have conducted. I first address some of the limitations that apply to our results, focusing on those issues that occurred in multiple investigations. I then discuss future directions for continuing research.

Limitations

The approach I have taken to making fine-grained inferences about people's understanding relies upon detailed models of people's learning and behavior. This approach allows the work to be applied across domains and to make use of the extensive modeling research in cognitive science, psychology, and education, but also means that there can be significant consequences if people diverge from our assumptions. In Chapter 4, we showed that some participants' algebra skills could not be accurately evaluated because they relied on actions that were not a part of our model, and in Chapter 5, we saw an example of how motivations that were not accounted for by the model could affect the accuracy of our predictions. There are several ways to address this limitation. First, continued testing and development of computational models is needed. As more accurate models are developed, the frameworks we have created can take advantage of these improvements. Our frameworks could also be used to recognize when a model may make incorrect assumptions or be inconsistent with human behavior. For example, we used the planning noise parameter to identify participants whose algebra data may have been poorly fit by our model; analysis of these people's behavior suggests ways to modify the model to improve its ability to model all participants. These applications can thus provide a feedback loop for improving cognitive models and highlighting potentially erroneous assumptions.

I have also shown evidence that not all deviations between our assumptions and the true characteristics of human learning are detrimental in practice. When using POMDPs for pedagogical action selection, we saw that learner models with very different assumptions were still effective at improving learning efficiency over a baseline policy. Because of these different assumptions, we know that it is not possible for all of the learner models to accurately reflect human cognition, meaning that some inaccuracy can be tolerated. This is encouraging since we would not expect any model to fully capture the intricacies of human learning, and it may be beneficial for computational reasons to create simpler models. We thus need not be strongly committed to our learner models being completely accurate reflections of human cognition, but must be aware of the potential for inaccuracies and recognize which objectives are particularly sensitive to these inaccuracies. In extending these frameworks, we hope to explore how to quantify the robustness of the model to deviations in learner behavior and further develop methods for identifying when a model is a poor fit for some or all learners.

Computational issues were another recurring theme in our explorations. Computing exact solutions to POMDPs with large state or action spaces is generally intractable. While MDPs are more efficient to solve, approximations must be used with continuous state or action spaces, as we saw in our application to algebra understanding. Computational tractability is likely to be a continuing concern as we move towards more adaptive systems based on complex learner models. However, this concern can frequently be addressed through modeling choices and balancing accuracy with efficiency of computation. POMDP planning becomes more tractable with certain types of structured models. Since our experiments provide evidence that POMDPs can be effective even when their assumptions about learners are not completely accurate, it may be beneficial to target exploration towards learner models that are efficient to solve or to relax existing learner models to meet the structured assumptions. Simulations can be conducted to determine the effect of these relaxations on learning outcomes if the original models were in fact more accurate. For knowledge diagnosis, the most costly part of the computation is computing the Q-functions. Caching these solutions and then adjusting the posterior approximation procedure to take advantage of these functions (rather than resampling from the continuous space) would result in much faster online computation of the diagnosis for an individual learner. The advantage of this approach is that the task may still vary to some degree across learners; for example, the algebra problems to solve need not be constant. The use of MCMC sampling methods also provides a natural way to balance accuracy and efficiency: fewer samples will likely produce less accurate and more variable results, but these results can be computed more quickly. In applications like the online algebra tutor, results with a small number of samples could be used for minor adjustments to the problems provided. A more accurate diagnosis could be computed after a set number of problems has been solved by a user, and after this diagnosis has been calculated, a larger intervention could be provided, such as linking the learner to existing materials and offering opportunities for scaffolded problem solving related to the misunderstandings. Such an approach aims to personalize the website as quickly as possible while only suggesting time-consuming interventions when significant evidence for their applicability is available. It thus takes into account both educational objectives and learners' individual experiences in determining how to cope with limitations of computational power.

Future Directions

This work begins from a theoretical standpoint of how to evaluate and respond to learners' behavior given particular models of cognition, focusing on developing approaches that can be applied across domains. While we have provided experimental evidence for the effectiveness of our frameworks, many of our applications are not yet as sophisticated as typical classroom material. One key next step is thus to apply these frameworks to a large, complex educational domain, providing an opportunity to investigate the frameworks' performance when teaching and assessing a much broader topic. The online algebra tutor provides a platform for beginning this investigation. Through continued experiments on linear equation solving and other algebra topics, we will be able to test whether the diagnoses from Bayesian inverse planning are consistent with conventional assessment. Additionally, the website allows us to reach a much wider range of learners: we can encourage students and other algebra learners to come to the website to improve their understanding, and test the effectiveness of interventions that are customized based on their problem solving performance. This platform also offers the potential to connect learners to existing resources; free online educational content is widespread, but it can be difficult for learners to recognize what content would be most helpful to them. Our customized interventions can bridge this gap.

Another way to test our frameworks in typical educational domains is to incorporate our models into existing systems. As we have touched on throughout this dissertation, there has been considerable work on designing learner models in the intelligent tutoring systems community, especially for structured domains. By combining these models with the frameworks we have proposed, we can explore both theoretical and practical questions. For instance, the probabilistic models of cognition that we have focused on are generally finer-grained than models used in intelligent tutoring systems; these models represent how learners might misunderstand rather than representing only whether a skill has been mastered. By comparing the optimal policies for each type of model, we could explore when representing misunderstandings has benefits, and when simpler models are effective. Combining existing curricula as already realized in instructional systems with our algorithms would facilitate broader exploration of the impact of personalization on learning, providing a way to make sense of the rich behaviors that learners' exhibit in many interactive activities.

In addition to considering how to apply our frameworks to typical educational domains, an important challenge is how to apply these frameworks to incorporate broader conceptions of understanding. We have primarily used assessments of understanding focused on accuracy. For instance, in Chapter 6, we assessed whether participants could identify an arithmetic mapping or accurately state whether numbers were part of the targeted concept. Incorporating more constructivist assessments and definitions of mastery that focus on understanding and integration of concepts would allow us to evaluate the effectiveness of our frameworks for learning objectives beyond accuracy. For example, rather than focusing on success in an individual task, we could take into consideration how likely students are to be able to transfer their knowledge; we discussed formulating such objectives in Chapter 6. Other learning objectives might be related to metacognition or integration of new ideas into a complete understanding. To consider these issues within our frameworks will require formulating them computationally. For instance, the learner model would need to take into account interactions between metacognitive skills and domain-specific skills and care would be required to determine an efficient way to represent the possible learner states. Despite these challenges, we believe our current approaches can be adapted to diagnose these more complex versions of understanding. Bayesian inverse planning can interpret students' freeform interactions with activities, which may carry rich information about their problem-solving abilities. The planning parameter we estimated in our assessment of algebra understanding (Chapter 4) could be interpreted as relating to metacognitive abilities, and in future research, we will investigate whether planning abilities are consistent across domains. Extending our frameworks to model many types of understanding is likely to lead to more engaging and effective activities and to clarify our current representations of knowledge.

Two of our frameworks bear on the question of how to personalize guidance to learners. Bayesian inverse planning can be used to identify misunderstandings to target with guidance (as in Experiment 3 of Chapter 3), and POMDP planning provides a way of choosing which guidance will be most helpful over the long term. However, designing guidance (and instructional materials in general) is an important and complex task. Further testing is needed to establish what types of guidance are most helpful, and to incorporate these findings into our models of learners. Design of guidance is an active area of research, although a review of the literature reveals some practices that tend to be more effective (see Shute, 2008, for an overview). There is considerable variation in which learners benefit most for different types of guidance, however. For instance, learners with lower prior knowledge tend to benefit from more specific guidance (Shute, 2008) and different types of scaffolding (Razzaq & Heffernan, 2009). These results suggest the potential for pedagogical action selection methods like the POMDP to improve student learning beyond simple heuristics that focus only on skills that have not yet been mastered.

Conclusion

Educational technologies offer significant promise: they can decrease the burden on teachers, who are tasked with instructing ever greater numbers of students; facilitate life-long learning; and provide access to material to diverse learners. In principle, these technologies can enable educational experiences that are more scalable and personalized to maximize individual learning. However, the abilities of current technologies are still limited, falling short of the flexibility and seamlessness of fictional inventions like the illustrated primer of *Diamond Age*. In the investigations in this dissertation, I have focused on developing computational frameworks that can be used to systematically create activities that are more intelligent and responsive to learners' behavior. My goal has been to create frameworks that allow learners the freedom to make choices in authentic environments while still closely monitoring their understanding. While the models developed here have not yet been applied in classrooms, the laboratory and online studies we have conducted provide support for their flexibility, accuracy, and effectiveness. By combining Bayesian probabilistic models of cognition and machine learning algorithms, I address questions relevant to education, cognitive science, and computer science. The work contributes both to creating practical frameworks for supporting learners and to better understanding how people learn and act based on their understanding, bringing us closer to the vision of the illustrated primer.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning ICML.* ACM Press.

Ajo-Franklin, J. B. (2009). Optimal experiment design for time-lapse traveltime tomography. *Geophysics*, *74*(4), Q27–Q40.

Amir, O., & Gal, Y. (2011). Plan recognition in virtual laboratories. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)* (p. 2392-2397).

Anderson, J. R. (1993). Problem solving and learning. *American Psychologist*, *48*(1), 35.

Andrade, G., Ramalho, G., Santana, H., & Corruble, V. (2005). Extending reinforcement learning to provide dynamic game balancing. In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th IJCAI* (pp. 7–12).

Åström, K. (1965). Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 174–205.

Atkinson, A., Donev, A., & Tobias, R. (2007). *Optimum experimental designs, with SAS* (Vol. 34). New York: Oxford University Press.

Atrash, A., & Pineau, J. (2006). Efficient planning and tracking in POMDPs with large observation spaces. In *AAAI-06 Workshop on Empirical and Statistical Approaches for Spoken Dialogue Systems.*

Baker, C. L., Saxe, R. R., & Tenenbaum, J. B. (2009). Action understanding as inverse planning. *Cognition*, *113*(3), 329–349.

Baker, C. L., Saxe, R. R., & Tenenbaum, J. B. (2011). Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (pp. 2469–2474).

Baker, C. L., Tenenbaum, J. B., & Saxe, R. R. (2006). Bayesian models of human action understanding. *Advances in Neural Information Processing Systems (NIPS)*, *18*, 99-106.

Baker, R. S., Corbett, A. T., & Koedinger, K. R. (2004). Detecting student misuse of intelligent tutoring systems. In *Proceedings of the 6th International Conference on Intelligent Tutoring Systems* (pp. 531–540).

Baker, R. S., Corbett, A. T., Koedinger, K. R., Evenson, S., Roll, I., Wagner, A. Z., ... Beck, J. E. (2006). Adapting to when students game an intelligent tutoring system. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems* (pp. 392–401).

Barnes, T. (2005). The Q-matrix method: Mining student response data for knowledge. In *Proceedings of AAAI 2005 Educational Data Mining Workshop.*

Barnes, T., & Stamper, J. (2008). Toward automatic hint generation for logic proof tutoring using historical student data. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (p. 373-382). Springer.

Becchio, C., Manera, V., Sartori, L., Cavallo, A., & Castiello, U. (2012). Grasping intentions: from thought experiments to empirical evidence. *Frontiers in Human Neuroscience*, *6*.

Beck, J. E., & 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.

Bellman, R. E. (1957). *Dynamic programming*. Princeton, NJ, USA: Princeton University Press.

Birenbaum, M., Kelly, A., & Tatsuoka, K. (1993). Diagnosing knowledge states in algebra using the rule-space model. *Journal for Research in Mathematics Education*, 442–459.

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

Breslow, L., Pritchard, D. E., DeBoer, J., Stump, G. S., Ho, A. D., & Seaton, D. (2013). Studying learning in the worldwide classroom: Research into edx's first MOOC. *Research & Practice in Assessment*, *8*, 13–25.

Brown, J., & VanLehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, *4*(4), 379–426.

Bruno, R., Hille, D., Riva, A., Vivier, N., ten Bokkel Huinnink, W. W., van Oosterom, A. T., ... Sheiner, L. (1998). Population pharmacokinetics/pharmacodynamics of docetaxel in phase II studies in patients with cancer. *Journal of Clinical Oncology*, *16*(1), 187–196.

Brunskill, E., Garg, S., Tseng, C., Pal, J., & Findlater, L. (2010). Evaluating an adaptive multi-user educational tool for low-resource regions. In *Proceedings of the International Conference on Information and Communication Technologies and Development.*

Brunskill, E., & Russell, S. (2010). RAPID: A reachable anytime planner for imprecisely-sensed domains. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence.*

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*. Boca Raton, FL: CRC Press.

Cassandra, A. (1998). A survey of POMDP applications. In *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes* (pp. 17–24).

Cavagnaro, D. R., Myung, J. I., Pitt, M. A., & Kujala, J. V. (2010). Adaptive design optimization: A mutual information-based approach to model discrimination in cognitive science. *Neural Computation*, *22*(4), 887–905.

Cen, H., Koedinger, K. R., & Junker, B. (2006). Learning factors analysis - A general method for cognitive model evaluation and improvement. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (pp. 164–175). Springer.

Chaloner, K., & Verdinelli, I. (1995). Bayesian experimental design: A review. *Statistical Science*, *10*(3), 273–304.

Chang, K., Beck, J., Mostow, J., & Corbett, A. (2006). A Bayes net toolkit for student modeling in intelligent tutoring systems. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (pp. 104–113). Springer.

Chater, N., & Manning, C. D. (2006). Probabilistic models of language processing and acquisition. *Trends in Cognitive Sciences*, *10*(7), 335-344.

Chater, N., Tenenbaum, J. B., & Yuille, A. (2006). Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences*, *10*(7), 287–291.

Chi, M., Jordan, P., VanLehn, K., & Hall, M. (2008). Reinforcement learning-based feature selection for developing pedagogically effective tutorial dialogue tactics. In *Proceedings of the 1st International Conference on Educational Data Mining.*

Common Core State Standards Initiative. (2010). *Common core state standards for mathematics*. Washington, DC: National Governors Association Center for Best Practices and the Council of Chief State School Officers.

Conati, C., & Maclaren, H. (2009). Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction*, *19*(3), 267–303.

Conati, C., & Muldner, K. (2007). Evaluating a decision-theoretic approach to tailored example selection. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (pp. 483–488).

Corbett, A. T. (2001). Cognitive computer tutors: Solving the two-sigma problem. In M. Bauer, P. Gmytrasiewicz, & J. Vassileva (Eds.), *User modeling 2001* (Vol. 2109, pp. 137–147). Springer.

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, *4*(4), 253–278.

Corbett, A. T., & Bhatnagar, A. (1997). Student modeling in the ACT programming tutor: Adjusting a procedural learning model with declarative knowledge. In *Proceedings of the 6th International Conference on User Modeling* (pp. 243–254). New York.

Dantas, L., Orlande, H., & Cotta, R. (2002). Estimation of dimensionless parameters of Luikov's system for heat and mass transfer in capillary porous media. *International Journal of Thermal Sciences*, *41*(3), 217–227.

Davis, E. A., Linn, M. C., & Clancy, M. (1995). Learning to use parentheses and quotes in LISP. *Computer Science Education*, *6*(1), 15–31.

Derlinden, E. V., Bernaerts, K., & Impe, J. V. (2010). Simultaneous versus sequential optimal experiment design for the identification of multi-parameter microbial growth kinetics as a function of temperature. *Journal of Theoretical Biology*, *264*(2), 347–355.

Desmarais, M. C., & Baker, R. S. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, *22*(1-2), 9–38.

Doucet, A., de Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo methods in practice.* New York: Springer.

Duschl, R. A., Schweingruber, H. A., & Shouse, A. W. (2007). *Taking science to school: Learning and teaching science in grades K-8.* National Academies Press.

Elvind, D., Asmund, H., & Rolf, V. (1992). Maximum information at minimum cost: A north sea field development study with an experimental design. *Journal of Petroleum Technology*, *44*(12), 1350–1356.

Emery, A. F., Nenarokomov, A. V., & Fadale, T. D. (2000). Uncertainties in parameter estimation: The optimal experiment design. *International Journal of Heat and Mass Transfer*, *43*(18), 3331–3339.

Erev, I., & Roth, A. (1998). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, *88*(4), 848–881.

Falmagne, J.-C., Cosyn, E., Doignon, J.-P., & Thiéry, N. (2006). The assessment of knowledge, in theory and in practice. In *Formal concept analysis* (Vol. 3874, pp. 61–79). Springer.

Falmagne, J.-C., & Doignon, J.-P. (2011). *Learning spaces*. Springer.

Feinberg, E., Shwartz, A., & Altman, E. (2002). *Handbook of Markov decision processes: Methods and applications*. Kluwer Academic Publishers.

Feldman, J. (2000). Minimization of Boolean complexity in human concept learning. *Nature*, *407*, 630-633.

Feng, M., Heffernan, N., & Koedinger, K. R. (2009). Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction*, *19*, 243-266.

Folsom-Kovarik, J., Sukthankar, G., Schatz, S., & Nicholson, D. (2010). Scalable POMDPs for diagnosis and planning in intelligent tutoring systems. In *AAAI Fall Symposium on Proactive Assistant Agents.*

Fu, W.-T., & Gray, W. D. (2006). Suboptimal tradeoffs in information seeking. *Cognitive Psychology*, *52*(3), 195–242.

Fu, W.-T., & Pirolli, P. (2007). SNIF-ACT: A cognitive model of user navigation on the world wide web. *Human–Computer Interaction*, *22*(4), 355–412.

Fujiwara, M., Nagy, Z. K., Chew, J. W., & Braatz, R. D. (2005). First-principles and direct design approaches for the control of pharmaceutical crystallization. *Journal of Process Control*, *15*(5), 493–504.

Gal, Y., Yamangil, E., Shieber, S., Rubin, A., & Grosz, B. (2008). Towards collaborative intelligent tutors: Automated recognition of users' strategies. In B. Woolf, E. Aïmeur, R. Nkambou, & S. Lajoie (Eds.), *Intelligent tutoring systems* (Vol. 5091, p. 162-172). Springer Berlin / Heidelberg.

Gelly, S., & Silver, D. (2007). Combining online and offline learning in UCT. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 273–280).

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721-741.

Gilks, W., Richardson, S., & Spiegelhalter, D. J. (Eds.). (1996). *Markov chain Monte Carlo in practice*. Suffolk, UK: Chapman and Hall.

González-Brenes, J. P., & Mostow, J. (2012). Dynamic cognitive tracing: Towards unified discovery of student and cognitive models. In *Proceedings of the Fifth International Conference on Educational Data Mining* (pp. 49–56).

González-Brenes, J. P., & Mostow, J. (2013). What and when do students learn? Fully data-driven joint estimation of cognitive and student models. In *Proceedings of the 6th International Conference on Educational Data Mining* (pp. 236–239).

Goodman, N. D., Baker, C. L., & Tenenbaum, J. B. (2009). Cause and intent: Social reasoning in causal learning. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society* (p. 2759-2764).

Gough, J. (2004). Algebra skills and traps and diagnostic teaching for the future. *Australian Senior Mathematics Journal*, *18*(2), 43–54.

Grèzes, J., Frith, C., & Passingham, R. E. (2004). Inferring false beliefs from the actions of oneself and others: an fMRI study. *Neuroimage*, *21*(2), 744–750.

Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., & Tenenbaum, J. B. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in Cognitive Sciences*, *14*(8), 357–364.

Griffiths, T. L., Christian, B. R., & Kalish, M. L. (2008). Using category structures to test iterated learning as a method for identifying inductive biases. *Cognitive Science*, *32*, 68-107.

Groen, G., & Atkinson, R. C. (1966). Models for optimizing the learning process. *Psychological Bulletin*, *66*(4), 309.

Hahn, U., & Oaksford, M. (2007). The rationality of informal argumentation: A Bayesian approach to reasoning fallacies. *Psychological Review*, *114*(3), 704.

Haines, L. M., Perevozskaya, I., & Rosenberger, W. F. (2003). Bayesian optimal designs for phase I clinical trials. *Biometrics*, *59*(3), 591–600.

Hatano, G., Amaiwa, S., & Inagaki, K. (1996). "Buggy algorithms" as attractive variants. *The Journal of Mathematical Behavior*, *15*(3), 285–302.

Hennessy, S. (1994). The stability of children's mathematical behavior: When is a bug really a bug? *Learning and Instruction*, *3*(4), 315–338.

Hoey, J., Poupart, P., Boutilier, C., & Mihailidis, A. (2005). POMDP models for assistive technology. In *Proceedings of the AAAI Fall Symposium on Caring Machines: AI in Eldercare.*

Hu, C., Lovejoy, W., & Shafer, S. (1996). Comparison of some suboptimal control policies in medical drug therapy. *Operations Research*, *44*(5), 696–709.

Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99-134.

Karush, W., & Dear, R. (1967). Optimal strategy for item presentation in a learning process. *Management Science*, *13*(11), 773–785.

Kautz, H., & Allen, J. F. (1986). Generalized plan recognition. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 32–37).

Khan, S. (2006-2014). *Khan Academy.* `http://www.khanacademy.org`.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Kluger, A., & DeNisi, A. (1996). The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, *119*(2), 254.

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

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, *36*(5), 757–798.

Koedinger, K. R., & MacLaren, B. A. (2002). *Developing a pedagogical domain theory of early algebra problem solving* (Tech. Rep. No. CMU-HCII-02-100). Carnegie Mellon University.

Kolowich, S. (2013). The professors who make the MOOCs. *Chronicle of Higher Education*, *59*(28), A20–A23.

Kramer, A. (2010-2014). *Symja library - Java symbolic math system.* `https://bitbucket.org/axelclk/symja_android_library/wiki/Home`.

Kujala, J., Richardson, U., & Lyytinen, H. (2008). A Bayesian-optimal principle for child-friendly adaptation in learning games. *Journal of Mathematical Psychology*, *54*(2), 247–255.

Kurniawati, H., Hsu, D., & Lee, W. (2008). SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems IV*.

Leacock, C., & Chodorow, M. (2003). C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, *37*(4), 389–405.

Lee, J. I., & Brunskill, E. (2012). The impact on individualizing student models on necessary practice opportunities. In *Proceedings of the 4th International Conference on Educational Data Mining* (pp. 118–125).

Lesh, N., Rich, C., & Sidner, C. L. (1999). Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modeling* (pp. 23–32).

Levine, M. (1970). Human discrimination learning: The subset-sampling assumption. *Psychological Bulletin*, *74*(6), 397–404.

Li, N., Cohen, W. W., Koedinger, K. R., & Matsuda, N. (2011). A machine learning approach for automatic student model discovery. In *Proceedings of the 4th International Conference on Educational Data Mining* (pp. 31–40).

Linn, M. C., Lee, H.-S., Tinker, R., Husic, F., & Chiu, J. L. (2006). Teaching and assessing knowledge integration in science. *Science*, *313*(5790), 1049-1050.

Liu, T.-C., Lin, Y.-C., & Kinshuk. (2010). The application of Simulation-Assisted Learning Statistics (SALS) for correcting misconceptions and improving understanding of correlation. *Journal of Computer Assisted Learning*, *26*(2), 143–158.

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, *4*(4), 590–604.

Martin, F. G. (2012). Will massive open online courses change how we teach? *Communications of the ACM*, *55*(8), 26–28.

Matz, M. (1982). Towards a process model for high school algebra errors. In D. Sleeman & J. Brown (Eds.), *Intelligent tutoring systems* (pp. 25–50).

Meisenhelder, S. (2013). MOOC mania. *Thought & Action*, 7–26.

Mintzes, J. J., Wandersee, J. H., & Novak, J. D. (2005). *Assessing science understanding: A human constructivist view*. Academic Press.

Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). On the structure of educational assessments. *Measurement: Interdisciplinary research and perspectives*, *1*(1), 3–62.

Monahan, G. (1982). A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 1–16.

Murray, R., VanLehn, K., & Mostow, J. (2004). Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence in Education*, *14*(3), 235–278.

Myung, J., & Pitt, M. (2009). Optimal experimental design for model discrimination. *Psychological Review*, *116*(3), 499–518.

Nelson, J., & Movellan, J. (2001). Active inference in concept learning. In *Advances in Neural Information Processing Systems 13* (pp. 45–51).

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 663–670).

Nicaud, J., Chaachoua, H., & Bittar, M. (2006). Automatic calculation of students' conceptions in elementary algebra from Aplusix log files. In *Intelligent tutoring systems* (pp. 433–442).

Nosofsky, R. M., Gluck, M., Palmeri, T. J., McKinley, S. C., & Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory & Cognition*, *22*, 352-369.

Oaksford, M., & Chater, N. (1994). A rational analysis of the selection task as optimal data selection. *Psychological Review*, *101*, 608-631.

Oxoby, R. J. (2006). Experiments and behavioral economics. *Handbook of Contemporary Behavioral Economics*, 441–454.

Pappano, L. (2012). The year of the MOOC. *The New York Times*, *2*(12), 2012.

Pardos, Z. A., Heffernan, N. T., Anderson, B., & Heffernan, C. L. (2010). Using fine-grained skill models to fit student performance with Bayesian networks. In C. Romero, V. S., M. Pechenizkiy, & R. S. Baker (Eds.), *Handbook of educational data mining* (pp. 417–426). CRC Press.

Patel, D., Fleming, S., & Kilner, J. (2012). Inferring subjective states through the observation of actions. *Proceedings of the Royal Society B: Biological Sciences*, *279*(1748), 4853–4860.

Pavlik, P., Yudelson, M., & Koedinger, K. R. (2011). Using contextual factors analysis to explain transfer of least common multiple skills. In *Proceeding of the 15th International Conference on Artificial Intelligence in Education* (pp. 256–263).

Payne, S., & Squibb, H. (1990). Algebra mal-rules and cognitive accounts of error. *Cognitive Science*, *14*(3), 445–481.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., & Thrun, S. (2003). Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, *42*(3), 271–281.

Polson, M. C., & Richardson, J. J. (2013). *Foundations of intelligent tutoring systems*. Psychology Press.

Puglisi, A., Baronchelli, A., & Loreto, V. (2008). Cultural route to the emergence of linguistic categories. *Proceedings of the National Academy of Sciences*, *105*(23), 7936–7940.

Pukelsheim, F. (2006). *Optimal design of experiments* (Vol. 50). Society for Industrial Mathematics.

Puterman, M. L. (2005). *Markov decision processes: Discrete stochastic dynamic programming*. Hoboken, NJ: Wiley.

Rafferty, A. N., Brunskill, E., Griffiths, T. L., & Shafto, P. (2011). Faster teaching by pomdp planning. In *Proceedings of the 15th Annual International Conference on Artificial Intelligence in Education* (pp. 280–287). Springer.

Rafferty, A. N., LaMar, M. M., & Griffiths, T. L. (in press). Inferring learners' knowledge from their actions. *Cognitive Science*.

Rafferty, A. N., Zaharia, M., & Griffiths, T. L. (2012). Optimally designing games for cognitive science research. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society* (pp. 893–898).

Ramachandran, D., & Amir, E. (2007). Bayesian inverse reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)* (p. 2586-2591).

Ratcliffe, S. J., & Shults, J. (2008). GEEQBOX: A MATLAB toolbox for generalized estimating equations and quasi-least squares. *Journal of Statistical Software*, *25*(14), 1–14.

Razzaq, L. M., Feng, M., Nuzzo-Jones, G., Heffernan, N. T., Koedinger, K. R., Junker, B., ... Rasmussen, K. (2005). The ASSISTment project: Blending assessment and assisting. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education* (pp. 555–562).

Razzaq, L. M., Heffernan, N., Feng, M., & Pardos, Z. (2007). Developing fine-grained transfer models in the Assistment system. *Journal of Technology, Instruction, Cognition, and Learning*, *5*(3), 289–304.

Razzaq, L. M., & Heffernan, N. T. (2009). To tutor or not to tutor: That is the question. In *Proceeding of the 14th International Conference on Artificial Intelligence in Education* (pp. 457–464).

Restle, F. (1962). The selection of strategies in cue learning. *Psychological Review*, *69*(4), 329–343.

Robison, J., McQuiggan, S., & Lester, J. (2009). Evaluating the consequences of affective feedback in intelligent tutoring systems. In *Affective Computing and Intelligent Interaction and Workshops* (pp. 1–6).

Ross, S. M., Chaib-draa, B., & Pineau, J. (2008). Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA).* IEEE.

Ross, S. M., Pineau, J., Paquet, S., & Chaib-draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, *32*(1), 663–704.

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (pp. 93–100).

Russell, S. (1998). Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (pp. 101–103).

Seidenberg, M. S., & MacDonald, M. C. (1999). A probabilistic constraints approach to language acquisition and processing. *Cognitive Science*, *23*(4), 569–588.

Shafto, P., Goodman, N. D., & Griffiths, T. L. (2014). A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive Psychology*, *71*, 55–89.

Shannon, C., & Weaver, W. (1948). The mathematical theory of computation. *Bell. Sys. Tech. J*, *27*, 379–423.

Shepard, R. N., Hovland, C. I., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs*, *75*. (13, Whole No. 517)

Shin, S., Han, S., Lee, W., Im, Y., Chae, J., Lee, D.-i., ... Urban, Z. (2007). Optimize terephthaldehyde reactor operations. *Hydrocarbon Processing*, *86*(4), 83–90.

Shute, V. (2008). Focus on formative feedback. *Review of Educational Research*, *78*(1), 153–189.

Shute, V. (2011). Stealth assessment in computer-based games to support learning. In S. Tobias & J. D. Fletcher (Eds.), *Computer Games and Instruction* (pp. 503–524). Charlotte, NC: Information Age Publishing.

Simon, R. (1989). Optimal two-stage designs for phase II clinical trials. *Controlled Clinical Trials*, *10*(1), 1–10.

Siorpaes, K., & Hepp, M. (2008). Games with a purpose for the semantic web. *Intelligent Systems*, *23*(3), 50–60.

Sleeman, D. (1984). An attempt to understand students' understanding of basic algebra. *Cognitive Science*, *8*(4), 387–412.

Sleeman, D. (1985). Basic algebra revisited: A study with 14-year-olds. *International Journal of Man-Machine Studies*, *22*(2), 127–149.

Sleeman, D., Kelly, A. E., Martinak, R., Ward, R. D., & Moore, J. L. (1989). Studies of diagnosis and remediation with high school algebra students. *Cognitive Science*, *13*(4), 551–568.

Sondik, E. J. (1971). *The optimal control of partially observable Markov processes*. Unpublished doctoral dissertation, Stanford University.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Van Der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *64*(4), 583–639.

Stacey, K., & Steinle, V. (1999). Understanding decimals: The path to expertise. In *Making the difference: Proceedings of the 22nd Annual Conference of the Mathematics Education Research Group of Australasia Incorporated* (pp. 446–453).

Stephenson, N. (1995). *The diamond age: Or, a young lady's illustrated primer*. Bantam Spectra.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning*. MIT Press.

Taber, K. (2000). Multiple frameworks?: Evidence of manifold conceptions in individual cognitive structure. *International Journal of Science Education*, *22*(4), 399–417.

Tan, C., & Cheng, H. (2009). IMPLANT: An integrated MDP and POMDP learning ageNT for adaptive games. In *Proceedings of The Artificial Intelligence and Interactive Digital Entertainment Conference* (pp. 94–99).

Tang, Y., Young, C., Myung, J., Pitt, M., & Opfer, J. (2010). Optimal inference and feedback for representational change. In *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society* (p. 2572-2577).

Tauber, S., & Steyvers, M. (2011). Using inverse planning and theory of mind for social goal inference. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (pp. 2480–2485).

Tenenbaum, J. B. (2000). Rules and similarity in concept learning. In *Advances in Neural Information Processing Systems 12* (pp. 59–65).

Tenenbaum, J. B., & Griffiths, T. L. (2001). Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences*, *24*, 629-641.

Tenenbaum, J. B., Griffiths, T. L., & Kemp, C. (2006). Theory-based Bayesian models of inductive learning and reasoning. *Trends in Cognitive Sciences*, *10*(7), 309–318.

Theocharous, G., Beckwith, R., Butko, N., & Philipose, M. (2009). Tractable POMDP planning algorithms for optimal teaching in "SPAIS". In *Proceedings of the IJCAI Workshop on Plan, Activity, and Intent Recognition (PAIR).*

Thompson, C. (2011). How Khan Academy is changing the rules of education. *Wired Magazine*, *126*.

Ullman, T. D., Baker, C. L., Macindoe, O., Evans, O., Goodman, N., & Tenenbaum, J. B. (2010). Help or hinder: Bayesian models of social goal inference. In *Advances in Neural Information Processing Systems (NIPS)* (Vol. 22, p. 1874-1882).

Van Der Linden, W., & Glas, C. (2000). *Computerized adaptive testing: Theory and practice*. Springer.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197–221.

Villano, M. (1992). Probabilistic student models: Bayesian belief networks and knowledge space theory. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems* (pp. 491–498). Springer.

Von Ahn, L. (2006). Games with a purpose. *Computer*, *39*(6), 92–94.

Walonoski, J. A., & Heffernan, N. T. (2006). Detection and analysis of off-task gaming behavior in intelligent tutoring systems. In M. Ikeda, K. Ashley, & T.-W. Chan (Eds.), *Intelligent Tutoring Systems* (Vol. 4053, pp. 382–391). Springer Berlin Heidelberg.

Walsh, T. J., & Goschin, S. (2012). Dynamic teaching in sequential decision making environments. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence* (pp. 863–872).

Waters, A. E., Lan, A., Studer, C., & Baraniuk, R. G. (2012). Learning analytics via sparse factor analysis. In *Personalizing Education With Machine Learning, NIPS 2012 Workshop.*

Young, S., Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., & Yu, K. (2010). The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, *24*(2), 150–174.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* (pp. 1433–1438).

Appendix A

Technical details for teaching via POMDP planning

In this appendix, we elucidate some of the technical details in our use of POMDPs for teaching. We first provide information about the free parameters in all experiments in Chapter 6, then provide details of our implementation of the particle filter for the continuous concept learning model, and finally describe how we fit the free parameters for the three concept learning models based on data from the control condition.

A.1 Free parameters in POMDP experiments

Both experiments in Chapter 6 involve several free parameters. In this appendix, we list all free parameters, describe their roles, and identify whether each parameter was optimized.

All learner models involve two free parameters, a production noise parameter and a transition noise parameter. Each of these parameters was optimized based on data from the control condition, and these parameters different across experiments and models. Small changes to the values of these parameters have little effect on the computed policies. There are two other free parameters specific to the learner models: a memory parameter for the discrete model with memory and the number of particles for the continuous model. Neither of these parameters was optimized, and both were kept constant across experiments. The memory parameter was set to two, meaning that the two previous examples were kept in memory. Changes to this parameter affect how quickly the discrete memory learner is able to learn; when the parameter is zero, the discrete memory learner is equivalent to the memoryless learner. The number of particles was set to 16 for the continuous learner. This value was chosen to be small enough to allow for relatively fast computations, and the planning algorithm is not sensitive to small changes in the number of particles.

The cost function also has several free parameters: the discount factor γ and the cost of each action type. The discount factor γ was set to 0.99 to reflect the fact that future costs should be about as costly as current costs; this parameter was not optimized. The cost of actions varied across experiments. It was set to be the median time for control participants to complete each type of action in order to reflect the relevant quantity (time) that we were trying to minimize.

Within planning, the lookahead horizon and the number of actions sampled at each step are free parameters. These parameters were again not optimized, and small changes in the number of actions sampled do not result in large changes in the policy. Adjusting the lookahead horizon can result in changes to the policy, with longer lookahead horizons allowing one to capture longer-term effects of actions. For each model, we experimented with how the length of computation was affected by different settings of these parameters, and chose their values such that to maximize the number of actions sampled with the constraint that computation generally took no more than three seconds (the maximum time allowed).

A.2 Particle filter implementation

For the continuous learner model in Chapter 6, we use a *particle filter* to maintain and update the belief state (Doucet, de Freitas, & Gordon, 2001). A particle filter approximates a distribution using a limited number of particles, each of which represents a sample from the distribution. Each particle has a weight, representing its mass in the distribution. In our case, the filter is approximating the teacher's distribution over possible knowledge states. Each particle thus corresponds to one knowledge state, which is itself a distribution over possible concepts. At the first time step, we initialize the filter to include two particles: one that has a uniform distribution over concepts and one that has the prior distribution over concepts. This provides some level of robustness in cases where a learner's prior does not match the prior assumed by the teacher. Each of these particles is given equal weight, so the belief state is a distribution with 0.5 of the mass on each of these possible knowledge states.

The particle filter is updated after each teacher action and student response. The update for the response is completed first, and is relevant in the case of a *quiz* or *question with feedback* action. For these actions, the weights on the particles are updated to account for the observed student response. If a student gives response z to action a, then each particle s with weight w_s is updated to have weight w'_s as follows:

$$w'_s = w_s \left((1-\varepsilon_p) p(z|a,s) + \varepsilon_p \frac{1}{|Z|} \right), \tag{A.1}$$

where $|Z|$ is the number of possible student responses and ε_p is the production noise parameter. Thus, the new weight is the product of the old weight and the probability that a learner with the knowledge state represented by particle s would have given the observed response. The probability $p(z|a,s)$ of giving a particular response is specified by the observation model. If all of the new weights are very small (less than 0.005), then none of the particles would be likely to produce the given response. This is an instance of *particle depletion*: The filter has no samples that seem to be good approximations of the learner's actual state. In that case, we re-initialize the distribution with a particle representing the prior distribution and a particle representing the state that would occur if the learner had begun with a uniform distribution and updated her state at each step to be consistent with the observed data. If particle depletion does not occur, we renormalize the weights to sum to one, since they form a probability distribution.

After adjusting for the observed response, the particles are updated to account for the example that the learner is shown. This update, relevant in the case of an *example* or *question with feedback* action, creates a new set of particles that represent how the existing states would change given the learner model. For each original particle s and weight w_s, we have two new particles s' with weight $w_{s'}$ and s'' with weight $w_{s''}$. Particle s' is the result of the knowledge state in s being updated and has entries s'_i as follows:

$$s'_i \propto \begin{cases} s_i & \text{if } c_i \text{ is consistent with } a \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

where c_i is the ith concept and a is the action taken at this time step. s'_i is then renormalized to sum to one. This corresponds to the update given in the main text for how the learner's knowledge state in the continuous model changes given an action. This particle has weight $w_{s'} = (1-\varepsilon_t)w_s$, which is the original probability of the particle weighted by the probability that the student updates her state. The other new particle s'' is identical to the original particle s as it is the result of the state not being updated. This particle has weight $w_{s''} = \varepsilon_t w_s$, where the original probability of the particle is now weighted by the probability of the state not updating. After constructing the new particles for the updated belief state, we check that the sum of the weights of these particles is not too small (again, less than 0.005). If particle depletion has occurred, we follow the procedure above.

If particle depletion has not occurred, we complete the state update by limiting the number of particles that we preserve and then renormalizing their weights. Particle filters are advantageous because they allow us to work with continuous distributions using only a finite number of samples. However, if the number of particles is allowed to grow without limit, they quickly become computationally infeasible. Thus, we limit the number of particles at each time step to N; the main text gives the values of N we used for each experiment. Particles are ordered by their current weights, and the particles with the top N weights are maintained. The particle weights are then renormalized to sum to one.

A.3 Fitting parameters for concept learning models

For each of the three model concept learning models in Chapter 6, we set the noise parameters ε_p and ε_t based on the data generated by participants in the control condition, who were taught using a random policy. The procedure for setting these parameters was the same for both experiments. For the memoryless model and the discrete model with memory, we used expectation maximization to find the values of ε_p and ε_t that would maximize the likelihood of the produced data. To maximize the expected likelihood L with respect to ε_t, we compute:

$$\hat{\varepsilon}_t = \underset{\varepsilon}{\operatorname{argmax}} \sum_k p(\mathbf{k}|\mathbf{z},\varepsilon) L(\mathbf{z},\mathbf{k}|\varepsilon), \tag{A.3}$$

where **k** is a vector in which $k_i = 0$ if the learner state did not update at time i and $k_i = 1$ otherwise, and **z** is the vector of observed student responses. Let T be the number of time steps where the state could have failed to update (the total number of example and question with feedback actions). If **k** was known, Equation A.3 would be easily maximized:

$$\hat{\varepsilon}_t = \frac{\sum_i I(k_i = 0)}{T}, \tag{A.4}$$

where $I(\cdot)$ is the indicator function that is equal to 1 if its argument is true and 0 otherwise. However, since **k** is unknown, we use a variation of the Baum-Welch algorithm for learning the parameters of a hidden Markov model to calculate the probability of no update occurring at each time step with an example or question with feedback action:

$$\eta_i = \frac{\sum_m \alpha_i(s_i = m)\varepsilon_t \beta_{i+1}(s_{i+1} = m) p(z_{i+1}|s_i = m)}{\sum_m \sum_n \alpha_i(s_i = m) p(s_{i+1} = n|s_i = m)\beta_{i+1}(s_{i+1} = n) p(z_{i+1}|s_{i+1} = n)}, \tag{A.5}$$

where the forward probability $\alpha_i(s_i = m)$ is the probability $p(s_i = m, z_{1:i})$ that the learner is in state m at time i and observations $z_{1:i}$ occurred in steps 1 through i and the backward probability $\beta_i(s_i = m)$ is the probability $p(z_{(i+1):N}|q_i = m)$ that observations $i+1$ through N, the final observation, occur given that the learner is in state m at time i. Intuitively, each η_i corresponds to how likely it is that the update at time i was ineffective. Note that to calculate Equation A.5, we use the estimate value of ε_t as well as the current estimate of ε_p. Since each η_i is an expectation over k_i being equal to 0 (i.e., an expectation over whether the update at time i was ineffective), we can replace the numerator in Equation A.4 with the expectation:

$$\hat{\varepsilon}_t = \frac{\sum_i(\eta_i)}{T}. \tag{A.6}$$

We alternate between calculating the η_i given the current estimate of ε_t, and calculating $\hat{\varepsilon}_t$ given the estimate of η_i.

We simultaneously estimate the probability of an error in production, ε_p. Let **p** be a vector where $p_i = 0$ if the observation at time i was due to a production error and $p_i = 1$ otherwise. Then if **p** was observed, the estimate would be:

$$\hat{\varepsilon}_p = \frac{\sum_i I(p_i = 0)}{T}, \tag{A.7}$$

where T is now the number of possible time steps in which production errors could have occurred (the number of quiz and question with feedback actions). Since **p** is unobserved, we again use a variation of the Baum-Welch algorithm to calculate the probability of a particular observation occurring due to production error:

$$\phi_i = \frac{\sum_m \alpha_i(s_i = m)\beta_i(s_i = m)\varepsilon_p p(z_i|s_i = m, p_i = 0)}{\sum_m \alpha_i(s_i = m)\beta_i(s_i = m)(\varepsilon_p p(z_i|s_i = m, p_i = 0) + (1-\varepsilon_p) p(z_i|s_i = m, p_i = 1))}. \tag{A.8}$$

This quantity is the expectation of the observation at time i occurring due to production error, so we can substitute it for p_i above:

$$\hat{\varepsilon}_p = \frac{\sum_i(\phi_i)}{T}. \tag{A.9}$$

At each expectation step, we calculate all ϕ_i and η_i given the current values of $\hat{\varepsilon}_p$ and $\hat{\varepsilon}_t$; we then maximize these parameters given the new values of ϕ_i and η_i. This algorithm is continued until the estimates converge. This algorithm is an expectation-maximization algorithm and is thus not guaranteed to find the global optima. However, restarting the algorithm did not generally result in substantially different solutions.

To calculate the estimates of the noise parameters for the continuous model, a slightly different algorithm is required. In the continuous case, we have an infinite state space, making the calculations above infeasible. Instead, we find an approximate solution using forward filtering. We model the learner's state using a particle filter, and perform updating until time i. For ε_t, we then calculate the probability that the observation at the next step was generated based on a failed transition:

$$\eta_i = \frac{\sum_p p(\text{particle}_p)\varepsilon_t p(z_{i+1}|\text{particle}_p)}{\sum_p p(\text{particle}_p)(\varepsilon_t p(z_{i+1}|\text{particle}_p) + (1-\varepsilon_t)p(z_{i+1}|\text{particle}_{q|p}))}, \tag{A.10}$$

where $\text{particle}_{q|p}$ is the updated version of particle p. At each step i, we consider all possible particles that could occur based on our current particles; this will be twice as many particles as are in the current state, since we must consider each particle being updated or failing to be updated. We maintain only a fixed number of particles when updating until the time i, though. This η_i is an approximation of the η_i above, allowing us to estimate ε_t using Equation A.6. Similarly, we can perform forward filtering to calculate an approximate version of ϕ_i:

$$\phi_i = \frac{\sum_p p(\text{particle}_p)\varepsilon_p p(z_i|s_i = m, p_i = 0)}{\sum_p p(\text{particle}_p)(\varepsilon_p p(z_i|s_i = m, p_i = 0) + (1-\varepsilon_p)p(z_i|s_i = m, p_i = 1))}. \tag{A.11}$$

We use this ϕ_i to calculate $\hat{\varepsilon}_p$ using Equation A.9.

For alphabet arithmetic, we found the following values for the learner model parameters: memoryless model: $\varepsilon_t = 0.15$ and $\varepsilon_p = 0.019$; discrete model with memory: $\varepsilon_t = 0.34$ and $\varepsilon_p = 0.046$; and continuous model: $\varepsilon_t = 0.14$ and $\varepsilon_p = 0.12$. For the number game, the fitted values were as follows: $\varepsilon_p = 0.14$ and $\varepsilon_t = 0.25$ for the memoryless model; $\varepsilon_p = 0.10$ and $\varepsilon_t = 0.18$ for the discrete model with memory; and $\varepsilon_p = 0.15$ and $\varepsilon_t = 0.21$ for the continuous model.