Complex decision-making

by

Sebastiaan van Opheusden

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Center for Neural Science

New York University

May 2019

Wei Ji Ma

I see only one move ahead, but it is always the correct one

José Raúl Capablanca, world chess champion, 1921–1927

Acknowledgements

This thesis represents not just the product of the research I did over the last few years, but the culmination of a journey that I started a decade ago. Throughout this journey I have been mentored by Helmut Schiessel, Frank Redig, Vincenzo Vitelli, Michael J. Berry II and Wei Ji Ma, my advisor at New York University. All of you have shared wisdom both scientific and personal, and I am grateful that you all encouraged me to explore, moving from the Netherlands to America, from Physics and Mathematics to Cognitive Neuroscience, and from a charming Dutch town in which I have lost a few too many bikes to a place known simply as “the city”.

I want to especially thank Wei Ji and the other members of my advisory committee Eero Simoncelli, Paul Glimcher, Nathaniel Daw and Matt Botvinick, for helping me shape the research in this thesis. I also want to credit Wei Ji's stroke of genius to write a grant to study people playing four-in-a-row, which has turned out incredibly fun, productive and will inspire many more PhD theses of future students. I would like to thank Nathaniel Daw for helping us adapt four-in-a-row for human neuro-imaging, Daeyeol Lee for collaborating on a non-human primate version and Erman Misirlisoy, Liron Jacobson, and everyone at Peak Brain Training for launching our large-scale online data collection project. Finally, thanks to Facebook Reality Labs, the cognitive science team at FRL and James Hillis for inviting me over to Washington for the best 6 months I've ever experienced.

It's the friends we meet who help us appreciate the journey. It's a cheesy quote but it's true. As I cannot possibly do justice to everyone, I apologize in advance. Daniel, I will forever be in awe of your insatiable curiosity and ability to see beauty in the most unlikely places. Benny, the same but then applied to Icelandic volcanoes. Will, you supported me when I needed it most, and simultaneously engaged me in the one conversation that most profoundly changed the way I view the world. Andra, you have helped me truly appreciate that one can never know these things. Kevin, you inspire me with your humility to always question yourself and your methods, and your dedication to advancing science. Chris, you are probably the only person I know who successfully pulls off a bowtie, or who has actually tried. Aspen, I love how you make me feel comfortable to express my genuine self. Maija, thank you for encouraging me to keep going and be the best I can be (currently a V6).

I'd like to thank all my other friends from Leiden, Princeton, New York, and all the friends I made recently in Redmond. But most importantly and always, I am indebted to my family, Loes, Ineke & Mieke, Joost & Margôt, Paul, Gerard, Lies & Greet.

Preface

In chapters 1 and 2, I present work with Gianni Galbiati, Zahy Bnaya and Yunqi Li. This work also greatly benefited form discussions with Maija Honig about Turing tests, Luigi Acerbi about model fitting and Andra Mihali about eye tracking. Chapter 3 presents work with Luigi Acerbi. I severely regret lacking the space to also include in this dissertation work with Andra Mihali on Bayesian Microsaccade Detection.

Abstract

In this thesis, I introduce a research program for studying increasingly complex decision-making, focused on sequential decision-making. To support this program, I present a case study investigating human decision-making in a board game in which two players compete to create 4-in-a-row on a 4-by-9 board. I argue that this game is at the optimal level of complexity for which behavioral modeling is tractable, but reveals generalizable insight into the computations underlying human cognition and people's strategies for decision-making in complex sequential environments.

Along with this task, I present a behavioral model that predicts individual players' choices when playing against other players or computers opponents. The model combines intuitive value judgments with a weighted linear sum of features, mental simulation of potential move sequences into a decision tree, and attentional oversights as dropping features. To justify our choice of model, I perform a Bayesian model comparison analysis with lesions, modifications, extensions and controls as alternative models. I demonstrate that inferred model parameters are reliable and that model rankings are robust to changes in experimental conditions. Finally, I perform multiple tests to assess the absolute goodness-of-fit of the behavioral model and to identify in what aspects it may fail to match people's distribution of choices. For one such test, I conduct a Turing test experiment in which human observers classify videos of game segments as occurring between two human players or between two computers with parameters inferred for those players.

For any model to be a valid hypothesis for the algorithm by which people arrive at their choices, it needs to generalize between subtasks within a domain and within data types. To test whether our model can do this, I ask it to predict people's choices in two-alternative forced choices (2AFC) trials and board evaluations. I estimate model parameters for each participant from their choice in games against computer opponents, and then predict 2AFC and evaluation data. Similarly, I use the model with estimated parameters to predict participants' response times and eye movements.

To predict response times, I modified the model to terminate its search algorithm after 50 consecutive iterations without changes in its preferred move. I can predict response times using the size of the decision tree on each trial as the predictor. Moreover, the model can predict how much time participants spend fixating on each square by counting the number of times that the model visits that square during the construction of its decision tree. Using a regression analysis, I find that the most informative square visits are those that occur on depth 1 to 3. Finally, I execute two experiments to demonstrate that the model can be used to address scientific hypotheses on the effects of time pressure and the nature of talent (individual skill differences between naive participants) and expertise (skill differences induced by practice). I find that time pressure reduces the size of the decision tree that people build, and both talent and expertise rely on increased tree search, decreased feature dropping but no change in feature weights.

The models that I fit in this thesis and many models used for other decision-making studies are too are too complex to derive a closed-form analytical expression for their log-likelihood. Therefore, I present inverse binomial sampling , a method to estimate log-likelihoods from sample responses applicable to any generative model and any task as long as the space of responses is discrete. Although the mathematics underlying IBS have been published before (Dawson, 1953; de Groot, 1959), I am the first to adapt the method for behavioral model fitting.

I compare IBS to a traditional method called fixed sampling and find that IBS has appealing theoretical properties: its estimates are uniformly unbiased, with standard deviation within 30% of a theoretical lower bound. I then compare IBS and fixed sampling in practice, with a parameter recovery study for an orientation discrimination or change localization task. In both tasks, IBS estimates model parameters more accurately with equally many or fewer samples, and therefore less computational cost. Finally, I discuss improvements to IBS with can decrease its standard deviation or computational cost considerably, and allow for multi-threaded implementations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We live in a complex world. Our senses convey an imperfect picture of the world around us, the consequences of our actions or inaction are hard to predict, and the choices we have to make on a daily basis are rarely directly related to our evolutionary goals - to survive and reproduce. What's worse, choices rarely come in isolation, but in sequences.

For example, consider a recent college graduate planning a career. She can opt to immediately start a job, or to continue her education with a professional degree, or perhaps to pursue an academic career. In the latter case, she will have to decide whether to apply for graduate school, a Master's program or a Research Assistant position, and then which university to enroll in and which lab to join. For the next decade, she will find herself moving between short-term positions and face difficult career choices each time. To know whether graduate school is right for her, she would need to consider the entire branch of career trajectories leading to different permanent jobs (of which the tenure-track faculty is neither the most likely or the most desirable), and judge whether the eventual payoff is worth the investment.

The complexity of the world poses a dual challenge, on the algorithms that people use to learn and make decisions, and on the experimental tasks that researchers can use to investigate human learning and decision-making. To understand what strategies people have evolved to deal with complexity, cognitive neuroscientists need to study complex tasks. Unfortunately, this is not the tradition. In fact, neuroscience has been argued to follow a reductionist program (Krakauer et al., 2017), and many studies ignore behavior altogether. Within behavioral neuroscience, the majority of studies focus on tasks where the dimension of complexity that causes computational difficulties is perceptual uncertainty. Representative examples are the infamous random-dot motion task (Britten et al., 1992; Newsome et al., 1989) and the entire discipline of psychophysics (Stevens, 2017).

1.1 Dissertation outline

In this thesis, I will explore human decision-making in a task where the primary difficulty is the requirement to plan ahead. Specifically, I will study a two-player board game, which has similar properties to chess, go, or tic-tac-toe, but lies at a sweet spot of complexity where behavior is rich but still possible to understand through cognitive modeling. In chapter 2, I will introduce a behavioral model, that is, an algorithm that predicts people's choices in the game, and whose parameters we can learn for individual players. In chapter 3, I will demonstrate that the model can generalize beyond predicting choice data and predict response times, eye movements, and reveal insight on the nature of expertise and talent in sequential decision-making. Chapter 4 presents inverse binomial sampling, a method that I used to fit the model in chapter 2, but one that can be formulated and applied generally for any behavioral modeling study.

1.2 Literature

Multiple disciplines have used games to study human decision-making, including model-based reinforcement learning, economic game theory, player modeling or the study of expertise in chess. Unfortunately, I lack the space and time to review these disciplines comprehensively, but below I briefly mention some theoretical and experimental results, specifically those that relate closest to the topic of this thesis.

1.2.1 Model-based reinforcement learning

The problem of how to play tic-tac-toe is used as a motivating example in the famous textbook on reinforcement learning (RL) by Sutton & Barto (Sutton and Barto, 1998). RL represents behavioral tasks as *Markov Decision Problems* (MDPs), which consist of a set of *states* and a set of available *actions* in each state. After taking an action a in a state s, the agent may receive a *reward* r and then transitions to a *successor state* s'. The dynamics are assumed to be Markovian, meaning that given the current state and action, the reward and successor state are independent of the history of states and actions. The solution to an MDP is a policy $\pi(a|a)$ which specifies a distribution of actions in a given state. To solve an MDP, one has to learn a *policy* $\pi(a|s)$, which specifies a distribution over actions in each possible state, so that the *value* $V_\pi(s)$ in each state is maximal. The value is the expected long-term cumulative discounted reward

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| s_t = s \right]$$

where $\gamma \in [0,1)$ is the discount rate. The optimal policy is always deterministic and *greedy*, and we can use the Markov property to derive a self-consistency condition for the optimal value function knows as the Bellman equation (Bellman, 2013)

$$V_*(s) = \max_a \sum_{s',r} \mathbb{P}(s', r|s, a) \left[\gamma V_*(s') + r\right]$$

Many algorithms exist to solve Bellman's equation, which can be separated into two broad classes: *model-free* versus *model-based* reinforcement learning. In model-free RL, the agent does not have access to $\mathbb{P}(s', r|s, a)$ (the transition rule), and learns purely by trial-and-error. These methods implicitly approximate $\mathbb{P}(s', r|s, a)$ by averages over experienced rewards and state transitions. By contrast, a model-based or *planning* agent possesses a *model* of the world, formally a belief state over the transition rule. Sutton & Barto distinguish *background planning*; using a model to *replay* or simulate experience off-line or as part of a learning rule, and *decision-time planning*; using the model to plan a sequence of actions starting in the state that the agent currently resides. This latter type of planning is most relevant to this thesis.

1.2.1.1 Heuristic search

In chapter 8.9, Sutton & Barto introduce the *heuristic search* framework for decision-time planning in determinstic two-player winner-takes-all games like chess, go or tic-tac-toe. In these games, the Bellman equation reduces to the *minimax* equations

$$V_*(s) = \begin{cases} \max_a V_*(s \oplus a) & \text{if player 1 to move} \\ \min_a V_*(s \oplus a) & \text{otherwise} \end{cases}$$

where $s \oplus a$ denotes the state resulting from action a in state s. Solving these equations recursively until a *terminal* or game-ending state requires constructing a *decision tree* whose size is exponential in the number of choice points. Heuristic search algorithms circumvent this problem by introducing a *heuristic value function*

$$H(s) = \sum_i w_i f_i(s)$$

where $f_i(s)$ are *features* of the state s and w_i are feature weights. These weights are gradually learned during experience to minimize the discrepancy between $H(s)$ and the optimal or *game-theoretic value* $V_*(s)$. However, since such learning is only tractable if the feature set is small, it will be impossible for $H(s)$ to equal $V_*(s)$. Therefore, to choose an action in a given state, heuristic search algorithms construct a partial decision tree with that state as the root node. One common method to construct a decision tree is *best-first search* (Dechter and Pearl, 1985), which is an iterative algorithm with fours steps: given a partial tree, *select* a sequence of actions by either player until a leaf node of the tree, *expand* the node with candidate moves, *evaluate* the resulting states with $H(s)$ and *back-propagate* the information to the root using Bellman updates.

1.2.1.2 Experimental results

One set of experimental data employs the *two-step task* (Daw et al., 2005), developed by Daw, Niv and Dayan. In this task, participants make a sequence of two choices between states represented with fractal images (see figure 1.1). In the first decision stage, the participant chooses between an “orange” or “blue” stimulus. The “orange” stimulus usually (70% of trials) leads to the “orange” state, and the “blue” stimulus to the “blue” state.

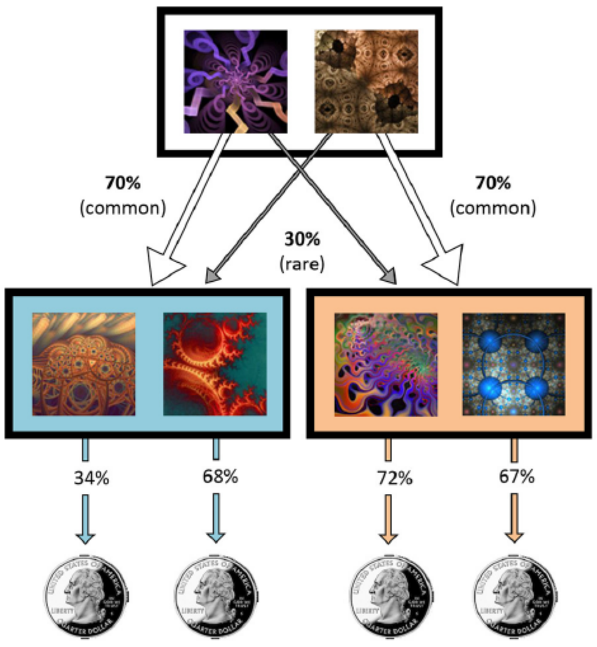


Figure 1.1 Illustration of the two-step task, adapted from Daw et al. (Daw et al., 2005).

However, occasionally (30% of trials), the “orange” stimulus leads to the “blue” state, or vice versa. On the second stage, participants choose between two stimuli, which are associated with a monetary reward. Crucially, these reward probabilities fluctuate slowly, so participants will have to constantly adapt the subjective value they associate to each image, and adjust their choices accordingly. Albeit complicated, this is the simplest task in which model-free and model-based RL make different behavioral predictions. The two-step task has been used to demonstrate that people engage in model-based RL and identified neural substrates for the computation of model-based value (Daw et al., 2011, 2005; Gläscher et al., 2010; Kool et al., 2016; Lee et al., 2014).

Another relevant article by Huys et al. investigates the algorithm by which people construct *“Bonsai trees in your head”* (Huys et al., 2012, 2015). In their task, participants make a sequence of multiple two-alternative decisions by which they traverse a graph of 6

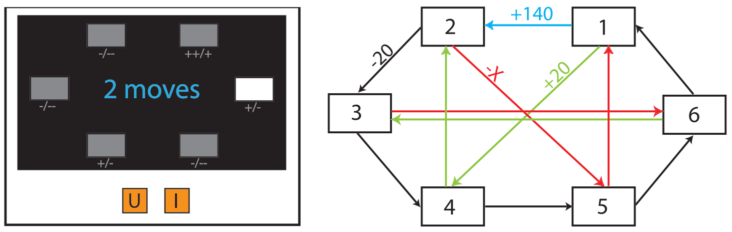


Figure 1.2 Illustration of the task used by Huys et al. (Huys et al., 2012).

states (see figure 1.2). Each transition incurs a reward which can be either positive or negative, and the task is designed such that the optimal policy requires taking large negative rewards to obtain positive rewards on future steps. They find that people plan trajectories multiple steps ahead, but they *prune* options leading to large negative rewards and thereby fail to act optimally.

Finally, Solway & Botvinick study a sequential choice task (see figure 1.3), and show that people's behavior can be captured by noisy evidence integration, which treats each path through the decision as an independent competitor in a bounded accumulation process. Specifically, the model instantiates one accumulator variable for each path from the root of the decision tree to a terminal node. Each time step, all accumulators update with drift rate proportional to the value of the items on the path, and when one accumulator variable exceeds all others by more then a given threshold, the process stops. Although Solway & Botvinick test their models in the context of a relatively small MDP with a known value function, they point to a striking similarity between the integration process in their algorithm and that of Monte Carlo Tree Search, which can serve as a starting point for a study of human decision-making in large state spaces.

Although the experimental literature on model-based reinforcement learning and its implementation in the brain is substantial, it has focused primarily on simple tasks, at least

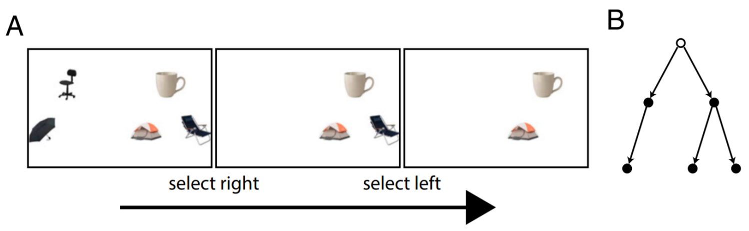


Figure 1.3 A. In the task used by Solway & Botvinick (Solway and Botvinick, 2015), people choose between a number of items by making a sequence of binary (left/right) decisions. In the first decision, participants choose between the items on the top row, but simultaneously choose which choice set they will receive on the second decision. In this trial, the participant chose the mug over the office chair, and subsequently the tent over the beach chair. **B.** Schematic of the decision tree corresponding to this trial.

when measured in state-space complexity. The two-step task has 3 states; the task used by Huys et al. has 6, or $\approx$ 50 if one incorporates time into the state representation. By contrast, tic-tac-toe has 765 states[I], and chess has approximately 10^{47} (Chinchalkar, 1996). Therefore, people's planning strategies in complex games remains an open problem.

1.2.2 Expertise in chess

Perhaps the most directly relevant literature to this thesis, and specifically chapter 3, is that on expertise in chess. This literature traces back to the seminal dissertation by Adriaan de Groot (de Groot, 1946a) titled *"het denken van de schaker"* which translates[II] to *"How chess players think"*. In his dissertation, de Groot proposed that strong chess players make decisions by constructing a decision tree through an *iterative deepening* algorithm. This algorithm starts by exploring candidate moves up to limited depth, then increasingly

[I] This number ignores states that are mirror images or reflections of these states, as well as illegal state with too many pieces for either player, or multiple simultaneous three-in-a-row patterns. However, it does count terminal states.

[II] The 1965 translation of de Groot's dissertation is titled *"Thought and choice in chess"*, which I like a lot less.

deeper as time progresses. He then studied the problem of "giftedness", the psychological underpinnings of exceptional chess ability, due to exceptional *talent*, *expertise*, or often both.

To study gifted chess players, de Groot conducted two experiments. In the first, he presented chess players with a pre-configured board position and instructed them to narrate their thought process while deciding on a move. In the second, he instructed players to memorize and reconstruct given chess positions and counted the number of pieces they placed incorrectly. In the first experiment, de Groot found no differences in the search strategy of stronger and weaker players in his experiment[III], whereas he did find strong differences in their ability to memorize chess positions.

The study of chess expertise in the 72 years between de Groot's dissertation and my own consists to a remarkably large extent of repetitions and replications of de Groot's experiments. In 1973, Simon & Chase repeated the reconstruction experiment but added a control condition in which player were tasked to reconstruct scrambled (often illegal) chess positions. They found that experienced players were better at memorizing de Groot's chess positions but not the scrambled ones, suggesting that their memory representation of chess positions must somehow be adapted to the statistics of positions that arise in human-vs-human games. Specifically, they hypothesized that players represent chess positions as an array of small patterns called *chunks*, allowing them to compress information and avoid capacity limits. Chunking theory has influenced memory research beyond the domain of chess (Black and Bower, 1979; Dowling, 1973). In 2012, Linhares & Mendes

[III] It is important to note, as has been all too forcefully pointed out by Bilalić (Bilalić et al., 2008), that de Groot did not compare experts with novices. Instead he compared extremely strong players (Dutch Masters, International Grand Masters and even two world champions), to strong players in the Dutch amateur leagues.

(Linhares et al., 2012) replicated the reconstruction experiment and analyzed which specific features of a chess position players remember incorrectly.

The narration experiment has also been replicated numerous times, with less consistent results, and the role of forward search in chess expertise has been hotly debated. Some studies found no difference in search between experts and novices (Charness, 2013; Gobet and Simon, 1998), whereas others (Campitelli and Gobet, 2004; Saariluoma, 1992) did. Charness (Charness, 1991) proposed that improved search may be responsible for one's trajectory from novice to expert, but the step to Grand Master level relies on pattern recognition. In my opinion, the debate on the role of search in chess ability is limited primarily by the inability of verbal report data to distinguish hypotheses on the cognitive nature of expertise or talent.

To provide more objective data on the same question, Holding conducted experiments in which participants played chess under *time pressure* (Holding, 1992) or while counting backwards (Holding, 1989a), in order to tax their central executive component of working memory. Both manipulations are assumed to selectively impair chess players' search and planning, while leaving their pattern recognition abilities intact. Both manipulations affect experts more than novices (Calderwood et al., 1988; Chabris and Hearst, 2003; Van Harreveld et al., 2007), suggesting that search plays a role in chess expertise.

In another experiment, Holding asked players to *evaluate* chess positions taken from Grand Master games, after which he showed them the next few moves in the game and asked them to re-evaluate (Holding, 1989b). He showed that weaker players were more likely than stronger players to change their evaluation after witnessing the Grand Master's moves. I especially like this study, since this "discrepancy" measure is conceptually identical to the objective that Silver et al, (Silver et al., 2017) optimized to develop AlphaZero, the strongest computer chess program that currently exists.

Although cognitive research on chess has addressed similar questions as I will in this thesis, it has not provided adequate answers. Multiple studies implicitly assume models of the cognitive processes by which chess players arrive at their decisions, these models are rarely specified in enough detail to predict an individual player's move in a given position. As such, one cannot infer parameters of these models from human data, or compare models that instantiate competing hypotheses. Moreover, many studies do not actually collect data on human chess play, but tangentially related tasks such as reconstruction. Finally, the continued reliance on verbal reports as data undermines the strength of conclusions regarding the nature of expertise and talent.

1.2.3 Economic game theory

Economics has a long tradition of analyzing games, dating back to at least John Nash (Nash et al., 1950). Following his tradition, I define economic games as an interaction where a number of players each choose an action, which is *hidden* to all other players, and the game specifies a *payoff* for each player given all players' choices. Famous examples of economic games include matching pennies, rock-paper-scissors, or predator-prey interactions. Nash's brilliant insight was to include non-deterministic strategies or *policies*, defined as probability distributions over actions, and consider the *expected payoff* under such a policy. He then showed that each two-player economic game contains a *Nash equilibrium*, a pair of policies, such that the first player's policy maximizes the first player's expected payoff given the second player's policy, and vice versa. The definition of economic games can be extended to include multiple decision stages (as in chess or go), information that is known to some players but hidden to others (as in poker or Starcraft[IV]), or repeated games, and Nash's theorem applies to these more general games.

Although Nash Equilibria have been successful at explaining market forces, they may be unsuitable as models for individual people's choices for three reasons. First of all, finding a Nash equilibrium can be computationally expensive. For two-player adversarial games of full information (combinatorial games (Beck, 2008)), the Nash Equilibrium is the minimax solution, whose computational complexity is exponential in the number of decision stages. Moreover, some games, such as the infamous prisoner's dilemma, have multiple Nash Equilibria, giving rise to the *equilibrium selection* problem. Finally, a Nash Equilibrium provides players with a policy that maximizes expected reward (or *utility*), but a growing body of literature (Kahneman and Tversky, 2013; Thaler, 2016) has shown that maximum-expected-utility strategies are generally poor predictors of human choice. The discipline of characterizing human decision-making in economic choice is called *behavioral economics*, or *behavioral game theory* (Camerer, 2010) when applied to study economic games.

1.2.3.1 Cognitive hierarchy

The most relevant to the content of this thesis are *cognitive hierarchy* models (Camerer et al., 2004). To illustrate the cognitive hierarchy, consider the p-beauty contest (Nagel, 1995). In this game, multiple players each pick a number between 0 and 100, and the player who picks closest to two-thirds of the average number wins (payoff \$1) and everyone else loses (no payoff). Therefore, in the Nash Equilibrium, each player's choice should be

[IV] StarCraft® is a trademark or registered trademark of Blizzard Entertainment, Inc., in the U.S. and/or other countries.

two-thirds of the average of all other players' choices, which defines a recurrence relation whose only solution is for all players to choose 0. However, this is not what most people choose, and it is almost never the winning option in experimental studies. Instead, people on average choose around 20-35 and the winning choice is therefore around 13-22.

To understand this behavior, the cognitive hierarchy considers *level-k* agents (Arad and Rubinstein, 2012; Costa-Gomes and Crawford, 2006). A level-0 agents picks randomly and uniformly from the choice set, which on average will be 50. A level-1 agent maximizes its expected reward, assuming that all other players are level-0. Therefore, it will choose 33. In general, the level-k player assumes that everyone else is level-$(k-1)$, and chooses two-thirds of their average response. This defines an easily solved recurrence relation, and the level-k player chooses $50 \times \left(\frac{2}{3}\right)^k$. In the limit $k \to \infty$, the level-k agent converges to Nash Equilibrium. To model individual players' choices, the cognitive hierarchy assumes that people maximize expected reward given their beliefs of the *distribution* of levels of all other players. In practice, they find that this distribution is mostly concentrated on levels 1 to 3.

The cognitive hierarchy framework has been useful in providing a rational account of seemingly irrational choices in economic games, but it does not immediately give rise to a behavioral model for human decision-making in board games. One limitation is that the iterated reasoning in the cognitive hierarchy framework operates over belief distributions over other player's beliefs or strategies. To me, this seems qualitatively different from iterated reasoning about decisions made in sequence, and I don't know how to implement a cognitive hierarchy model for games like chess, go or even tic-tac-toe.

1.2.4 Artificial Intelligence and player modeling

Since the inception of artificial intelligence research (Machinery, 1950; Shannon, 1950), programming a computer to play board games and chess in particular has been a motivating example and a concrete benchmark to drive progress. Many innovations in artificial intelligence were inspired by the study of human cognition, but I will focus on those algorithms developed at least partly by incorporating data from human players, a practice known as *player modeling*.

Yannakakis et al. (Yannakakis et al., 2013) define player modeling as *"the study of computational models of players in games"*. To illustrate the breath of work captured by this definition, Smith et al. (Smith et al., 2011) provide a taxonomy of player modeling studies by 4 facets that determine the kind of model: its scope, purpose, domain and source.

The scope of a player model describes the set of people whose behavior it is meant to capture, which can range from individual to class-specific (for example, based on player profiles) to universal (everyone) or hypothetical (no one). The purpose of a model can be generative, meaning that one can expose them to novel game states and generate outputs, or descriptive, meaning that the model summarizes statistics of a player's game play. The domain of a player model is the output that it predicts, which can be the actions that a player takes during the game, or the *experience* that a player feels while playing. Finally, the source of a player model refers to the methods used to develop the model. This can be induced (developed through statistical learning on human game play data), interpreted (informed by data as interpreted by the developer), analytic (optimized for an objective performance metric) or synthetic (developed through purely theoretical constraints). By this taxonomy, the models developed in this thesis and the majority of cognitive neuroscience are individual generative induced action models.

Another distinction between player models, and particularly one that distinguishes cognitive neuroscience, is the goal or use case of the model. The cognitive neuroscientist's goal is to use game play data to adjudicate between scientific hypotheses about the nature of human cognition. Therefore, each model instantiates a competing hypothesis about the cognitive process by which players reach in-game decisions, and determining the best-fitting model is an end in itself. Instead, in player modeling, the understanding of human decision-making or affect embedded in a player model is a means to an end.

1.2.4.1 Applications of player modeling

One use of player modeling is the development of AI agents that achieve high or even superhuman performance. A common strategy for the development of Go algorithms has been to train a Bayesian pattern learner (Stern et al., 2006) or convolutional neural network (Clark and Storkey, 2015) to predict human players' moves in a large database of expert games. Silver et al. improved on this by using human expert data to *pretrain* a network to use as a starting point for AlphaGo (Silver et al., 2016), the first AI agent that achieved superhuman performance on Go. Note that the later version AlphaZero (Silver et al., 2017) did not use human data but instead learned to play Go entirely from scratch. In another study, Runarsson & Lucas (Runarsson and Lucas, 2014) used preference learning to train an evaluation function for Othello from a database of human games, and created a strong agent that used only this value function and no tree search.

Player modeling is also useful to create AI agents that learn an *opponent model* in real time and use that to adapt their strategy to exploit the opponent's weaknesses or to infer hidden information known by the opponent. Opponent modeling has been used extensively to create AI agents for Poker (Billings et al., 1998; Davidson et al., 2000) and is a component of Libratus (Brown and Sandholm, 2017), an AI agent that beat top professionals in Texas No Limit Hold'Em. Another domain in which opponent modeling has been used successfully is real-time strategy (RTS) games (Bakkes et al., 2009), and specifically StarCraft 2.

One key component of StarCraft strategy is to use units to scout the map to gather information and determine the *build order* that the opponent is using. This information is invaluable to decide whether to invest resources in long-term growth by training additional workers or researching technologies or to invest in short-term power by building an army. These decisions are crucial, since expanding one's economy while the opponent is preparing an attack is often a lethal mistake. The learning problem is also incredibly hard, since high-level players often use a number of different build orders and often try to reveal misleading information that fools the opponent into making incorrect decisions. Finally, the decision is urgent, since players will often use *all-ins* or *timing attacks*, where are specifically designed to attack the opponent at inconvenient times in their build order and may be indefensible even when recognized early. For a review of opponent modeling in Starcraft 2, see Ontanon et al. (Ontanón et al., 2013).

Other uses of player modeling include the development of algorithms for non-player characteristics (NPCs), which behave as human-like as possible (for example, see Ortega et al. (Ortega et al., 2013)). This research is closest to the behavioral modeling methodology presented in this thesis.

Finally, one application of player modeling that excites me tremendously is to use player models to optimize game design or the content in games. For example, Pedersen (Pedersen et al., 2010) and Shaker (Shaker et al., 2010) optimized levels in Super Mario Bros[V] to be most enjoyable. Similarly, Togelius (Togelius et al., 2006) developed methods to design race tracks in a driving game tailored to the skill level of individual players. These studies either rely on a computational theory of fun (Malone, 1981), or on models that are trained to predict experience or affect rather than actions. For a through review on player modeling in all its facets, see Yannakakis & Togelius (Yannakakis and Togelius, 2017), chapter 5.

Although player modeling is a large and growing literature, it has a different focus than cognitive neuroscience and does not address our question of how people make sequential decisions in board games. In my view, the most important distinction is that, since player modeling uses models as means to an end, it is less important that player models are "correct", only that the algorithms or game design choices motivated by the model are successful. Therefore, player models can often be cognitively unexciting *"my opponent raises* 30% *of hands when facing a 3-bet from the button"*, and little care is taken to ensure that model parameters are estimated accurately, or that multiple models are compared.

1.2.5 Other studies

I would like to mention a few more studies that I found difficult to classify but that I believe are highly relevant for the purpose of this thesis. The first article (Snider et al., 2015) studies "prospective optimization", which as far as I can tell is synonymous with planning at action time. In their task, participants watched a triangular lattice of disks of different sizes scroll down a touchscreen, and traced the most rewarding path (see figure 1.4).

V Super Mario Bros. ©1985 Nintendo Co., Ltd.

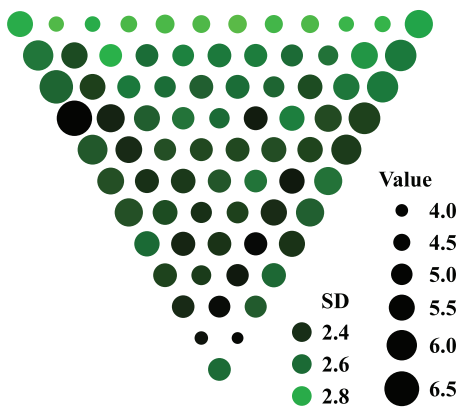


Figure 1.4 Illustration of the task used by Snider et al. (Snider et al., 2015).

Moreover, I have been hugely inspired by literature on mental simulation (Battaglia et al., 2013; Hamrick et al., 2016, 2015). Finally, I would like to mention Xiahong Wan et al. (Wan et al., 2011), who studied the neural basis of *intuitive* (meaning too fast for substantial search) move generation in Shogi.

Chapter 2

Modeling human decision-making in 4-in-a-row

2.1 Introduction

How do people think ahead, or plan a sequence of future actions in search of a reward? Planning and sequential decision-making are ubiquitous in almost any aspect of life, and pose a core difficulty in developing cognitive algorithms that scale to the full complexity of real-world decision problems. However, laboratory experiments (Daw et al., 2005; Huys et al., 2012; Snider et al., 2015; Solway and Botvinick, 2015) that probe planning often restrict the size of the state space (or *state space complexity*), in the hope that behavioral modeling remains tractable. Other studies such as those on expertise in chess (Chase and Simon, 1973; de Groot, 1946a) or player modeling (Silver et al., 2016; Togelius et al., 2006; Yannakakis et al., 2013) do embrace complexity but compromise on the level of detail or accuracy in their methods or models. (see chapter 1.2.5).

Instead, we aim for the best of both worlds, and develop a behavioral task which is complex enough to that it provides rich data on the computational principles underlying human sequential-decision making, but at the same time simple enough to allow for detailed and precise behavioral modeling. Specifically, we study human decision-making in an instance of an (m, n, k)-game (Beck, 2008), in which two players alternate placing pieces on an m-by-n board, and the first player to obtain k pieces in any row, column or diagonal, wins the game. When the board is full and neither player has won, the game is a draw. Tic-tac-toe is $(3, 3, 3)$, we use $(4, 9, 4)$ (see figure 2.1A).

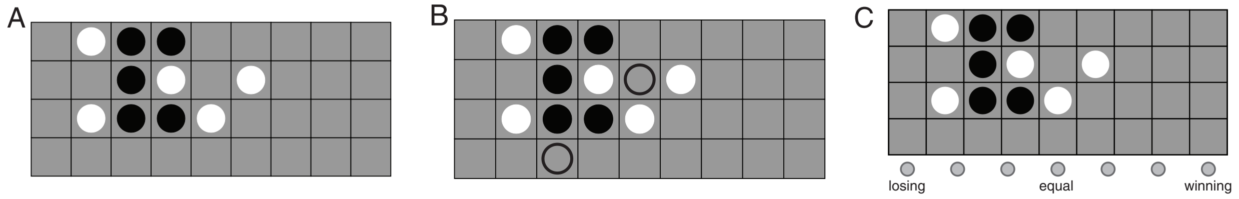


Figure 2.1 A. Example board position in the 4-in-a-row game. Two players, black and white, alternate placing pieces on the board, and the first player to achieve 4-in-a-row wins the game. **B.** Example board position on a trial of the 2AFC task. The participant indicates which of the two candidate moves (dashed circles) they would most likely make in a hypothetical game against a computer. **C.** Example board position on an evaluation trial, in which the participant indicates their expecting winning chances in a hypothetical game starting from this position on a 7-point scale.

Our game has $1.22309(36) \cdot 10^{16}$ non-terminal states (see below), which prevents any exhaustive search or brute force algorithms from being successful. Therefore, people as well as artificial agents who can play this game need to address the challenge of efficient search, which finds promising moves without excessive computational resources. We hope that understanding the algorithms that people have developed to solve this computational problem provides generalizable insight into human cognition and sequential decision-making specifically.

Since the game is deterministic, winner-takes-all and without hidden information, it is solvable. We have weakly solved the game (unpublished work) by demonstrating that with perfect play from both sides, the first player wins. We could use the same algorithm that provided the weak solution to find the game-theoretic value in each position, but in practice we use the main model with all sources of noise set to zero and the tree size fixed to $200,000$. This produces the same values in almost all positions.

2.2 Number of non-terminal states

To determine the number of non-terminal states (N_{nts}) in our game, we need to calculate the number of configurations of black and white pieces on a 4-by-9 board such that:

- The number of black pieces is either equal to the number of white pieces (when black is to move), or one higher (when white is to move).
- Neither player has 4 pieces in a row, column or diagonal.

The first condition allows us to write

$$N_{\text{nts}} = \sum_{n=0}^{36} N_s(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor) p_{\text{nt}}(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor)$$

where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote the ceil and floor functions, respectively, $N_s(m,n)$ is the number of configurations of m black and n white pieces on the board and $p_{\text{nt}}(m,n)$ is the probability that a random configuration with that number of pieces is non-terminal. The number of states reduces to a multinomial coefficient $N_s(m,n) = \frac{36!}{m!n!(36-m-n)!}$, which we can evaluate analytically. To estimate the probability that a random state is non-terminal, we use a Monte Carlo procedure: for a given value of m and n, we generate $1,000$ random configurations and count how many are non-terminal. We do this for each value of m and n satisfying the first constraint, multiply by $N_s(m,n)$ and sum. Finally, we repeat this process $1,000$ times and compute the mean and standard error of N_{nts} across these repeats, resulting in a final estimate of $1.1812 \cdot 10^{16} \pm 3.1 \cdot 10^{13}$ non-terminal states. In figure 2.2, we plot the graphs of $N_{\text{s}}(n)$, $p_{\text{nts}}(n)$ and $N_{\text{nts}}(n)$.

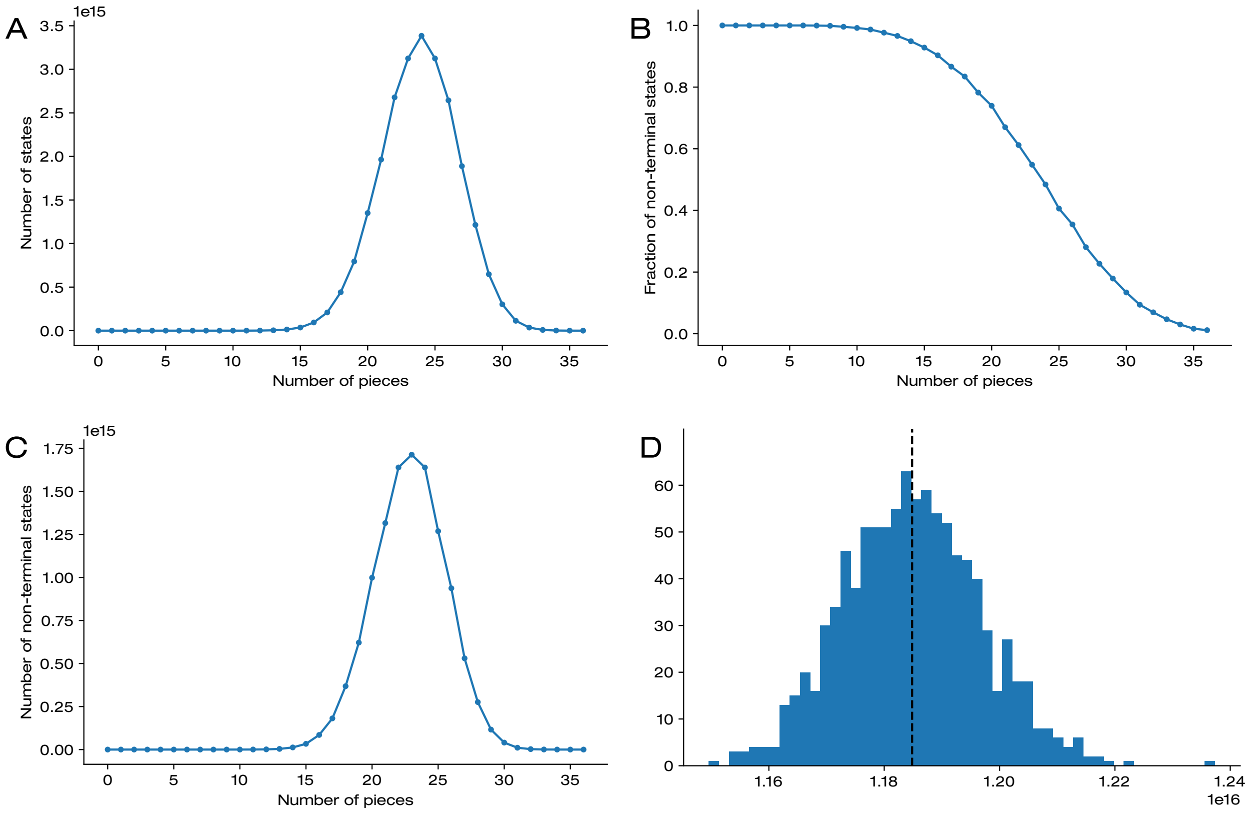


Figure 2.2 A. Graph of $N_{\text{s}}(n)$, the number of possible board configurations as a function of the number of pieces on the board (counting both black and white). **B.** Graph of $p_{\text{nts}}(n)$, the fraction of such states that are non-terminal. **C.** Graph of $N_{\text{nts}}(n)$, the number of non-terminal board states. This is the product of the graphs in **A** and **B**. **D.** Histogram of estimates of the total number of non-terminal states across $1,000$ simulations. The vertical dashed line indicates the mean estimate.

2.3 Experiments

We conducted six experiments: human-vs-human ($N = 40$ participants), generalization ($N = 40$), eye tracking ($N = 10$), learning ($N = 30$), time pressure ($N = 30$) and a Turing test ($N = 30$). We describe these experiments in detail in section 2.5. All experiments except human-vs-human and eye tracking can be played online on our website `basvanopheusden.github.com`. On this website, we also share all experiment and analysis code and all data.

2.4 Model

The primary contribution of this chapter is the introduction of a behavioral model for human sequential decision-making and specifically the 4-in-a-row task. We assume that people's choices on each move are independent and generated by the same decision-making process with the same parameters, at least within a session. Therefore, we cannot model serial dependencies or within-session learning. Additionally, this implies that our model cannot capture strategy adaptations to an opponent's strategy, and therefore the model treats the game as a "game against nature" rather than an economic game.

We first describe the model in conceptual terms and focus on literature support for our chosen model components. Our model is based on the heuristic search framework (Sutton and Barto, 1998), and consists of a value function and a tree search algorithm. Additionally, we include sources of noise to capture variability in human play and specifically mistakes that people make.

2.4.1 Value function

The core of our model is a value function $V(s)$, by which the player estimates the game-theoretic value of a board state. Because the game has so many states, we assume that people use value function approximation (Sutton et al., 2000) to reduce the dimensionality of the space of all possible value functions and to allow for faster learning. Specifically, we assume that people's value function is a weighted sum of features

$$v(s,\vec{w}) = \sum_{i=1}^{5} w_i \phi_i(s,\text{self}) - \sum_{i=1}^{5} w_i \phi_i(s,\text{opponent}) \tag{2.1}$$

where ϕ_i denote the features and w_i the weights. We use 5 features: center, connected 2-in-a-row, unconnected 2-in-a-row, 3-in-a-row and 4-in-a-row. The center feature assigns a value to each square, and sums up the values of all squares occupied by the player's pieces. This value of each square is inversely proportional to its Euclidean distance from the board center. The other features count how often particular patterns occur on the board (horizontally, vertically, or diagonally):

Connected 2-in-a-row: two adjacent pieces with enough empty squares around them to complete 4-in-a-row.

Unconnected 2-in-a-row: two non-adjacent pieces which lie on a line of four contiguous squares, with the remaining two squares empty.

3-in-a-row: three pieces which lie on a line of four contiguous squares, with the remaining square empty. This pattern represents an immediate winning threat.

4-in-a-row: four pieces in a row. This pattern appears only in board states where a player has already won the game.

Whenever the model evaluates a state, the weights of features belonging to the active player are multiplied by a scaling constant c_{act}. For example, the three-in-a-row feature is more valuable when it's the player's own move (it's an immediate win) than on the opponent's (it can be blocked). Active scaling does not apply to the center feature.

These features are a results of a process of manual selection and refinement. We originally chose features that "made sense" to us as researchers, and that aided in the development of strong computer players. We then iteratively added features to improve the model's fit of the human-vs-human data. We also aimed to keep the feature set small, and removed features that proved unnecessary for the model to fit well. In section 2.10, we compare our main model against versions of the model with more or fewer features and demonstrate that these 5 features constitute a minimal set for predicting human choice.

In the future, we aim to pursue more sophisticated methods of feature selection or learning. We intend to train artificial neural networks or other game-playing agents to play 4-in-a-row and investigate their internal state representations. Additionally, we have already collected data for an fMRI study using the same game, and we plan to use representational similarity analysis (Kriegeskorte et al., 2008) to investigate the state representation of ventro-medial prefrontal cortex and thereby extract people's features from the brain rather than behavior.

Tree search. The evaluation function guides the construction of a decision tree with an iterative best-first search algorithm (Dechter and Pearl, 1985). Each iteration, the algorithm chooses a board position to explore further, evaluates the positions resulting from each legal move, and prunes all moves with value below that of the best move minus a threshold (similar to the pruning mechanism proposed in Huys, 2012 (Huys et al., 2012)). After each iteration, the algorithm stops with a probability γ, resulting in a geometric distribution over the total number of iterations.

Tree search or forward planning is well supported as a computational principle in human and animal neural and cognitive processing. For example, Pfeiffer & Foster (Pfeiffer and Foster, 2013) found that hippocampal place cells in rodents fire in sequences such that the place fields sweep paths along trajectories that the animal might take. Johnson & Redish (Johnson and Redish, 2007) found a similar prospective activity in the precession of theta waves, particularly when an animal stops at a choice point. In other work, Miller, Botvinick & Brody (Miller et al., 2016) and Akam et al. (Akam et al., 2015) implemented a version of the two-step task (Daw et al., 2005) for rodents and found substantial model-based planning, which is consistent with (but not necessarily indicative of) forward planning at decision time. Finally, as elaborated in the introduction, tree search is also supported by many human cognitive modeling studies (Arad and Rubinstein, 2012; de Groot, 1946a; Snider et al., 2015).

Noise. To account for variability in people's choices, we add three sources of noise. We model selective attention by randomly dropping features (at specific locations and orientations) before constructing the decision tree, we randomly drop features, which are then omitted during the calculation of $V(s)$ anywhere in the tree. During tree search, we add Gaussian noise to $V(s)$ in each node. Finally, we include a lapse rate λ.

2.5 Experimental methods

2.5.1 Participants

We recruited participants through the NYU psychology research participant system, flyers, a sign-up link on our lab webpage or personal communication. We did not collect demographic data. We compensated participants $12 per hour, but did not incentivize task performance.

2.5.2 Procedure

In the human-vs-human experiment, we recruited 40 participants in pairs. Participants in each pair played against each other for 60 minutes, switching colors after every game. In the generalization experiment, participants performed three tasks: playing against a computer opponent for 30 minutes, 82 trials of a two-alternative forced-choice (2AFC) between moves in given board positions (figure 2.1B), and 82 board evaluation trials, in which they rated their winning chances in given board positions on a 7-point scale (figure 2.1C). The eye tracking experiment consisted of 40 minutes playing against computer and 82 2AFC trials, while we recorded participants' eye movements with an infra-red eye tracker (EyeLink II). The learning experiment consisted of 5 sessions, spaced no more than 3 days apart. In sessions 1, 3 & 5, participants played against computers for 30 minutes, then completed 60 trials each of the 2AFC and evaluation tasks. In session 2 & 4, they played against computers for the entire 50-minute session. In the time pressure experiment, participants played against the computer for 60 minutes. For each game, we added a time limit randomly selected between 5, 10 or 20 seconds per move. If participants exceeded the time limit, they lost the game. Finally, on the Turing test experiment, participants played for 60 minutes against a computer opponent on one session. The next day, they performed a second session on which they viewed 180 videos of excerpts from games played either between two human players or two computers following the model with parameters inferred for those players. Participants classified each video as human or computer and reported their confidence on a continuous scale.

2.6 Experimental methods

2.6.1 Human-vs-human

For our human-vs-human experiment, we recruited participants in pairs. For each pair, we provided consent forms and instructed participants on the task together, after which we separated them into different rooms. We manually started the task for each participant, then they played games against each other through an online interface (similar to the interface on `basvanopheusden.github.io`). After 50 minutes had expired and they finished their last game, we manually proceeded them to a post-task questionnaire (see below), during which we provided them with payment ($12 in cash). Only after completing the survey and receiving payment did participants leave their respective rooms. Thus, participants interacted socially before and often after the experiment, but not during the task.

Participants played games against each other, switching colors after every game. After each game, we presented both participants with a pop-up showing both players' names, current score and a button to continue to the next game. The interface proceeded only after both players had clicked the "continue" button. Every time the participant or their opponent moved, the interface made a faint "clicking" noise. During games, instead of making a move, participants could offer a draw to their opponent, which caused a pop-up prompt to appear on the other participants' screen to accept or reject the offer. If the opponent accepted the draw, the game ended immediately, otherwise the pop-up disappeared and the player who made the offer could make a move instead. We did not restrict how many draw offers participants could make (including multiple draw offers on the same move), but in general participants used the draw offers quite sparingly. In this experiment, we never imposed any time limits, the only reason for participants to play at similar speed as their opponent was out of social desire.

The post-task questionnaire consisted of 11 questions:

Q1 *What is your name?* Open-ended answer.

Q2 *What is the highest level of education you completed?* Multiple choice answer, with options: *"Did not complete high school"*, *"High school"*, *"Bachelor student"*, *"Bachelor's degree"*, *"Master student"*, *"Master's degree"*, *"Graduate student"*, *"Graduate degree"*, *"I prefer not to answer this question"* or *"I'm not sure how to answer this question"*.

Q3a (only presented if the respondent answered Question 2 with *"Bachelor student"*, *"Master student"* or *"Graduate student"*): *What is your major?* Open-ended.

Q3b (presented if the respondent answered Question 2 with *"Bachelor's degree"*, *"Master's degree"* or *"Graduate degree"*): *What major is your degree in?*

Q4 *Can you describe how you decided which moves to make? Did you follow a particular strategy?* Open-ended.

Q5 *Did you feel you were adapting your strategy to your opponent's game play? If so, can you explain how?* Open-ended.

Q6 *How much previous experience do you have playing the following board games?* Response grid, with rows *Chess*, *Checkers/Draughts*, *Go*, *Connect-four*, *Go-moku*, *Xiangqi*, *Backgammon*, *Stratego* and *Risk*, and columns *"Never played"*, *"Played once or twice"*, *"I play occasionally"*, *"I play often"* and *"I'm an expert"*.

Q7 *Are there other board games you have previous experience with?* Open-ended.

Q8 *How much previous experience do you have playing card games? Which games did you play?* Open-ended.

Q9 *How much previous experience do you have playing video games? What games did you play?* Open-ended.

Q10 *How much experience do you have solving puzzles?* Open-ended.

Q11 *Did you ever play games competitively or professionally? If so, which games?* Open-ended.

2.6.2 Generalization

In this experiment and all following ones, participants performed the task on their own. Each session started with the participant providing informed consent, after which we instructed them on the details of the task. We always compensated participants \$12 at the end of their session.

In the generalization experiment, participants played against computer opponents for 30 minutes, after which they automatically continued to the 2AFC condition. They completed 82 trials in this condition, then proceeded to the Evaluation condition and after again 82 trials we debriefed them and provided payment. The interface for the play-against-computer task was identical to the human-vs-human experiment, except for two modifications: the between-game pop-up did not display any names or current score, and we removed the “offer draw” button.

In all human-vs-computer games, the computer’s algorithm is similar to the behavioral model (see section 2.7.2.1) with 3 modifications: we used the pruning rule from the **Fixed branching** model, we included scale factors for weights of features belonging to the opponent (as in the **Opponent scaling** model) and for features of different orientation (the **Orientation-dependent weights** model) but not between "active" and "passive" feature weights (as in the No active scaling model). Finally, the algorithm used a slightly different feature set. We artificially added a "thinking time" to each computer move, which was $0.5 * \sqrt{\text{iterations}} + 1$ seconds or 2 seconds when the algorithm lapses. We chose to use a thinking time proportional to the number of iterations to ensure that the computer played faster in easy positions than in harder ones.

We created 30 agents that all used the same algorithm but with different parameters. We started by fitting the behavioral model on all but 2 participants in the human-vs-human experiment, resulting in 38×5 agents. We obtain 5 times more agents than participants because we fit models using 5-fold cross-validation. For each of these agents, we then generated 7 additional ones by either dividing the feature drop rate by 2, doubling the mean tree size, halving the value noise or any combination thereof. We then ran an all-vs-all tournament between these $38 \times 5 \times 8$ agents and ranked their performance using the Elo system (see 2.7.1). Finally, we selected 30 agents such that their Elo ratings uniformly cover an interval ranging from slightly weaker than the worst human players to slightly stronger than the best. We tuned this interval using pilot experiments. We divided the set of 30 agents into 6 levels with 5 agents per level, and matched participants with computer opponents using a one-up, one-down staircase, starting at the level 3 (just below the middle). That is, each time a participant one, we increased the level of their opponent, we decreased it when they lost and kept the same level for draws. We then randomly selected an opponent from the 5 agents on that level.

On a 2AFC trial, we presented a participant with a board position and two candidate options, and they indicated their preference by clicking on the corresponding candidate move (see figure 2.1B). We did not impose any time limits on participants' choices. To present participants with "interesting" choices, we selected board positions using a combination of two criteria. For a candidate position, we simulated moves from the behavioral model (with the modifications above) for each of the 38×5 agents fitted to human behavior. We then used inverse binomial sampling (see chapter 4) to measure the entropy of the move distribution for each agent as well as the mutual information between agent identity and the moves they made. We then selected positions for which the entropy, averaged across agents, exceeded a threshold and ranked them by mutual information. For each position, we determined the two most preferred moves across all agents, to present as candidate moves in that position. We computed the game-theoretic value of those moves, and selected the set of 82 positions that maximized the average mutual information while ensuring that each trial type (win-win, win-draw, etc) is represented equally (14 times). We presented the same positions to each participant, in shuffled order.

In the evaluation experiment, we presented participants with pre-arranged board positions and instructed them to indicate their expected winning chances on a 7-point scale (see figure 2.1C). Participants entered their rating by clicking one of 7 buttons. We labeled the first, middle and last button with "losing", "equal" and "winning", respectively. For the evaluation experiment, we selected positions using the same procedure as in the 2AFC experiment, except that in the final selection stage, we ensured that the game-theoretic values of the presented positions were equally distributed across winning, losing or drawn (28 each). Note that the 2AFC position set is biased towards objectively winning positions, since the position is winning if at least one move wins (so win-win, win-draw, or win-loss all count), whereas for position to be losing both candidate moves have to be losing. In the final selection, 61 positions are presented in both the 2AFC and evaluation experiments.

2.6.3 Eye tracking

In the eye tracking experiment, participants play against computer opponents for 20 minutes and performed 82 trials of the 2AFC experiment, with settings for both experiments identical to the generalization experiment above. For the entire experiment, we recorded their eye movements with a remote infrared video-oculographic system (EyeLink 1000; SR Research, Ltd., Mississauga, Ontario, Canada (Cornelissen et al., 2002)) with a 1 kHz sampling rate and $\approx$ 0.01 degree precision. We acquired eye position data with the EyeLink software using the "Heuristic filter ON" option. We displayed stimuli on a 21-inch Sony GDMF520 CRT monitor (resolution: 1280 $\times$ 960 pixels, refresh rate: 100 Hz). Subjects used a headrest located approximately 57 cm from the screen. We set the eye tracker to record events only, so that our data set consists of a time series of fixations, saccades and blinks.

On each session, we first calibrated the eye tracker with the built-in 9-point calibration method, but we also added to calibration conditions directly before and after the play-against-computer component of our experiment. In this calibration procedure, we presented an empty board with a white piece with a fixation cross on top of it on the bottom left tile (see figure 2.3A). We instructed participants to fixate on the cross and press the space bar when they felt their fixation was steady. After they pressed the space bar, the piece and the cross moved one tile to the right, instructing the participant to fixate on the next tile, which they again indicated with a space bar press. We continued moving the cross accordingly across all 36 squares, obtaining fixation coordinates and time stamps for the space bar presses for each square.

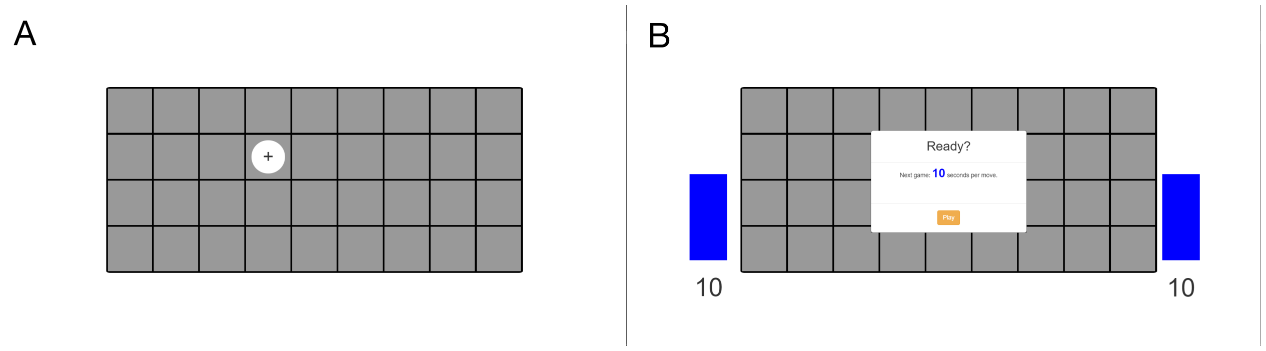


Figure 2.3 A. Screenshot of the calibration component in the eye tracking experiment. We instructed participants fixate on the black cross in the white circle, and press the space bar. The white circle and the cross then move to the next tile, until we have acquired fixation coordinates for all 36 tiles. **B.** Screenshot of the pre-game pop-up in the time pressure experiment. The time limit is indicated on the pop-up, and underneath the visual timer on the right of the board. Additionally, the height and color of the colorbar inform the participant of the time limit. During each participant's move, the colorbar on the right gradually shrinks while the text beneath counts down to zero. The same happens for the left timer during each computer's move.

2.6.4 Learning

The learning experiment consisted of 5 sessions. We required participants to schedule consecutive sessions with no more than 2 days in between. On the first, third and fifth session, participants played against computer opponents for 30 minutes and completed 60 trials each of the 2AFC and evaluation conditions. On the second and fourth sessions, participants only played against computer opponents for the entire 60-minute session. In all these sessions, the computer opponents were identical to those of the generalization experiment and the positions were selected using the same information criteria, with one difference. We selected 180 positions that we divided into 3 groups of 60, and ensured that the order of the days on which we presented these positions was counterbalanced across participants. The learning experiment contained one more session, on which participants performed a position memory recall task, which we do not analyze here. We compensated participants \$12 per session, with a \$12 completion bonus at the end.

2.6.5 Time pressure

In the time pressure experiment, participants played against computer opponents for 50 minutes, again with the identical procedure as the generalization experiment. However, in each game, both the human participant and the computer opponent had to obey a time limit of 5, 10 or 20 seconds per move. The time constraint was constant within each game and varied randomly between games. If a participant exceeded this time limit, the game ended immediately and counted as a loss. We also amended the "thinking time" for the computer, which we set to 2.5 seconds if the algorithm lapsed and otherwise chose according to the following equations

$$\begin{aligned}
t_{\text{base}} &= 4\sqrt{\gamma \cdot \text{iterations}} + \frac{1}{2} && (2.2)\\
t_5 &= 4 - \log\left[1 + \exp\left(4 - t_{\text{base}}\right)\right] && (2.3)\\
t_{10} &= 8 - 2\log\left[1 + \exp\left(4 - \frac{3}{5}t_{\text{base}}\right)\right] && (2.4)\\
t_{20} &= 16 - 4\log\left[1 + \exp\left(4 - \frac{3}{8}t_{\text{base}}\right)\right] && (2.5)
\end{aligned}$$

The functions relating t_{base} to t_5, t_{10} and t_{20} (plotted in figure 2.4) are all monotonically increasing, starting close to the origin and saturating at 4, 8 or 16 seconds, respectively. Therefore, these choices ensured that the computer would never use more than 80% of its allowed "thinking time".

To inform participants of their time constraint, we indicated the time limit for each game

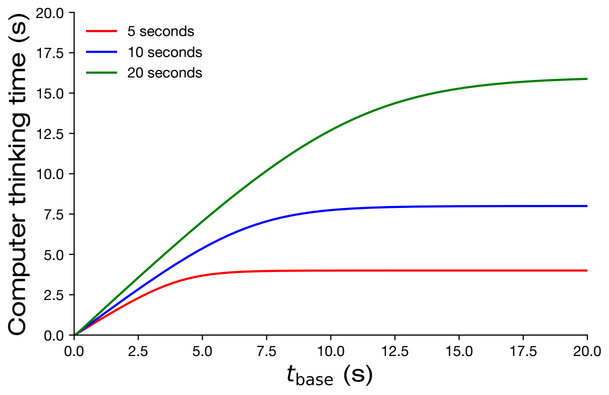


Figure 2.4 The functions that determine the computer's "thinking time" in the time pressure experiment. We first calculate t_{base} with the same formula we used to determine thinking times in the unconstrained experiments, then map t_{base} to a thinking time using one of three condition-dependent curves.

in a pop-up before the start of that game. Additionally, directly to the right of the board we displayed a timer, namely a colored bar that shrunk gradually as participants were contemplating their move and directly below it, we displayed a text-based count-down with the remaining thinking time in seconds (see figure 2.3B). In the 20-second condition, at the start of each move the colorbar was equally high as the board and linearly decreased to zero in 20 seconds. Initially, the bar was green, but when the participant had 10 seconds left it changed color to blue, and 5 seconds before the end it changed color to red. To warn participants even more of the passage of time, we played three warning sounds (short beeps) when the participant had 2, 1 or 0 seconds left, with increasingly higher pitch as time counted down. In the 5 or 10-second condition, we started the timer in the same state it would be in the 20-second condition after 10 or 15 seconds had elapsed (10 seconds: blue colorbar, half as high as the board; 5 seconds: red bar, quarter board height). When the computer was "thinking", we displayed a timer to the left of the board, with the identical behavior. The time warnings were largely effective, and participants lost on time in only 1.87% (33 out of 1766) of their games.

2.6.6 Turing test

The Turing test experiment consisted of two sessions, which we required to occur on consecutive days. On the first session, participants played against computer opponents for 60 minutes. In this experiment, the computer opponents used our exact behavioral model (the `main` model) with parameters inferred for participants in the human-vs-human experiment. As before, we ran an all-vs-all tournament between the 200 agents (5 agents each for 40 participants), and estimated each agent's playing strength with the Elo system. Likewise, we again selected 30 agents such that their Elo ratings approximately follow an arithmetic sequence, and divided them into 6 levels of 5 agents each. We then matched participants with computer opponents using the same one-up, one-down staircase procedure as in the generalization experiment.

On the second session, participants performed 180 trials of a classification task. On one trial of the task, we presented participants with a movie of a segment of a game played either by two players in the human-vs-human experiment, or two computers following the behavioral model with parameters inferred for those players. Participants could start the movie at any time by pressing a button labeled "Play" on the first trial or "Play next" on the remaining 179 trials. The video then played at a constant speed of 1.8 seconds per move. At the end of the video, we presented a slider with color ranging from black (on the left) to red (on the right). We labeled the left, middle and right of this slider with "Certainly computers", "No clue" and "Certainly humans", respectively. Participants moved the slider to some value, after which a button appeared with the label "Submit". Participants could submit their choice by clicking that button, or re-adjust the slider as often as they pleased. We converted the slider position into a classification judgment by measuring whether they positioned the slider closer towards “Certainly computers” or “Certainly humans”, and we defined confidence as the distance of the slider position from “No clue”. Note that we used discrete slider positions so that participant could never position the slider exactly at “No clue” and the classification judgment was always well-defined. After each trial, we provided participants with feedback whether their classification judgment was correct (“Correct!”) or not (“Incorrect.”).

We selected game segments to use for human-vs-human videos from games played in human-vs-human experiment, and computer-vs-computer videos using a similar sampling method. First, we created one video excerpt for each game in the human-vs-human experiment. For each game, we drew a number from a geometric distribution with rate 0.15 and selected the position that occurred in the game after that many moves. We then drew a maximum length for the segment from another geometric distribution with rate 0.1, and added moves from the game until the segment exceeded that maximum length or until the end of the game. For each game, we also generated a computer-vs-computer segment, starting from the same position, using the same maximum length. As before, we added moves from a simulated computer-vs-computer game until that segment exceeded the maximum length or the game ended. In other words, all computer-vs-computer video segments start from a position that occurred in a human-vs-human game, but all moves are made by the behavioral model. Finally, we selected a random subset of 90 games to use for human-vs-human videos and 90 others for computer-vs-computer videos.

After participants completed all 180 trials, we presented them with a post-task questionnaire, which consisted of one open-ended question: *How did you distinguish between humans and computers?*, followed by a closed-answer question: *For each of the following events, please indicate how much and in what direction it mattered in your decisions*

Q1 *A player made an irrational move*

Q2 *A player missed an obvious threat*

Q3 *A player won very quickly*

Q4 *A player was very defensive*

Q5 *A game began in an odd position*

Q6 *A player was making geometric patterns*

Q7 *A player's moves were very spread out*

Q8 *A player could have won but did not*

Q9 *A player made poor moves*

Q10 *A player's moves did not seem coherent*

2.7 Analysis methods

2.7.1 Playing strength estimation using Bayeselo

To estimate a player's playing strength from games against computer opponents, we use the Elo system (Elo, 1978), implemented using the Bayeselo algorithm. This algorithm treats the problem as a Bayesian parameter estimation problem, with a model that specifies the probability of a win, loss or draw given in a game where player i plays black against player j as:

$$\begin{aligned}
\mathbb{P}\left(\text{player } i \text{ wins}\right) &= f(R_i - R_j - R_b + R_d) && (2.6)\\
\mathbb{P}\left(\text{player } j \text{ wins}\right) &= f(R_j - R_i + R_b + R_d) && (2.7)\\
\mathbb{P}\left(\text{draw}\right) &= 1 - \mathbb{P}\left(\text{player } i \text{ wins}\right) - \mathbb{P}\left(\text{player } j \text{ wins}\right) && (2.8)
\end{aligned}$$

where R_i and R_j are ratings of players i and j, respectively, R_b captures the relative advantage of playing with the black pieces, R_d captures the tendency for games to end in a draw, and

$$f(\delta) = \frac{1}{1 + 10^{\delta/400}}$$

The algorithm takes as input a database of game results and estimates all ratings R_i as well as R_b and R_d jointly by maximum-likelihood estimation. Maximizing the log-likelihood is challenging and Bayeselo uses the Minorization-Maximization algorithm (Hunter, 2004). Note that the model predictions only depends on rating differences between players, so it is unaffected if we add a constant offset to all players' ratings. The Bayesian approach to playing strength estimation has the advantage of working for arbitrary amounts of game results (as opposed to other approaches that require all players to play equally many games), automatically computing and incorporating uncertainty in opponent's ratings, and dates back almost a century (Zermelo, 1929).

To ensure that the ratings of all players' ratings are calibrated across all experiments, we run Bayeselo on a database containing all human-vs-computer games and a simulated computer-vs-computer tournament in which each computer plays once against every other computer, including itself. The database contains 7897 human-vs-computer and 900 computer-vs-computer games. We also append to this database the outcome of games between the behavioral model with parameters inferred for human players and the computer opponents used in our experiments (note that the model and the computer opponents use a slightly different algorithm, as well as different parameter settings). In these games, we match the model with computer opponents using the same staircase procedure as used in all our experiments, resulting in 41451 model-vs-computer games. Therefore, the final database is considerably large (50247 games between 1770 real and virtual players) and we can estimate individual players' Elo ratings with sufficient precision to use it as a valid proxy for playing strength.

2.7.2 Specification of all 25 models

2.7.2.1 Main model

Our main model is an algorithm that instantiates our hypothesis for the thought process by which a human player decides on a move in a given board state. The core of the model is an evaluation function $V(s)$, which assigns a value to a board state s. The higher this value, the more likely the black player is to win from that state. The value function guides the construction of a decision tree with an iterative best-first search procedure.

Value function. The value function consists of two terms, the first of which measures whose pieces are closer to the board center:

$$V_{\text{center}} = \sum_{\vec{x} \in \text{Pieces(black)}} \frac{1}{\|\vec{x} - \vec{x}_{\text{center}}\|} - \sum_{\vec{x} \in \text{Pieces(white)}} \frac{1}{\|\vec{x} - \vec{x}_{\text{center}}\|}$$

where Pieces(p) enumerates the locations of all pieces that player p owns, $\vec{x}_{\text{center}}$ denotes the coordinate of the board center, and $\|\cdot\|$ is the Euclidean distance.

The second term counts how often particular patterns occur on the board (horizontally, vertically, or diagonally). A feature is a binary function $f_{t,x,y,o}(s)$ which returns 1 if a pattern of type t occurs at location (x, y) with orientation o, and 0 otherwise. We use the following 4 patterns:

1 *Connected 2-in-a-row:* two adjacent pieces with enough empty squares around them to complete 4-in-a-row.

2 *Unconnected 2-in-a-row:* two non-adjacent pieces which lie on a line of four contiguous squares, with the remaining two squares empty.

3 *3-in-a-row:* three pieces which lie on a line of four contiguous squares, with the remaining square empty. This pattern represents an immediate winning threat.

4 *4-in-a-row:* four pieces in a row. This pattern appears only in board states where a player has already won the game.

We define F to be the set of all such features (one for each type, orientation and board location), and associate a weight w to each feature in this set. The feature weight depends only on its type, not the orientation or location. Finally, we write the value function as:

$$V_F(s) = w_{\text{center}} V_{\text{center}} + c_{\text{black}} \sum_{i \in F} w_i f_i(s, \text{black}) - c_{\text{white}} \sum_{i \in F} w_i f_i(s, \text{white}) + \mathcal{N}(0, 1)$$

where $c_{\text{black}} = C$ and $c_{\text{white}} = 1$ whenever black is to move in state s, and $c_{\text{black}} = 1$ and $c_{\text{white}} = C$ when it is the opponent's move. The scaling constant C captures value differences between "active" and "passive" features. For example, a black three-in-a-row feature signals an immediate win on the black's move, but not on white's. The last term $\mathcal{N}(0, 1)$ represents additive Gaussian noise with mean zero and unit variance.

Search algorithm. -The search algorithm constructs a decision tree, consisting of nodes that contain a state s, the color of the active player in that state and a value associated to the state. Upon initialization, the value of a new node is set by calling the feature-based evaluation function. However, this value changes as the algorithm investigates the consequences of future play from that state. The algorithm starts with a single-node decision tree and gradually grows the tree. Each iteration, the algorithm selects a leaf node, expands it by adding one child node each for a number of candidate moves, and backpropagates the value of these new nodes recursively into the leaf node as well as its parents.

Algorithm 1: MakeMove(State s)

```
if Lapse($\lambda$) then
    return RandomMove($s$);
else
    DropFeatures($\delta$);
    root ←node($s$);
    while !Stop($\gamma$) and !Determined(root) do
        $n$ ←SelectNode();
        ExpandNode($n$);
        Backpropagate($n$);
    end
end
return $\text{argmax}_{c \in \text{children(root)}} c.val$;
```

Here, `Lapse` and `Stop` represent stochastic functions which return `true` with probability λ and γ, respectively, and `Determined` checks if the value of the root node (winning, losing or drawn) has been determined with certainty. `RandomMove`(s) returns a random legal move in state s. The `SelectNode` function determines the order by which nodes are added to the tree. We use best-first search, which selects a node by following the principal variation, in which both players always make the best moves according to the currently estimated values, starting from the root to a leaf node. Because the value of nodes in the tree changes constantly, so does the principal variation, and therefore the search algorithm can switch between exploring different branches of the tree on consecutive iterations.

Algorithm 2: SelectNode()

```
$n \leftarrow$ root;
while $children(n) \neq \emptyset$ do
    if $n.color = black$ then
        $n = \arg\max_{c \in \text{children}(n)} c.val$
    else
        $n = \arg\min_{c \in \text{children}(n)} c.val$
return $n$;
```

After the search algorithm has selected a leaf node to explore, it expands it by adding one child node for each legal move in the associated state. As it initializes the children, it automatically evaluates their states using $V(s)$ as defined above. The algorithm does not yet check whether either of these states is terminal (that is, either player has achieved 4-in-a-row or the board is full), but it effectively does so if $w_{\text{4-in-a-row}}$ is high enough. Next, the algorithm prunes unpromising children; those whose value difference with the best candidate move exceeds a threshold θ. Only afterwards does it assign $V = 10,000$ to each child state in which black has won, $V = -10,000$ if white has won and $V = 0$ for draws. It is therefore possible that, if $w_{\text{4-in-a-row}}$ is too low, or if the algorithm has dropped a 4-in-a-row feature in a relevant location, it will prune away an immediately winning move which can result in gross (but human-like) blunders.

Algorithm 3: ExpandNode(node n)

```
s← $n.state$;
foreach legal move $m$ in $s$ do
    n.AddChild(node($s+m$));
if $n.color = black$ then
    $V_{\max} = \max_{c \in \text{children}(n)} c.val$
else
    $V_{\max} = \min_{c \in \text{children}(n)} c.val$
for $c \in children(n)$ do
    if $|c.val - V_{max}| > \theta$ then
        RemoveChild($c$)
```

Next, the search algorithm incorporates the value of the newly created nodes into the decision tree with minimax backpropagation. After backprogagation, the value of each state reflects the search algorithm's best estimate of the result of a game starting in that state with perfect play from both sides.

Algorithm 4: Backpropagate(node n)

```
if $n.color{=}black$ then
    $n.val \leftarrow \max_{c \in \text{children}(n)} c.val$
else
    $n.val \leftarrow \min_{c \in \text{children}(n)} c.val$
if $n{\neq}root$ then
    Backpropagate (n.parent)
```

The search algorithm continues to run until the `Stop` routine returns `true`, after which it makes the best move according to its estimated values. Since the `Stop` routine is random and independently drawn each iteration, the total number of iterations follows a geometric distribution with parameter γ. When implementing our model as an AI algorithm to play against human opponents, we convert the number of iterations N into a 'thinking time' for the AI by $t = a\sqrt{N\gamma} + b$, where $a = 4$s and $b = 0.5$s.

The main model has 10 parameters: the pruning threshold θ, the stopping probability γ, the lapse rate λ, the feature drop rate δ, the active scaling constant C and the feature weights $w_{\text{center}}, w_{\text{connected 2-in-a-row}}, w_{\text{unconnected 2-in-a-row}}, w_{\text{3-in-a-row}}, w_{\text{4-in-a-row}}$. We do not add a parameter for the variance of the value noise, since changing the noise distribution from $\mathcal{N}(0,1)$ to $\mathcal{N}(0,\sigma^2)$ has the same effect as changing $\theta \rightarrow \frac{\theta}{\sigma}$ and $w \rightarrow \frac{w}{\sigma}$ for each feature. Therefore, adding σ would over-parametrize the model and by construction cause σ, θ and $\{w_i\}$ to be unidentifiable from data.

2.7.2.2 Lesions

Our first set of alternative models are lesion models, obtained by removing components from the main model, each of which can be implemented by fixing a parameter to a constant. The **No center**, **No connected 2-in-a-row**, **No unconnected 2-in-a-row**, **No 3-in-a-row** and **No 4-in-a-row** models are obtained by setting the respective feature weight to zero. The **No feature drop** model is obtained by fixing δ to zero, and the **No active scaling** model results from fixing C to 1. To obtain the **No pruning** model, we fix θ to $20,000$, which is larger than any value difference that occurs in practice and causes the model to never prune. Note that the model cannot compensate by increasing feature weights since their order of magnitude is yoked by fixing the value noise to have unit variance. Finally, the **No tree** model is achieved by fixing γ to 1. Therefore, the algorithm stops after 1 iteration, in which case it will have expanded only the root node, and its choice will be the highest-value child. Pruning lower-value children does not affect this choice, so θ is not a parameter in this model. We do not consider the model with λ fixed to 0, since a non-zero lapse rate is necessary to avoid halting problems in inverse binomial sampling.

2.7.2.3 Modifications

In our first modified model, **Fixed iterations**, we change the stopping routine `Stop` from a stochastic function to a deterministic function that returns `true` whenever the number of iterations has exceeded a constant N. In the **Fixed depth** model, we amend the search process to explore every branch up to a fixed depth D. In this model, the order is which nodes are selected to explore does not impact the algorithm's output. Moreover, we use alpha-beta pruning to eliminate branches from the decision tree whose evaluation can be proven not to affect the algorithm's output, in order to decrease runtime. In the **Fixed branching** model, we amend the pruning rule to keep the K highest-value children in each node (lowest-value when white is to move). If the expanded node has less than K children, the algorithm prunes nothing. Next, we consider removing the feature drop mechanism and instead applying a function in which each child is pruned with a probability ε in `ExpandNode` before the value-based pruning, resulting in the **Tile dropping** model. For the **Optimal weights** model, we restrict fix the feature weights $\{w_i\}$ a constant vector, which we chose by maximizing the Pearson correlation between $\tanh\left(\frac{V(s)}{20}\right)$ and the game-theoretic value $\tilde{V}(s)$ across all states s encountered by any player in the human-vs-human experiment.

Finally, we consider **Monte Carlo Tree Search** (MCTS). In this algorithm, instead of evaluating a state with $V(s)$, we perform a rollout; a simulated game starting from state s

between two agents that follow a myopic policy. That is, in state s', the agent chooses the move m that maximizes $V(s' + m)$, or the one that minimizes it when white is to move. We then assign a value of 1 to state s if the rollout results in a win for black, 0 for white wins, and $\frac{1}{2}$ if the game is a draw. Note that, since the evaluation function contains noise, the myopic policy and the outcome of the rollout are also stochastic. Note also that we perform only a single rollout when evaluating a state.

After performing a rollout, MCTS backpropagates by averaging rather than minimax, ensuring that the value of each intermediate node of the tree is equal to the average outcome of the rollouts conducted in all descendants of that node. We also amend the best-first selection rule $n = \arg\max_{c \in \text{children}(n)} c.\text{val}$ to the UCB formula

$$n = \arg\max_{c \in \text{children}(n)} c.\text{val} + C_{\text{exp}} * \sqrt{\frac{\log(n.N_{\text{rollouts}})}{c.N_{\text{rollouts}}}}$$

where $n.N_{\text{rollouts}}$ counts the number of rollouts that have been conducted in node n or any of its descendants, and C_{exp} is a parameter that controls the balance between exploitation (investigating high-value children) and exploration (investigating children that haven't been investigated much yet). Finally, after the tree search terminates, the algorithm makes a move by maximizing N_{rollouts} across all children of the root node.

2.7.2.4 Extensions

We create the **Orientation-dependent weights** by multiplying the weight of vertically or diagonally oriented features by scaling constants c_{vert} and c_{diag}, respectively. For the **Orientation-dependent dropping** model, we allow the feature drop rate for horizontally, vertically or diagonally oriented features to vary, whereas in the **Type-dependent**

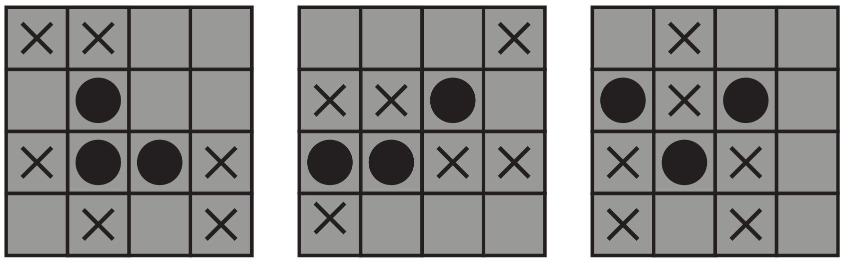


Figure 2.5 The triangle feature. A player is said to possess a triangle feature at a location of the board if their pieces form any of these three patterns (indicated with black pieces), while at least 3 out of the 6 relevant neighboring squares (indicated with crosses) are unoccupied. These features can occur in any orientation.

dropping, we let the drop rate depend on the feature type. In the **Triangle** model, we include a feature which counts how many times any of a set of 3-piece patterns (see figure 2.5) occurs on the board. Each of these patterns also includes a set of relevant neighboring squares, 3 of which need to be empty for it to count. Finally, the **Opponent scaling** model extends the main model by adding a scaling constant c_{opp} that multiplies weights of features belonging to the opponent. Note that opponent scaling and active scaling are dissociated since the former multiplies weights of the opponent's features regardless of whose move it is, whereas the latter is adaptive.

2.7.2.5 Controls

Our first control model, `Random-playout MCTS`, is identical to the MCTS model described above except that the playout policy is completely random (that is, in each state, all legal moves have equal probability). The `soft-max` model is given by

$$\mathbb{P}(\text{move on } x, y|s) = \frac{\exp\left[-\beta U(x,y)\right]}{\sum\limits_{(x',y')\in\text{Empty-squares}(s)} \exp\left[-\beta U(x',y')\right]}$$

where Empty-squares(s) iterates over all unoccupied squares in state s and $U(x, y)$ are fitting parameters which control the "utility" of moving on each square. We constrain these utilities to be symmetric, so that $U(x, y) = U(9 - x, y)$ and $U(x, y) = U(x, 4 - y)$, and note that we can fix the inverse temperature β to 1. without loss of generality. The resulting model has 10 parameters. For the `Opt-rand` model, we first compute the game-theoretic value of all states that can occur after one move. The prediction for a player's move in a given state is a mixture between a uniform distribution across all "optimal" moves (those that preserve the game-theoretic value), and a uniform distribution across all legal moves. The model has one parameter, since the mixing coefficients a_{opt} and a_{rand} are constrained to sum to 1. Finally, we compare our models to `Chance`, a zero-parameter model whose probability distribution of a player's move in a given state is uniformly random across legal moves.

2.8 Parameter estimation

The main model has 10 parameters: the 5 feature weights, the active-passive scaling constant C, the pruning threshold θ, stopping probability γ, feature drop rate δ and the lapse rate λ. We infer these parameters for individual participants and individual learning sessions or time limit conditions with maximum-likelihood estimation. Unfortunately, deriving the log-likelihood analytically requires marginalization of all latent variables (which features are dropped, the value at each node and the number of iterations in the search algorithm), which is intractable. Instead, we estimate the log-likelihood with inverse binomial sampling (de Groot, 1959), which is unbiased but noisy and computationally expensive (See chapter 4). Moreover, we cannot calculate gradients of the log-likelihood, so we optimize the log-likelihood function with multilevel coordinate search (Huyer and Neumaier, 1999), a gradient-free algorithm.

2.9 Computational time

One primary difficulty in data analysis is the computational time required for model fitting. Our end goal is to infer parameters for 25 models for a total of 330 data sets (human-vs-human: 40 participants, generalization: 40 participants, eye tracking: 10 participants, learning: 30 participants in 5 sessions, time pressure: 30 participants in 3 conditions). Since we use 5-fold cross-validation to estimate out-of-sample predictive accuracy, we need to optimize a log-likelihood $25 \times 330 \times 5 = 41,250$ times. The total computational time required to find the maximum-likelihood estimate for one data set is a product of 4 terms: the number of function evaluations used by MCS, the number of trials in the data set (note that each cross-validation splits contains $\frac{4}{5}$ of the trials in each original data set), the number of samples used by IBS to estimate the log-likelihood on a single trial and the time required for the model to make a single move in a single position. Each of these terms varies across data sets and across different runs on the same data set. In figure 2.6, we show the distribution of the number of function evaluations (mean and standard error: 1132 ± 38), the number of trials per data set (163 ± 2) and the number of samples per trial used by IBS (129.9 ± 0.6).

The final term that controls computational time of model fitting is the time used by the algorithm to make a move. The most time-consuming component of the algorithm is to compute the value function in each node of the decision tree, which requires a summation over all feature types, orientations and locations. Therefore, the algorithm's run time scales with the size of the decision tree (mean and standard error: 312.0 ± 0.4) times the

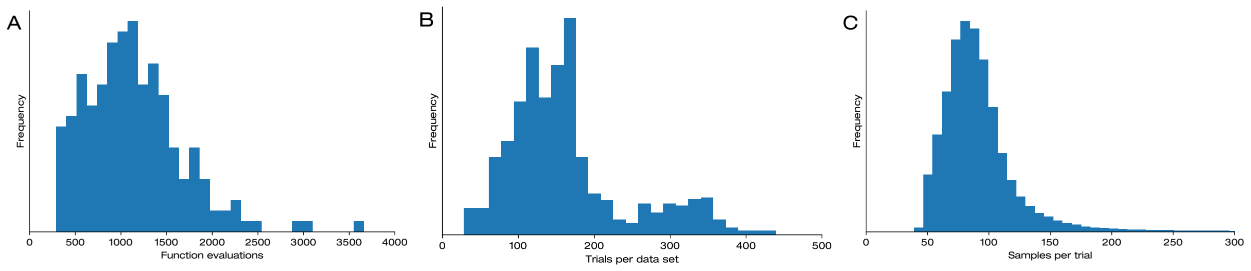


Figure 2.6 A. Histogram of number of function evaluations that MCS takes on a single optimization run of the main model. **B.** Histogram of the number of trials per cross-validation split across all 6 experiments we conducted. **C.** Histogram of the average number of samples that IBS uses to estimate the log-likelihood for a given trial and parameter vector. This average incorporates the effects of both the threshold and repeated sampling mechanisms in IBS.

number of feature instances of all types, orientations and locations (732 total). To speed up the computation, we do not compute the value of child nodes from scratch each time we expand a move, but compute the difference in value with the parent, which requires a summation over only features created or blocked by that move. In our implementation, detecting a feature at a certain location and orientation on the board takes equally many operations as determining its absence. Therefore, the computational time does not scale with the number of features present on any given board. Figure 2.7 shows the distribution of run times across moves made by the main model in all positions in the human-vs-human data set, and the correlation between run time and decision tree size (Spearman correlation coefficient: $\rho = 0.943, p < 0.001$, slope: 14.1μs per node). For this figure, we ran the model on a laptop with an Intel(R) Core(TM) i7-6700k CPU @ 4.0 GHz and 32GB SDRAM.

Together, these results imply that fitting all models to all data sets will require at least $25 \times 330 \times 5 \times 1132 \times 163 \times 129 \times 312 \times 14.1\mu\text{s} = 138.5$ years on the laptop. However, this total run time is an underestimate, since estimating parameters for most alternative models is more time-consuming than for the main model, partly because some models require

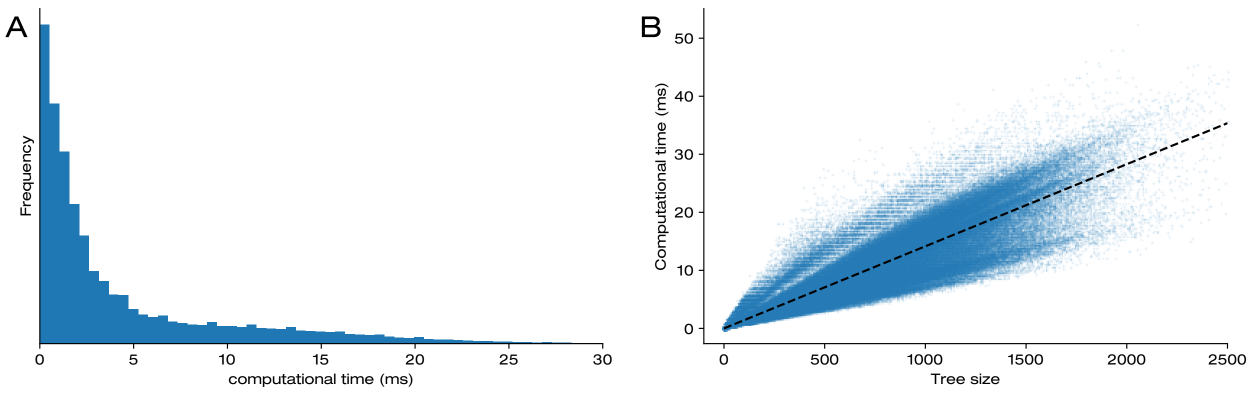


Figure 2.7 **A.** Histogram of computational time in milliseconds that the main model uses to make a single move in a single position. **B.** Correlation between computational time used to make a move on a single trial and the number of nodes in the decision tree that the model built on that trial. The dashed black line indicates the result of a linear fit, which has slope 14.1μs per node.

more computations (for example, the **Fixed depth** model in its worst case builds a decision tree of $\frac{36!}{31!} = 42497280$ nodes), partly because their log-likelihood is lower (for example in the **No 3-in-a-row** model) and IBS takes more samples. To run these computations in a reasonable amount of time (though still multiple months), we use a multi-threaded implementation of IBS with 20 CPUs running in parallel. Finally, we run these computations on the High Performance Cluster (HPC) at NYU, which processes approximately 10 such jobs simultaneously.

2.10 Reliability of parameter estimates

Before showing the model comparison results, we first validate our modeling tools by analyzing the reliability of estimated model parameters for individual players. Parameter estimates are noisy since inverse binomial sampling provides a noisy estimate of the log-likelihood given a parameter vector, and this noise causes the maximum-likelihood estimates found by multi-level coordinate search to vary. Uncertainty arises from the data being finite and relatively small (163 ± 2 trials per participant on average).

To estimate noise, we compute the Spearman correlation between inferred parameters on two independent fits of the main model to the human-vs-human data across participants (Figure 2.8A). All parameters except the lapse rate are correlated across fitting runs, but the correlations are far from perfect: the most reliable parameter estimates are w_{center} ($\rho = 0.93, p < 0.001$) and γ ($\rho = 0.83, p < 0.001$) and the least reliable is $w_{\text{4-in-a-row}}$ ($\rho = 0.44, p < 0.005$).

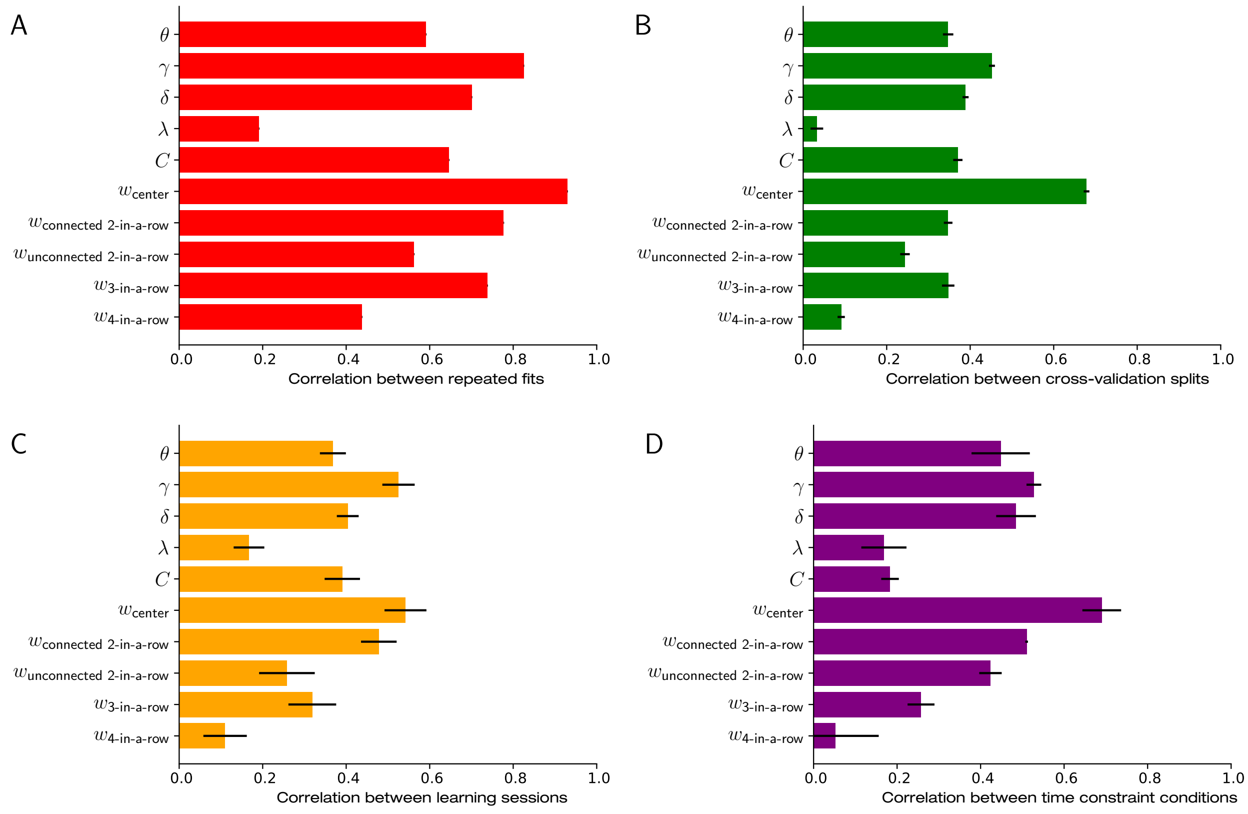


Figure 2.8 Spearman correlation across participants between model parameters estimated in **A.** two independent fits. **B.** different cross-validation splits. **C.** different sessions in the learning experiment. **D.** different time pressure conditions. The errorbars in **B**-**D** denote standard error of the correlations computed for each pair of splits, sessions or conditions.

To estimate uncertainty, we ideally would like to compute a posterior distribution over parameters given the data, or to perform maximum-likelihood estimation on bootstrap-resampled versions of the data. However, the former is impossible without an exact log-likelihood function, whereas the latter is computationally costly. Instead, we compute the Spearman correlation across participants of parameters estimated in the main model between data sets generated by cross-validation splits in all experiments (Figure 2.8B). This yields $\frac{4\times5}{2} = 10$ values for each parameter. To test whether these values are on average larger than zero, we perform a one-sample T-test. Again, all parameters except λ show positive correlations, and as before the most reliable parameters are w_{center} ($\rho = 0.679 \pm 0.007$ (mean and standard error), $p < 0.001$) and γ ($\rho = 0.452 \pm 0.007, p < 0.001$) whereas the least reliable is $w_{\text{4-in-a-row}}$ ($\rho = 0.091 \pm 0.009, p < 0.001$). Note that this analysis slightly overestimates parameter uncertainty, since cross-validation splits are $\frac{4}{5}$ the size of the original data whereas bootstrap-resampled data sets are equally large.

The correlations computed above are encouraging, but it is not surprising to find similar parameter estimates when fitting the same data twice. Additionally, each pair of cross-validation splits share $\frac{3}{4}$ of their trials, since both are $\frac{4}{5}$ the size of the original and $\frac{3}{5}$ of the original data is contained in both splits. To estimate the reliability of parameters estimated for each participant rather than each data set, we compute the Spearman correlation across participants between different learning sessions (Figure 2.8C) or different timing conditions (Figure 2.8D). As before, we test whether the average correlation is positive with a T-test. We now observe small or non-significant correlations for λ (learning: $\rho = 0.17 \pm 0.04, p < 0.005$, time constraints: $\rho = 0.17 \pm 0.05, p = 0.13$) and $w_{\text{4-in-a-row}}$ (learning: $\rho = 0.11 \pm 0.05, p = 0.076$, time constraints: $\rho = 0.05 \pm 0.10, p = 0.73$). As usual, the most reliable parameters are w_{center} (learning: $\rho = 0.54 \pm 0.05, p < 0.001$, time constraints: $\rho = 0.69 \pm 0.05, p < 0.001$) and γ (learning: $\rho = 0.52 \pm 0.04, p < 0.001$, time constraints: $\rho = 0.53 \pm 0.02, p < 0.001$).

Together, these results show that the parameters of the main model, and γ in particular, can be estimated reliably despite the small data size and noisy log-likelihood estimation. If we interpret the method of collecting data from a participant and inferring model parameters from that data as "measuring" a participant's tree size, the correlation across learning sessions demonstrates the test-retest reliability of this measurement.

2.11 Trade-offs between estimated model parameters

Aside from noise and uncertainty in parameter estimates, one may worry about parameter identifiability. The ideal method to investigate whether model parameters are identifiable given a finite data set D is to compute the posterior distribution $\mathbb{P}(\theta|D)$ or to sample from it using Monte Carlo Markov Chain (MCMC) methods. However, our inverse binomial sampling method to estimate log-likelihoods is noisy and cannot be used inside standard MCMC sampling algorithms. Alternatively, we could try computing a confusion matrix by performing a parameter recovery analysis:

- For each $i = 1 \ldots N_{\text{params}}$, generate parameter vectors θ^{probe} around $\hat{\theta}_{\text{real}}$ by varying θ_i while keeping all other coordinates fixed.
- For each such parameter vector, generate multiple fake data sets of the same size as D from the model.
- For each such data set, compute the maximum-likelihood estimate $\hat{\theta}_{\text{probe}}$.
- Compute the correlation across fake data sets between θ_i^{probe} and $\hat{\theta}_j^{\text{probe}}$.

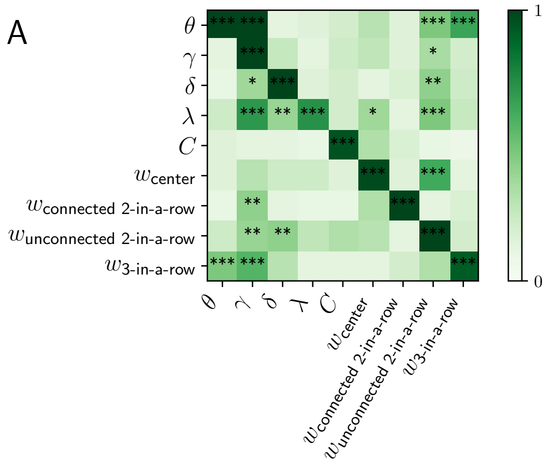
However, running a parameter recovery analysis requires $N_{\text{params}} \times N_{\text{probes}} \times N_{\text{fake data sets}}$

log-likelihood optimizations for each original data set D, which for our model is too computationally expensive. Instead, we investigate trade-offs by analyzing parameter estimates in the lesion models, each of which is equal to the full model with one parameter fixed to a constant. If two parameters i and j trade off against each other, the optimization algorithm will compensate for the lesioning of parameter i by increasing parameter j. Specifically, we make two predictions:

1. Fixing θ_i to a constant changes the distribution (across participants) of θ_j.
2. The change in $\hat{\theta}_j$ that results from fixing θ_i is correlated (across participants) with the value of θ_i in the unconstrained model.

To test for such trade-offs, we compute for each pair (i, j) the 2-sample Kolmogorov-Smirnov (KS) test statistic between the distribution of $\hat{\theta}_j^{\text{lesion } i}$ and $\hat{\theta}_j^{\text{full}}$ across participants in the human-vs-human experiment (see figure 2.9A), and the Spearman correlation coefficient between $\hat{\theta}_i^{\text{full}}$ and $\hat{\theta}_j^{\text{full}} - \hat{\theta}_j^{\text{lesion } i}$ (see figure 2.9B). The KS test reveals a mutual interaction between the 4-in-a-row feature weight and pruning, and unidirectional effects of lesioning the three-in-a-row feature or the decision tree on almost every other parameter. Additionally, the active scaling constant is affected by lesioning the feature drop mechanism or the connected two-in-a-row feature. The Spearman correlation reveals 7 additional significant interactions, only slightly more than the 5 expected from our chosen false discovery rate (100 tests with $\alpha_{\text{FDR}} = 0.05$). Finally, there are three parameter pairs for which both tests are significant ($\alpha_{\text{FDR}} = 0.05$): lesioning the three-in-a-row feature affects the feature drop rate and the weight of the connected two-in-a-row, and lesioning the decision tree affects the three-in-a-row feature weight.

These results show that parameters in our model are not fully distinguishable given our



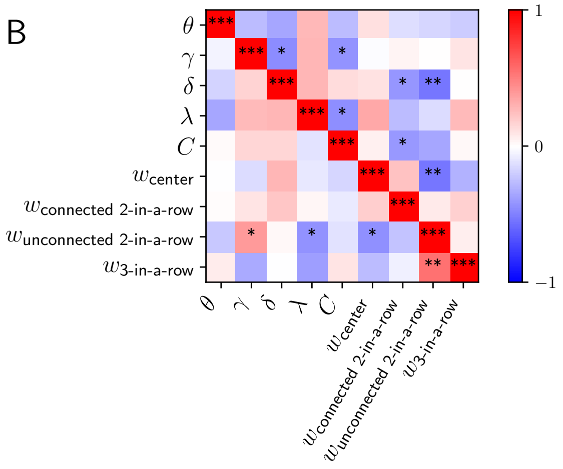


Figure 2.9 We detect trade-offs between model parameters using two approximate methods, since computing the full confusion matrix in a parameter recovery analysis is too computationally costly. **A.** 2-sample Kolmogorov-Smirnov (KS) test statistic between the distribution of $\hat{\theta}_j^{\text{lesion } i}$ and $\hat{\theta}_j^{\text{full}}$ for each pair of parameters. We indicate statistics that are significant at $\alpha = 0.05$, $\alpha = 0.01$, $\alpha = 0.001$ (all FDR-corrected) with one, two or three stars, respectively. **B.** Spearman correlation between $\hat{\theta}_i^{\text{full}}$ and $\hat{\theta}_j^{\text{full}} - \hat{\theta}_j^{\text{lesion } i}$ for each pair of model parameters, with significance levels as in **A**.

finite data size. This limits the strength of conclusions we can draw by model comparison and especially those derived from lesion models.

2.12 Model comparison on choice data

We now present the main result of this chapter: comparing the cross-validated log-likelihood of the main model averaged across all 6 experiments to that of 25 alternative models (figure 2.10). We test four categories of alternative models: *lesions*, generated by removing model components; *extensions*, generated by adding new model components; *modifications*, generated by replacing a model component with a similar implementation; and *controls*, which are structurally different from the main model. In section 2.7.2.1, we provide detailed descriptions of all 25 models.

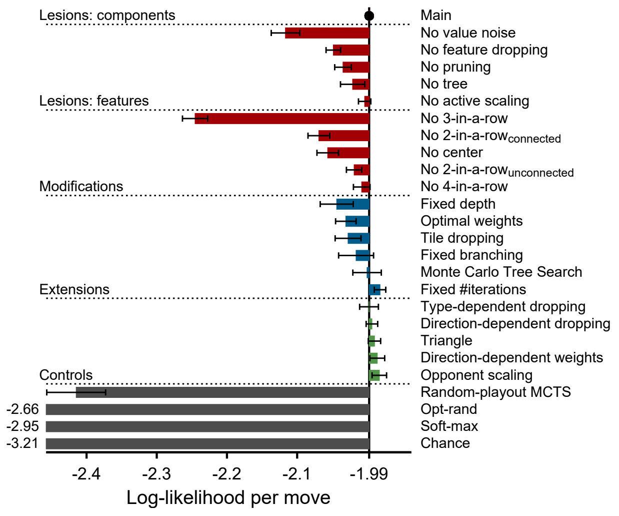


Figure 2.10 Mean log-likelihood across data sets

All lesioned models fit worse than the full model. The most impactful lesions are specific features (3-in-a-row, connected 2-in-a-row and center) and sources of variability (value noise and feature dropping). Lesioning the pruning mechanism or the entire tree search algorithm has a less dramatic effect, primarily due to parameter trade-offs (see section 2.9). Finally, some lesions (active scaling, unconnected 2-in-a-row and 4-in-a-row features) cause only small but significant impairments. Modifications also reduce the model's log-likelihood, except for Monte Carlo Tree Search (which removes the evaluation function and instead estimates state values by aggregating outcomes of simulated games) and a model with uses a fixed number of iterations in the tree search algorithm instead of the main model's geometric distribution. The extensions all increase the model performance, but only slightly. Finally, all control models fit much worse than the main model.

Together, these results show that participants' choices are consistent with a broad class of planning algorithms, which need to contain a feature-based evaluation function, tree search, pruning and a mechanism to capture attentional oversights. We select our main model as a representative of this class, which balances parsimony with predictive power.

Next, we show that log-likelihood differences between models are highly correlated between data from different experiments ($\rho_{\text{intraclass}} = 0.31$), demonstrating this conclusion is robust to experimental conditions (playing against other humans or computers, under time pressure, while head-fixed with the experimenter present for eye tracking).

2.13 Log-likelihood correlations across experiments

We selected our main model based on its high log-likelihood on the human-vs-human data, especially compared to alternative models. This raises two concerns. First, the main model and its alternatives are the result of a gradual tuning process, and we did not settle on final model specifications until after collecting the human-vs-human data set. This introduces researcher degrees of freedom (Simmons et al., 2011) into our analysis and may have caused us to unintentionally "overfit" to idiosyncracies of this data set. Assessing if and how much overfitting has taken place is impossible without conducting a registered replication (Simons et al., 2014). Second, there is no a priori guarantee that the thought process by which participants decide on their moves is the same between all experiments, given the variety of experimental conditions.

We can address both concerns by replicating the model comparison analysis on our other data sets: generalization, eye tracking, learning and time pressure. We conduct the model comparison analysis independently for each session of the learning experiment and each condition in the time pressure experiment (figure 2.11). We find that the cross-validated log-likelihoods are highly correlated across models between these different experiments (Spearman correlation: $\rho = 0.903 \pm 0.005, p < 0.001$, see figure 2.12). We next compute the log-likelihood of each model in each learning session relative to the log-likelihood

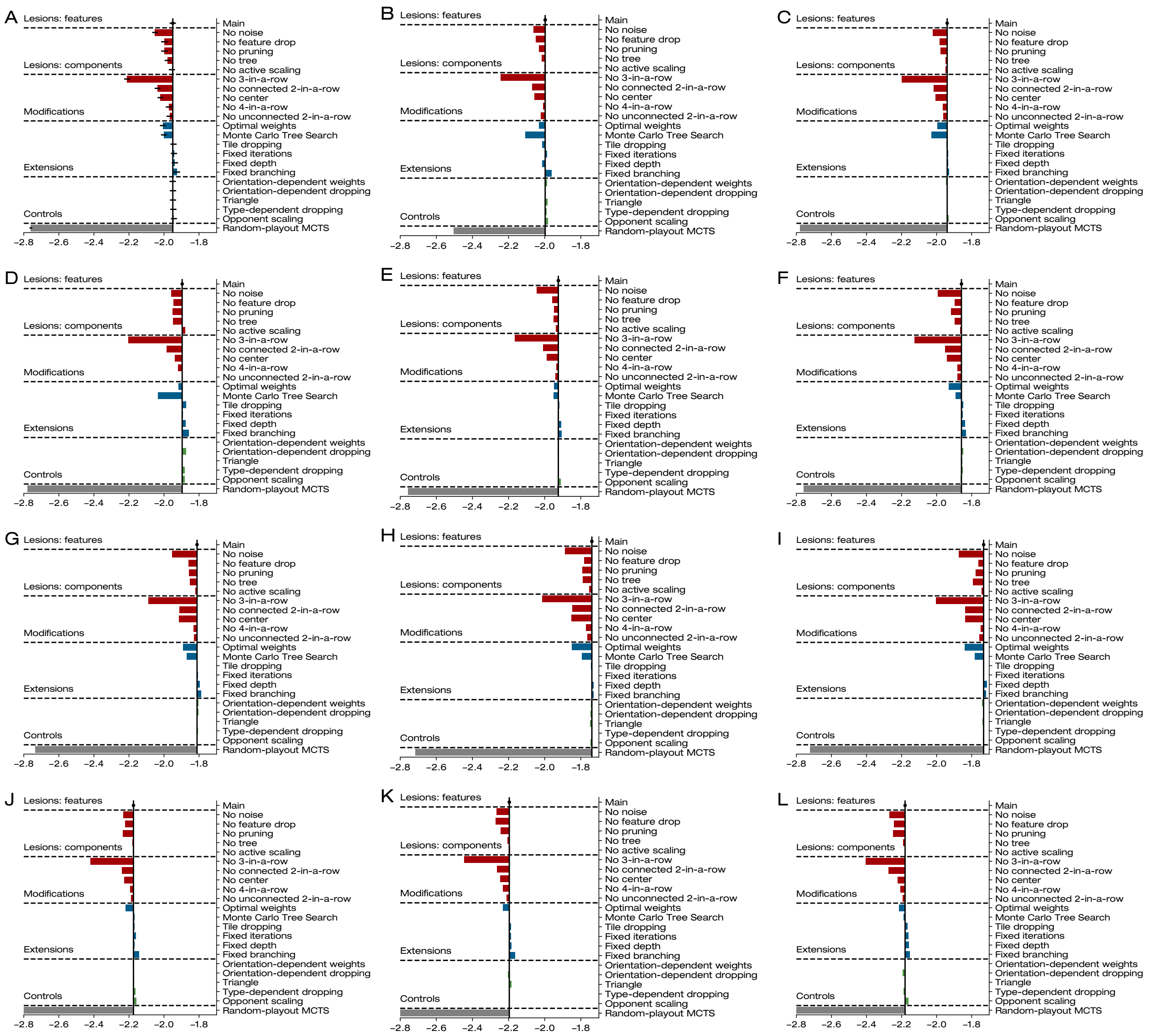


Figure 2.11 Breakdown of model comparison results across all data sets. **A.** Log-likelihood per move for each model across all data sets. This figure is identical to figure 2.10. **B.** Data from the human-vs-human experiment. **A.** Generalization experiment. **D.** Eye tracking. **E-I.** Learning session 1-5. **J-K.** Time pressure 5, 10 and 20-second conditions.

of the same model on the first session (figure 2.13A), or of the main model on the same session (figure 2.13B). We find that log-likelihood differences between models are stable as participants gain experience, and that all models have higher log-likelihoods for later sessions (since this data presumably has lower entropy). In the time pressure experiment, we

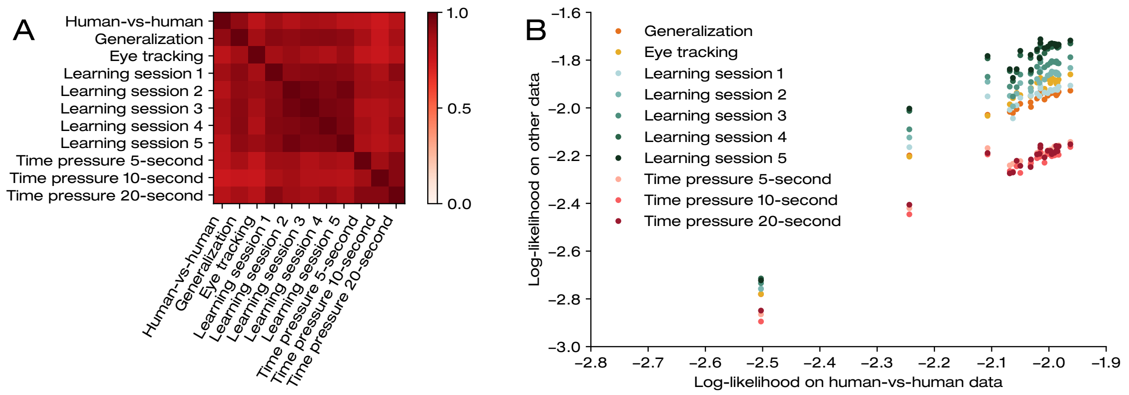


Figure 2.12 A. Spearman correlation across all 25 models of the log-likelihood per move on each pair of data sets. **B.** Scatterplot with the log-likelihood per move of each model on the human-vs-human data set versus its log-likelihood per move on each other data sets.

observe that log-likelihood differences are again stable with respect to time pressure conditions (figure 2.13C and figure 2.13D).

Together, these results show that our model comparison analysis replicates and that we did not abuse researcher degrees of freedom. Moreover, the main model provides an accurate yet parsimonious description of participants' choices throughout all experimental conditions.

2.14 Is the model good?

The model comparison analysis demonstrates that the main model fits human game play equally well or better than alternative models. This raises two questions: could it be that the main model fits the data as well as it possibly can, and if not, how does it fail to match the human-vs-human data? The upper limit of the log-likelihood equals the negative entropy of that participant's true distribution of moves given the board states they encountered. Unfortunately, we cannot estimate the data entropy, because participants

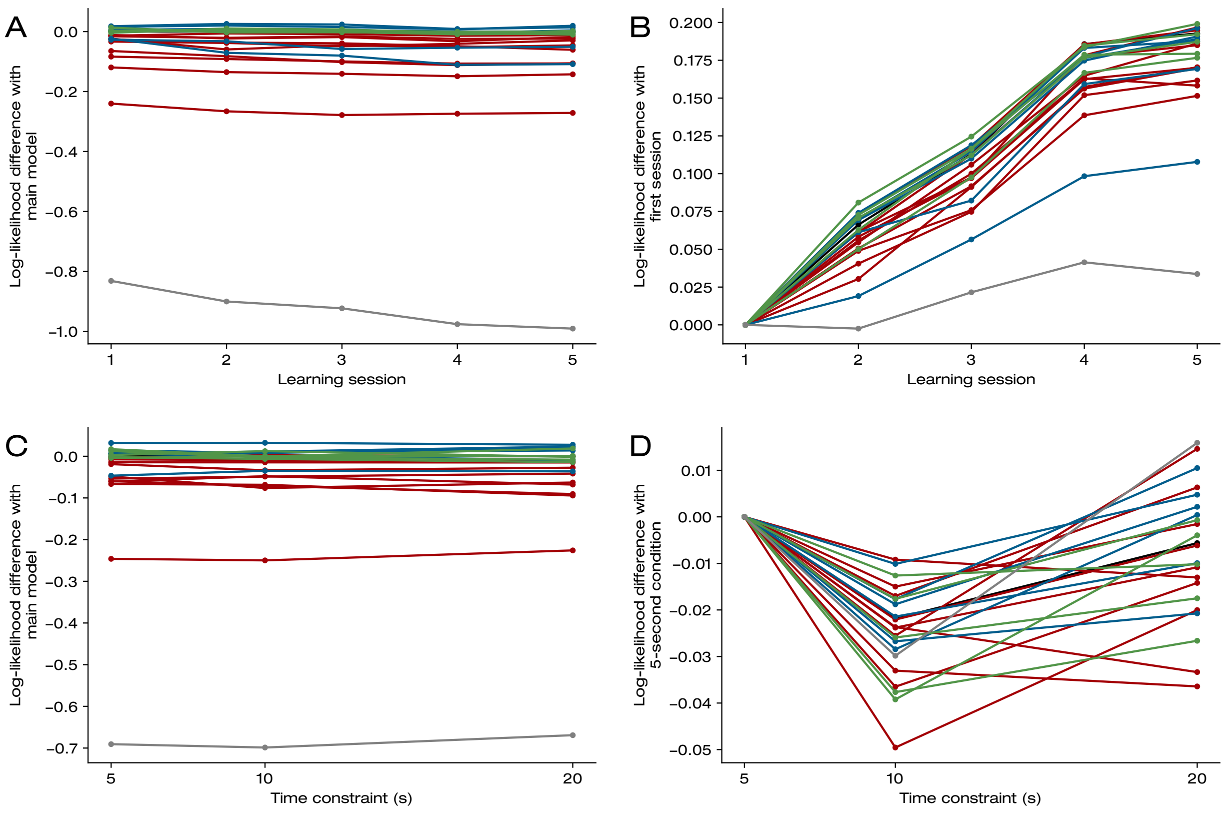


Figure 2.13 A. Difference in log-likelihood per move between the main model and all other models in all sessions of the learning experiment. **B.** Difference in log-likelihood per move between the first learning session and all other session, for all models. **C,D.** Same for the time pressure experiment.

rarely encounter the same board state twice. Therefore, we use a number of approximate techniques to assess the model's *absolute goodness-of-fit* and potential *model mismatch.*

2.15 Summary statistics predicted by the model

First, we measure summary statistics of the human-vs-human data as well as predictions from the main model or a random agent. These statistics serve both purposes. If we find a mismatch between the model and the data on one or more statistics, that would prove that the model provides an imperfect fit. Moreover, we can investigate the deviations be-

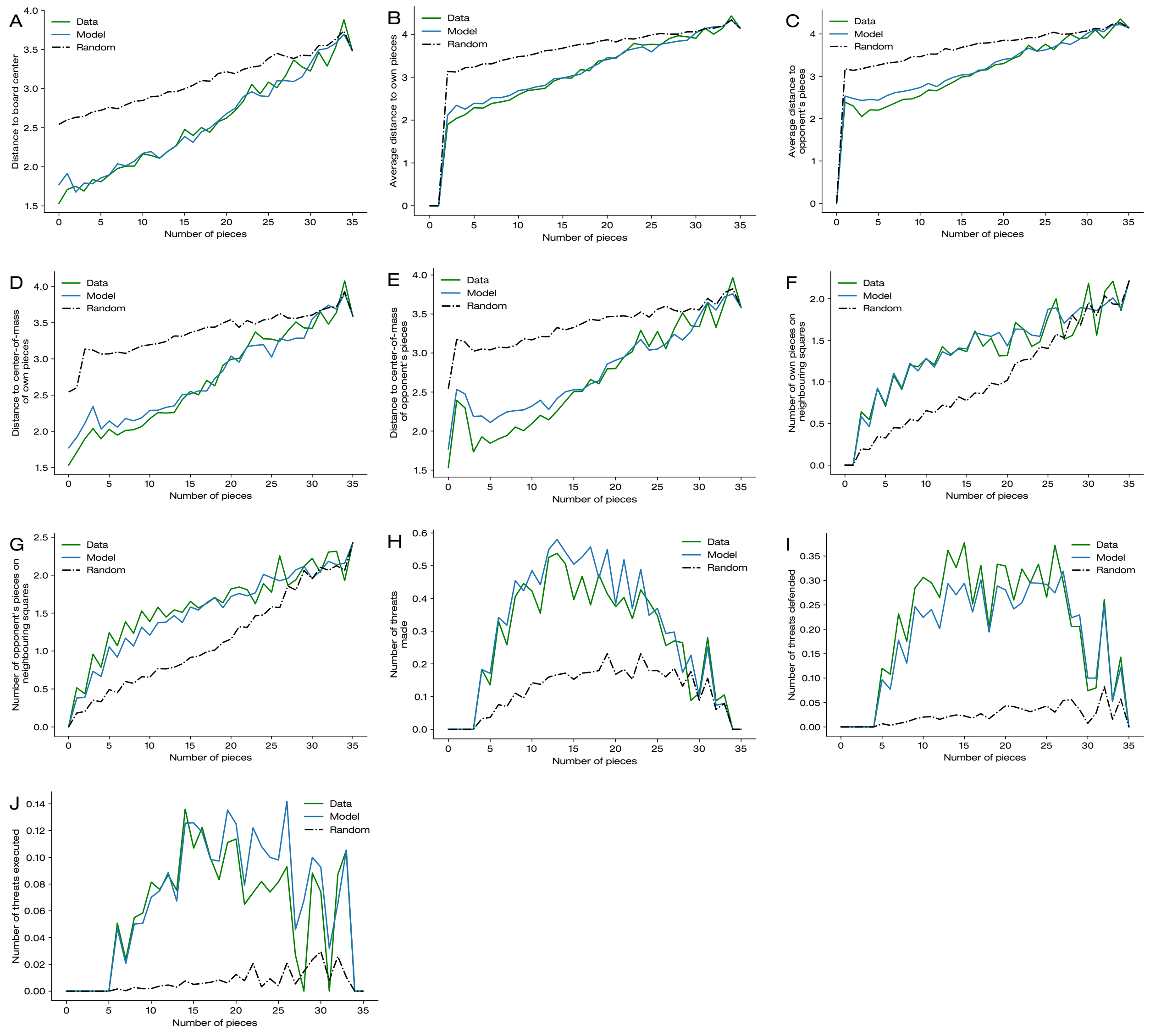


Figure 2.14 Summary statistics computed as a function of number of pieces on the board, for moves made by human players (green solid lines), the behavioral model with inferred parameters (blue lines) or random moves (black dashed lines). These graphs depict cross-validated predictions. For all statistics, the model prediction is much closer to the human data than random, although some deviations remain.

tween the model and data to design model improvements or generally understand why the model fails.

For each move made by each human player, we compute 7 statistics: the distance from the chosen square to the nearest pieces owned by that player, the distance to the nearest piece of the opponent, the distance to the the center of mass of that player's pieces, the distance to the center of mass of the opponent's pieces, the distance to the center of the board, the number of that player's pieces on the 8 squares neighboring the chosen square (including diagonals), and the number of opposing pieces on neighboring squares. Additionally, we indicated whether with their chosen move, the player created a threat to make 4-in-a-row on the next move, parried a direct threat from their opponent, or executed a threat that the opponent failed to defend on their previous move. We also computed these statistics for moves made by the main model in the same position, as well as random moves. For the main model, we use parameters inferred for that player on the cross-validation split that excludes that particular trial. We report the average of these 10 statistics as a function of the number of pieces on the board, aggregated across all participants. All of these summary statistics are substantially different from the prediction based on random moves, and the main model predicts that difference almost exactly (figure 2.15).

Furthermore, we calculate the mean and standard error of each of these summary statistics for each player, averaging across all positions they encounter. As before, we also compute the mean summary statistic for each player from moves made by the model in the same positions. However, since different players encounter different positions, a direct comparison between the model prediction and the data is confounded. Therefore, we plot the difference between the data and the random prediction against the difference between the model and random (figure 2.15). We then observe strong correlations across participants between the model prediction and the data (Spearman correlation: $\rho \in [0.70, 0.94], p < 0.001$ for all summary statistics).

Although the model captures variance in these summary statistics between participants

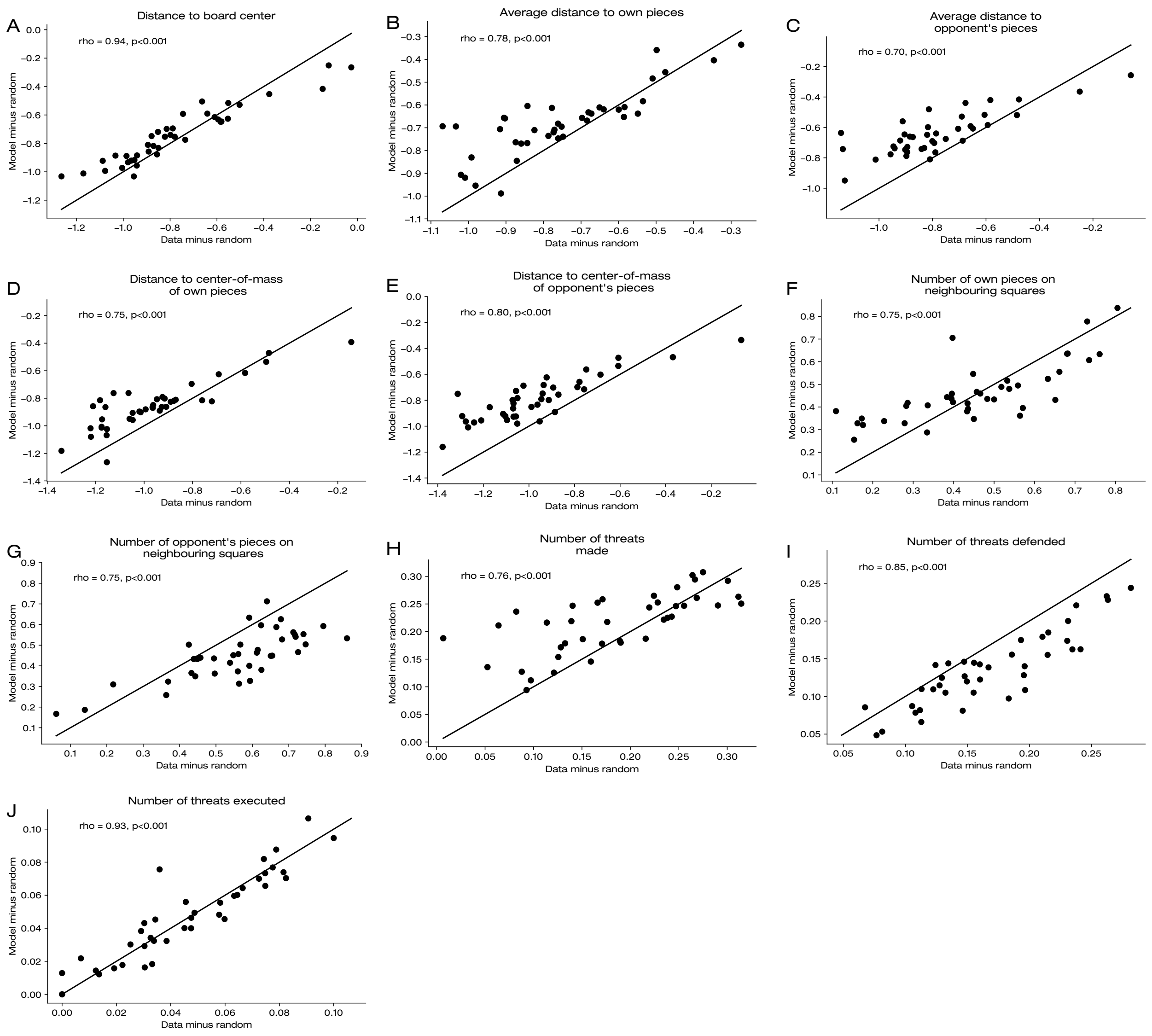


Figure 2.15 The model predicts individual differences in summary statistics. Each panel shows a scatterplot for each summary statistic, where each point represents a participant in the human-vs-human experiment, the x-coordinate the statistic computed on that participant's moves and the y-coordinate the statistic computed on moves made by the model. For both the x and y-axis, we subtract the statistic computed on random moves. The inset in each panel shows the Spearman correlation coefficient for that scatterplot.

(figure 2.15), and their evolution over the course of a game (figure 2.14), we also observe multiple consistent deviations from the data. The model moves further away from already present pieces on the board, especially the opponent's pieces, and especially in the early game (see figures 2.14B-G and 2.15B-G). This can partly be explained by regression to the mean: the model over-estimates participants' decision noise which causes its predicted move distribution to be too diffuse, and contain more probability mass away from already placed pieces.

Additionally, the model makes more threats but is less likely to defend existing threats (see figures 2.14H,I and 2.15H,I). These summary statistics reveal an even more striking mismatch between the model and the data in a specific scenario. Namely, when the opponent forms two immediate winning threats, people invariably ($87.7 \pm 0.7\%$ of the time) block one of the threats. By contrast, the model (assuming it has not dropped the features associated with either of these threats) will realize that the position is lost and chooses another move - often making a threat of its own. Since participants can easily identify these positions as lost, and end up losing $96.1 \pm 1.4\%$ of these games, their choice to block one of the two threats is not particularly rational.

Second, we compare each participants' task performance (estimated with the Elo system (Elo, 1978), see Methods) with that of the model with parameters inferred for that participant (figure 2.16A). The model consistently underestimates the playing strength of stronger participants. This is expected, as the parameter optimization algorithm regards predictable strategies outside the model space as unpredictable (i.e., random) and therefore overestimates sources of noise (value noise, feature dropping and lapses).

Next, we calculate the log-likelihood of all models as a function of the number of pieces on the board (figure 2.16B). All models fit much better than chance in the mid-game, but only a little above chance in the early game (moves 1-6) and approximately at chance in the endgame (move 30 onwards). Our main model outperforms the other models at all stages of the game, suggesting that the model mismatch is shared between all models, in-

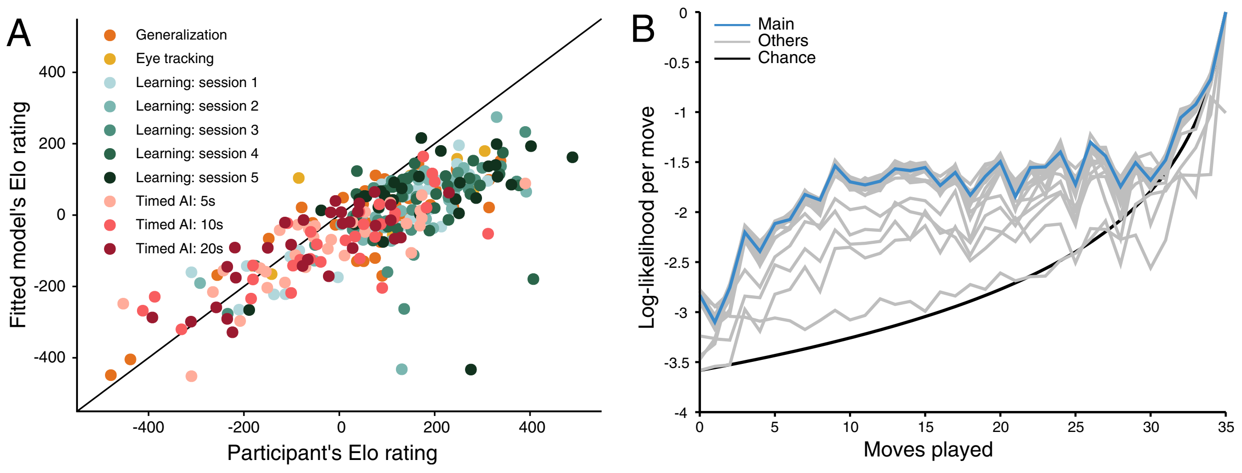


Figure 2.16 A. Scatterplot, in which each point represents an individual participant, or one participant in one learning session or time constraint condition, the horizontal coordinate indicates their Elo rating and the vertical coordinate the Elo rating of the main model with parameters estimated for that participant. The solid black line indicates equality. **B.** Log-likelihood per move as a function of the number of moves played, for the main model (blue), all alternative models (gray) and chance (black). The main model outperforms or matches all other models at every stage in the game.

cluding modifications, extensions and controls.

2.16 Opening moves mismatch

One specific pattern of the data that the model fails to capture is the distribution of moves people choose in the initial position (Figure 2.17A). First of all, in the human-vs-human experiment, people start the game by moving in a corner 13.6 $\pm$ 3% (mean and standard error across participants) of the time, and 3 participants do so over 80% of the time. By contrast, the non-corner squares on the short edge of the border (just above or below a corner) are much less popular: 38 out of 40 participants never move there at the start of the game, one participant does so only once, and one participant 3 times (out of 10 games). However, for the model the frequency of moves on corner squares is 4.8 $\pm$ 0.6% and that of non-corner short-edge squares is 7.7 $\pm$ 0.9% (see figure 2.17B). In our other

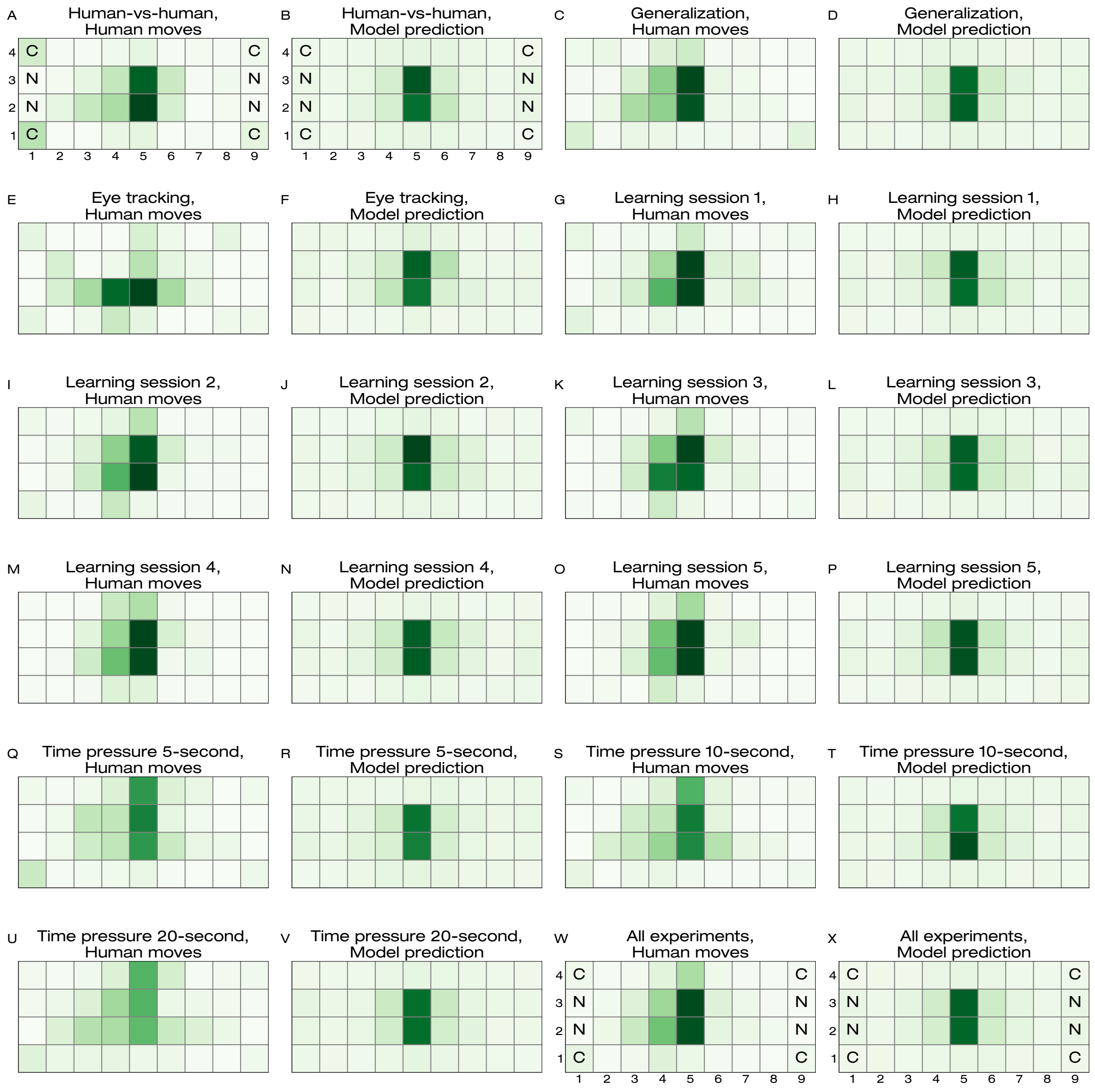


Figure 2.17 Distribution of opening moves in real data (first and third column) and predicted by the model (second and fourth) for each data set as well as averaged across all data (bottom right). In the first two and last two panels, we indicate with the letter "C" and "N" the corner squares and non-corner short-edge squares, respectively. Note that, across all experiments, people tend to prefer "C" squares over "N" squares, whereas the model fails to capture that preference.

experiments, we observe the same pattern, although to a lesser extent: people consistently prefer corner squares over non-corner short-edge squares, whereas the model predicts the opposite (figure 2.17C-X). Finally, the model fails to capture individual participants' preference for corner moves (figure 2.18A), non-corner short-edge squares (figure 2.18B) or the difference thereof (figure 2.18C)

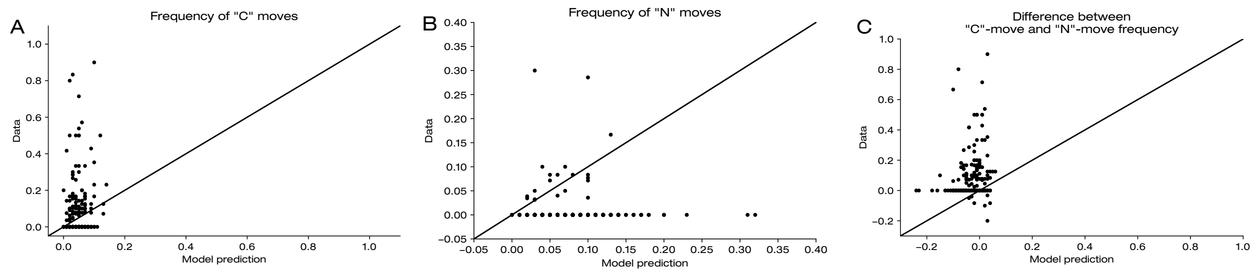


Figure 2.18 A. Scatterplot of the frequency of moves on corner squares ("C" moves) predicted by the model for each participant in the human-vs-human experiment, and that observed in the data. **B.** Same for the frequency of "N" moves. **C.** Same for the difference between the frequency of "C" and "N" moves.

People's preference for corner squares on the initial move is irrational (it is among the weakest possible moves) and idiosyncratic: it is inconsistent across participants or experimental conditions, it disappears for later moves in the game, and it is not obvious what the analogue of corner moves would be in other games. Moreover, it is unclear how to modify the planning algorithm of our model so that it would play on corner squares, so adapting the model to explain people's preference would likely require a special case for the initial position.

Aside from this unusual preference of playing on corner squares, people also are biased to start their games by playing towards the left-hand side of the board (Figure 2.17A). Across all participants in all experiments, the mean x-coordinate of their initial moves is 4.59 ± 0.04 (mean and standard error), which is significantly different (one-sample T-test, $p < 0.001$) from the board center located at $x = 5$ (figure 2.19). There is no substan-

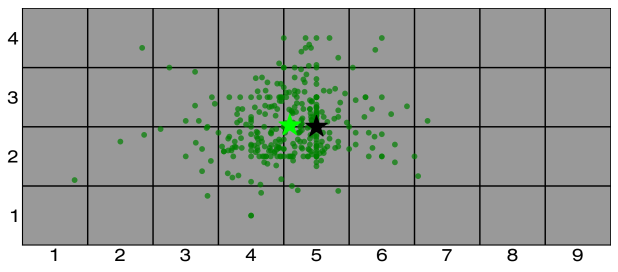


Figure 2.19 People's opening moves are biased towards to left of the board. Each green dot indicates the mean location of one participant's opening moves in any of our experiments, the black star the board center, and the light green star the mean location averaged across all participants.

tial bias in the vertical dimension. This bias too is irrational, since the game is symmetric. Since this symmetry is hard-coded into our model, it (or any variants created by modifying the planning algorithm) cannot capture people's leftward bias. Instead, one would need to add asymmetric features. It is interesting to speculate what causes this bias and why people prefer specifically the left-hand side. One hypothesis is that it may depend on their handedness or the direction of writing in their native language. Unfortunately, we did not record either data, but it is worth noting that all our participants were fluent in English, and the majority preferred using a right-hand computer mouse.

These results demonstrate the the main model does not match all aspects of the data and therefore can be improved. However, the specific patterns that the model fails to capture, the corner-move preference and the leftward bias in initial moves, seem unrelated to the search process. Therefore, it is unlikely that extending the model to account for these statistics will alter our conclusions about the computational principles underlying sequential decision-making.

2.17 Turing test

Finally, we conducted a "Turing test" experiment in which human observers classified videos of game segments taken from human-vs-human or computer-vs-computer games and reported the confidence of each of their classifications (see methods). To provide observers with maximal opportunity to learn any distinguishing criteria between human and computer play, we ensured that observers had prior experience with the computer opponents (60 minutes of game play), and we showed feedback after each Turing test trial. We presented all videos at a constant playback speed and ensured that the initial positions are balanced between the two groups. On average, human observers classify video segments correctly as human-vs-human or computer-vs-computer $55.4 \pm 0.9\%$ of the time (inter-quantile range: 52.9-59.3%). Moreover, observer's accuracy correlates on single trials with their reported confidence (logistic regression predicting accuracy given confidence: $\beta = 1.11, p < 0.001$, see figure 2.20).

The accuracy-confidence correlation suggests that, if we aggregate classification judgments across observers by a voting rule and especially one that gives preference to more confident observers, we can leverage the wisdom of the crowds and the accuracy of the group will exceed that of the average observer. We try three such voting rules. First, we set the group response in each position to that of the majority of observers. Second, we use a majority rule in which each observer casts an amount of votes proportional to their confidence. Finally, we set the group response to that of the majority of the 5 most confident observers. Indeed, all these voting rules yield group accuracies above the average observer (majority: $57.2 \pm 3.7\%$, confidence-weighted majority: $61.1 \pm 3.6\%$, most confident observers: $65.0 \pm 3.6\%$).

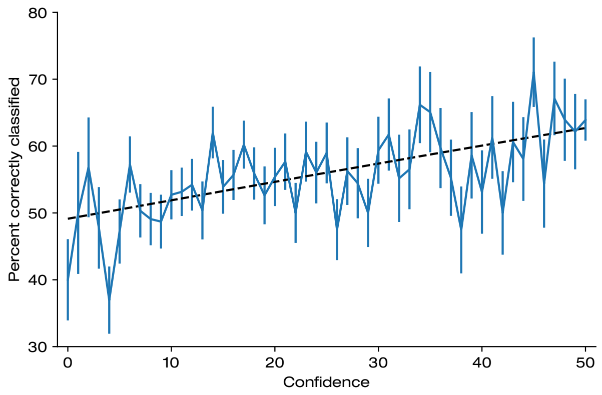


Figure 2.20 Confidence-accuracy correlation in classification judgments in the Turing test. We bin all trials from all observers by confidence ratings in bins of size 1, and compute accuracy for each bin. Error bars denote standard error across all trails in each confidence bin. The dashed line is a logistic curve fit of the raw data.

To test whether the variability in classification accuracy across observers reflects intrinsic differences the quality of their classification judgments or could have arisen by chance, we compare the distribution of accuracies against the prediction from a null model (figure 2.21A). In the null model, each observer independently guesses the identity of each video by consulting an oracle which returns the correct answer with probability 0.554. Since these distributions match almost exactly, we find no evidence that some participants are better observers than others. For example, one might hypothesize that participants who excel at playing the game are also better at classifying segments, but we find no correlation across participants between Elo rating and classification accuracy (Spearman $\rho = 0.161, p = 0.40$). More generally, it bears emphasizing that the results of the Turing test depend crucially on the observer's skill, and if we for example had included more human-vs-computer playing sessions, the results could have changed.

Using the same method, we find that judgments of different observers for the same video segment are not independent (figure 2.21B). There are 5 video segments that almost all

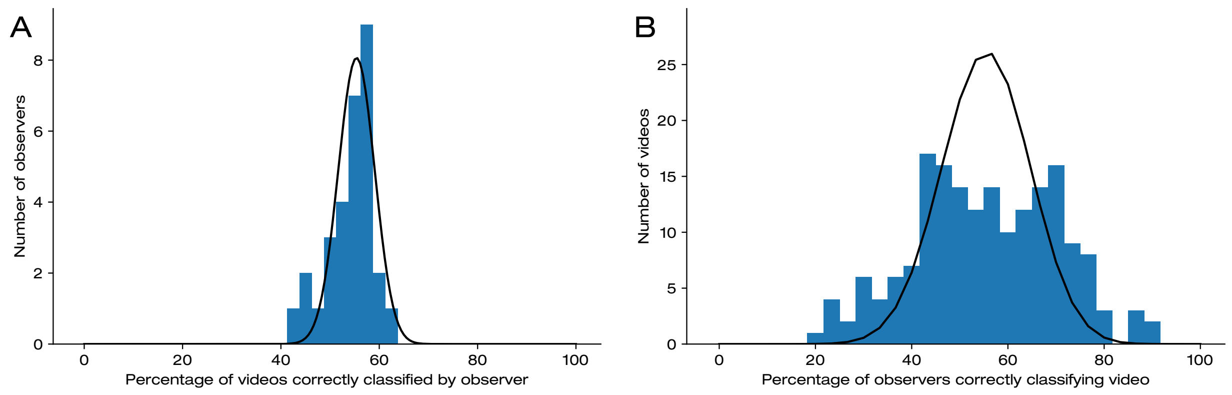


Figure 2.21 A. Histogram of classification accuracy (the percentage of correctly classified videos) for each observer (blue bars) and the probability density function of a binomial with $p = 0.554$ and $N = 180$. The histogram matches the curve, demonstrating that individual variability in the classification accuracy does not exceed that expected by chance. **B.** Histogram of classification accuracy for each video (blue bars), with a binomial pdf with $p = 0.554$ and $N = 30$ (black line). We do find that some videos are correctly classified by more participants than expected from chance alone.

observers identify correctly, and there is a striking lack of segments on which observers as a group are uncertain. This suggests that human observers base their classification decisions on some shared features of these video segments (not to be confused with features in the value function of the behavioral model). One such feature is the video length; observers reliably classify longer game segments as coming from human players. Human observers are correct to use this feature, since human-vs-human games are indeed on average longer than computer-vs-computer ones (10.73 ± 0.69 versus 8.02 ± 0.61 moves, independent 2-sample t-test: $p < 0.005$).

To discover what other features human observers base their judgments on, we perform a multiple logistic regression to predict each observer's classification judgments given 6 manually defined candidate features:

- The video length

- The number of pieces on the first board in the video
- Whether either player failed to block an immediate winning threat
- Whether either player failed to execute a possible immediate win
- The fraction of errors (from either player) in the game segment
- The average log-likelihood of both players' moves given the best-fitting behavioral model, which measures how surprising or irrational the players' moves are
- The average Euclidean distance between consecutive moves, which measures how "spread out" the players' moves are

Before running the logistic regression, we normalize each feature by subtracting its mean across all video's and dividing by the standard deviation. The logistic regression correctly predicts % (mean and standard error) of human observers' classification judgments, showing that participants indeed use at least some of these features. Investigating the regression coefficients (see table 2.1), we find that participants base their decision mostly on video length and failure to block threats, and largely ignore the Euclidean distance and log-likelihood. We next compare these coefficients from the logistic regression that predicts observers' judgments to those from a logistic regression that predicts the true video type (human-vs-human or computer-vs-computer), and find a strong correlation (Spearman $\rho = 0.857, p < 0.05$, see figure 2.22A). Therefore, people's usage and weighting of different features is rational.

Finally, we compare these regression coefficients to observer's responses in the post-task questionnaire (see methods). Out of the 30 observers, only one did not complete the questionnaire. To summarize observers' self-reported classification strategies, we show a word

Feature	β_{judgment}	β_{true}	Self-reported use
Irrational move	-0.047	0.367	0.228
Failure to block	0.075	0.382	0.383
Number of starting pieces	0.177	0.207	0.124
Euclidean distance	0.079	0.098	0.231
Failure to win	-0.168	0.271	0.410
Error rate	0.027	-0.300	0.117

Table 2.1 Values of the coefficients in the regressions predicting observers' classification judgments (first column), or the true video type (second column). The third column shows the values of self-reported feature use, normalized to a scale from -1 to 1.

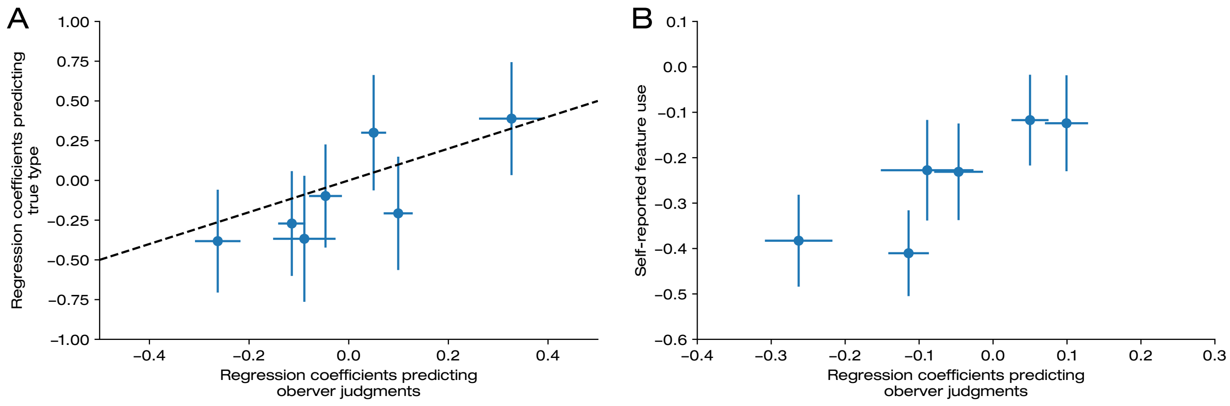


Figure 2.22 **A.** Scatterplot in which each point represents a video feature, the x-coordinate that feature's coefficient in the logistic regression predicting participants' responses using these features and the y-coordinate the regression coefficient when predicting the correct video type. Horizontal errorbars indicate standard error across observers, vertical errorbars indicate 95% confidence intervals in the logistic regression. The dashed line denotes equality. **B.** Same as **A**, except the y-coordinate represents observers' self-reported feature usage in the post-task questionnaire. In this plot, all errorbars indicate standard error across observers. We omit an equality line regression coefficients and self-reported feature do not share the same units. Note that the both the x and y axis cover asymmetric ranges.

cloud indicating their most commonly used words (figure 2.23), and in Appendix A we include all observers' responses. For each of these 9 options, participants responded by moving a slider between "*Humans*" and "*Computers*". Most of these questions are linked to specific features in our logistic regression (Q1: log-likelihood, Q2: failure to block threats, Q5: number of starting pieces, Q7: Euclidean distance, Q8: failure to win when possible, Q9: Error rate), whereas others (Q3, Q4, Q6 and Q10) we did not operationalize as fea-

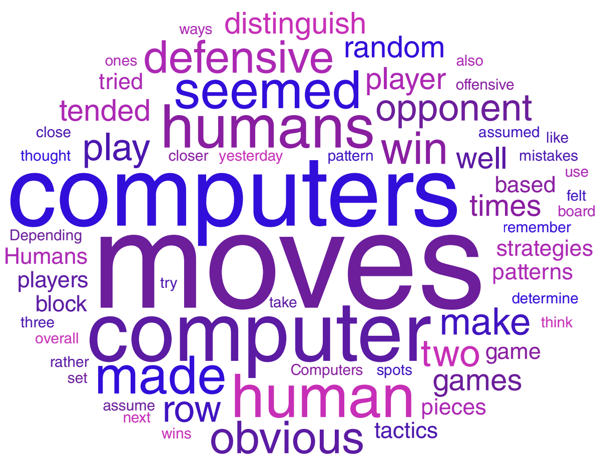


Figure 2.23 Word cloud of the most common words used by our participants in the Turing test post-task questionnaire. The font size of each word is proportional to its frequency in participants' responses. This figure was created using `wordclouds.com`.

tures. For the former 6 features, the weights in the logistic regression that predicts observer's classification judgments correlates with the average amount that observers report using that feature ($\rho = 0.829, p < 0.05$, see figure 2.22B). However, regression weights and self-reported feature use don't correlate across individuals. Moreover, for each feature, observers indicated that its presence was more likely to cause them to report "human", demonstrating a bias in their self-reported feature use. Therefore, participants are aware of the features they use to classify videos, but only to a limited extent.

Finally, we calculate a lower bound on the model mismatch between the behavioral model and the human-vs-human data using the length-accuracy curve for two classifiers: the logistic regression optimized for predicting true video type, and the human observers' responses aggregated by confidence-weighted voting. We assume that observers extract I units of information per move shown in a video, so that after seeing n moves, the classifier

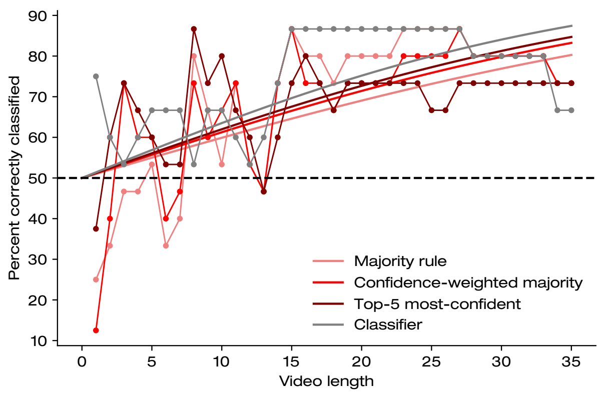


Figure 2.24 Accuracy of classification judgments as a function of video length for majority-rule voting among observers (pink), confidence-weighted majority voting (red), majority among 5 most confident observers (brown) or the artificial classifier (gray). The points lines depict the data (smoothed with a windowed average of size 5), the solid lines fits with a logistic function. The black dashed line indicates chance.

then correctly reports the true video type with probability

$$\mathbb{P}(\text{correct}) = \frac{1}{1 + \exp(nI)}$$

We fit this curve to the length-accuracy curve of the three human voter models (majority: $I = 0.0401$, confidence-weighted majority: $I = 0.0458$, most confident observers: $I = 0.0489$) and the artificial classifier ($I = 0.0554$, see figure 2.24). Therefore, we conclude that the model does indeed mismatch the data, but the size of the mismatch is relatively low. The extracted information I is a lower bound for the KL divergence between the model and the data-generating distribution, which is a metric of model mismatch. The result is a lower bound rather than an estimate since we cannot know has close our observers are to extracting all available information that distinguishes humans from computers.

2.18 Discussion

In this chapter, we introduced 4-in-a-row as a rich experimental task to study sequential decision-making or planning at decision time. Together with the task, we introduced a behavioral model and extensively tested its ability to fit human game-playing data. The model is based on the heuristic search framework and combines a feature-based value function, tree search, pruning and multiple sources of variability or noise. By comparing the model's log-likelihood to that of alternative models, we found that each component of the model is necessary but the specific implementation is not fully constrained by the data.

Although we have validated the model only for our game, it is constructed generically and can be applied to any deterministic two-player game with hidden information. One only has to specify a set of suitable features for value function approximation in that game. Additionally, both our log-likelihood estimation method (IBS) and optimization (MCS) are black-box and don't require any analytical calculations. Therefore, replicating our study using a different game is conceptually straightforward. However, modeling a more complex game such as chess will require more computational time and may be practically unfeasible.

In this article, we introduce a new sequential decision-making task with a state space complexity orders of magnitude higher than previously studied cognitive psychology tasks, but in which human behavior is still tractable to model. The combination of this task and our model opens the door for researchers to study human sequential decision-making in all its complexity. We have focused on the nature of expertise and talent, but our task could be used to investigate how people adapt their strategic decisions to specific opponents, communication between team players (as in Bahrami,2010 (Bahrami et al., 2010)) or problem solving (using puzzles).

Appendix A: Turing questionnaire responses

Responses to questionnaire after Turing test: *How did you distinguish between humans and computers?*

- I tried to find strategies that the humans would use that maybe computers wouldn't. I also felt like computers tended to place squares right next to ones they'd already made rather than seemingly at random on the board.
- i picked computer when both made very silly mistakes or not taking an opportunity to win the game. I tend to pick human when a player made an outplay or think ahead
- Humans were more defensive in choosing their next move, as they work to stop their opponent from gaining 4 in a row. Computers were more random and didn't catch on too quickly on places where their opponent can get 4 in a row.
- Generally if there was a obvious mistake in play, I would assume that it was the computer, as a human would likely not make those errors. Also, if the moves made by the players were similar to what I would do, then I would assume human.
- The human players tended to be more "defensive" with their moves & block their opponent's strategies, and the computers, while occasionally blocking opponent's moves, tended overall to be more "offensive" and lay out strategies for themselves outside of their opponent's moves.
- patterns of randomness/non randomness
- I tried to distinguish between humans and computers through logical thinking, though I wasn't correct all the time.
- I thought about which moves I made during the game, as well as those that the computer made and linked them to the current study to help me distinguish between the two.
- I thought about what moves I would make/what moves were logical to make and at-

tributed to those to "human" moves, and ones that deviated, as "computer" moves! Depending upon the patterns & moves

- I noticed that humans followed their opponents and mirrored their moves while computers just moved wherever they wanted to go.
- If one player won too easily or if a player missed out on an obvious chance to win, I felt it was a computer. I was also basing my decisions on what moves I would make myself.
- Humans would play more on the defensive, while computers would sometimes be unaware of how close the opponent is to winning. based on the sequences
- I answered human players for games where moves mostly immediate blocks or obvious wins were overlooked, and computers for games that ended up with multiple possible wins or moves that were planned several moves in advance. This distinction didn't seem to be particularly accurate, though
- Humans block their opponent. Computers play randomized moves.
- The humans seemed to be more strategic. One giveaway for the computers was that sometimes they seemed to pick very random spots on the board not close to each other, or not block a very obvious 4-in-1 row.
- I looked at how well each player blocked the other player's attempts of getting four in a row and I judged them based on what actions I would take. Other times, I selected based on intuition or guessing.
- Mostly through the defensive tactics. The computer seemed to be very focused on the attack and would not see obvious tactics trying to win, while the human defensive tactics were almost more cunning, as if they were on the defensive as well as the offensive at the same time. following the pattern
- I assumed that all of the games where the spots were clustered closer together were probably done by computers.
- I could not tell the difference
- What I found was that the humans tended to play more carefully and defensively, maybe

even anxiously, while the computers were less afraid to take risks. Depending on the length of the video and where it started in the match, it was easier sometimes than other times to apply this "theory" of mine.

- i looked for certain moves I remember the computer making yesterday
- Judging by patterns of three that humans would use, as well as larger clumps of pieces, as opposed to the more scattered approaches of the computers which would at times try to set up the ends of the win first.
- it was rather difficult to distinguish between the two. the computers seemed to try to make a horizontal row of three so that it could have two options to win. i tried to look for that pattern to determine if it was a computer. it often seemed like the humans placed their pieces closer to the opponent, while the computer seemed to spread its pieces out more. overall, i don't think i could determine a definite way to distinguish the two.
- by the amount of obvious mistakes made
- There were a few ways I remember the computer winning yesterday: they would set up two ways to win was the most successful, so I assumed this was the computers' game when this occurred. Also, human games seemed to be more "random" when placing their tiles.
- Defensive play was better from humans. In regards to offense, the computer would make moves that had little purpose at times.

Chapter 3

Generalization, time pressure and expertise in 4-a-in-row

3.1 Introduction

Having validated the behavioral model for our 4-in-a-row game extensively, we now demonstrate that it can be used to test scientific hypotheses and obtain generalizable insights. First, we demonstrate that our model not only predicts people's choices, but that its algorithm mimics the cognitive process by which people reach their decision. To do so, we show that it can generalize between subtasks within the 4-in-a-row domain, and that it can predict participants' response times and eye movements.

We then apply our task and model to investigate the nature of talent and expertise. Based on studies of chess experts (de Groot, 1946b), it is commonly believed that sequential decision-making and planning relies primarily on domain-specific knowledge and not on innate differences in computational abilities such as tree search. The idea of a limited role of search in expertise or talent has been generalized beyond the domain of chess and made its way into the textbooks and popular science discourse. However, the chess expertise literature is much less conclusive than needed to warrant such universal acceptance.

First of all, the chess expertise literature has often been misunderstood and evidence incorrectly interpreted. The presence of a role of pattern recognition in chess expertise has been demonstrated convincingly by experiments in which grandmasters recall chess positions taken from real games (but not scrambled ones) more accurately than their weaker counterparts. However, that in itself does not prove the absence of a role for search in expertise. Studies that focus specifically on search are mixed and actually most recent work supports a role of tree search in expertise (Campitelli and Gobet, 2004; Van Harreveld et al., 2007).

Additionally, almost all research on chess expertise relies on verbal reports, which raises important question on the validity of such data and specifically to what extent people's introspective beliefs about their decision-making strategies align with reality. Concretely, it may be that grandmasters search large decision trees but they underestimate how much they search, for example because it has become effortless or automatic. Our game and model are perfect to address these concerns and investigate the nature of expertise and talent in sequential-decision making with precise behavior modeling.

3.2 Generalization

To demonstrate that our model can generalize across tasks, we conducted a second experiment in which participants perform three tasks: playing against computer opponents, a 2AFC choice between moves in a given position and a board evaluation task. Although we developed our behavioral model to predict participants' moves during games, we can adapt it to make predictions for the other tasks.

To extend the model to choose an action in a 2AFC trial, in which two pre-determined candidate moves are presented in a given board position, we make two adjustments to the `Makemove` routine (algorithm 1). First, we adjust the `Lapse` procedure to randomly pick between the two candidate options, with probability 0.5 each. Next, after creating the root node (line 5) containing the given state, we attach two child nodes containing the states resulting after either candidate move. We then run the search process as usual, and return the highest-value child of the root node. Note that this modified algorithm will always return either of the two candidate moves as its choice. To extend the model for evaluation trials, we run the `Makemove` algorithm to choose a move, but instead of returning $\text{argmax}_{c \in \text{children(root)}} c.val$, we return $\max_{c \in \text{children(root)}} c.val$. This yields a value $v \in \mathbb{R}$, which we convert to a report on the 7-point scale by $v \to 4 + 3 \tanh(v/20)$ and rounding to the nearest integer. We chose the constants in this equation to ensure that the model report is always and integer between 1 and 7 and that a value $v = 0$ gets mapped to report 4 (label "equal" in the experimental interface). The constant 20 inside the hyperbolic tangent is mostly arbitrary, and we could probably achieve better predictions by fitting this parameter for individual participants.

3.2.1 Predicting single-trial 2AFC and evaluation responses

We estimate model parameters for each participant by maximizing the log-likelihood of their moves during games and predict their 2AFC choices and board evaluation. We measure the model's goodness-of-fit on a participant's 2AFC data by the percentage of correctly predicted choices, and on evaluation data by the Pearson correlation between predicted and observed evaluation scores. Participants' choices are predicted better than chance by the model (2AFC: $p_{\text{correct}} = 55.3 \pm 1.0\%, p < 0.001$ (mean and standard error across participants, one-sample T-test); evaluation: $\rho = 0.297 \pm 0.034, p < 0.001$).

3.2.2 Comparison with oracle and mean-across-participants

Next, we compare the performance of our main model on the 2AFC and evaluation trials against two baselines. First, we compare against an oracle model, which has access to the game-theoretic value of each move. On 2AFC trials where one move is objectively better than the other, the oracle model picks that move. On trials where both moves are equally strong, the model picks randomly. On evaluation trials, the oracle model responds 1 when the position is lost, 4 when it's a draw and 7 for winning positions. Second, we compare against a model that has access to other participants' responses. Its prediction for a participant's choice on a 2AFC trial is the move chosen by the majority of all other participants. On an evaluation trial, it responds with the average evaluation across all other participants rounded to the nearest integer. Note that both these models are unrealistic as models for a people's decision-making process, since participants do not have access to an oracle that provides the objectively correct moves, or to the choices made by other participants.

Both the oracle (2AFC: $p_{\text{correct}} = 55.3 \pm 1.0\%, p < 0.001$; evaluation: $\rho = 0.297 \pm 0.034, p < 0.001$) and the mean-across-participants baseline (2AFC: $p_{\text{correct}} = 60.5 \pm 1.4\%, p < 0.001$, evaluation: $\rho = 0.39 \pm 0.04, p < 0.001$, see figure 3.1) predict participants' choice above chance. However, the model explains variance in the data beyond the two baselines. To demonstrate this, we ran a multiple linear regression to correlate participants' evaluations with a linear combination of the predictions of the main model, the oracle and the mean-across-participants, and note that the regression coefficient for the main model is on av-

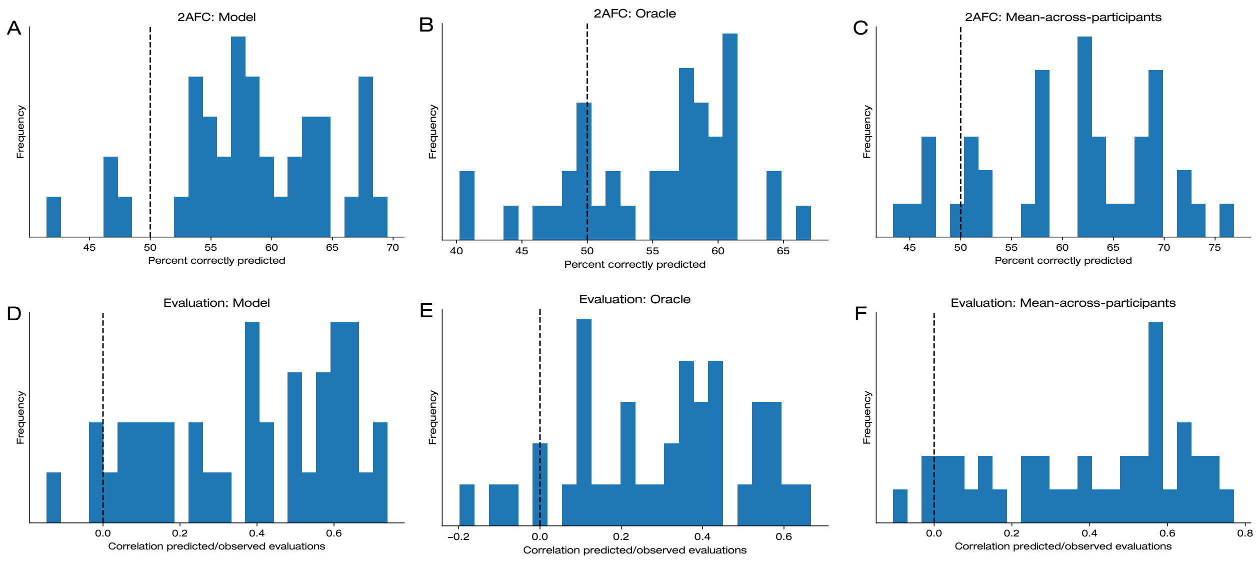


Figure 3.1 **A.** Histogram of percent correctly predicted 2AFC choices by the behavioral model across participants in the generalization experiment. We fit the model on the same participants' moves in games against computer opponents. **B,C.** Histogram of percent correctly predicted 2AFC choices by the oracle model (**C**), and the mean-across-participants model (**C**). **D-F.** Same as **A-C**, for the correlation between predicted and observed responses on the board evaluation task.

erage positive ($\beta = 0.35 \pm 0.07, p < 0.001$). Therefore, our model predicts participants' subjective preferences beyond the group level.

Next, we show that the goodness-of-fit of the main model is correlated across participants with these baselines on 2AFC (oracle: Spearman correlation $\rho = 0.62, p < 0.001$, mean-across-participants: $\rho = 0.56, p < 0.001$) and even more so one evaluation data (oracle: $\rho = 0.96, p < 0.001$, mean-across-participants: $\rho = 0.95, p < 0.001$, see figure 3.2). For all three tasks, the goodness-of-fit for each participant is bounded from above by the noise in that participant's decision-making process, as measured by variance or entropy of their choices. We cannot estimate the noise, since we never present the same position to the same participant twice. However, the large correlation between goodness-of-fit across these predictions suggests that the noise level varies across participants, and that this causes the large variance in goodness-of-fit across participants. In other words, participants whose

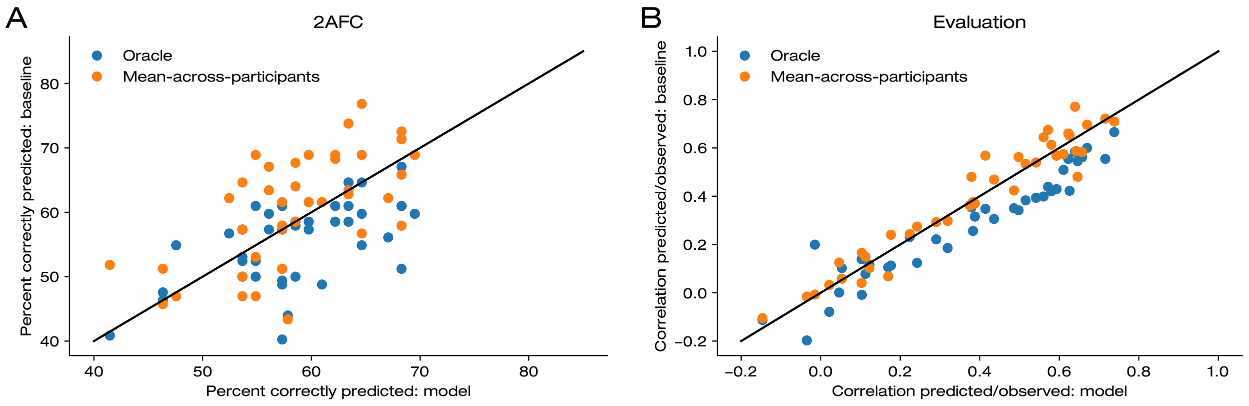


Figure 3.2 A. Scatterplot showing correlation across participants between percent correctly predicted by the behavioral model and the oracle model (blue dots), and between the behavioral model and the mean-across-participants model (orange). The black dashed line indicates equality. **B.** Same for the board evaluation task.

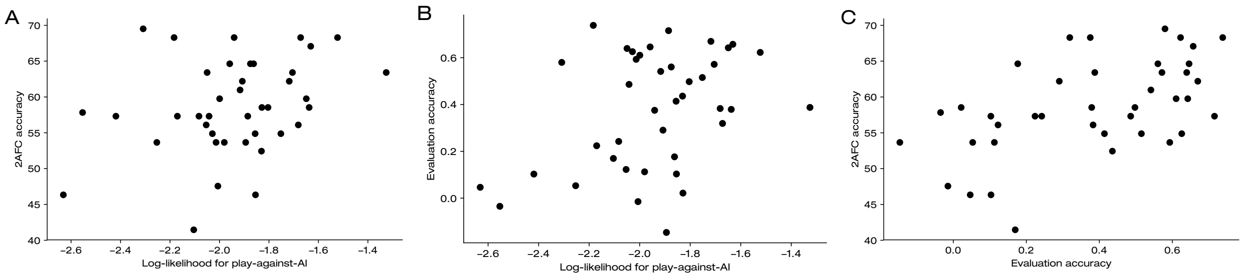


Figure 3.3 Scatterplots showing correlation across participants between **A.** Log-likelihood per move of the main model on games against computer opponents and percent of 2AFC trials correctly predicted by the model **B.** Log-likelihood per move and correlation predicted/observed board evaluations **C.** percent correctly predicted 2AFC trials and correlation predicted/observed board evaluations.

choices are poorly predicted by the model may simply be less predictable. To support this hypothesis, we observe that goodness-of-fit is also correlated across participants between the 2AFC task, evaluations and games against the computer (2AFC-games: $\rho = 0.29, p = 0.07$; evaluation-games: $\rho = 0.30, p = 0.06$; 2AFC-evaluation: $\rho = 0.51, p < 0.001$; see figure 3.3).

3.3 Response times

To test our model's ability to predict process data, we start by predicting response times. Figure 3.4A shows a histogram of response times across all participants in human-vs-human games. On average, participants play quickly in the opening and endgame and longer in the middle game (figure 3.4B). We obtain a response time prediction using the Hick-Hyman law (Hick, 1952; Hyman, 1953), which states that a participant's response time on a single trial is proportional to the entropy of their choice distribution (more predictable moves are made faster). Unfortunately, we cannot measure this distribution, as the participant's chosen move reflects only one sample. However, if our model is correct, we can approximate a participant's single-trial move distribution with that of the model with inferred parameters, from which we can draw arbitrarily many samples. We estimate the entropy of the model's move distribution with inverse sampling, and fit a proportionality constant for each individual participant.

Specifically, to estimate the entropy of the distribution of moves made by the behavioral model with parameters inferred for a particular participant in a given position. To do so, for each position that occurred in human-vs-human games, we first sample 100 moves from the behavioral model, then for each sampled move we estimate $\log \mathbb{P}\left(\text{move}|\text{board}, \theta\right)$ using inverse binomial sampling (see chapter 4). We run this estimator once for each sampled move in each position, and estimate the entropy by averaging across all samples and positions.

In figure 3.4A-B, we compare the model prediction obtained through the Hick-Hyman law with the data. The model successfully predicts the response time histogram and the response time decrease in the endgame, but it fails dramatically in the opening. This, to-

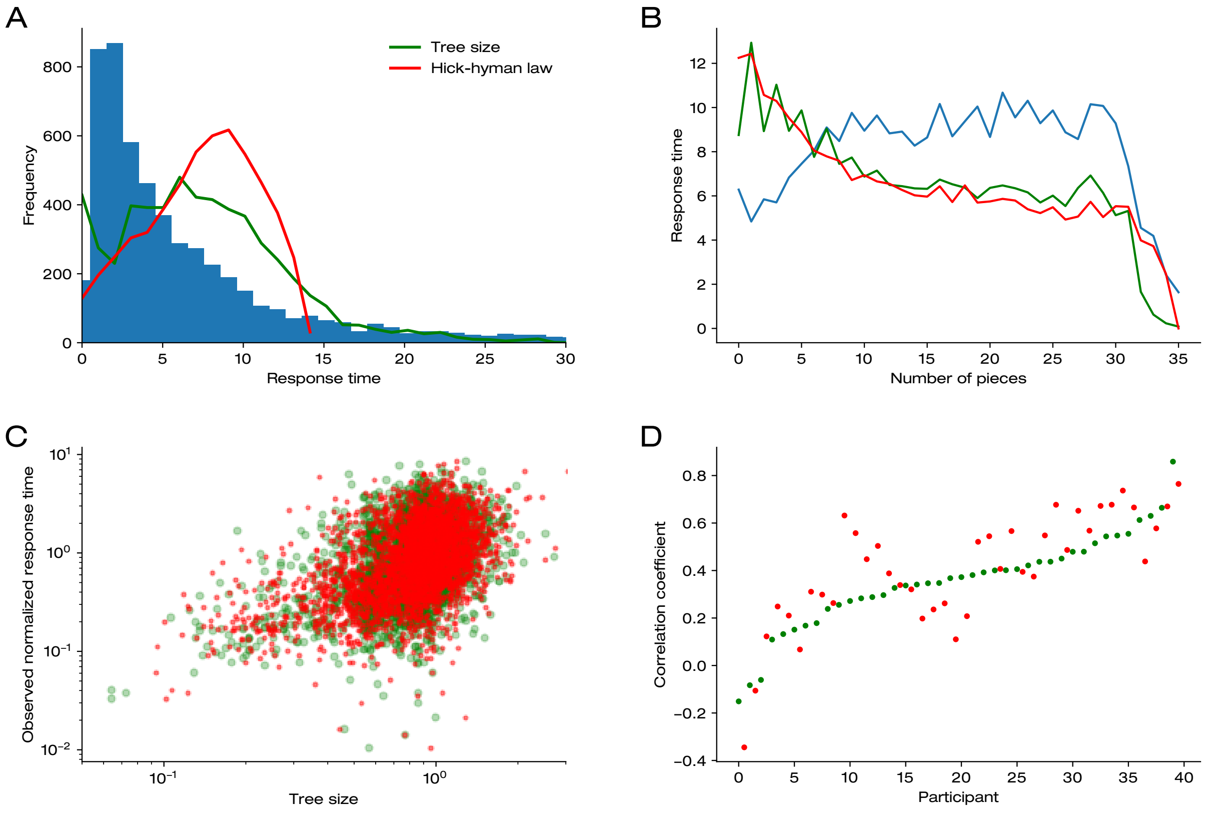


Figure 3.4 The behavioral model predicts response times on single trials. **A.** Histogram of people's response times, aggregated across all participants in the human-vs-human experiment (black bars), together with predicted distributions using the Hick-Hyman law (red) or tree size (green). **B.** Average response time as a function of number of moves played in each position, with the same color scheme as in **A**. **C.** Scatterplot in which each point represents one trial, the x-coordinate the normalized response time as predicted by the Hick-Hyman law (red) or using tree size (green), and the y-coordinate the normalized response time as observed in human data. **D.** Correlation between predicted and observed log response times for each individual participant, ranked from lowest to highest correlation averaged between the two predictions.

gether with our earlier observations that the model predicts opening moves not much better than chance and fails to match several summary statistics (figure 2.15), suggests that people use a different decision process than the goal-directed planning described by our model. For example, people may choose their starting moves through habitual formation or perseveration. To support this hypothesis, we demonstrate below that participants' starting moves are correlated with the starting move they chose on their previous game, in particular if they won that game. A detailed description of the cognitive process underlying people's opening choices is beyond the scope of this article. Instead, we will simply ignore the first 6 moves of each game when evaluating response time predictions.

3.3.1 Serial dependencies in opening moves

First, we show that participant's starting moves in consecutive games are correlated. On average, participants chose the same starting move in consecutive games $32.6 \pm 1.4\%$ of the time (mean and standard error across participants in all experiments). To compare this frequency to a baseline, we compute for each participant the coincidence probability based on the histogram of their starting moves. We denote by n_k the number of games on which a given participant opens on square k, N their total number of games played, and define the coincidence probability as

$$P_{\text{coin}} = \frac{1}{N(N-1)} \sum_{k=1}^{36} n_k(n_k - 1) \tag{3.1}$$

The coincidence probability is equal to the average frequency of repeated starting moves in consecutive games after shuffling the order of that participant's games. If participant's starting moves on each of their games were an independent and identically distributed sequence, the coincidence probability should on average equal the frequency of repeats. However, it is considerably lower: $P_{\text{coin}} = 28.9 \pm 1.2\%$, 2-sample T-test across participants, $p < 0.001$ (see figure 3.5A).

Next, investigate whether participants are more likely to repeat their chosen starting move after winning a game than after losing. For this analysis, we use only the generalization, eye tracking and learning data. The frequency of repetition after wins is $38.1 \pm 2.1\%$ and

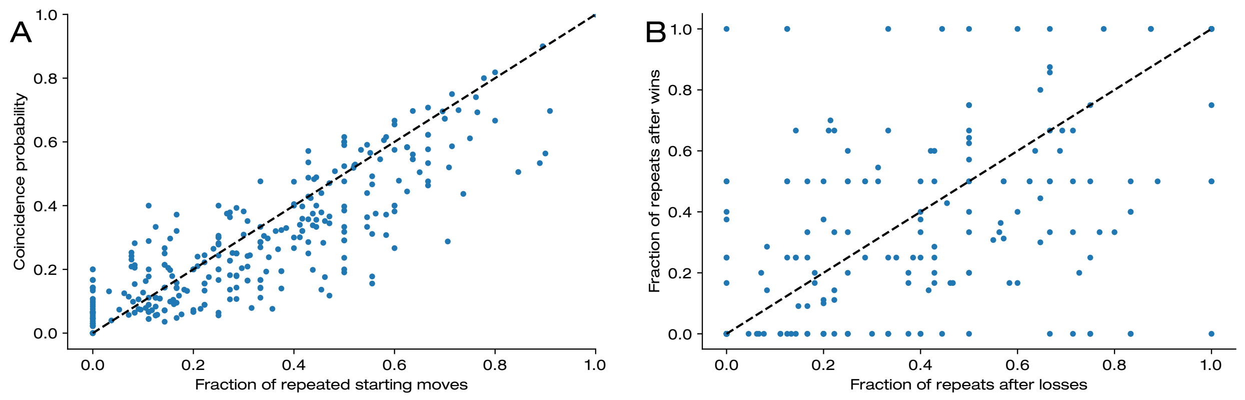


Figure 3.5 A. Scatterplot in which each point represents one participant in any of our 6 experiments, the x-coordinate the coincidence probability (see equation 3.1) and the y-coordinate the fraction of repeated moves on consecutive games. The points lie below the equality line (dashed black line), indicating that participants repeat their chosen opening move more often than expected by chance. **B.** Scatterplot of fraction of repeated moves in consecutive games after participant win a game versus that after losses.

it is $33.9 \pm 2.2\%$ after losses (2-sample T-test, $p < 0.05$, see figure 3.5B). However, since each participant plays only 15 games as black on average, these frequencies are difficult to estimate reliably and figure 3.5B looks noisy. Additionally, the repetition frequency is even higher after draws ($40.7 \pm 3.5\%$) than wins, which would contradict our hypothesis but this difference is not significant (2-sample T-test, $p = 0.70$). In summary, we conclude that, even though we find some evidence that participant's choice of opening move is influenced by their result on the previous game, we cannot draw strong conclusions.

3.3.2 Single-trial response time predictions

To test single-trial model predictions, we first divide each participant's response time by the mean, then correlate the logarithm of predicted and observed normalized response times. We use the logarithm rather than raw response time values because the response time distribution (figure 3.4A) resembles a log-normal, and because human time perception and production approximately obey Weber law (Bizo et al., 2006; Getty, 1975; Grondin, 1992). The average correlation across participants is $\rho = 0.405 \pm 0.037$ (Figure 3.4C). In figure 3.4D, we plot the predicted and observed response time for all trials across all participants.

Although the Hick-Hyman law extracts a response time prediction from our model, it only depends on the model's choice distribution, not the algorithm by which it chooses a move. To link the algorithm to people's cognitive process, we first modify the model by adding a stopping rule: the tree search algorithm terminates when the model's preferred move in the root node remains unchanged for 50 consecutive iterations. To obtain a response time prediction from the model, we measure the size of its decision tree on each trial, which is approximately proportional to the computational time the algorithm uses to make a move (see section 2.9). We sample 100 moves from the model in each human-vs-human position, measure the decision tree size for each sample and average across samples. In figure 3.4, we show that the model's tree size correlates with participant's response times on individual trials. This suggests that model reflects not just people's moves, but also their decision-making process.

3.3.3 Robustness of response time results

To examine the robustness of our response time prediction results, we first repeat the analysis with different choices for the cutoff between opening, middle and endgame moves (see figure 3.6). In these figures, we report both Spearman correlations and Pearson correlation between the logarithm of predicted and observed response times. Additionally, we report either the correlation across trials on aggregate data (in figure 3.6A,B) or the average correlation coefficient across participants (in figure 3.6C,D). With all these metrics, we find a significant correlation between predicted and observed response times given an opening cutoff at 5 moves, and an endgame cutoff at 30 moves. Moreover, in all cases we find that varying the endgame cutoff does not change this correlation, but that discarding more opening increases the quality of the response time prediction. Therefore, our results are robust to arbitrary choices in our analysis except for the opening cutoff, which we chose conservatively.

To predict response times, we modified the model to terminate its search algorithm after 50 consecutive iterations without changes in its preferred move. The choice to use 50 iterations as a stopping threshold is arbitrary and may affect the quality of our prediction. We therefore analyze models with stopping thresholds varying from 2 to 200 iterations and plot the mean tree size (figure 3.7A), the log-likelihood on choice data (figure 3.7B), and the trial-to-trial correlation between tree size and repsonse time (figure 3.7C,D). With lower stopping thresholds, the algorithm more often stops prematurely, and therefore builds smaller decision trees. Moreover, since lower thresholds are exceeded more often, lower thresholds cause the model prediction to deviate more from the unmodified model and therefore decrease the log-likelihood. However, the quality of the response time prediction (which requires a stopping threshold) peaks for stopping thresholds around 30 and varies by only 0.02 for thresholds between 10 and 100. Therefore, the exact choice of stopping threshold does not change the qualitative result: tree size correlates with response time on individual trials. The "optimal" choice of stopping threshold depends on a weighting between goodness-of-fit on choice versus response time data and one's prior belief of how large decision trees humans can plausibly build.

In this section, we have focused on using the behavioral model with parameters inferred from choice data to predict response times. We did this only for the main model, but it is

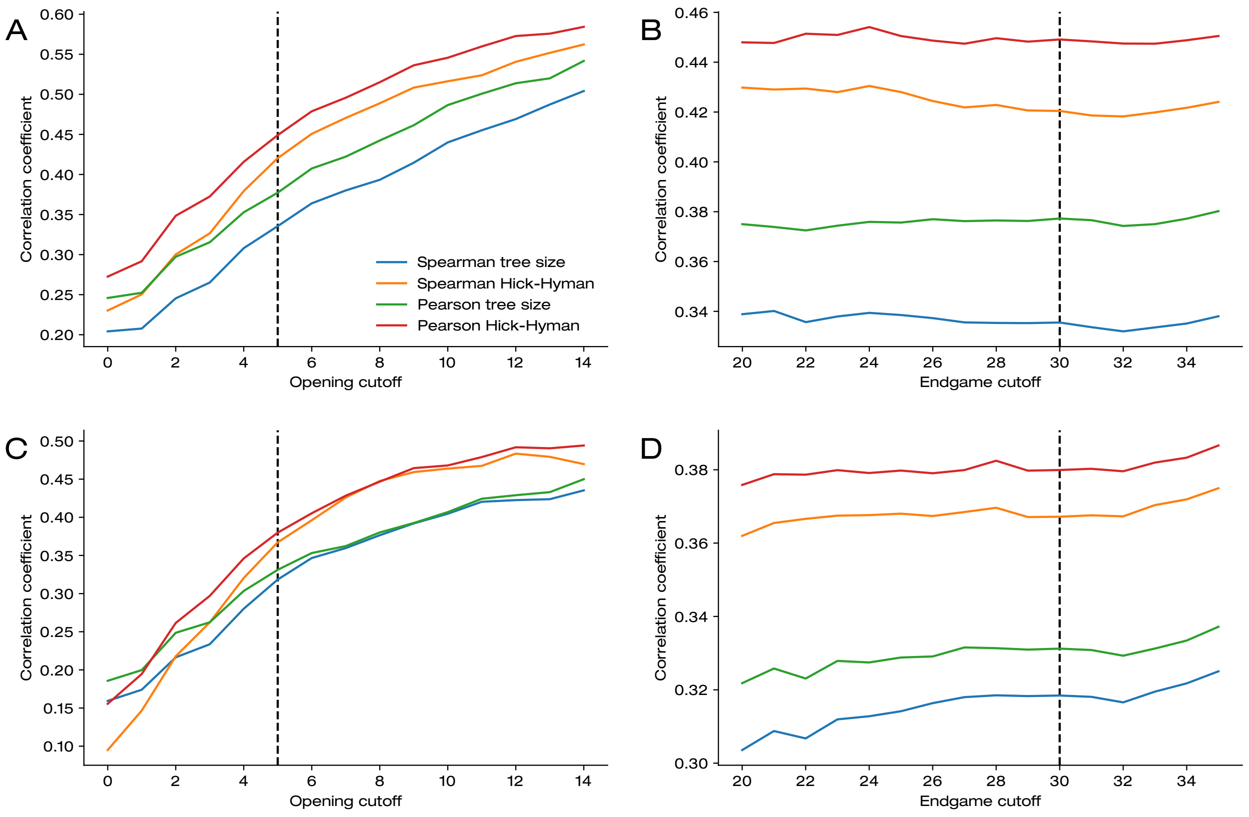


Figure 3.6 Response time results are robust to the choice of cutoff for opening and endgame moves, and metrics used to compute the correlation. **A.** Correlation coefficient between predicted and observed response times as a function of opening cutoff. In this panel, we aggregate response time across all participants before measuring the correlation coefficient. We compute either the Spearman correlation between predicted and observed response times (blue and orange curves) or the Pearson correlation between the logarithms of predicted and observed response times (green and red). For prediction, we either use the Hick-Hyman rule (orange and red curves), or tree size (blue and green). The dashed black line indicates the opening cutoff (5) that we choose to use for all other analyses. **B.** Same as **A**, expect we plot correlation coefficients against the endgame cutoff. The dashed black line indicates the default endgame cutoff (30). **C,D.** Same as **A,B**, except we now compute the correlation coefficients for each individual participant and average across participants.

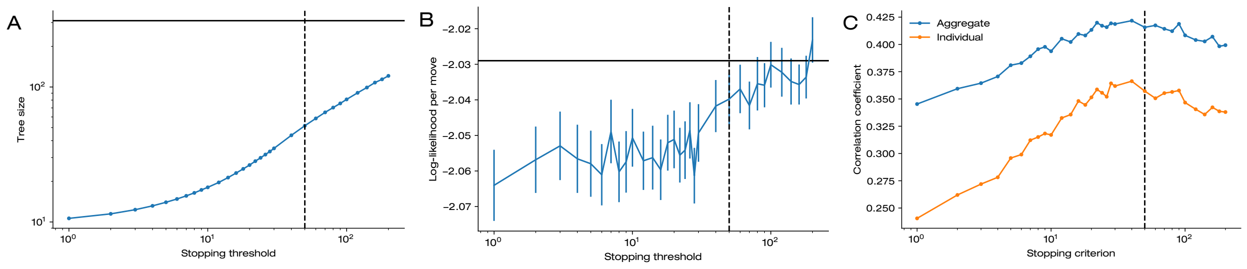


Figure 3.7 A. Average size of the decision tree that the main model with a stopping rule builds, as a function of the stopping threshold. The vertical black line indicates the average tree size of the main model without any stopping rule. **B.** Log-likelihood per trial of the main model with a stopping rule. Errorbars indicate standard error across participants of the difference between the log-likelihood per trial of the model with and without stopping rule. The vertical line indicates the log-likelihood per trial of the main model without a stopping rule. **C.** Correlation coefficient between predicted and observed response times (using tree size as predictor) for the main model with a stopping rule. We compute the correlation either across trials aggregated for all participants (blue curve) or for individual participants and report the average (orange).

likely that alternative models, at least those modifications or extensions that fit the choice data equally well, could also predict response times. However, to perform a model comparison analysis on response time data, we would need to examine in more detail the assumptions required to extract a response time prediction from the model, and whether the same prediction based on tree size is correct for all models. For control models, or the **No tree** lesion model, we would need to determine another response time predictor. Therefore, the results of a model comparison study on response time data would be difficult to interpret as it combines a measurement of the goodness-of-fit of each model as well as how appropriate the response time read-out is. In other words, the results in this section do not demonstrate evidence for our main model over alternatives, but they support an interpretation of the main model - being selected based on choice data - as reflecting not just people's decisions, but reflecting the process by which they arrive at those decisions.

3.4 Eye movements

Aside from response times, our model can also predict people's eye movements. To demonstrate this, we conducted another experiment in which participants played against computer opponents while we tracked their eye movements with a infra-red video-based eye tracker. This data requires some pre-processing before it is usable for testing model predictions.

3.4.1 Eye tracking pre-processing and calibration

To pre-process eye tracking data, we first convert the output files from `edf` to `ASCII` format using the `edf2asc.exe` program provided by SR Research. This results in a time series of fixations and saccades for each participant, with time measured relative to the moment we turned on the eye tracking software, and position in a coordinate system defined by the eye tracker calibration. We start by re-calibrating both time and space, using data from the calibration that participant complete before and after playing against computer opponents. In this calibration, they fixate on all 36 squares consecutively, pressing the space bar to provide a time stamp for each fixation. We ensured that both the eye tracking and behavioral recording computers were synchronized to the same time server (`tock.nyu.edu`). However, to account for small deviations in time stamps due to clock drift, we found for each participant the time offset that maximized the quality of the calibration. Specifically, we varied a temporal offset between the behavioral and eye tracking time stamps and maximized the Pearson correlation between the eye coordinate at the (corrected) times of the space bar press and the corresponding tile coordinate, summed across horizontal and vertical dimensions. This resulted in offsets between -3.28 and 0.23

seconds.

While re-calibrating space, we avoid the assumption that the transformation from eye tracker to board coordinates is globally linear. Instead, we divide the eye coordinate space into 36 tiles using a Voronoi tesselation with as centroids the eye tracker coordinates at the time of space bar presses, averaged across the pre-task and post-task calibration. In figure 3.8A-D, we show the eye tracker coordinates for an example participant with and without temporal re-calibration. In figure 3.8A, horizontal and vertical eye position (as returned by the eye tracker) are plotted in orange and green, respectively, and time stamps of space bar presses by the participants are indicated by vertical gray lines. In this figure, we depict both the pre-game and post-game calibration data, and excise all game-playing data. Note the typical staircase-like pattern that arises when a participant scans each row of the board, starting at the bottom, from left to right. In figure 3.8B, note that the board is distorted in the top left corner. Additionally, the tile numbers are mismatched with the board location (for example, the tile at the bottom right corner should be number 9, but the fixation coincides with the 10th space bar press). This suggests at temporal offset between the eye tracking and behavioral data. In figure 3.8B, note that the Voronoi tesselation now closely resembles a square grid, although the eye tracker still introduces some distortions.

This allows us to associate a square for eye fixation in the time series, but not to determine where on that square the participant is looking. For that purpose, we compute, for each fixation in the time series, the distance from that fixation's eye tracker coordinates to the centroid of its corresponding square as well as the 4 squares neighboring it (see fig-

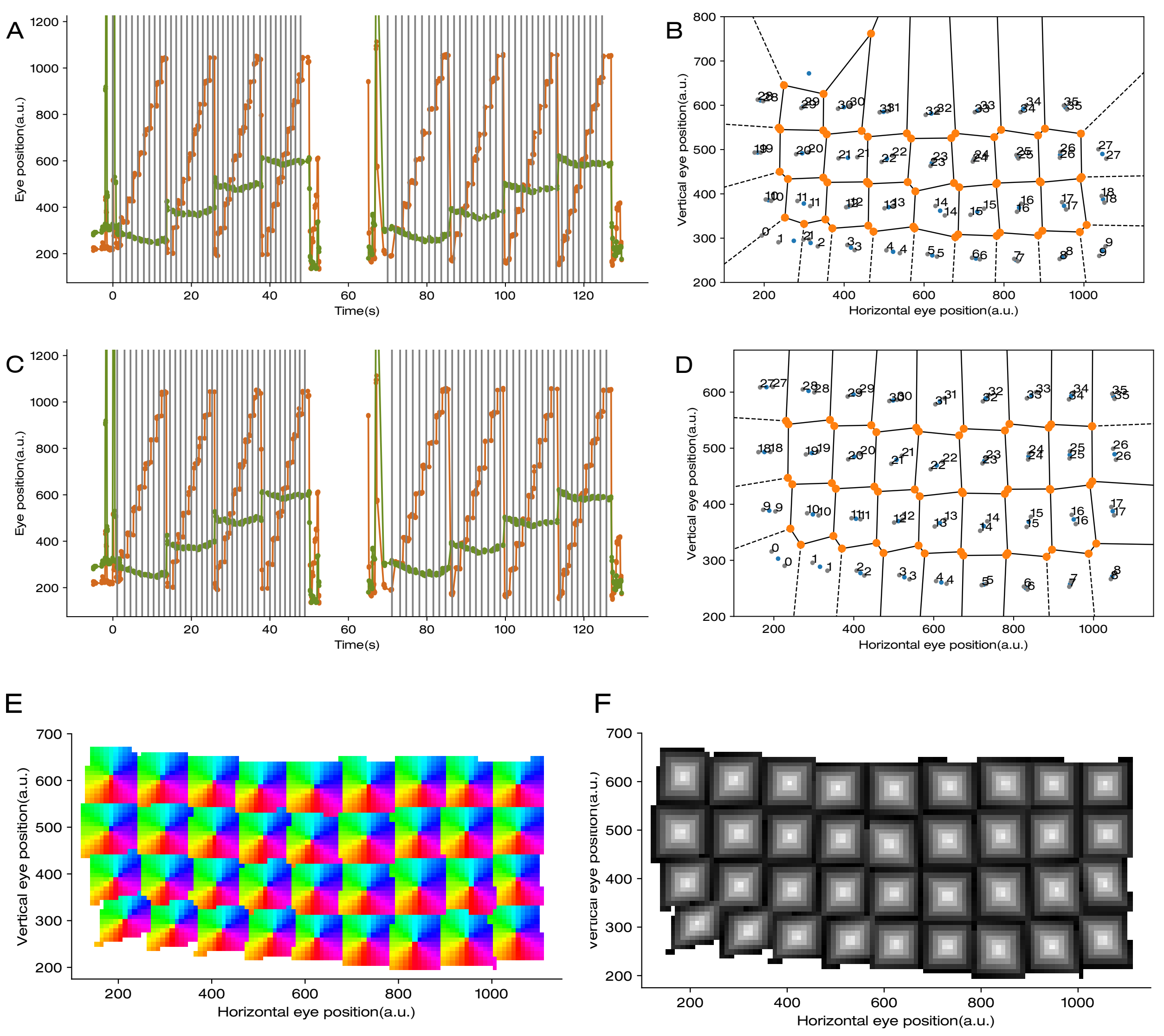


Figure 3.8 Eye calibration procedure. **A.** Behavioral and eye tracking data during the calibration stage. **B.** Eye coordinates at the time of each space bar press in both calibration stages (gray dots with text labels), and a Voronoi tesselation (black solid & dashed lines, orange dots) with centroids (blue dots) at the average of the two calibration coordinates for each tile. **C.** Same as **A**, after correcting for the temporal offset. Note that the gray lines are slightly displaced relative to **A**. D. Same as **B** after correcting for the offset. **E,F.** For each point in eye coordinate space, we compute fractional board coordinates using equation 3.2 and display the angle formed by x and y (**E**, in hue) as well as the distance from the square center (**F**, using $d = \max(|x|, |y|)$, in grayscale). The regions in eye tracker space that lie outside the board edges is colored white. Note the correspondence between the Voronoi tesselation centroids and edges with the pinwheels and discontinuities in these maps.

ure 3.9). For each neighbor, we then compute

$$f_{\text{neighbor}} = \frac{1}{2} + \frac{1}{2}\frac{\|\vec{e} - \vec{c}_{\text{tile}}\|^2}{\|\vec{c}_{\text{tile}} - \vec{c}_{\text{neighbor}}\|^2} - \frac{1}{2}\frac{\|\vec{e} - \vec{c}_{\text{neighbor}}\|^2}{\|\vec{c}_{\text{tile}} - \vec{c}_{\text{neighbor}}\|^2} \tag{3.2}$$

where $\vec{e}$ denotes the coordinates of the fixation, $\vec{c}_{\text{tile}}$ the coordinates of the corresponding tile centroid and $\vec{c}_{\text{neighbor}}$ the coordinates of the neighboring tile centroid, all measured in eye tracker space. This equation defines a linear function from the Voronoi cell to real numbers, which is zero at the cell centroid ($\vec{c}_{\text{tile}}$) and $\frac{1}{2}$ on the boundary between the Voronoi cells of the tile and its neighbor. We then define the fractional coordinates in board space as

$$x = \begin{cases} f_{\text{right}} & \text{if } |f_{\text{right}}| < |f_{\text{left}}| \\ -f_{\text{left}} & \text{otherwise} \end{cases} \qquad y = \begin{cases} f_{\text{up}} & \text{if } |f_{\text{up}}| < |f_{\text{down}}| \\ -f_{\text{down}} & \text{otherwise} \end{cases}$$

This results in a piecewise linear mapping between board space and eye tracker coordinates within each tile, but the slope varies between squares. For squares on the edge of the board, where f_{right}, f_{left}, f_{up} or f_{down} may be ill-defined, we use the case in the equation above that is well-defined. Finally, we disregard in the remaining analysis any fixations for which the absolute value of x or y exceeds $\frac{1}{2}$, since these lie either beyond the edges of the board, or in a region in eye tracker space where the assumptions of our calibration procedure fail (for example, if $\vec{c}_{\text{up}}$ actually lies beneath $\vec{c}_{\text{tile}}$). In figure 3.8F-G, we show the resulting mapping from eye tracker space to board coordinates for an example participant.

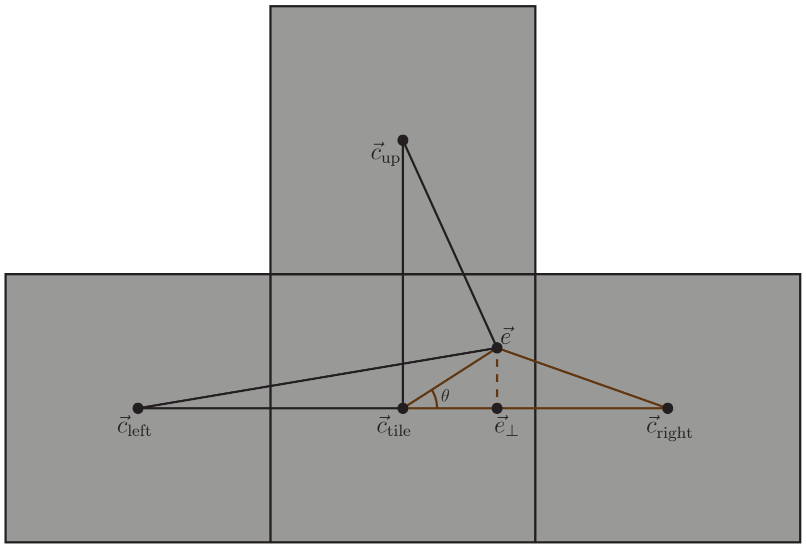


Figure 3.9 Graphical illustration of the geometry underlying equation 3.2. To derive this equation, we calculate the value of the angle θ twice by applying the cosine rule either to the triangle formed by $\vec{c}_{\text{tile}}$, $\vec{c}_{\text{right}}$ and $\vec{e}$ or the triangle formed by $\vec{c}_{\text{tile}}$, $\vec{e}$ and $\vec{e}_\perp$ and equate the results. The result then follows by straightforward algebra and the definition $f_{\text{right}} = \frac{\|\vec{e}_\perp - \vec{c}_{\text{tile}}\|}{\|\vec{c}_{\text{right}} - \vec{c}_{\text{tile}}\|}$.

3.4.2 Single-trial eye movement predictions

In figure 3.10A, we show one participant's fixations on an example trial. On each trial, we convolve this fixation trajectory with a Gaussian filter to estimate the participant's attention distribution (figure 3.10B, and rank all unoccupied squares by the amount of attention. In figure 3.10C, we show the distribution across trials of the attentional rank of the square on which the participant chooses to move. This distribution is concentrated on low ranks and specifically, the chosen square most often has attentional rank 1. In other words, predicting a participant's choice to be the square near which they fixate the most yields the correct result on 37.3 ± 2.7% of trials. Therefore, participants' eye movements contain information about their upcoming move.

We now show that our behavioral model can predict the estimated distribution of atten-

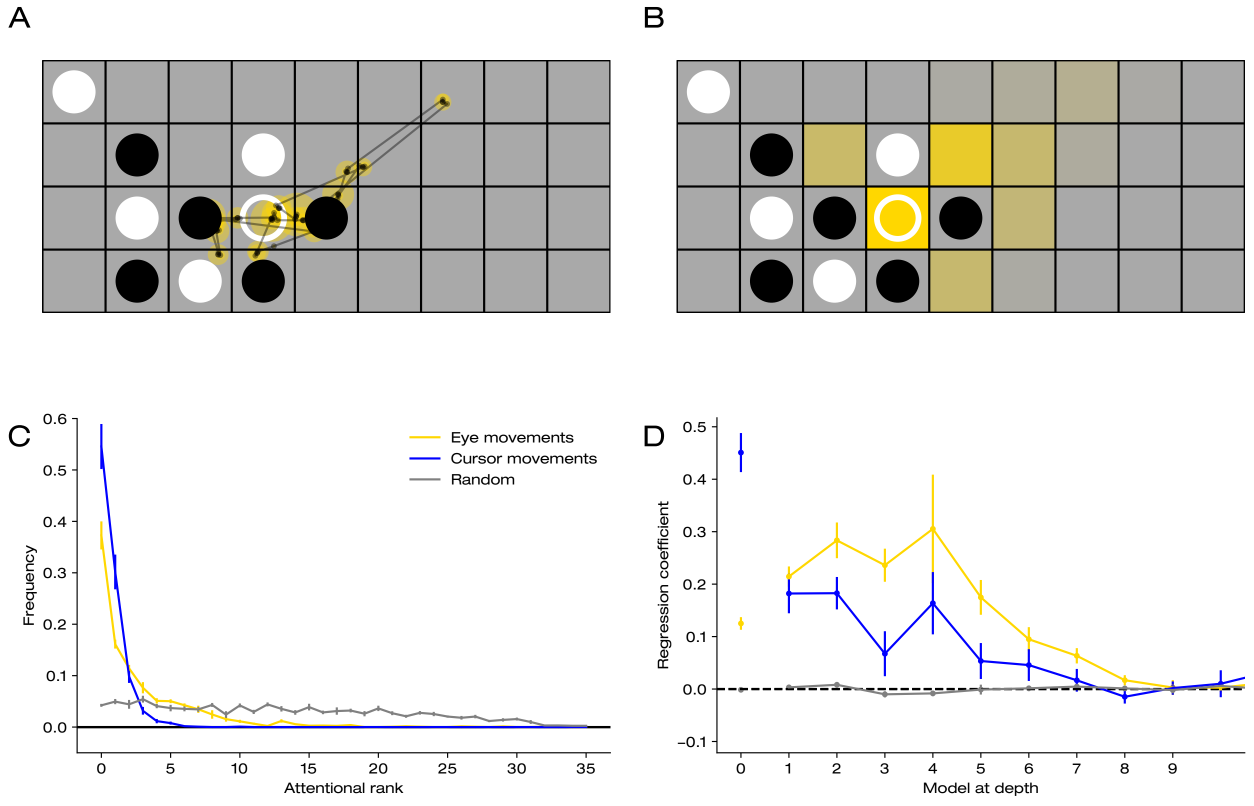


Figure 3.10 The behavioral model predicts eye movements. **A.** Trajectory of eye movements on one example trial. The black lines represent saccades, the golden circles fixations with duration proportional to the area of the circle. The white open circle indiciates the move that the participant ended up choosing in this position. Note that the participant played with the white pieces. **B.** The estimated distribution of the participant's attention across all unoccupied squares in the same position. We obtain this estimate by convolving the trajectory in **A** with a Gaussian filter (see text). **C.** Distribution of attentional rank of the people's chosen move across all positions encountered by all participants. We compute the attentional rank by sorting all unoccupied squares by their amount of participant's attention (as in **B**). We plot the distribution of ranks for attention estimated from eye movements (gold), cursor movements (blue) or a random prediction (gray) to provide a null baseline. **D.** Coefficients for a linear regression predicting participants' attentional distribution from the distribution of squares that the model includes in its principal variation at each depth, and the move participants' chosen move. As before, we plot the regression coefficients for either eye (gold) or cursor (blue) movements, or random regressors. Error bars denote mean and standard error across participants.

tion on individual trials. To do so, for position encountered by each participant we simulate moves from the behavioral model with parameters inferred for that participant, and count for each square and each depth how often the principal variation in the selection step of the search algorithm includes a move on that square at that depth. This results in a histogram over unoccupied squares for each position in our data set and each depth, which we normalize into probability distributions of the model's square visits. For each position, we also create a probability distribution which is 1 for the square on which the participant actually moved and zero otherwise. We then regress the model's square visit distributions and the actual move against participants' distribution of attention estimated from eye movements, which results in a Pearson correlation between predicted and observed attention of $\rho = 0.520 \pm 0.022$ (mean and s.e.m. across participants, one-sample T-test $p < 0.001$). Crucially, the regression coefficients are highest for the model at depth 1 to 4, and significantly larger than zero (one-sample T-test across participants) for depth up to 7 (see figure 3.10D). The regression coefficient for the actual move is also positive ($\beta = 0.125 \pm 0.012, p < 0.001$) but smaller than that for the regressors from the behavioral model (for example, $\beta_1 = 0.214 \pm 0.019, p < 0.001$).

3.4.3 Single-trial cursor movement predictions

In all our experiments, whenever a participant moves the mouse cursor to any unoccupied square, that square lights up faintly. Anectodally, have observed participants hovering the cursor above squares they consider clicking on, or to trace out possible move sequences on the board. This suggests that the distribution of time spent on each unoccupied square serves as another proxy for participants' distribution of attention. Therefore, we repeat the eye tracking analysis on the cursor data, with similar results. The chosen square on each trial has low attentional rank (figure 3.10C) and the behavioral model can predict the distribution of attention ($\rho = 0.633 \pm 0.040, p < 0.001$). However, the regression coefficient for the actual move is much higher ($\beta_{\text{actual}} = 0.451 \pm 0.037, p < 0.001$), and the coefficients for model regressors are significantly positive only for depth 1,2 and 4 (figure 3.10D).

3.5 The nature of expertise and talent

Having established a connection between our model and participants' decision-making process, we can now use it to answer our main question: how do expert players differ from novices? First, we show that participants' playing strength gradually increases during the 5 sessions of the learning experiment (figure 3.11A), whereas their response time decreases (figure 3.11B). The playing strength of the model with inferred parameters also increases, and this increase is correlated with all model parameters except for the pruning threshold and the weights of the linear features (data not shown). To directly test whether the superior performance of experts relies on more tree search, fewer lapses of attention or better feature weights, we convert the set of 10 parameters into 3 metrics: the average tree size, the feature drop rate, and the heuristic quality, which measures how closely the heuristic value function (Equation 2.1) approximates the game-theoretic value.

As noted before, we could try to use eye movement or cursor movement data to distinguish between models. For the same reason as mentioned before, doing so would require a more thorough investigation of exactly which aspects model's algorithm are predictive of eye or cursor movements than we have done in this thesis. Additionally, performing model comparison with eye or cursor data might be harder since we only have this data for 10 participants. However, given the rich nature of this data it is possible that examining it more closely can provide more detailed insight into the nuances of people's search process and their attentional mistakes. Most importantly, we could likely improve on the results shown here by considering eye movements as a time series rather than a fixed distribution.

3.5.1 Computing tree size, heuristic quality and feature drop rate

To investigate the nature of talent, expertise, we convert the set of 10 parameters inferred for each player in each session to 3 statistics: tree size, heuristic quality and feature drop rate. The feature drop rate is simply one of the parameters of the model (δ), but the tree size and value quality are more complicated function. To compute tree size, we simulate moves from the behavioral model with a given parameter vector in each position that occurred in human-vs-human games that have between 5 and 30 pieces. For each position, we sample 100 moves from the model, measure the size of the decision tree built by the model after each move, and average across all samples and boards. To compute the heuristic quality, we evaluate the feature-based heuristic value function $V(s)$ (see equation 2.1) without any feature dropping in each position in and compute the Pearson correlation between $\tanh\left(V(s)/20\right)$ and the game-theoretic value (1, 0 or -1 for objectively winning, drawn and losing positions, respectively). Note that by construction, the feature drop rate does not depend on any parameters except δ, the heuristic quality is independent of δ, γ or θ, but tree size depends on all model parameters (although in practice it is most affected by changes in γ or θ).

These procedures define how to compute feature drop rate, heuristic quality and tree size for the main model, and they extend naturally to alternative models with only a few exceptions: for the **Tile dropping** model, we consider the tile drop rate in place of the feature drop rate, for the **Orientation-dependent dropping** or **Type-dependent dropping** models, we define feature drop rate as the average drop rate across all orientations or types, and for **Monte Carlo Tree Search** we define tree size as the total number of rollouts performed by the algorithm.

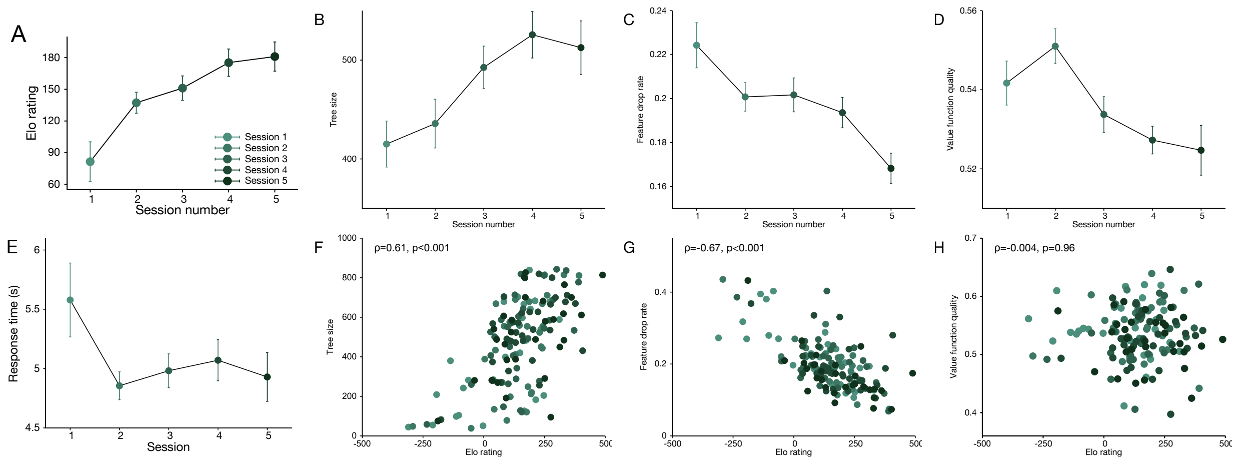


Figure 3.11 A. Average Elo rating of participants in the learning experiment, as a function of session number. In this panel and all others, error bars denote mean and standard error across participants. **B.** Average size of decision tree that participants build, as estimated by the behavioral model (see text). **C.** Same, for feature drop rate. **D.** Same, for heuristic value quality. **E.** Mean response time across all trial, as a function of session number. **F.** Scatterplot in which each point represents one participant in one session, the horizontal coordinate their Elo rating and the vertical coordinate the estimated tree size. **G.** Same as **F**, for feature drop rate. **H.** Same, for heuristic value quality.

Before investigating how these 3 statistics change across sessions, we ask how precisely they are constrained by the data for each individual participant in an individual session. To do so, we re-analyze the parameter estimates form section 2.10, and find that these statistics are correlated across cross-validation splits (feature drop rate: $\rho = 0.389 \pm 0.007$, tree size: $\rho = 0.410 \pm 0.010$, value quality: $\rho = 0.480 \pm 0.012$). Additionally, we can express the reliability in estimates of these statistics as confidence interval, at least if we assume that the posterior over statistics given data is Gaussian with constant variance for all participants and sessions. This yields a standard error of 0.033 for feature drop rate, 89.38 for tree size and 0.016 for value quality. Therefore, we can conclude that their is considerable uncertainty in the estimates of statistics for a single participant in a single session, but with $N = 30 \times 5$ data points we have enough statistical power to reliably detect changes with expertise or talent.

In figure 3.11C-E, we show that tree size increases across sessions and the feature drop rate decreases, whereas the heuristic quality remains roughly constant. Next, we investigate what underlies talent, i.e. differences in playing strength for individuals with the same amount of experience. Within each learning session, playing strength is correlated with tree size and feature dropping, but not with the heuristic quality (figure 3.11F-H). Note that the variance in playing strength across participants within each session strongly exceeds the within-participant across-session variance. Moreover, the effect of learning session on playing strength (figure 3.11A) is mediated by across-session changes in tree size and feature drop rate.

3.5.2 Performance improvements are not caused by increased thinking time

One relatively unexciting explanation for an increase in tree size with experience is if experienced players spend more time per move. However, the opposite is true: participants make faster decisions on later sessions (see figure 3.13), in games against computer opponents as well as the 2AFC and evaluation block. We define a participant's performance on the 2AFC task as the percentage of trials on which they choose the candidate option with higher game-theoretic value. To compute 2AFC performance, we ignore all trials on which both candidate moves have identical game-theoretic value. On the evaluation task, we define performance as the Pearson correlation between a participant's responses and the game-theoretic values of the presented positions. Participants' performance on the evaluation task increases over time, but their 2AFC performance decreases (see figure 3.12). Participants improve across sessions on the evaluation task but their performance on the 2AFC task decreases. This suggests that participants prioritize speed over accuracy on the 2AFC task in later sessions.

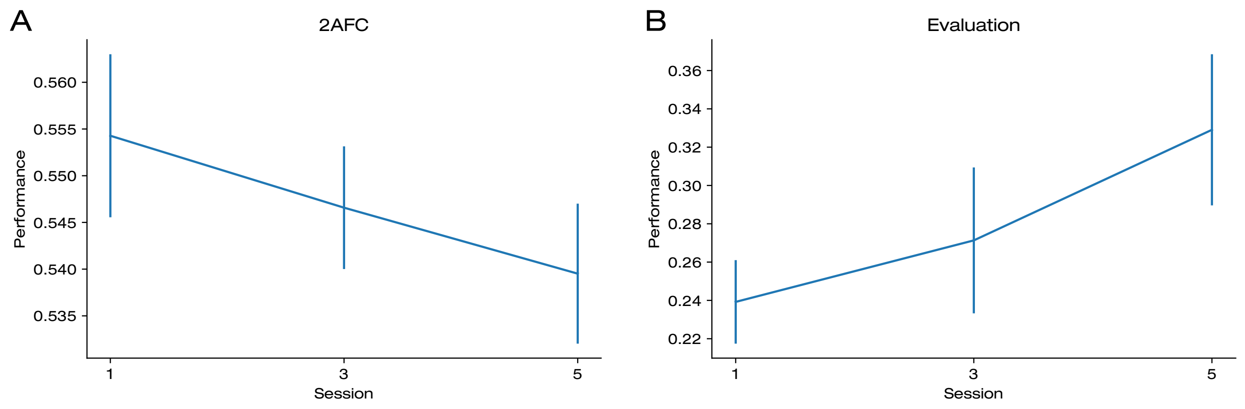


Figure 3.12 Performance on different sessions of the learning experiment, for the **A.** 2AFC and **B.** evaluation task. Errorbars indicate standard error of the difference between a participant's performance in a given session and that participant's average response time across all sessions.

3.5.3 Main result: Stronger players search faster

Finally, neither tree size nor playing strength correlates with participants' mean response time. Therefore, our results suggests that expertise and talent share a common mechanism: stronger players search faster and make fewer attentional mistakes.

Since we had to make some compromises in our model specification and parameter estimation methods, one may worry that these negatively affect the confidence with which we can draw this conclusion. First of all, the noise and uncertainty in parameter estimation might limit the statistical power in the subsequent subsequent analyses relating tree search to expertise and talent. However, the relevant parameters (primarily γ) have test-retest reliability of around 0.5, and we have $N = 150$ data points in fig. 3.11, which provides enough power at least for the positive conclusions of correlations between talent and feature drop rate or tree size (the smallest Spearman correlation that is detectable at 80% power is $\rho = 0.24$, the observed correlations are 0.61 and 0.67, respectively). However, the negative conclusion of no evidence for feature learning should not be conflated with evidence for the absence of feature learning, it is always possible that there is a small effect which we do not have enough power to detect.

More importantly, one might worry that experience-dependent changes in people's decision-making process can be captured by switching to different model specifications rather than adjusting model parameters, or that people may use an algorithm that fundamentally differs from our main model or any of the ones we tested. Specifically, the most plausible concern is that if our model lacks a certain feature that people use and which experts use more than novices, the parameter estimation algorithm might wrongly infer the resulting playing strength increase to stem from increased tree search or reduced feature dropping.

Although it is theoretically impossible to dismiss these concerns without testing all possible extensions of our model, or at least an exhaustive set of features to add, we did perform multiple analyses to address these concerns. First of all, our results in figure 2.13 argue against the first concern of expertise-dependent changes in models rather than parameters. To address the second concern, we note that although we did not explore all possible features, we did test a number of plausible candidates and found no evidence that people use them. Additionally, our trade-off analyses in figure 2.9 suggest that if we found features that improved the model's goodness-of-fit, tree size estimates in that improved model are likely similar to those of the current main model, at least as long as the improved model has the same parameter confusability matrix. Therefore, although we cannot rule out that our observed effects of expertise and talent on tree search and feature dropping are feature learning effect that masquerade as search in our models, we have some evidence against it. Crucially, we do not have evidence in favor.

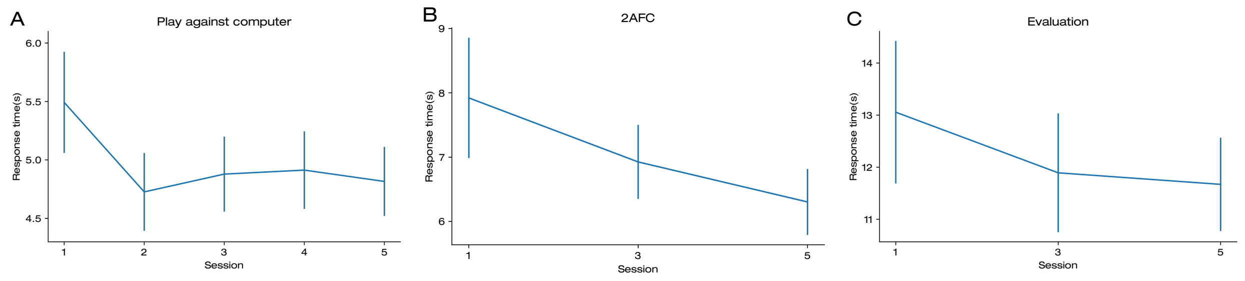


Figure 3.13 Average response time across all trials and all participants in different sessions of the learning experiment, in **A.** games against computer opponents, **B.** 2AFC trials and **C.** board evaluations. Errorbars indicate standard error of the difference between a participant's response time in a given session and that participant's average response time across all sessions.

3.5.4 Experience-dependent performance gains are mediated by model statistics

The playing strength increase across sessions can be predicted by combining the across-session changes in tree size, feature drop rate and heuristic quality with the within-session correlation between those statistics and Elo rating. For each session in the learning experiment, we run a multiple linear regression to predict Elo rating given the three statistics, and average the regression slopes across sessions. We find that 1 Elo point corresponds to 4.0 additional nodes in the decision tree, 0.10% lower feature drop rate or 0.0020 higher heuristic quality. We then predict the across-session effect on Elo rating by multiplying the average tree size, feature drop rate and heuristic quality across participants for each session with these regression slopes. As intercept, we again use the average intercept across sessions. The predicted Elo rating increases across sessions, qualitatively matching the main effect in our data (see figure 3.14). We also predict across-session Elo ratings using the slopes and intercept estimated on the first session, which again matches the data qualitatively. These analyses show that the playing strength increase due to experience is mediated by changes in model parameters, specifically the tree size, feature drop rate and

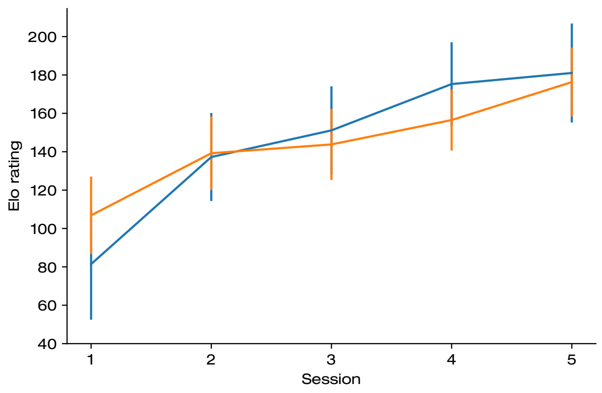


Figure 3.14 Elo rating of participants in each session of the learning experiment (blue curve) and Elo rating predicted using the model statistics (see text, in orange). Error bars indicate standard error across participants of Elo rating in each condition minus that participant's Elo rating averaged across conditions.

heuristic quality.

3.5.5 Robustness of expertise results to alternative model specifications

To test for robustness, we first repeat the above analyses for all 25 alternative models. Second, we ask whether expertise or talent could correlate with heuristic quality in a potential improved model. Specifically, our model may omit features which people use, stronger players may assign those features higher weights, and our parameter optimization may confuse those missed features with additional tree search. However, in section 2.9, we analyze parameter estimates in lesion models and find no such trade-offs. Therefore, even if our model could be improved by adding features, it is unlikely to change our main results.

To demonstrate that our results on the nature of expertise and talent are robust to arbitrary choices in the specification of our model, we repeat the analysis for all lesions, modifications and extensions (figure 3.15). For nearly all models, the Elo rating of individual

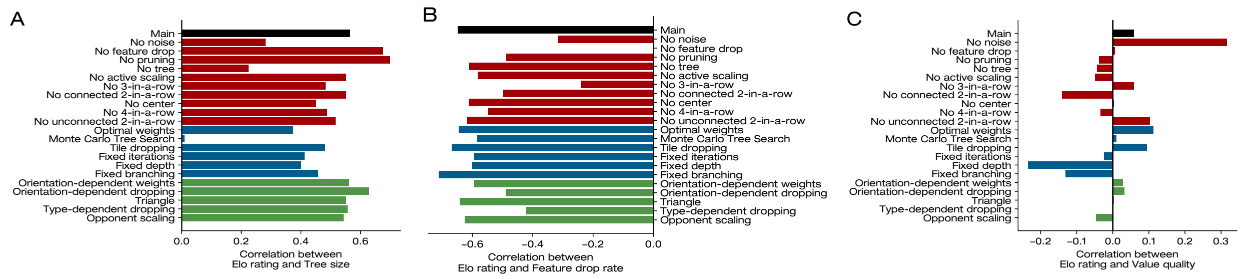


Figure 3.15 Replication of expertise results for alternative model specifications. **A.** Correlation between playing strength and tree size in all alternative models. **B.** Correlation between playing strength and feature drop rate. **C.** Correlation between playing strength and heuristic quality.

participants is correlated with tree size (one-sample T-test across models: $\rho = 0.475 \pm 0.032, p < 0.001$), inverse correlated with feature drop rate ($\rho = -0.559 \pm 0.025, p < 0.001$) and uncorrelated with heuristic quality ($\rho = 0.004 \pm 0.022, p = 0.87$). There are some notable exceptions: the **No feature drop** and No tree are by construction incapable of discovering a correlation between Elo rating and feature drop rate or tree size, respectively, the **Orientation-dependent dropping** and **type-dependent dropping** models result in a relatively smaller correlation between Elo rating and feature drop rate, the MCTS model does not find a correlation between Elo rating and tree size (equivalent to total number of playouts, since we always use 1 playout per node), and finally the **No value noise** model uncovers a modest correlation ($\rho = 0.316, p < 0.001$) between value quality and tree size where the main model does not.

However, these 6 examples are the exceptions that confirm the rule: in the 60 remaining cases, the correlations between Elo rating and tree size, feature drop rate and heuristic quality are remarkably robust. This includes models with substantially different algorithms, such as the **Tile dropping** model which exhibits a correlation between Elo rating and the tile drop rate, and the **Fixed depth**, **Fixed iterations** and **Fixed pruning** models which reveal correlations with tree size. Additionally, the results are robust

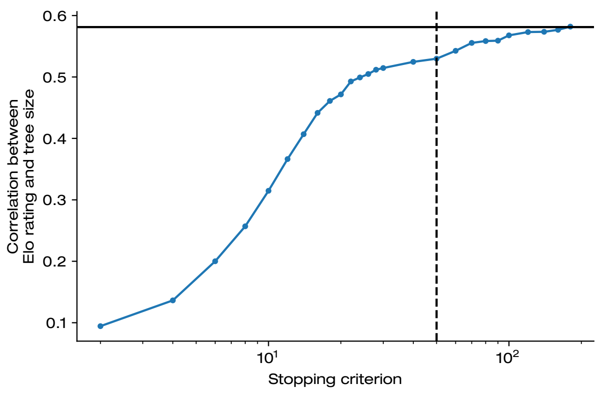


Figure 3.16 Correlation between playing strength and tree size for the main model with a stopping rule, as a function of the stopping threshold. The vertical black indicates the correlation in the main model without any stopping rule (as in 3.11F).

to using models with considerably lower log-likelihood. For example, the **No connected 2-in-a-row** model is 0.086 ± 0.005 log-likelihood units per move worse than the main model, but the results are qualitatively the same: Elo rating correlates with tree size ($\rho = 0.550, p < 0.001$), negatively with the feature drop rate ($\rho = -0.498, p < 0.001$), but not with heuristic quality ($\rho = -0.140, p = 0.086$). In other words, our results hold up even if we use a provably incorrect model specification. It is therefore unlikely that improving our main model for example by adding special cases for opening moves will alter our conclusions on the nature of expertise and talent.

Finally, we show that our conclusions are robust to the addition of a stopping rule, which we included when predicting response times. Note that, since we do not refit model parameters with the stopping rule, the feature drop rate and heuristic quality are independent of the stopping threshold. In figure 3.16), we show the correlation between Elo rating and tree size for models with different stopping thresholds. As for predicting choices and response times, one should not use too low a stopping threshold, but with a threshold of 20 or higher, the correlation is largely constant. In particular, with our choice of

50 iterations, we uncover as always a large correlation between Elo rating and tree size ($\rho = 0.540, p < 0.001$). Note that, since changing the stopping threshold changes the mean tree size built by the model (figure 3.7A), the stopping rule also allows one to incorporate a prior belief over plausible sizes for decision trees that humans can build without changing any conclusions about the nature of expertise and talent.

3.5.6 Survey responses

The common basis for talent and expertise is supported by results from participants' responses in the post-task survey in the human-vs-human experiment. Out of 40 participants, 36 answered the survey; in the remaining cases either the participant declined or we omitted the survey to avoid the session running over time. Even though none of the questions were mandatory, participants answered nearly all of them. One participant left the answer to question 5 (adapting to the opponent) blank; one other participant declined to answer question 7 (experience with other board games). To summarize main themes in people's introspection, we show a word cloud indicating the words that participants use most often in response to question 4 (describing their strategy) in figure 3.17A, or in response to question 5 (adapting to the opponent) in figure 3.17B. In appendix A, we include the full text of all our participant's responses on both questions. Many participants indicate trying to create patterns similar to the features used in our model, such as *"two dots horizontally with one space in between"*, *"a triangle"* or *"a V shape"*. Additionally, participants report trying to *"block the opponent"*, which our model achieves by assigning negative weight to the opponent's features. Curiously, a number of participants indicate not using any strategy at all.

Next, we investigate whether participants' playing strength on our 4-in-a-row game cor-

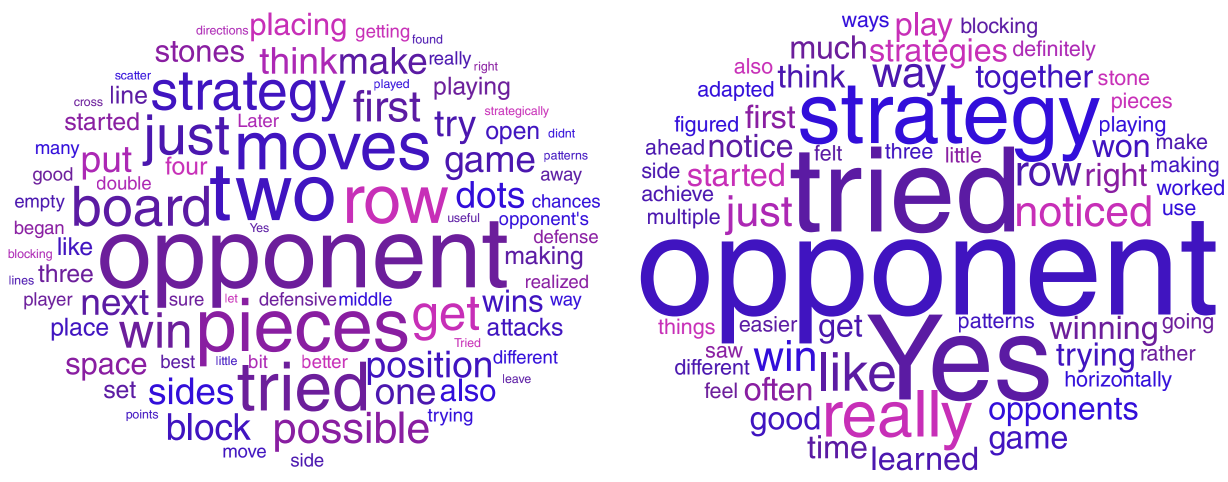


Figure 3.17 A. Word cloud of most common words that participants use to answer question 4 of the post-task questionnaire: *Can you describe how you decided which moves to make? Did you follow a particular strategy?*. **B.** Word cloud of most common words in responses to question 5: *Did you feel you were adapting your strategy to your opponent's game play? If so, can you explain how?*. We created both panels using `wordclouds.com`

relates with their self-reported experience playing other board games or their education level. Unfortunately, we cannot estimate a participants' Elo rating on human-vs-human games, since we estimating Elo ratings requires participants to play games against the same collection of opponents, to serve as a common benchmark. However, we know that in all other experiments, a participant's Elo rating correlates highly with the Elo rating of the model with inferred parameters for that participant. Therefore, we use the model's Elo rating as a stand-in for a participant's playing strength.

To score answers to question 2 (education level), we convert *"Did not complete high school"* to 1, *"High school"* to 2, etcetera. Likewise, on question 6, we converted *"Never played"* to 0, *"Played once or twice"* to 1 and so on. We scored questions 7 to 11 (game-playing expertise) by counting the number of unique games mentioned by a participant.

We find that playing strength does not correlate with education level (Spearman: ρ =

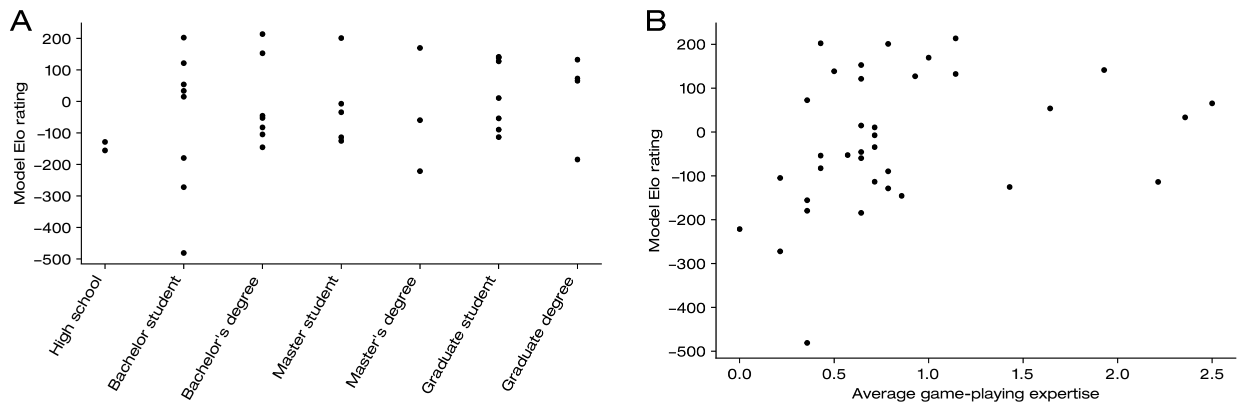


Figure 3.18 A. Scatterplot in which each point represents one participant in the human-vs-human experiment, the x-coordinate their level of education, and the y-coordinate the Elo rating of the main model with parameters inferred from that participant's moves. **B.** Scatterplot of average game-playing expertise reported by each participant against their Elo rating.

$0.21, p = 0.21$, see figure 3.18A), or self-reported experience on any of the games (highest correlation for chess, with $\rho = 0.29, p = 0.088$). This lack of correlation is unsurprising given the noise caused by the approximation of participants' playing strength by the model's Elo rating and the inherent subjective nature of self-reports. However, playing strength does correlate with the average self-reported experience across all games ($\rho = 0.40, p < 0.05$, see figure 3.18B). This suggests that individual differences in talent in our game may be due to expertise in related games.

3.6 Time pressure

To validate these analyses, we conducted another experiment in which participants play with a time limit of 5, 10 or 20 seconds per move. We group all games by time limit condition and independently estimate performance and model parameters for each participant and condition. Response times in this experiment are lower compared to the learning, generalization and eye tracking experiments, and participants play faster with additional time pressure (figure 3.19A). Participants in the time pressure experiment play worse than the other experiments but surprisingly, performance does not scale with time limit (figure 3.19B).

Based on behavioral economics studies which find that people switch to less computationally demanding strategies under time pressure in economic games (Lindner and Sutter, 2013) and chess literature which conceptualizes tree search and pattern recognition as fast and slow processes respectively (Calderwood et al., 1988; Chabris and Hearst, 2003; Van Harreveld et al., 2007), we expected time constraints to affect tree size. Figure 3.19C confirms this prediction: tree size is lower in the time pressure experiment and decreases with more severe limits. The introduction of time constraints did not affect heuristic quality, but the feature drop rate is higher. Surprisingly, the feature drop rate is highest for the least severe 20-second condition (figure 3.19D. We speculate that the stress of potentially losing on time causes participants to pay more attention with shorter time limits, whereas with 20 seconds, they are more relaxed and make more attentional lapses.

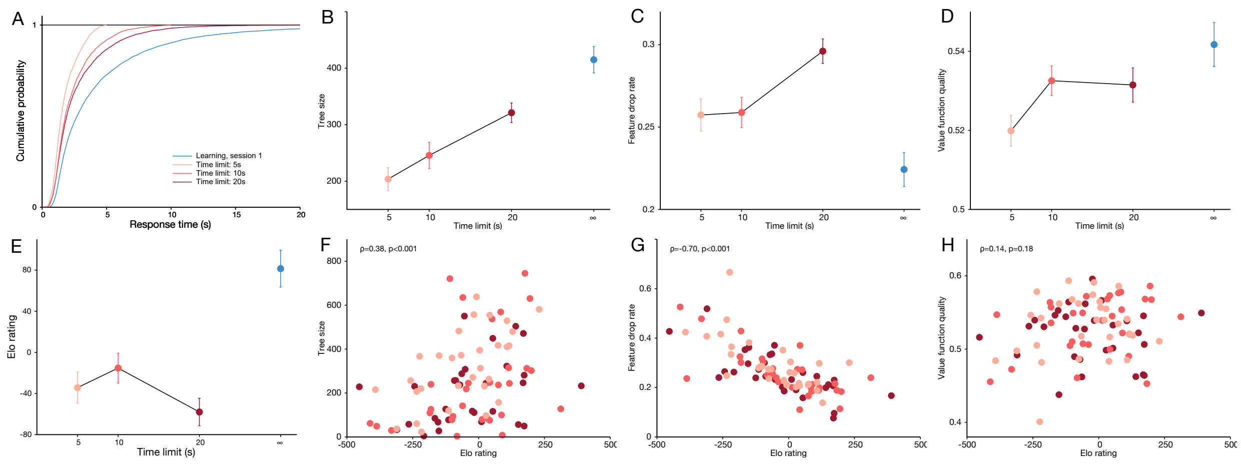


Figure 3.19 Same as figure 3.11, except for the time pressure experiment, and with Elo rating, model statistics and response time plotted as a function of time constraint condition.

As in the learning experiment, performance correlates across individuals with tree size and

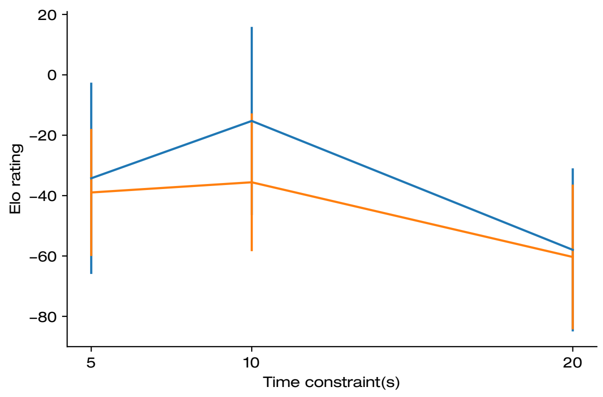


Figure 3.20 Same as figure 3.14, except for the time pressure experiment.

feature drop rate, and the effect of time limit on performance is mediated by these model parameters. In the 20-second condition, participants build larger decision trees but also drop more features. These opposite effects happen to be approximately equal, which explains the lack of correlation between time limit and performance.

Next, we repeat the same mediation analysis on the time pressure experiments and show that within-condition correlations can predict the across-session trend (or absence thereof) in Elo rating. We find similar regression coefficients for tree size (4.29 nodes/Elo point) and feature drop rate (0.087% per point) but the heuristic quality is inconsistent (1 Elo point $\sim$ 0.0027 decrease). As before, the prediction matches the data qualitatively (see figure 3.20), in that there is no correlation between time constraint condition and playing strength. Note that this is non-trivial since the time limit does correlate with tree size and feature dropping, but those opposite effects cancel out.

3.7 Discussion

In this chapter, we showed that the behavioral model developed in the previous chapter can predict response times, eye movements and cursor movements, thereby supporting its interpretation as a process model. We did not perform model comparison on non-choice data, partly for technical reasons outlined above, partly due to a lack of time. Therefore, the data in this chapter do not speak to any preference between behavioral models, but only to the interpretation of our main model which we selected purely based on choice data. Since we found that choice data does not distinguish between various modifications of the main model, this leaves open the possibility that predict response times, eye movements or cursor movements could provide a more refined distinction.

We also find that experts or talented naive players build larger decision trees, drop fewer features but use similar features and weights. However, this conclusion is subject to caveats. First, we have analyzed parameter estimates in a behavorial model, and this model is imperfect. One limitation is that we hand-picked a small set of features with no a priori justification. We already showed that our results are robust to adding some plausible features, such as the 'triangle'. A more systematic approach would be to construct feature sets by fitting neural networks to people's choices or training auto-encoders on the set of all board states in our data. It is however, not clear that adding features to the model would affect its estimates of people's tree size, as our analysis in section 2.9 argues against such parameter trade-offs.

Another imperfection is our assumption that model parameters are constant within each session, and change discontinuously from day to day. Adapting our inference procedure to estimate continuously varying parameters will turn the discrete learning curves in Figure 3.11 into continuous ones, but is unlikely to change their shapes. Alternatively, one could construct a reinforcement learning agent that learns to play our game using policy gradient descent with proper inductive biases (Lake et al., 2017; Silver et al., 2016).

Second, even if the model were correct, our log-likelihood estimation and parameter inference procedures are not. Although log-likelihood estimation with inverse binomial sampling is unbiased, its variance prevents us from making fine distinctions between models such as MCTS or heuristic search (figure 2.12). In the future, we plan to increase our statistical power by collecting large-scale data using Amazon Mechanical Turk or a custom Android/iPhone app.

Moreover, since we estimate parameters by maximizing these noisy log-likelihood estimates, one may worry that parameters get confused, in particular tree search, feature dropping and feature weights. In the learning and time pressure experiments, these specific parameters are dissociated: tree search increases with experience or thinking time, feature dropping decreases in the former but increases in the latter, and value approximation quality remains unchanged. Therefore, these parameters capture different aspects of people's performance. To more clearly link parameter estimates with psychological constructs, we plan to conduct a study in which participants play our game as well as common tests of attention, working memory or reasoning. Furthermore, we intend to support our model by investigating neurophysiological correlates of components of our model, specifically value function approximation and multivariate BOLD activity in the ventromedial prefrontal cortex.

Another potential concern is that our conclusions might not generalize beyond our specific game, and the nature of expertise may be different in other games. One way in which our game is special (for example compared to chess or go) is that finding a good set of features and weights to approximate the value function is easy, also for novices. Therefore, our observed lack of correlation between value approximation quality and expertise or talent might be a ceiling effect. However, other games or real-life sequential decision-making tasks often have even more possible states than our game, and the correlation between expertise and amount or depth of search is potentially even larger than in 4-in-a-row. Specifically, our research suggests that the sometimes negative findings on tree search in chess experts may be due to the field's reliance on verbal reports as data, and the role of computational processing speed in general cognitive expertise or talent may be underestimated.

Appendix A: Responses to open-ended survey questions

Responses to question 4 on survey after human-vs-human experiment: *Can you describe how you decided which moves to make? Did you follow a particular strategy?*

- Looking for win-win situations
- I tried to make a triangle with tree stones, with a whole in the middle. Later on, I tried to put two stones next to eachother on a horizontal line and get enough space on both sides, so the opponent could only stop me at one side.
- I would try to block the other opponent.
- I don't think I followed a particular strategy. I tried to think a couple moves ahead and try to win diagonally, which catching diagonal wins seems to be the most difficult, at least for me.
- I play this game a lot with my little brother it's like Connect 4
- Mostly just blocking my opponent's moves while at the same time trying to set up my own moves.
- My strategy was to switch up my moves from game to game, and not constantly make the same decisions. I also tried to scatter my pieces around the board a bit more than the previous game.
- Yes- what worked best was placing two dots next to each other horizontally (I found this after she started winning with placing two dots horizontally with one space in between), because then it is easy to get three dots in a row unless one side has been blocked by a color of the opposite player.
- I initially made a V shape. It optimizes all the possible variations of 4. I was also very cognizant of where my opponent chose to put her piece. put middle.or not really sure
- Started in the middle and worked my way out.

- One obvious win condition was “three in a row with empty sides”, which makes “two in a row with empty sides” a very strong position. I would check first if I imminently needed to defend before considering other strategies: possibility of opponent creating a four, or opponent has established two in a row with empty sides. I tried lots of different strategies at different times and the following are just a few of them: - Establish a neighborhood for myself, with lots of adjacent pieces and promising directions that might bear fruit later (in general, I started towards the end to think of any adjacent pair of two pieces as “pointing” in some direction, perhaps two directions if two sides are open, and in general I think the board would be well re-represented mentally as a field of intersecting four-long line segments radiating from/between pairs of stones that are adjacent or near enough to establish a “line of attack”. Then placing stones that create more such lines, and destroy opponents’ lines, is a useful reframing) - Establish coincidence-rich zones where many of my pieces are adjacent or one away, rather than scattering my pieces around - Divert attention by playing somewhere else; don’t fall victim to simply defending, but if the defense isn’t *immediately* critical, consider taking the chance to enhance your position somewhere else just a tiny bit before returning to put out fires - Play moves that look like “just” defending, so the opponent explains them away as defense and fails to read them as offense (this didn’t work against current opponent, though) - I didn’t like forcing “trivial” moves like one-for-one exchanges that just filled up board space, but in the last game my opponent did that while making sure his “forcing my hand” moves were actually strategically good for him and created coincidences. - Make attacks that force my opponent to defend, in such a way specifically as to put my own pieces in a better position, and the opponent’s defensive moves in a fairly useless position. - The “widthwise” attacks have more flexibility because the edges of the board don’t stop you (unlike crosswise and heightwise attacks), but the “crosswise” attacks are less perceptually salient and so easier to sneak under the opponent’s noise.
- Damage limitation, getting 3 in a row appears to be key to getting 4 in a row....

- I am not consciously using any strategies..
- i mainly played not to lose versus playing to win...and that really hurt me
- I did not follow a particular strategy in the beginning. But then I noticed that I would win if I set up the pieces so that more than one option would win. So I began placing my pieces more strategically in the beginning of the session–trying to spread them out a little bit. But sometimes I just followed my instinct and didn't think much about where I was placing them.
- Tried to block opponents potential paths, tried to scatter my moves for more opportunities
- At first it was trial and error. Then I realized that I was on defense and knew I better play offensively. But then I recalled my childhood when I played connect four and remembered that three dots next to each togeher with open ends on both sides is an automatic win. Then my opponent caught on. And I began starting with the same move and the exact same strategy, if my opponent tried something differently. I thought hard then moved.
- Tried to get double plays (2 possible simultaneous wins)
- Avoid letting partner get 3 points together, and try to make two pairs of 3 points. the best strategy at first was to place as many dots horizontally in a row as possible. if they let me get away with 2 dots and not place any of theirs next to mine they were doomed.
- did not begin with a strategy, then it became clear that due to the unique dimensions of the board a horizontal 3 in a row with space either side was a good position to get into. honestly, i didn't really think before i started playing...
- i didnt have a particular strategy. i just tried my best and that didnt work out fine
- In the first 10 min I had no strategy. Later I realized that having 2 in a row with no adjacent opponent stones can give you an advantage. Also having 4 down and 4 oblique sometimes leaves the opponent defenseless.
- yes I always try to keep two chances to win two chances
- The trick was to have a double cross in whatever way possible. My specific strategy was to have my lead on all the four corners, which wold increase my chances to achieve a double

cross.

- I mostly attempted to block my opponent from winning.
- I tried a few different strategies. Towards the beginning I was trying to cut off paths my opponent was making, but doing so at a distance from his pieces so that I would have more space to make my moves. Later on, I realized that it was good to use the middle of the board because that left more options open. When starting first,
- I would also try to spread out my pieces so that there were two spaces in between each of them. This would also leave more possible combinations open. Also though considering blocking his movements.
- At the beginning, was exploring the game. Then I saw the value of three neighbouring pieces as a branch point, and tactics were devoted to setting these up and preventing them. I found that spreading out on the board (as per Go) was a useful idea. However, many games were won on the basis of blunders (unforced errors).
- It seems important on each move to maximize the possible combinations which can be extended to a row of four. Some patterns are better suited for this than others.
- I took the defensive approach, essentially trying to block her ability to make a 4-block line, while also, ideally, making my own set.
- Not exactly. In the first 2-3 games I was going blindly and off of luck, but after that I tried to make moves that would leave the opponent with no other choice but losing.
- NO
- Initially, I was just playing it by ear and making sure to block him when he was getting close to 4 and adapt if he blocked me, but as time went on I began to notice patterns and realize that I could lull him into a false sense of security than I could set up a situation where he couldn't block me in just one move so I would automatically win as long as I started on the right place on the board and he didn't catch on too quickly.
- Yes! Horizontal has more degrees of freedom then vertical. Three in a row horizontal with nothing flanking guarantees a win. I try to stagger pieces maximizing possible wins with an

emphasis on horizontal wins.

- Start at the center of the board. Try to get on the offense and force him to place his chips where I want them. When I was on the defensive, don't let him get two chips next to each other horizontally. If he did this, I tried to see it and put mine right next to his.

Responses to question 5: *Did you feel you were adapting your strategy to your opponent's game play? If so, can you explain how?*

- Yes, she eventually recognized my tricks and then consistently beat me :(
- Yes, so I saw that the "two-stone-in-an-empty-horizontal-line" worked pretty good. Also, i tried to play not a stone directly to my opponents last stone, so he would be confused about what i was planning to do
- I felt that the strategy worked at most times.
- If anything I feel like the opponent was adopting my strategy, particularly of where to place the first piece.
- Not really
- Yes, by keeping an eye on her strategy and predicting my opponent's next move
- After playing a game or two, I tried to make it harder for her to get three shapes in a row going horizontally. I noticed that when she achieved this, it was much easier for her to get 4 in a row.
- Yes, I learned from her strategy after she won several in a row. It was also always easier to win when I started.
- At all times I tried to be aware of all combinations where she could win. If she was closest to winning I would focus on blocking her as opposed to advancing my pieces
- no
- Yes. She won the first few games and then I started to notice the way she played and it altered my choices.

- My opponent often didn't claim obvious conversions from two-with-empty-sides to three-with-empty-sides, and often didn't notice them when I made them, so I learned that those were often good ways to go. In exploring the space of strategies, I definitely tried a few that *neither* of us had tried, so that's another sense of adapting - like jumping entirely to the other side of the board, or crowding right up to my opponent, as things we hadn't tried together yet. I avoided moves that together we had "exhausted", like the stone-empty-stone second move, and tried other things.
- Not really
- not really
- Yes, I noticed that the opponent did not always catch it when I was about to win with a diagonal, so I tried to use them.
- A little. I learned to recognize his steps, but sometimes I forget or don't realize he is setting up a trap in a different way and ends up winning.
- No, not really. I caught myself trying to play defensively until I was able to be on offense again. I had to think ahead. I kept blocking my opponents wins only to regain control and play my same strategies hoping my opponent was not thinking ahead also. Memories of my childhood playing games with my father resurfaced.
- Not really
- No
- sort of, they were mostly just doing a three in a row (diagonally or horizontally or vertically) which i would have to block; they seemed to use little strategy.
- definitely adapted. seeing how she won helped to avoid that situation in the future. some very careless mistakes on my part though. yes somewhat but not really im not a very good game player at all. i was trying to just stay right where he was to stop him from winning
- No. I played it safe. I assumed the opponent had similar strategies and offered a draw when I saw there was no way out for me.
- Yes I tried to deviate his concentration in different ways

- My opponent was tying single line strategy so in this way he could achieve a double cross from either left or right side.
- Yes because I was a step behind most of the time.
- Yeah, I was. Or, I noticed that he would miss more opportunities than I initially thought he would. So, I adapted to this by just going for completed paths rather than trying to make multiple opportunities and corner the opponent.
- Yes, because we were both figuring it out at the beginning. We both figured out the branch point thing at roughly the same time.
- At first, it seemed like the game would not offer much room for strategy, but rather it would depend on the opponent making a mistake. After a while I realized this is not the case, and I could see me and my opponent picking up the strategy over time. I think both of us recognized how some patterns inevitably led to a win of one of the opponents, so we both started to try to achieve these patterns and prevent them for the opponent.
- Yes, I noticed early on she liked making larger squares and then working inside those areas, I tried imitate or find holes in that method.
- Yes, I tried to predict how the opponent would think/see the board and I made according moves so that I would be unexpected
- I go with the flow
- Over the course of multiple rounds I began to get accustomed to what he would notice and what he wouldn't and got smarter about placing pieces to distract him and knowing what he would and would not anticipate.
- Nah, I think we were both learning together
- I didn't feel much adaptation. It just felt like we both figured out how to be more cautious, leading to a lot more draws.

Chapter 4

Unbiased log-likelihood estimation with inverse binomial sampling

4.1 Introduction

The most important mathematical object in behavioral modeling studies is arguably the likelihood function, which measures how well a behavioral model with a given set of parameters can explain data. Specifically, for a dataset of discrete values, the likelihood has the simple interpretation of the probability that a random sample from the model matches the data, up to a normalization constant. The behavioral modeling approach to the study of human cognition adjudicates scientific hypotheses by measuring differences in model parameters between participant groups or experimental conditions, or by measuring evidence for competing behavioral models. The likelihood function is indispensable in both approaches. A principled method to find best-fitting model parameters for a given participant is maximum-likelihood estimation (MLE), which entails optimizing the likelihood function over parameter space. Other common parameter estimation methods, such as maximum a posteriori (MAP) estimation or posterior sampling, still involve the likelihood function. Additionally, computing model evidence to compare models entails integrating the likelihood function over parameter space, and common approximations to this integration (cross-validated log-likelihood, AIC, BIC, DIC) all require to compute likelihoods.

However, the likelihood of complex models is often intractable to compute analytically or numerically. In those cases, one commonly resorts to sampling. A sampling method consists of two components: a "sampling policy", which is a rule that determines how long to keep drawing samples for, and an "estimator", which converts the samples to a real-valued number. To estimate the likelihood of a participant's response on a single trial, the most obvious sampling scheme is to draw a fixed amount of samples from the behavioral model, and the most obvious estimator is the fraction of samples that match the participant's response. However, most applications require one to compute or estimate not just the likelihood, but its logarithm (see section 4.2.1 for underlying technical reasons), and the "fixed sampling" method above cannot provide uniformly unbiased estimates for the log-likelihood (see section 4.3). For most common estimators, the bias vanishes in the asymptotic limit of infinite samples, but drawing samples from the model can be computationally expensive, especially if the behavioral model is complex. In practice, the bias introduced by these estimators can cause considerable biases in estimates of model parameters or even reverse the outcome of model comparison analyses. In other words, using poor sampling methods can cause researchers to draw incorrect conclusions about scientific hypotheses.

In this chapter, we introduce inverse binomial sampling (IBS), a sampling method that provides uniformly unbiased estimates of the log-likelihood. Its variance is uniformly bounded and achieves the minimum variance for any unbiased estimator (given by the information inequality (de Groot, 1959), the analogue of the Cramer-Ráo bound for non-fixed sampling). We compare IBS and fixed sampling, used for maximum-likelihood estimation of parameters of simple behavioral models in simulated data for two example tasks: orientation discrimination and change localization. In both tasks, IBS produces lower root mean squared error in estimated parameters than fixed sampling with the same average number of samples. Our results demonstrate the potential of IBS as a practical, robust, and easy-to-implement method for log-likelihood evaluation when exact techniques are unavailable.

4.2 Definitions and notation

The two necessary ingredients to run IBS are

1. A data set $\mathcal{D} = \{(s_i, r_i)\}_{i=1}^N$ consisting of N trials on which a participant observes a stimulus s_i and makes a response r_i.
2. A behavioral model: a stochastic function that takes as input a stimulus s and a parameter vector $\vec{\theta}$ and outputs a response r.

Implicit in this definition is the requirement that the model outputs on different trials are independent, and that model parameters are constant during the experiment. The definition can easily be relaxed by incorporating time into the "stimulus" s and including time-dependent parameters explicitly in the model specification. However, the assumption of independence is required for IBS to work. These assumptions allow us to write the likelihood function as

$$\mathbb{P}\left(\mathcal{D}\middle|\vec{\theta}\right) = \prod_{i=1}^{N} \mathbb{P}\left(r_i|s_i, \vec{\theta}\right) = \prod_{i=1}^{N} p_i$$

where we introduce p_i as a shorthand for $\mathbb{P}\left(r_i|s_i,\vec{\theta}\right)$ when both the model and parameter vector are fixed, which will be the case in the majority of this chapter.

Crucially, we assume that the model is too complex to derive an analytical expression for the likelihood function. Therefore, the only way to estimate the likelihood or any function thereof is by drawing samples $r \sim \mathbb{P}\left(\cdot|s_i,\vec{\theta}\right)$ on each trial and matching them to the response r_i. This approach requires that there is a nonzero probability for a random sample r to match r_i, so for the remainder we will assume that the space of responses is discrete and reasonably small. However, it may be straightforward to extend IBS to large or continuous response spaces using approximate Bayesian computation (ABC, see section 4.7.3). Finally, we assume that drawing a sample from the behavioral model is at least moderately computationally expensive (i.e., time-consuming – for example, of the order of a fraction of a second or more), and that the computational cost is independent of the stimulus, response or model parameters.

4.2.1 Reduction to Bernouilli sampling

Regardless of the sampling method, the independent-trial assumptions implies that, to estimate the likelihood of the entire data set, we cannot do better than to estimate p_i on each trial independently, and combine the results. However, combining estimates $\hat{p}_i$ into an estimate of $\prod_{i=1}^{N} p_i$ is non-trivial, since the distribution of $\prod_{i=1}^{N} \hat{p}_i$ only converges if $\hat{p}_i$ is almost surely nonzero for each trial i and even then it converges to a log-normal distribution. Instead, it is easier to estimate $L_i = \log p_i$ for each trial and estimate the log-likelihood

$$L\left(\vec{\theta}\right) = \log \mathbb{P}\left(\mathcal{D}\,|\theta\,\right) = \sum_{i=1}^{N} \log p_i = \sum_{i=1}^{N} L_i$$

We can estimate this log-likelihood by simply summing estimates $\hat{L}_i$ across trials, in which case the central limit theorem guarantees that the distribution of $\hat{L}\left(\vec{\theta}\right)$ is normally distributed.

We can make one additional simplification. A behavioral model specifies a probability distribution $\mathbb{P}\left(r|s_i, \vec{\theta}\right)$ for each trial. However, to estimate the log-likelihood, we do not need to know the full distribution, only the probability for a random sample r from the model to match the participant's response r_i. Therefore, we can convert each sample r to

$$x = \begin{cases} 1 & \text{if } r = r_i \\ 0 & \text{otherwise} \end{cases}$$

and lose no information relevant for estimating the log-likelihood. By construction, x follows a Bernoulli distribution with probability p_i. Note that this holds regardless of the type of data, the structure of the behavioral model or the model parameters. The only difference between models and data sets is the distribution of the likelihood p_i across trials[I]. Therefore, we can reduce the problem of estimating the log-likelihood of a behavioral model by sampling to a smaller problem: Given a method to draw samples $(x_1, x_2, \dots)$ from a Bernoulli distribution with parameter p, estimate $\log p$ with as precisely and accurately as possible using on average as little samples as possible.

[I] In this paragraph and in the remainder of this chapter, we consider distributions and likelihoods over variables that themselves are (log)-likelihoods. Although this may occasionally result in confusing language, these mathematical objects are well-defined. We can interpret p as a parameter of the Bernouilli distribution, and apply standard Frequentist or Bayesian statistical reasoning. For the Bernoulli distribution, the mapping from this parameter to the probability mass function is exceptionally simple, but this imposes no conceptual difficulties.

4.2.2 Sampling policies and estimators

A *sampling policy* is a function that, given a history of samples $(x_1, x_2, \ldots, x_k)$, decides whether to draw an additional sample or not. In this chapter, we compare two sampling policies:

1. The *fixed* policy: draw a fixed number of samples M, then stop, regardless of the value of the samples. This is the simplest sampling policy.
2. The *inverse binomial sampling policy*: keep drawing samples until $x_k = 1$, then stop.

For the fixed sampling policy, since all samples are independent and identically distributed, a sufficient statistic for estimating p from the samples $(x_1, x_2, \ldots, x_M)$ is the number of correct samples $m = \sum_{k=1}^{M} x_n$. The most obvious estimator is then

$$\hat{L}_{\text{naive}} = \log\left(\frac{m}{M}\right),$$

but this has infinite bias, since as long as $p \neq 1$, there is always a nonzero chance that $m = 0$, in which case $\hat{L}_{\text{naive}} = -\infty$. This divergence can be fixed in multiple ways; we use

$$\hat{L}_{\text{fixed}} = \log\left(\frac{m+1}{M+1}\right). \tag{4.1}$$

For inverse binomial sampling we note that, since x is a binary variable, the policy will always result in a sequence of samples of the form $(0, 0, \ldots, 1)$, but the length of the sequence is a stochastic variable, which we label K. Moreover, since each sample is independent and 'correct' with probability p, the length K follows a geometric distribution with parameter $1-p$.

$$\mathbb{P}\left(K=k\right)=p(1-p)^{k-1}$$

We convert a value for K into an estimate for $\log p$ using

$$\hat{L}_{\text{ibs}}=-\sum_{k=1}^{K}\frac{1}{k}, \tag{4.2}$$

which we will explain in section 4.4. First, however, let us understand why fixed sampling is not a good policy.

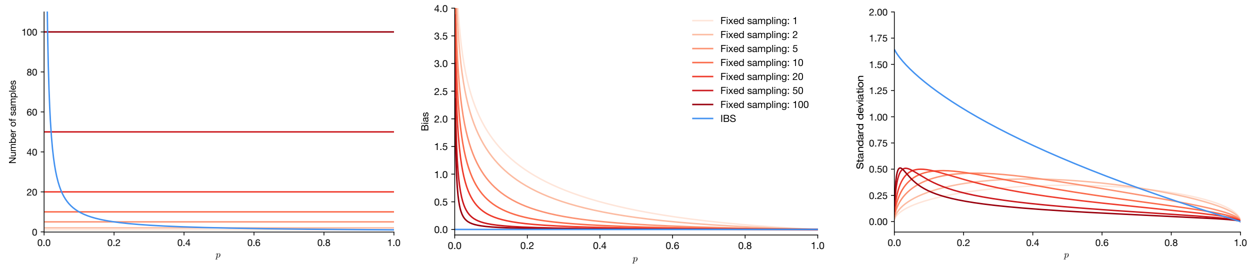


Figure 4.1 A. Number of samples used by fixed (red curves) or inverse binomial sampling (blue) to estimate the log-likelihood $\log p$ on a single trial with probability p. IBS uses on average $\frac{1}{p}$ trials. **B.** Bias of the log-likelihood estimate. The bias of IBS is identically zero. **C.** Standard deviation of the log-likelihood estimate.

4.3 Why fixed sampling fails

The primary disadvantage of the fixed sampling policy is that its estimates of the log-likelihood are biased (see figure 4.1). The bias decreases as one takes more samples, but for $p \to 0$, the estimator remains biased. More precisely, in a joint limit where $M \to \infty$, $p \to 0$ and $pM \to \lambda$ for some constant λ, the bias collapses onto a single master curve (see figure 4.2). This follows since in this limit, the binomial distribution $\text{Binom}\left(\frac{\lambda}{M}, M\right)$

converges to a Poisson distribution $\mathrm{Poiss}(\lambda)$ and therefore the bias converges to

$$\mathbb{E}\left[\hat{L}_{\text{fixed}} - \log \frac{\lambda}{M}\right] \to \exp(-\lambda) \sum_{m=0}^{\infty} \frac{\lambda^m}{m!} \log(m+1) - \log \lambda \tag{4.3}$$

From figure 4.2, we observe that the bias is close to zero for $\lambda \gg 1$ and that it diverges when $\lambda \ll 1$, or equivalently, for $M \gg \frac{1}{p}$ and $M \ll \frac{1}{p}$, respectively. This asymptotic behavior is not a coincidence. In fact, it is mathematically guaranteed since the Fisher information of $\mathrm{Poiss}(\lambda)$ equals $\frac{1}{\lambda}$ and the reparametrization identity for the Fisher information (Lehmann and Casella, 2006) yields $I_f(\log \lambda) = \lambda$. In the limit of $p \ll \frac{1}{M}$, which corresponds to $\lambda \ll 1$, this Fisher information vanishes and the outcome of fixed sampling simply provides zero information about $\log \lambda$ or $\log p$. Therefore, any estimates of $\log p$ are not informed by the data and instead are a function of the regularization chosen in the estimator $\hat{L}_{\text{fixed}}$ (equation 4.1). Note that the argument above does not invoke the specific form of the estimator, and therefore holds for any choice of regularization.

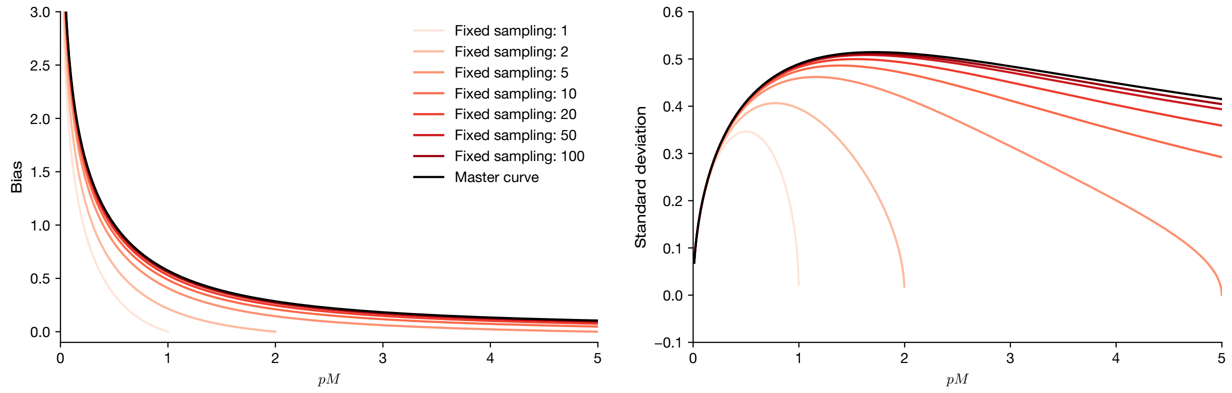


Figure 4.2 A. Bias of fixed sampling estimators plotted as a function of pM, where p is the likelihood on a given trial, and M the number of samples. As $M \to \infty$, the bias converges to a master curve (equation 4.3). **B.** Same, but for standard deviation.

We can express the problem with fixed sampling more clearly using Bayesian statistics. The “correct” belief about $\log \lambda$ given the outcome of fixed sampling (m) is quantified by the posterior distribution $\mathbb{P}(\log\lambda|m)$, which is a product of the likelihood $\mathbb{P}(m|\log\lambda)$ and a prior $\mathbb{P}(\log\lambda)$. In the limit $\lambda \ll 1$, the Poisson distribution converges to a Kronecker delta distribution concentrated on $m = 0$. In other words, almost surely none of the samples taken by the behavioral model will match the participant's response. When $m = 0$, the likelihood equals $\exp(-\lambda)$, which is mostly flat (when considered as a function of $\log\lambda$, see figure 4.3) for $\log\lambda \in [-\infty, -2]$ and therefore our posterior belief ought to be dominated by the prior $\mathbb{P}(\log\lambda)$. In a sense, the choice of regularization in $\hat{L}_{\text{fixed}}$ reflects a prior over the values of $\log p_i$ in the experiment. If that prior mismatches the true (but unknown) distribution, the estimator will be biased.

Finally, to convey the intuition of the argument above, we provide a gambling analogy. Imagine playing a slot machine and losing the first 100 bets you make. You can now deduce that this slot machine likely has a win rate less than 1%, but there is no way of knowing whether it is 1%, 0.1%, 0.01% or even 0% apart from any priors you may have (for example, you expect that the house has stacked the odds in their favor but not overwhelmingly so). This uncertainty is unlikely to affect your decision whether to continue playing the slot machine, since the expected value of the slot machine depends linearly on its win rate. However, if your goal is to estimate the logarithm of the win rate, the difference between these percentages is infinitely large.

4.3.1 Why fixed sampling cannot be fixed

The asymptotic analyses above suggest an obvious solution to prevent fixed sampling estimators from becoming biased: make sure to draw enough samples so that $M \gg \max_{i=1\ldots N} \frac{1}{p_i}$. Although this solution will succeed in theory, it has practical issues. First of all, choosing M requires knowledge of p_i on each trial, which is equivalent to the prob-

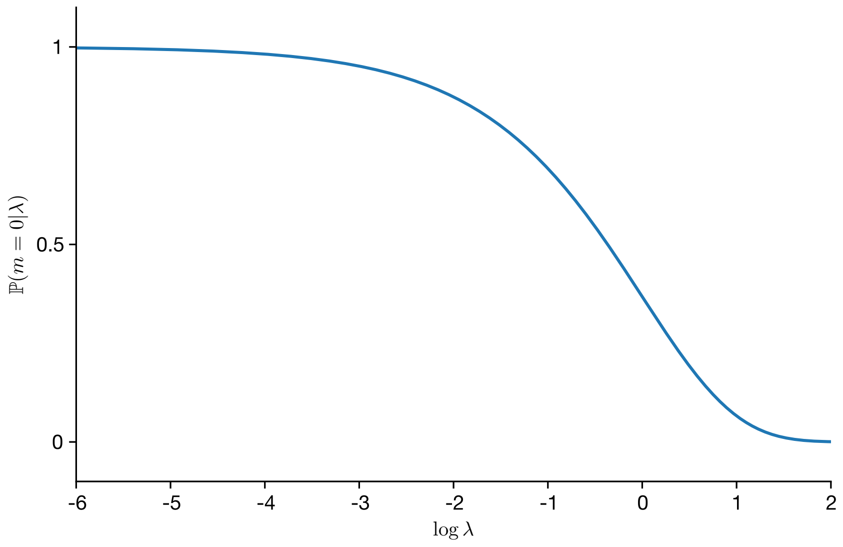


Figure 4.3 Likelihood function of $\log\lambda$ given that fixed sampling returns $m = 0$ (none of the samples from the model match the participant's response). The likelihood is approximately flat for all $\log\lambda \leq -2$. Since λ is defined as $\frac{p}{M}$, this implies that the posterior distribution over p will be dominated by a prior rather than evidence, as quantified by the likelihood.

lem we set out the solve in the first place. Moreover, even if one can derive or estimate an upper bound on $\frac{1}{p_i}$ (for example, in behavioral models that include a lapse rate), fixed sampling will be inefficient. As shown in figure 4.2, the bias in $\hat{L}_{\text{fixed}}$ is small when $\lambda \approx 1$ or $M \approx \frac{1}{p}$ and increasing M even further has diminishing returns, at least for the purpose of reducing bias. If we choose M inversely proportional to the likelihood on the trial where model is least likely to match the participant's response, we will draw many more samples than necessary for all other trials.

One might hope that in practice the likelihood is approximately constant across trials, but the opposite is true. For example, imagine a orientation discrimination task with a behavioral model that includes measurement noise and lapses (see example 4.6.1 below). Imagine moreover, that the experiment contains ≈ 500 trials, and the participant's true lapse rate is around 1%. The model will always assign more probability to correct responses than errors, so for all correct trials, p_i will be at least 50%. However, there will likely be handful of trials where the participant lapses and makes a grave error (responding incorrectly to a stimulus very far from the decision boundary), in which case p_i will be 0.5%. This hypothetical scenario is not exceptional, in fact it is almost inevitable in any experiment where participants occasionally make unpredictable responses, and perform hundreds or more trials.

A more sophisticated solution would be relax the assumption that M needs to be constant and choose M as a function of p_i on each trial. However, since p_i is unknown, one would need to first estimate p_i by sampling, choose M_i for each trial, then re-estimate L_i. Such an iterative procedure would create a non-fixed sampling scheme, in which M_i adapts to p_i on each trial. This approach is promising and it is how we originally arrived at the idea for inverse binomial sampling.

Finally, a less sophisticated solution would be to disregard any statistical concerns, pick M based on some intuition or copy the value used in a previously published research article, and hope that the bias turns out to be negligible. We do not intend to dissuade researchers from using such pragmatic approaches if they work in practice. Unfortunately, this one does not. As figure 4.2 shows, estimating log-likelihoods with fixed sampling can cause biases of 1 or more points of model evidence if the data set contains even a single trial on which $p_i \leq \frac{1}{2M}$. Since Bayes factors larger than 5 are often regarded as strong evidence (Jeffreys, 1998; Kass and Raftery, 1995), it is well possible for such biases to affect or reverse the outcome of a model comparison analysis[II]. This point bears repeating; if one

[II] Model comparison would still be fair if the log-likelihood estimate of each model were biased, but equally so. This would occur when the distribution of p_i across trials, and in particular the number of trials in which $p_i \ll \frac{1}{M}$, are the same for each model. Unfortunately, this is rarely true.

uses fixed sampling to estimate log-likelihoods and the number of samples is too low, one risks drawing conclusions about scientific hypotheses that are not supported by the experimental data one has collected.

4.4 Why inverse binomial sampling works

We start by showing that the inverse binomial sampling policy combined with the estimator $\hat{L}_{\text{ibs}}$ (equation 4.2) yields a uniformly unbiased estimate of $\log p$:

$$\begin{aligned}\mathbb{E}\left[\hat{L}_{\text{ibs}}\right] &= -\,\mathbb{E}\left[\sum_{k=1}^{K}\frac{1}{k}\right] = -\mathbb{E}\left[\sum_{k=1}^{\infty}\frac{1}{k}1_{k\leq K}\right] = \\ &= -\sum_{k=1}^{\infty}\frac{1}{k}\mathbb{E}\left[1_{k\leq K}\right] = -\sum_{k=1}^{\infty}\frac{1}{k}\mathbb{P}\left(k\leq K\right) = \\ &= -\sum_{k=1}^{\infty}\frac{1}{k}(1-p)^{k} = \log p.\end{aligned}$$

The first equality is the definition of $\hat{L}_{\text{ibs}}$, in the second equality we introduce the indicator function $1_{k\leq K}$ which is 1 when $k \leq K$ and 0 otherwise. The third equality follows by linearity of the expectation and the fourth directly from the definition of the indicator function. The fifth and second-to last equality is the formula for the cumulative distribution function of a geometric variable, and the final equality is the definition of the Taylor series of $\log p$ expanded around $p = 1$. Note that this series converges for all $p \in (0, 1]$.

In the derivation above, we can replace $\frac{1}{k}$ by an arbitrary set of coefficients a_k and show that

$$\mathbb{E}\left[\sum_{k=1}^{K}a_k\right] = \sum_{k=1}^{\infty}a_k(1-p)^{k}.$$

For all p for which the resulting Taylor series converges. This immediately proves two corollaries. First, we can use the inverse binomial sampling policy to estimate any analytic function of p. Second, since we can rewrite any estimator $\hat{L}(K)$ as $\sum_{k=1}^{K} a_k$, and since Taylor series are unique, $a_k = \frac{1}{k}$ is the only choice for which $\hat{L}(K)$ converges to $\log p$. In other words, $\hat{L}_{\text{ibs}}$ is the only uniformly unbiased estimator of $\log p$ with the inverse sampling policy, and as such, it is automatically the minimum-variance unbiased estimator.

4.5 Is inverse binomial sampling really better?

The law of conservation of misery[III] implies that the upside of zero bias in IBS should come at a cost. The two obvious candidates to absorb the misery are computational time (number of samples drawn) or variance. Surprisingly, below we show that IBS is sample-efficient and has low variance, so it violates the law and the lack of bias really comes for free.

4.5.1 Computational time

The number of samples that IBS takes on a trial with probability p_i is geometrically distributed with mean $\frac{1}{p_i}$. We saw earlier that for fixed-sampling estimators to be approximately unbiased, one needs at least $\frac{1}{p_i}$ samples, and IBS does exactly that. Moreover, since IBS adapts the number of samples it takes on different trials, it will be considerably more sample-efficient than fixed sampling with constant M across trials. For example, in the aforementioned example of the orientation discrimination task, when most trials have a likelihood $p_i \geq 50\%$, IBS will often take just 1 or 2 samples on those trials. Therefore,

[III] Communicated to me in personal communication with prof. Frank Redig, my bachelor thesis co-adviser and brilliant probability theorist.

it will allocate most of its samples and computational time on trials where p_i is low and those samples are needed.

4.5.2 Variance

We can derive the variance of IBS using a similar calculation as in equation 4.4:

$$\mathrm{Var}\left[\hat{L}_{\mathrm{IBS}}\right] = \sum_{k=1}^{\infty} \frac{1}{k^2}(1-p)^k = \mathrm{Li}_2(1-p)$$

where we introduce the dilogarithm or Spence's function $\mathrm{Li}_2(z)$ (Maximon, 2003). The variance (plotted in figure 4.1) increases when $p \to 0$, but it does not diverge. Instead, it converges to $\frac{\pi^2}{6}$. This may be surprising, since the variance of K (the outcome of IBS) does diverge, but luckily the function that maps K to $\hat{L}_{\mathrm{IBS}}$ has a decreasing derivative and the effects balance each other out. Therefore, IBS is not only uniformly unbiased, but its variance is uniformly bounded. We already saw that $\hat{L}_{\mathrm{fixed}}$ is the minimum-variance unbiased estimator of $\log p$ given the inverse binomial sampling policy, but it also comes close to saturating the *information inequality*, the analogue of a Cramer-Ráo bound for an arbitrary function $f(p)$ and a non-fixed sampling policy.

$$\mathrm{Std}(\hat{f}|p) \geq \sqrt{\frac{p(1-p)}{\mathbb{E}\left[K|p\right])}} \left|\frac{\mathrm{d}f(p)}{\mathrm{d}p}\right| \tag{4.4}$$

In our case, where $f(p) = \log p$, the information inequality reduces to $\mathrm{Std}(\hat{L}_{\mathrm{IBS}}) \geq \sqrt{1-p}$. In figure 4.4, we plot the standard deviation of IBS compared to this lower bound.

It may be disappointing that IBS does not match the information inequality. Kolmogorov showed that the only functions $f(p)$ for which the fixed sampling policy with M samples

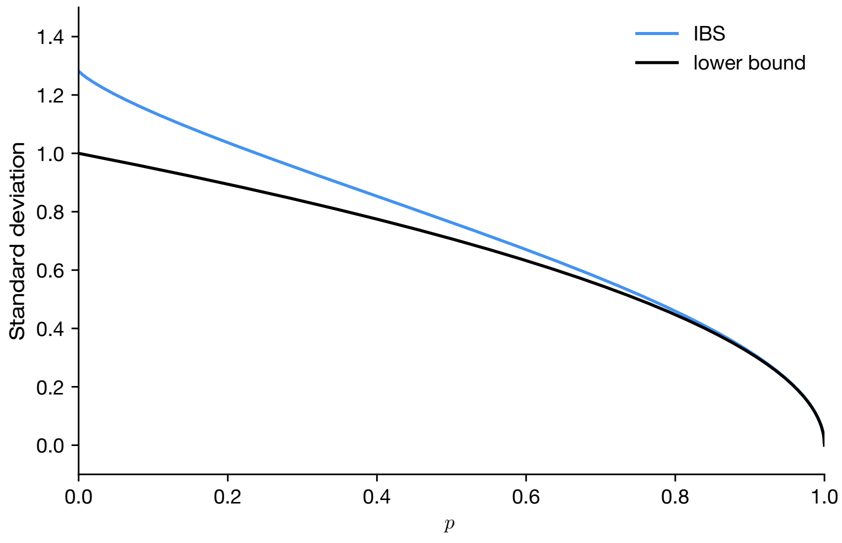


Figure 4.4 Standard deviation of IBS (Blue curve) and the lower bound given by the information inequality (black, see equation 4.4). The standard deviation of IBS is within 30% of the lower bound across the entire range of p.

allows an unbiased estimator are polynomials of degree up to M, and those estimators can saturate the information equality. Dawson (Dawson, 1953) and later de Groot (de Groot, 1959) showed that if an unbiased estimator of a non-polynomial function $f(p)$ exists and it matches the information inequality, it must use the inverse binomial sampling plan. Moreover, de Groot derived necessary and sufficient conditions for $f(p)$ to allow such estimators. Therefore, the standard deviation in IBS is close (within 30%) to its theoretical minimum.

To compare the variance of IBS and fixed sampling on equal terms, we use the scaling behavior of $\hat{L}_{\text{fixed}}$ as $M \to \infty$. Specifically, for fixed sampling, we plot $\sqrt{M} \times \text{std}(\hat{L}_{\text{fixed}})$ and for IBS we plot $\frac{1}{\sqrt{p}} \times \text{std}(\hat{L}_{\text{IBS}})$ (see figure 4.5). With this scaling, the curves for fixed sampling again collapse onto a master curve[IV]. Note that if we create additional estimators by

[IV] These curves converge pointwise on $(0,1]$ and uniformly on any interval $(\varepsilon, 1]$, but not uniformly on $(0,1]$. The limits $M \to \infty$ and $p \to 0$ are not exchangeable.

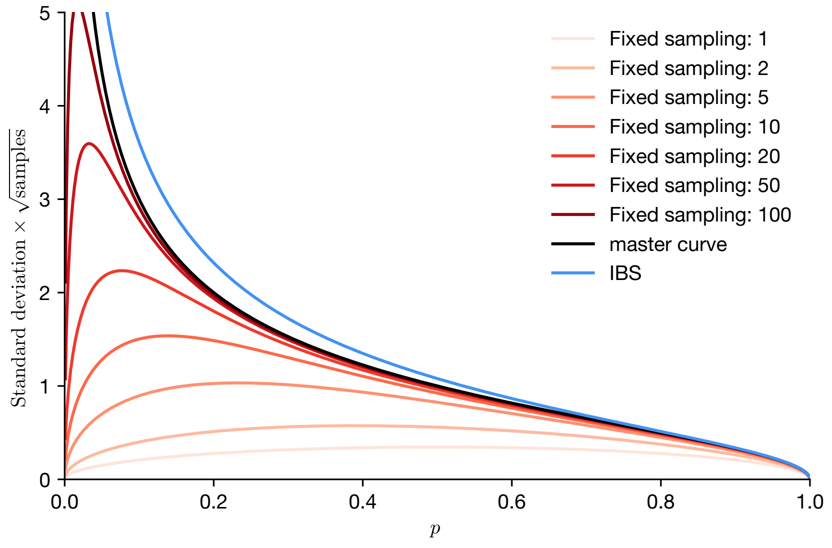


Figure 4.5 Standard deviation times square root of the expected number of samples drawn by IBS (blue) and fixed sampling (red), and the master curve (black) that fixed sampling converges to when $M \to \infty$.

running $\hat{L}_{\text{IBS}}$ multiple times and averaging the results, those would overlap with the curve for regular IBS.

All these curves increase and diverge as $p \to 0$, reflecting the fact that estimating log-likelihoods for small p is hard. The standard deviation of fixed sampling is always lower than that of IBS, especially when $p \to 0$ (specifically when $p \ll \frac{1}{M}$). In other words, fixed sampling produces low-variance estimators exactly in the range in which its estimates are biased, as guaranteed by the Cramer-Ráo bound. However, in the large-M limit, fixed sampling does saturate the information inequality, so its master curve lies slightly below IBS. In other words, if one is able to draw so many samples that bias is no issue, then fixed sampling provides a slightly better trade-off between variance and computational time. However, in section 4.7.2, we discuss an improvement to IBS which decreases its variance by a factor 2-20, in which case IBS is clearly superior.

4.5.3 Why high-order moments don't matter

So far, we have considered the mean (or bias) and variance of $\hat{L}_{\text{fixed}}$ and $\hat{L}_{\text{IBS}}$ in detail, but ignored any higher-order moments. This is justified since to estimate the log-likelihood of a model with a given parameter vector, we will sum these estimates across many trials. Therefore, the central limit theorem guarantees that the distribution of $L\left(\vec{\theta}\right)$ is Gaussian with mean and average determined by the mean and variance of $\hat{L}_{\text{fixed}}$ or $\hat{L}_{\text{IBS}}$ on each trial, at least as long as the distribution of p_i across trials satisfies a regularity condition. A sufficient but far from necessary condition is that there exists a lower bound on p_i, which is the case for behavioral model with a lapse rate. Using the same argument, the number of samples that IBS uses to estimate $L\left(\vec{\theta}\right)$ is also Gaussian.

4.5.4 Bias or variance?

In the previous sections, we have seen that IBS is always unbiased, whereas fixed sampling can be severely biased with too few samples. However, with the right choice of M, fixed sampling can have slightly lower variance. We now list 4 reasons why bias matters more than variance.

1. To use IBS or fixed sampling to estimate the log-likelihood of a given data set, we sum estimates of L_i across trials. The central limit theorem implies that, as $N \to \infty$, the standard deviation of $\hat{L}\left(\vec{\theta}\right)$ will grow proportional to $\sqrt{N}$, whereas the bias grows linearly with N.
2. Behavioral modeling studies typically use data from multiple participants, providing a second level of averaging that can increase signal relative to noise.

3. One primary reason to estimate log-likelihoods is to infer parameters of a behavioral model with maximum-likelihood estimation. For that purpose, one uses an optimization algorithm that calls the routine that estimates $\hat{L}\left(\vec{\theta}\right)$ many times with different candidate values of $\vec{\theta}$ and combines the results together to estimate the value that maximizes $L\left(\vec{\theta}\right)$. Clever optimization algorithms, for example those which build a Gaussian process approximation of the objective function (Acerbi and Ji, 2017; Snoek et al., 2012), can operate with noisy objectives by effectively averaging function values for nearby parameter vectors. By contrast, no optimization algorithm can handle bias.

4. On a conceptual level, bias is much more dangerous than variance. Bias can causes researchers to confidently draw false conclusions, variance causes decreased statistical power and lack of confidence. Recently, it has been emphasized that underpowered studies combined with questionable research practices or invalid statistical techniques such as p-hacking or HARKing can inflate effect sizes and create false positive results, but bias is impossible to recognize or correct no matter what statistical tools one uses.

4.6 Practical demonstrations that IBS outperforms fixed sampling

To demonstrate that IBS outperforms fixed sampling in realistic conditions, we simulate data from an orientation discrimination (section 4.6.1) and a change localization (section 4.6.2) task, and compare the accuracy of both sampling methods on parameter recovery through maximum-likelihood estimation. Since these methods provide noisy and possibly biased estimates of $L\left(\vec{\theta}\right)$, the estimates of $\vec{\theta}_{\text{MLE}}$ that result from optimizing the log-likelihood will also be noisy and biased. In both cases, we compare fixed sampling with 1, 2, 5, 10, 20, 50 or 100 samples, and we create versions of IBS by running the estimator multiple times and averaging the results. In the orientation discrimination experiment, we also compare the performance of IBS and fixed sampling to parameter estimation with the exact log-likelihood function, for which we can derive a closed-form expression.

We maximized the log-likelihood with Bayesian Adaptive Direct Search (BADS (Acerbi and Ji, 2017), `github.com/lacerbi/bads`), a hybrid Bayesian optimization algorithm based on the mesh-adaptive direct search framework (Audet and Dennis Jr, 2006), which affords a fast, robust exploration of the function landscape via Gaussian process (GP) surrogates. When estimation parameters using IBS or fixed sampling, we enabled "uncertainty handling" in BADS, informing it to incorporate measurement noise into its model of the objective function.

4.6.1 Orientation discrimination

The first task we simulate is an orientation discrimination task, in which a participant observes an oriented patch on a screen, and indicates whether they believe it was to the left or right of a reference line (see figure 4.6A). We simulated data from 600 trials, and on each trial we draw the stimulus s from a Gaussian with mean 0° (the reference) and standard deviation 3°. The behavioral model assumes that participants make a measurement x of the stimulus, which is normally distributed with mean s and standard deviation σ. They then respond "right" if x is larger than μ (a parameter which captures response bias, or an incorrect memory of the reference line) and "left" otherwise. However, a fraction of the time, given by a lapse rate α, the participant guesses randomly. We visually illustrate the model in figure 4.6B , and we can derive the likelihood analytically:

$$\mathbb{P}(\text{response right}|s, \mu, \sigma, \alpha) = \frac{\alpha}{2} + (1-\alpha)\Phi\left(\frac{s-\mu}{\sigma}\right) \tag{4.5}$$

Where $\Phi(x)$ is the cumulative normal distribution function. When optimizing the log-likelihood with BADS, the lower bound, plausible lower bound, plausible upper bound and upper bound for σ are $0°$, $2°$, $5°$ and $10°$, respectively. For μ, the bounds are $-2°$, $-1°$, $1°$ and $2°$. For α, they are 0.01, 0.01, 0.5 and 1. The upper and lower bounds are hard constraints, whereas the plausible bounds provide additional information to the algorithm to where the global optimum is likely to be, and are used by BADS, for example, to draw a set of initial points to start building the surrogate Gaussian process, and to set priors over the GP hyperparameters. In all these simulations, we extend IBS with a threshold of $\log 2 \approx 0.693$ (see section 4.7.1).

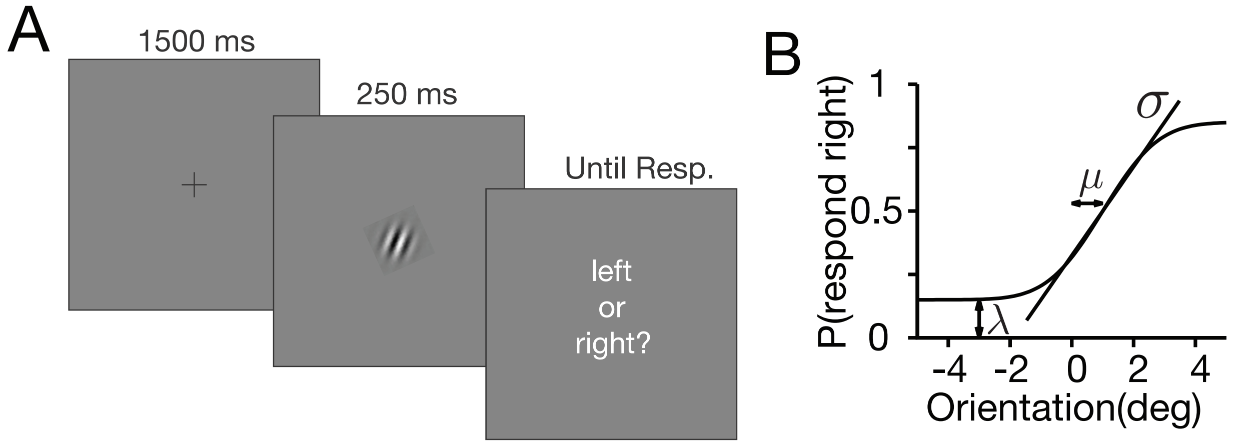


Figure 4.6 A. Trial structure of the orientation discrimination task. A oriented patch appears on a screen for 250 ms, after which participants decide whether it is to the left or right of a vertical reference. Note that in this chapter, we only use simulated data. **B.** Graphical illustration of the behavioral model, which specifies the probability of choosing left as a function of the true stimulus orientation. The three model parameters μ, σ and α correspond to the horizontal offset, slope and asymptote of the graph.

To estimate parameters for a given data set, we run BADS 8 times with different starting points, all combinations of $\sigma \in \{3°, 4°\}$, $\mu \in \{-0.333°, 0.333°\}$, $\alpha \in \{0.173, 0.337\}$, each of which returns a candidate for $\vec{\theta}_{\mathrm{MLE}}$. We then evaluate $\hat{L}(\vec{\theta})$ for each of these 8 candidates and select the candidate with highest log-likelihood. Note that the starting points always lie on one-thirds or two-thirds of the distance between the plausible upper and lower bound for each parameter. We generate 120 fake data sets. In the first 40, we fix μ to 0.1°, α to 0.1 and vary σ linearly over a range from 0° to 5°. In the second 40 data sets, we fix σ to 2°, α to 0.1 and vary μ from $-1°$ to 1°. Finally, the last 40 have σ equal to 2°, μ equal to 0.1° and α between 0.01 and 0.2 (again linearly spaced). We then repeat this process 100 times, resulting in 12000 data sets. Note that in this section we use Gaussian distributions for circularly distributed variables, which is justified since both the stimulus distribution and the measurement noise are small.

In figure 4.7, we show the parameter recovery using fixed sampling, IBS or the exact log-likelihood function from equation 4.5. First, we show that IBS can estimate σ and α more accurately using on average the same or fewer samples (figure 4.7A,B). As baseline, we also plot the mean and standard deviation of exact maximum-likelihood estimation, which is imperfect due to the finite data size (600 trials) and approximations made in the optimization algorithm. We omit results for estimates of μ, since even fixed sampling can match the performance of exact MLE with only 1 sample per trial.

Next, we fix $\sigma = 2, \mu = 0.1, \alpha = 0.1$ and plot the mean and standard deviation of the estimated $\hat{\sigma}$ and $\hat{\alpha}$ across 100 simulated data sets as a function of the number of samples per trial used by IBS or fixed sampling (figure 4.7C,D). Fixed sampling is highly sensitive to the number of samples, and with less than 20 samples per trial, its estimate of σ is severely biased. Estimating α remains impossible even with 100 samples per trial. By

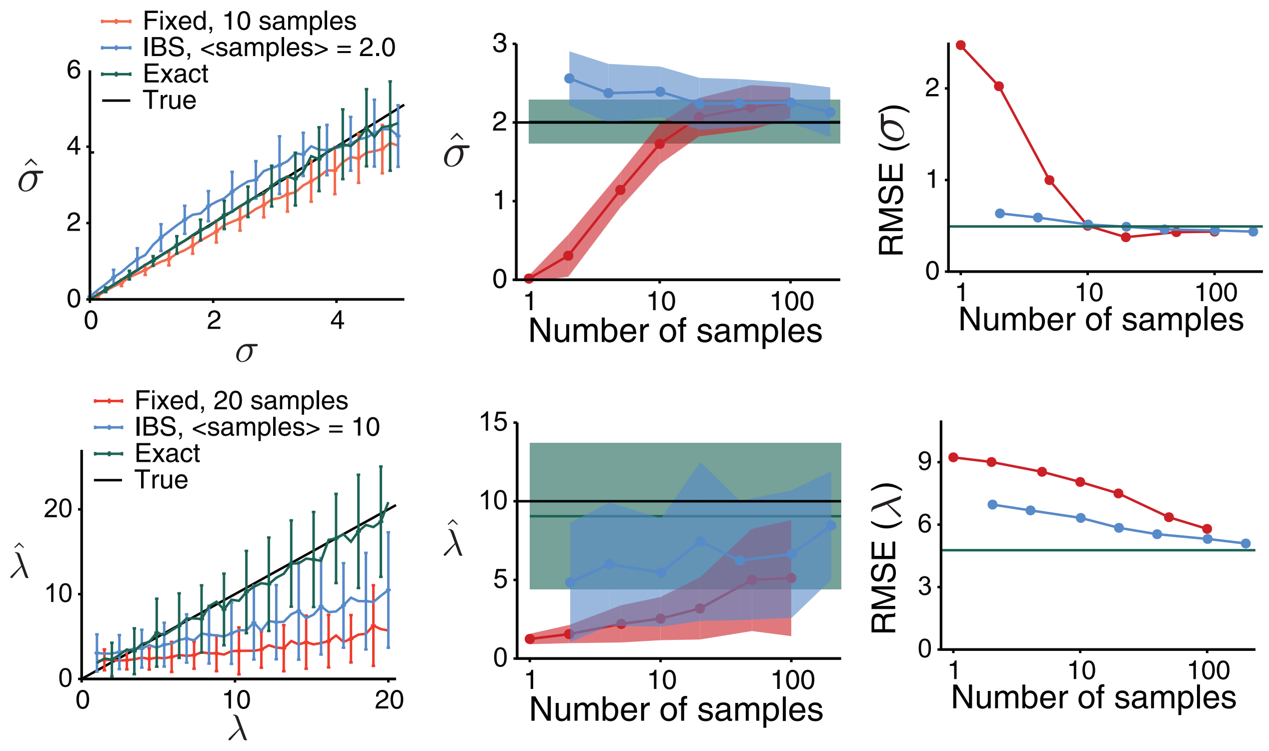


Figure 4.7 A. Estimated values of σ as a function of the true σ in simulated data using IBS (blue), fixed sampling with $M = 10$ (red) or the exact likelihood function (green). The black line denotes equality. Error bars indicate standard deviation across 100 simulated data sets. IBS uses on average 2.0 samples per trial. **B.** Mean and standard deviation (shaded regions) of estimates of σ for 100 simulated data sets with $\sigma_{\text{true}} = 2^{\circ}$, using fixed sampling, IBS or the exact likelihood function. For fixed sampling and IBS, we plot mean and standard deviation as a function of the expected number of samples used. **C.** Root mean squared error (RMSE) of estimates of σ, averaged across the range of σ_{true} in **A**, as a function of the number of samples used by IBS or fixed sampling. We also plot the RMSE of exact maximum-likelihood estimation, which is nonzero since we simulated data sets with only 600 trials. **D-F** Same, for α. These results demonstrate that IBS estimates parameters of the behavioral model for orientation discrimination more accurately than fixed sampling using equally many or even fewer samples.

contrast, IBS estimates σ reasonably accurately with 2.037 ± 0.013 (mean and standard error) samples per trial, and α with 20.179 ± 0.032 samples per trial. IBS has a tendency to overestimate σ while underestimating α, thereby incorrectly attributing behavioral variability to measurement noise rather than lapses. This bias results from an interaction of the uncertainty handling in BADS with our choice of model parametrization and parameter bounds. In general, estimating lapse rates is notoriously difficult.

Finally, we measure the root mean squared error (RMSE) of IBS, fixed sampling and the exact solution, averaged across all simulated data sets (figure 4.7E,F). This confirms the same pattern: fixed sampling makes severe errors in estimating σ with fewer than 10 samples, and it never succeeds in estimating α. IBS, even with 2 samples on average, estimates σ equally well as exact MLE, and it is better than fixed sampling at estimating α, when measured with the same average number of samples.

4.6.2 Change localization

We also conduct a parameter recovery study for simulated data from a change localization task (see figure 4.8A), in which participants observe a display of 6 oriented patches, and after a short inter-stimulus interval, a second display of 6 patches. Of these patches, 5 are identical between displays and one will have changed orientation. The participant responds by indicating which patch they believe changed orientation (for example, which a mouse click). We simulate data from 400 trials, the patches on the first display are all independently drawn from a uniform distribution. For the second display, we randomly select one of the patches and change its orientation by an amount drawn from a von Mises distribution centered at 0 with concentration parameter $\kappa_\mathrm{s} = 1$.

Our behavioral model assumes that participants independently measure the orientation of each patch in both displays. For each patch, the measurement distribution is a von Mises centered on the true orientation with concentration parameter κ. The participant then selects the patch for which the absolute circular difference of the measurements between the first and second display is largest. This model too includes a lapse rate α. By symmetry, we can write the likelihood of this model as

$$\mathbb{P}(\text{respond } i|j \text{ changed}) = \begin{cases} P_{\text{correct}}\left(\kappa, \alpha, \Delta_s^{(j)}\right) & \text{if } i = j \\ \frac{1}{5}\left(1 - P_{\text{correct}}\left(\kappa, \alpha, \Delta_s^{(j)}\right)\right) & \text{otherwise} \end{cases}$$

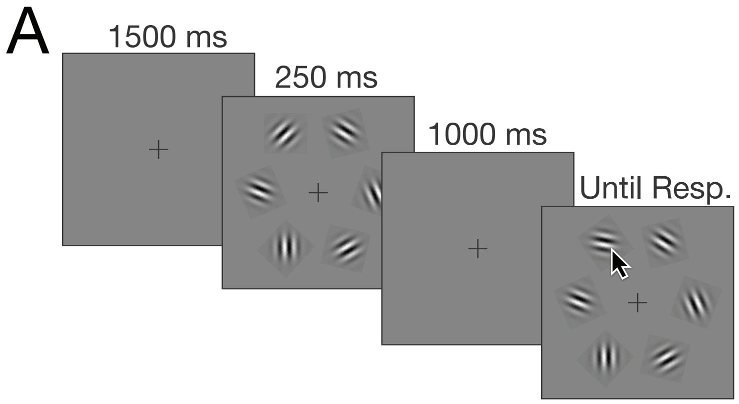
where $\Delta_s = \left|d_{\text{circ}}(s_j^{(1)}, s_j^{(2)})\right|$ is the absolute circular distance between the true orientations of patch j in the first and second display. We can derive an expression for $P_{\text{correct}}(\kappa, \alpha, \Delta_s^{(j)})$ by marginalizing over the circular distance between the respective measurements

$$P_{\text{correct}}(\kappa, \alpha, \Delta_s^{(j)}) = \frac{\alpha}{6} + (1-\alpha)\int_0^\pi \mathbb{P}\left(\Delta_x^{(j)}|\Delta_s^{(j)}\right)\mathbb{P}\left(\forall i \neq j : \Delta_x^{(i)} \leq \Delta_x^{(j)}|\Delta_s^{(i)} = 0\right)\mathrm{d}\Delta_x^{(j)}$$

where we have defined $\Delta_x^{(i)} = \left|d_{\text{circ}}(x_i^{(1)}, x_i^{(2)})\right|$ and we suppress the dependence on κ to simplify the notation. The first term in this equation is the probability density function (pdf) of the circular distance between two von Mises random variables whose centers are $\Delta_s^{(j)}$ apart. The second term simplifies, since $\Delta_x^{(i)}$ for all $i \neq j$ are all independent and identically distributed. Therefore, we can rewrite this equation as

$$P_{\text{correct}}(\kappa, \alpha, \Delta_s) = \frac{\alpha}{6} + (1-\alpha)\int_0^\pi \mathbb{P}\left(\Delta_x^{(j)}|\Delta_s^{(j)}\right)\mathbb{P}\left(\Delta \leq \Delta_x^{(j)}\right)^5 \mathrm{d}\Delta_x^{(j)} \tag{4.6}$$

where Δ is an auxiliary variable generated by taking the absolute circular difference between two von Mises random variables that are centered at 0 with concentration parameter κ. The second term of the integrand, therefore, is the fifth power of the cumulative distribution function (cdf) of Δ. Since the distribution of the circular distance between two von Mises random variables is itself von Mises, we can calculate the pdf analytically but the cdf is non-analytic. Moreover, the integral in equation 4.6 is analytically intractable as well. We can, however, evaluate it numerically and we plot $P_{\text{correct}}(\kappa, \alpha, \Delta_s^{(j)})$ as a function of $\Delta_s^{(j)}$ in figure 4.8B.



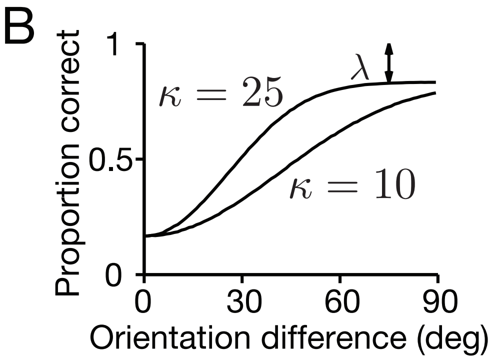


Figure 4.8 A. Trial structure of the change localization task. While the participant fixates on a cross, 6 oriented patches appear for 250 ms, disappear and the re-appear for another 250 ms. In the second display, one patch will have changed orientation, in this example the top left. The participant indicates with a mouse click which patch they believe changed. Note again that all data in this chapter in simulated. **B.** The behavioral model is fully characterized by the proportion correct as function of model parameters and circular distance between the orientations of the changed patch in its first and second presentation (see text). Here we plot this curve for two values of κ. In both curves, $\alpha = 0.2$. We can read off κ from the slope and α from the asymptote.

We use the same procedure and settings for BADS as for the orientation discrimination task. The lower bound, plausible lower bound, plausible upper bound and upper bound for κ are 1, 1, 100 and 100, and for α they are 0.01, 0.01, 0.5 and 1. We use a threshold of $\log 6 \approx 1.792$. We run BADS 4 times, with starting values of $\kappa \in \{34, 67\}$ and $\alpha \in \{0.173, 0.337\}$. We generate 40 parameter vectors with $\kappa = 11.11$ and α linearly spaced from 0.01 to 0.5 and 40 data sets with $\alpha = 0.03$ and κ between 1 and 100. Again, we create 100 data sets for each such parameter combination.

In figure 4.9, we compare the performance of IBS and fixed sampling. Since we have no closed-form analytic expression for the log-likelihood, we cannot compare to exact maximum-likelihood estimation. As before, IBS estimates both κ and α more accurately with fewer samples than fixed samples (figure 4.9A,B), fixed sampling is highly sensitive to the number of samples and can cause severe downwards biases in both κ and α (figure 4.9C,D), and overall the RMSE of IBS is lower than fixed sampling when compared on equal terms (figure 4.9E,F).

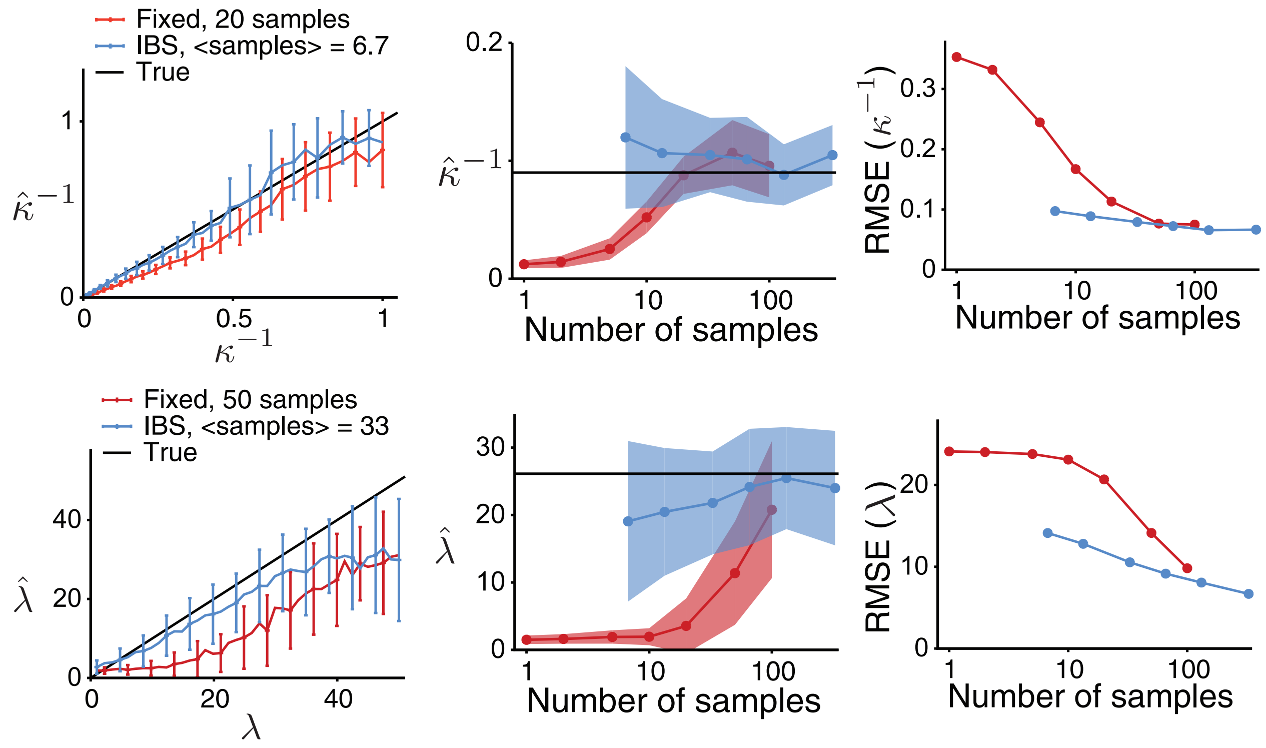


Figure 4.9 Same as figure 4.7, for the change localization experiment and estimates of κ and α. Note that we do not include exact maximum-likelihood estimation on any of these panels, since the likelihood is non-analytic.

These results demonstrate that in realistic scenarios, maximum-likelihood estimation with too few samples causes severe biases in parameter estimates, whereas inverse binomial sampling is much more accurate and robust to the number of samples used. With enough samples, fixed sampling can recover model parameters accurately, but the required number of samples per trial depends on the task, model and the parameter of interest. For high-dimensional models with a large response space such as the one we used in chapters 2&3, accurate parameter estimation will require many more samples per trial than are feasible given the computational time needed to generate them. Therefore, accurate and efficient parameter estimation is only possible with IBS.

4.7 Improvements and extensions

4.7.1 Early stopping threshold

One downside of inverse binomial sampling is that the computational time it uses to estimate the log-likelihood is approximately equal to the exponent of the negative log-likelihood. In other words, IBS spends exponentially more time on estimating log-likelihoods of poorly-fitting models or bad parameters. This implies that an optimization algorithm that uses IBS allocates more computational resources to estimating the objective function $L\left(\vec{\theta}\right)$ for parameter vectors $\vec{\theta}$ where the objective is low. However, the value of the objective at such poor parameter vectors are unlikely to affect its estimate of the location or value of the maximum, so the optimizer (BADS in our case) is wasting time. It may be possible to develop optimization algorithms that take into account the exponentially large cost of probing points where the objective function is low, but we can circumvent the problem by amending IBS with a criterion that stops sampling when it realizes that $\hat{L}\left(\vec{\theta}\right)$ will be low.

In section 4.2.2, we introduced the inverse binomial sampling scheme for a single trial, but it generalizes trivially to multiple trials. For each trial, draw samples from the behavioral model until one matches the human participant's response, then move to the next trial. This yields a value of K_i on each trial i, which IBS converts to an estimate $\hat{L}_i$, and then sums across trial. However, another way to implement multi-trial IBS is to draw one sample from the behavioral model for each trial, then set $K_i = 1$ for each trial where the sample matches the participant's response. For all other trials, draw a second sample from the behavioral model, and if that matches the response, set $K_i = 2$. Finally, repeat this process until no more trials remain. We illustrate this sampling scheme graphically in figure 4.10. After each iteration, we then compute

$$\hat{L}_K = -\frac{1}{N}\sum_{i\in\mathcal{I}_{\text{match}}}\sum_{k=1}^{K_i}\frac{1}{k} - \frac{N_{\text{remaining}}}{N}\sum_{k=1}^{K}\frac{1}{k}$$

where K is the iteration number, $\mathcal{I}_{\text{match}}$ is the set of trials where we found a matching sample and $N_{\text{remaining}}$ is the number of remaining trials. This value $\hat{L}_K$ is decreasing and by construction converges to $\frac{1}{N}\sum_{i=1}^{N}\hat{L}_{i,\text{IBS}}$ as $K \to \infty$. Therefore, whenever $\hat{L}_K$ exceeds a lower bound L_{lower}, we are guaranteed that $\hat{L}_{\text{IBS}}$ will exceed that bound too. When it does, we stop sampling and return L_{lower} as estimate of $L\left(\vec{\theta}\right)$. This does introduce bias into the estimate, but since we bound the *average* log-likelihood, the bias will be exponentially small in N as long as the true value $L\left(\vec{\theta}\right) - L_{\text{lower}}$ is nonzero.

In practice, we recommend using as lower bound the average log-likelihood of the data under a "chance" model, which assigns uniform probability to each possible response on each trial. In the orientation discrimination and change localization examples, the log-likelihood is $-\log 2$ and $-\log 6$, respectively, while for the 4-in-a-row game presented in chapters 2&3 the log-likelihood of chance depends on the number of pieces

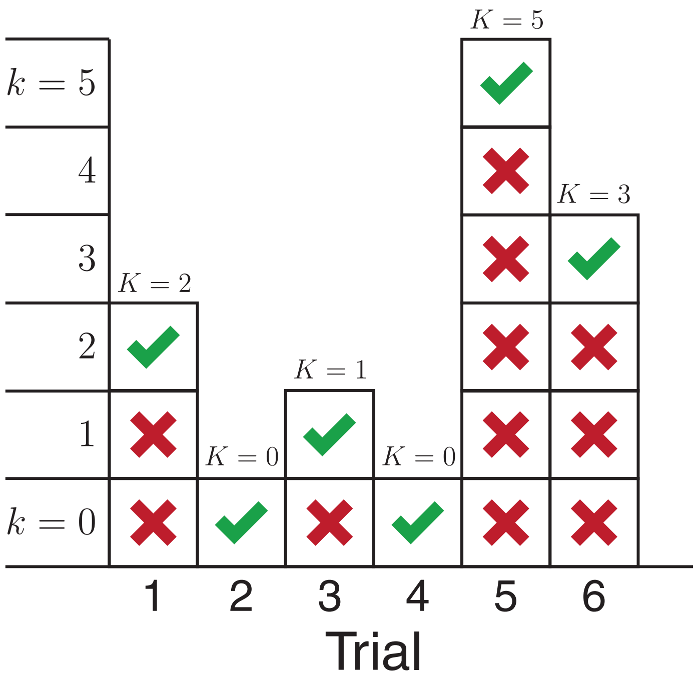


Figure 4.10 Graphical illustration of the two methods to implement IBS with multiple trials, in this case 6. In this figure, each column represents a trial, each box above the trial number represents a successive sample from the model from that trial, with red crosses for samples that do not match the participant's response and green checkmarks for ones that do. Above each column, we in indicate K, the number of "unsuccessful" samples before a successful one. For trials 2 and 4, $K = 0$ so $\hat{L}_{\text{IBS}} = 0$. The most obvious implementation of multi-trial IBS is "columns-first", to sample model responses for each trial until a success, and only then move to the next trial. However, a more convenient sampling method is "rows-first", and sample one response for each trial with $k = 0$, then one response for each trial with $k = 1$, excluding trials 2 and 4 since the first sample was successful, and continue increasing k until all trials have one successful response. This method allows for early stopping and a parallel processing.

on each board position. This new sampling scheme has an additional advantage: since each iteration, we independently sample from the behavior model on multiple trials, we can run these computations in parallel in a multi-threaded implementation. We provide a multi-threaded implementation of IBS in C++ using the `pthread` library on `github.com/basvanopheusden/ibs`.

4.7.2 Reducing variance by repeated sampling

Another method to improve IBS is to run the estimator multiple times and average the results. This will preserve the zero bias but reduce variance inversely proportional to the number of repeats R. We can improve on this by varying the number of repeat R_i between trials, and define

$$\hat{L}_{\text{IBS}}^{\text{repeat}} = \sum_{i=1}^{N} \frac{1}{R_i} \sum_{r=1}^{R_i} \hat{L}_i^{(r)}$$

where $\hat{L}_i^{(r)}$ denotes the outcome of the r-th run of IBS on trial i. This estimator is unbiased regardless of the choice of R_i (as long as $R_i > 0$ for all trials), and we can analytically compute both its variance and expected number of samples (see equation 4.7). This defines a constrained optimization problem, namely to minimize the variance of $\hat{L}_{\text{IBS}}^{\text{repeat}}$ given that its expected number of samples does not exceed a budget B:

$$\min_{R_i, R_2 \ldots, R_N} \left\{ \frac{1}{N} \sum_{i=1}^{N} \frac{\text{Li}_2(1-p_i)}{R_i} \middle| \sum_{i=1}^{N} \frac{R_i}{p_i} \leq B \right\} \tag{4.7}$$

We can solve this optimization problem using a Lagrange multiplier and find a closed-form expression for the optimal number of repeats per trial

$$R_i^* = B \left(\sum_{j=1}^{N} \frac{\text{Li}_2(1-p_j)}{p_j} \right)^{-1} \sqrt{p_i \text{Li}_2(1-p_i)} \tag{4.8}$$

By inspection, we observe that the optimal choice of repeats entails dividing the budget B across trials, where trial i is allocated repeats proportional to $\sqrt{p_i \mathrm{Li}_2(1-p_i)}$. We plot this function in figure 4.11 and see that, to minimize variance, we should allocate resources primarily to trials where p_i is close to $\frac{1}{2}$ and avoid trials where $p_i \approx 1$ (since the variance of IBS is already small for those trials) or where $p_i \approx 0$ (since those use up a larger share of the budget). We can also solve exactly for the reduction in variance when using the optimal choice of repeats R_i^* compared to a constant R which divides the budget equally across trials.

$$\frac{\mathrm{Var}\left[\hat{L}|R_i = R_i^*\right]}{\mathrm{Var}\left[\hat{L}|R_j = R\right]} = \left(\sum_{i=1}^{N} \sqrt{\frac{\mathrm{Li}_2(1-p_i)}{p_i}}\right)^2 \times \left(\sum_{i=1}^{N} \mathrm{Li}_2(1-p_i)\right)^{-1} \times \left(\sum_{i=1}^{N} \frac{1}{p_i}\right)^{-1}$$

This equation implies that the gain in precision from this method depends on the distribution of p_i across trials. If this distribution is uniform and $N = 500$, the median precision gain is 1.584 and the inter-quartile range (IQR) is 1.375 to 2.090. For a Haldane distribution (HALDANE, 1956), the median and IQR are 14.00 and 7.127-34.80, respectively. Note that the gain is always greater than 1, unless the likelihood is constant across trials.

The derivation above has some issues. First of all, we have treated R_i as continuous variables, and we need to convert R_i^* to integers. Since the method is only unbiased if R_i is nonzero for all trials, we recommend rounding R_i^* up to the nearest integer. This will reduce the gain in precision, but it is still better for both uniformly distributed p_i (median: 1.567, IQR: 1.374-2.002) or the Haldane distribution (median: 3.569, IQR: 3.042-3.988). More importantly, computing R_i^* requires knowledge of p_i on each trial, which we do not have. In practice, we recommend the following iterative scheme:

1. Choose a default parameter vector $\vec{\theta}_0$, and run IBS 100 times to estimate the (log)-

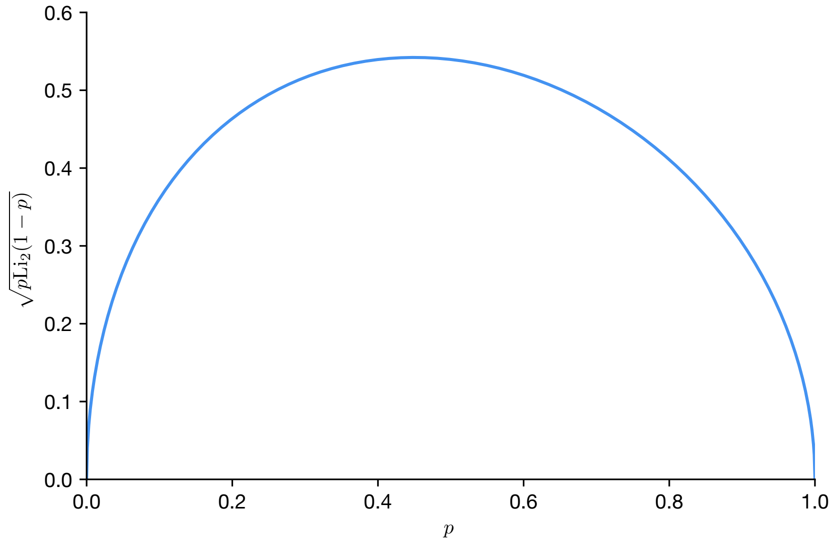


Figure 4.11 Graph of $\sqrt{p\mathrm{Li}_2(1-p)}$, which is proportional to the optimal number of repeats for a trial with likelihood p (see equation 4.8). We observe that the optimal allocation of computational resources entails repeated sampling for trials with $p \approx \frac{1}{2}$ and to avoid $p \approx 0$ or $p \approx 1$.

likelihood of the behavioral model on each trial.

2. Compute the optimal repeats R_i^* given the estimated likelihoods $\hat{p}_i$, and round up.

3. Run IBS with those repeats per trial on each iteration of the optimization algorithm

This implicitly assumes that the log-likelihood will be correlated across trials between the behavioral model with parameter vector $\vec{\theta}_0$ and any other vector $\vec{\theta}$ probed by the optimization algorithm. This usually the case, since low-probability trials are often those where the participant lapsed or otherwise made an error. In our experience, this scheme considerably reduces the variance of IBS for a given computational time budget.

4.7.3 Extension to continuous responses

So far, we have assumed that the space of possible responses is discrete. This assumption is necessary, since for continuous responses, the probability that a sample from the behavioral model matches the participant's response exactly is zero and IBS will never finish. To extend IBS to tasks with continuous response spaces, we can combine it with Approximate Bayesian Computation (ABC). Given a metric $D(\cdot,\cdot)$ to measure distances in response space, and a tolerance level ε, we can use IBS to estimate

$$L_\varepsilon\left(\vec{\theta}\right) = \prod_{i=1}^{N} \log \mathbb{P}\left(D(r, r_i) \leq \varepsilon | s_i, \vec{\theta}\right)$$

and use BADS or another optimizer to find the parameter vector θ that maximizes this objective. As $\varepsilon \to 0$, we recover exact maximum-likelihood estimation[V]. However, the expected number of samples used by IBS diverges in that limit, so in practice there is a lower bound for ε that is feasible and one needs to extrapolate to the $\varepsilon = 0$ limit. Note that this implementation of ABC is somewhat different from the more common rejection sampling based on a summary statistic.

4.8 Applications

We developed inverse binomial sampling for log-likelihood estimation of behavioral models, for the purpose of model comparison or fitting model parameters with maximum-likelihood estimation, but IBS has more general uses.

[V] This statement almost surely requires some technical regularity conditions, which we do not explore in this chapter.

4.8.1 Checking analytical or numerical log-likelihood calculations

We focused on behavioral models for which the log-likelihood is intractable to compute analytically or numerically. However, for behavior models where the log-likelihood has a closed-form solution, deriving that equation often involves a fair amount of algebra and mistakes in the calculation or implementation of the resulting equations are easily made. To check for such mistakes, one can compare the analytically derived log-likelihood against estimates of IBS and test for deviations. Note that one can perform these tests for any data set (real or simulated) and any model parameter vector. Another common scenario is that the log-likelihood cannot be derived exactly, but approximated through analytic techniques such as variational inference or numerical methods such as Riemann Integration for marginalization. In that case, one can estimate the quality of the approximation by comparing against IBS.

4.8.2 Estimating entropy and other information theoretic quantities

We can also use inverse binomial sampling to estimate the entropy of an arbitrary probability distribution $\mathbb{P}(x)$. To do, first draw a sample x from the distribution, then use IBS to estimate $\log \mathbb{P}(x)$. The first sample and the samples in IBS are independent, and therefore we can calculate the expected value of the outcome of IBS

$$\mathbb{E}\left[\hat{L}_{\text{IBS}}\right] = \mathbb{E}_{x\sim\mathbb{P}(\cdot)}\left[\log \mathbb{P}(x)\right] = \sum_{x\in\chi} \mathbb{P}(x)\log \mathbb{P}(x)$$

which is the definition of the negative entropy of $\mathbb{P}(x)$. We can use this trick to estimate the entropy of the predicted choice distribution of a behavioral model with a given parameter vector on any trial, for example to use in the Hick-Hyman law (see chapter 3). Moreover, we can generalize the trick to estimate the cross-entropy between two distributions (sample from one, estimate log-likelihood with the other), or the KL divergence between distributions.

We can also use this trick to estimate the marginal log-likelihood of a behavioral model given a prior of parameters $\mathbb{P}\left(\vec{\theta}\right)$. Marginalization is often difficult, but IBS does not require an analytic expression, only a method to draw samples. We can sample from

$$\mathbb{P}\left(r|s\right) = \sum_{\vec{\theta}} \mathbb{P}\left(\vec{\theta}\right) \mathbb{P}\left(r|s, \vec{\theta}\right)$$

by first sampling $\vec{\theta}$ from the prior, then r from the model conditioned on those parameters. Estimating marginal log-likelihoods therefore only requires that we perform this two-step sampling in each iteration of IBS. Studying the practical application of this method for estimation of the model evidence, and comparing it to other approximation techniques, is venue for future work.

Finally, we can use IBS to estimate the mutual information between the parameters of a behavioral model and the choices it makes on a given trial. To do so, we note that the mutual information is the average KL divergence between the model conditioned on a parameter vector and the model after marginalizing over parameters.

$$I\left(r, \vec{\theta}\right) = \mathbb{E}_{\vec{\theta}}\left[D_{\mathrm{KL}}\left(\mathbb{P}\left(r|s, \vec{\theta}\right); \mathbb{P}\left(r|s\right)\right)\right]$$

Therefore, estimating mutual information between model parameters and model responses is a straightforward combination of the tricks above. We provide example code for estimating entropy, cross-entropy, KL divergence, log-marginal-likelihood and mutual information on `github.com/basvanopheusden/ibs`. All these estimates are uniformly unbiased[VI].

4.9 Conclusion

In this chapter, we introduced inverse binomial sampling (IBS) as a tool for estimating the log-likelihood of models given an experimental data set. Estimates from IBS are uniformly unbiased and their variance is uniformly bounded. IBS is sample-efficient and, when combining with an early stopping criterion, combines naturally with gradient-free optimization algorithms that handle stochastic objective functions, like Bayesian Adaptive Direct Search (BADS).

We compared IBS to fixed sampling and showed that the bias inherent in fixed sampling can cause researchers to draw false conclusions in model comparison studies. Moreover, we showed that maximum-likelihood estimation of model parameters is more accurate with IBS than with fixed sampling with the same average number of samples.

This leads to the following recommendations for researchers who want to estimate the log-likelihood of a behavioral model. First, try to derive a closed-form analytic expression for the log-likelihood. If that is intractable, find an approximate analytic or numerical solution, for example using variational inference or Riemann integration. Before using an approximate solution, measure the quality of the approximation by comparing its log-likelihood estimates against IBS with well-chosen test trials and model parameters. Fi-

[VI] The lack of bias in entropy estimates by IBS may be surprising in light of a theorem from Paninski (Paninski, 2003) that uniformly unbiased estimators of the entropy given a finite set of samples cannot exist. This theorem does not apply to IBS since its sample size is a stochastic variable. It does, however, prove that one cannot estimate entropy (or similar information-theoretic quantities) with fixed sampling.

nally, if the model is too complex for analytical or numerical approximations, estimate the log-likelihood using inverse binomial sampling.

Chapter 5

Conclusion

5.1 Summary of results

In this thesis, I have introduced 4-in-a-row as a rich task to study computational principles underlying complex sequential decision-making. Along with the task, I have presented a behavioral model, which can capture individual players' strategies and characteristics as model parameters (chapter 2). By comparing the model to a range of alternatives, I conclude that human decision-making in this game relies on tree search, pruning, feature-based value function and attentional oversights.

To finish the assessment of the model as a choice prediction, I conduct a number of absolute goodness-of-fit tests, including a Turing test in which observers attempt to discriminate between human-vs-human game excerpts and games between computers. These tests show that the model can be improved, but that the mismatch between the model and human-vs-human data is small and along dimensions that are likely orthogonal to the space spanned by varying the search or planning algorithms. Hence, the faults of the model are unlikely to shed doubt on conclusions I obtain regarding tree search or planning in human sequential decision-making.

To support the model as a hypothesis for people's cognitive process, I show that it can generalize and predict people's choices in a variety of subtasks in the 4-in-row domain, as well as response times, eye movements and cursor movements (chapter 3). This requires some additional assumptions on which model components ought to predict people's response time or attention, but these assumptions are minimal. Finally, I demonstrate that the model can provide insight into the nature of human cognition and especially the difference between experts and novices, or individuals with different level of talent. I obtain a remarkably robust and strong conclusion: stronger players make fewer attentional oversights and perform more tree search.

The primary challenge in modeling human decision-making in 4-in-a-row was to develop accurate methods to estimate model parameters for individual participants. To do so, I use inverse binomial sampling (IBS, chapter 4), a method which has been studied mathematically (Dawson, 1953; de Groot, 1959) but which has to my knowledge never been used for behavioral modeling. I show that IBS is not only capable of estimating log-likelihoods without bias, but it also makes efficient use of computational time by using a low average number of samples, and it is precise since its log-likelihood estimates have low standard deviation.

I compare IBS to fixed sampling in simulated data from two tasks and demonstrate that IBS combined with a gradient-free optimization algorithm recovers model parameters much more accurately than fixed sampling. IBS is easy to implement, it works for every task and every model, as long as the response space is discrete, and it always outperforms fixed sampling. I recommend using IBS for any behavioral modeling study in which closed-form analytical likelihoods or numerical approximations are intractable. When likelihoods can be computed or approximated, I recommend using IBS to verify the quality of the approxmation or to localize mistakes in analytical derivations.

5.1.1 Future work

I'd also like to discuss three projects that I am currently working on which extend the work presented in this thesis, or address limitations of the current experiments and analyses.

First, I'd like to discuss two projects aimed at better understanding possible neural implementations of model-based planning algorithms, in particular the behavioral model presented in this thesis for the 4-in-a-row task. I am collaborating with prof. Nathaniel Daw on a version of the task in which we present participants with a pre-configured board positions and ask them to report what move they would play in the hypothetical scenario that they're playing a game against a computer and encounter that position. We present 180 such positions, and afterwards we *actualize* a randomly selected trial, thus ensuring that participants are properly incentivized to search for the best move in each position. Participants perform these choose-the-best-move trials in a functional MRI scanner, and we have collected data from 35 participants with this paradigm. Using this data, we can test several hypotheses generated by the work in this thesis.

In particular, the behavioral modeling suggests that people's decision-making algorithm relies on two main components: an evaluation function which people can query to obtain value judgments for given positions, and a search process, by which people simulate future outcomes of possible moves. Based on previous literature which has found prospective activity in rodent hippocampus (Ainge et al., 2007; Johnson and Redish, 2007; Pfeiffer and Foster, 2013), we hypothesize that for sequential decision-making in 4-in-a-row, the human hippocampus plays an important role. Furthermore, based on literature on economic choice (Levy and Glimcher, 2012), we hypothesize that the ventromedial prefrontal cortex (vmPFC) encodes values of board states, either those presented currently on a screen, or ones suggested by the search algorithm as possible future states which are likely to arise. To test these assumptions, we plan to run univariate searchlight regressions with single-trial regressors taken from the behavioral model, in particular the predicted size (breadth and depth) of the decision tree, as well as the state value.

More importantly, though, our model predicts that the evaluation function assigns value to states by counting the presence of relatively simple features, and multiplying the counts by feature weights. To test whether vmPFC indeed computes such a weighted sum of features, we intend to perform a representational similarity analysis (Kriegeskorte et al., 2008). Namely, if vmPFC indeed represents the features that determine state values, the trial-to-trial representational similarity matrix should have low rank, equal to the number of features. However, in brain areas that only represent value but not the features going in to the computation, the similarity matrix should have rank 1, whereas brain areas that represent board states in their entirety should have high (up to full) rank. Thus, a variability of the rank of the representational similarity matrix across brain areas should reveal insight into the computations performed in different parts of the brain.

In conjunction with the human fMRI study, I am also collaborating with prof. Daeyeol Lee to implement a version of the 4-in-a-row task for non-human primates. This project is in an early stage, we have been able to secure funding but have not started any monkey training or preparatory work. The goal of the project is to reveal insight in the neural computations underlying planning, specifically, to test the hypothesis that the primate dorsomedial prefrontal cortex (dmPFC) contributes to the evaluation of alternative strategies and learning algorithms and arbitration between them (Lee and Seo, 2016; Seo et al., 2014), while the dorsolateral prefrontal cortex (dlPFC) might be more involved in the computation and representation of integrated values of alternative choices (Lee et al., 2014). To this end, we plan to simultaneously record single-neuron and local-field potential (LFP) activity of neurons in dlPFC and dmPFC using two independent multi-electrode recording systems (Thomas Recording). For the dlPFC, we will use a recently developed 16-channel AMEP device, while activity from the dmPFC will be recorded using a traditional 16-channel (or 5-channel) "Eckhorn" multi-electrode recording system with which the Lee lab has extensive (>17 years) experience. We hope that this project will help us better understand the complementary role of primate dlPFC and dmPFC in sequential decision-making and planning.

Finally, I'd like to discuss a behavioral experiment we intend to conduct with a completely different task than 4-in-a-row, to address a limitation of our task. One downside of 4-in-a-row is that, even though it is a two-player game, it is fully deterministic without hidden information and as such, there exists a pure strategy which dominates all others, namely the minimax policy. Additionally, the algorithm in the behavioral model converges to this strategy in the limit of no noise and infinite iterations, and no pruning. Therefore, participants' best chance to increase their winning chances is to try to reduce their noise or attentional oversights, and perform as much search as possible in limited time, while generally ignoring their opponent's strategy and adapting their own strategy to it. Another way to phrase this limitation is that the optimal strategy of the first player given knowledge of the second player's strategy does not differ much from the optimal strategy without suck knowledge. However, such *opponent modeling* is clearly a feature of many two-player games as well as real-life decision-making.

Therefore, we intend to study human decision-making and planning in a two-player game called *Perudo*, or *Liar's dice*, in which two players roll a set of dice, but keep the outcome of those rolls private. The first player then makes a statement about the collection of all dice, of the form *"there are at least n dice with outcome m"*, after which the second player can call a bluff. If they do, all dice get revealed and if the first player's statement is correct, they win, otherwise the second player wins. If the second player decides not to call a bluff (for example because they know the statement to be true given their own die rolls), they need to reply with a statement of their own, in which n has to be higher - hence making the statement a priori less likely. This process alternates until either player calls a bluff. Perudo is a two-player game which is deterministic given the die rolls, but players have hidden information, which means that a crucial component of the game is to infer from a player's statements what outcomes they are likely to have rolled. This game is therefore ideal to study how people form models of their opponent and how they adapt strategies based on those models.

5.2 Complex decision-making: tasks, models and the ultimate victory

I would like to finish this thesis with a collection of my thoughts, some of which are a direct result of the experiments and analyses I presented in the previous chapters, some of which result from reading literature related to my thesis topic or cognitive neuroscience in general, and some for which I do not know the origin.

5.2.1 What are good experimental tasks?

The first and most important choice in a behavioral modeling study is the experimental task to use to investigate a cognitive phenomenon of interest. Tasks vary on many dimensions, but a particularly influential design choice is the task complexity. As mentioned in the introduction, there are many dimensions of complexity on which experimental tasks differ. In this thesis, we explored a task with a high state space complexity, but one could consider tasks with perceptual uncertainty, stochasticity, multi-agent interaction, a hierarchy of time scales and many other attributes.

The default method for choosing task complexity is to run the *least complex task that probes the cognitive phenomenon of interest.* The rationale behind this principle seems obvious: the more complex the task, the harder it is to understand. Examples of such minimally-complex tasks are orientation discrimination to study acuity, random-dot motion to study evidence integration, or the two-step task to study model-based planning.

Instead, I propose a framework in which the optimal choice of experimental task is dictated by a trade-off between a number of desirable properties which translate into *design pressures* in the space of tasks[I]. Some pressures push towards simpler tasks, some towards more complex ones, and the optimal trade-off lies somewhere in the middle. The relative importance or weights one should associate to each of these pressures depends on one's research goal, model animals and ultimately also personal preference. However, in my view, the neuroscience literature has generally overestimated the importance of pressures away from complexity and therefore used much too simple tasks. I believe that, at least for cognitive neuroscientists with a view towards understanding naturalistic decision-making, the

[I] This framework is partially inspired by (Alon, 2009; Juavinett et al., 2018)

4-in-a-row task in much closer to optimal.

5.2.1.1 Tasks that generalize to natural environments

The most important requirement of any experimental task is that it provides insight into the computations underlying a cognitive process in real-world decision-making, and not just in the task itself. This requires that the strategies that people or animals use to solve the task are *scalable* and that predictions made by behavioral models can *generalize* to different versions of the task and ultimately to real-world problems. For example, I cannot imagine how to convert a bounded accumulation algorithm for evidence integration in the random-dot motion task to an algorithm that can play chess or go, let alone more complex real-world problems.

By contrast, the algorithm presented in this thesis readily generalizes to the entire class of combinatorial games including chess and go. That is, the algorithm can be used to play such games given a suitable set of features and weights. It is an open question to what extent the conclusions from this thesis, specifically about the nature of talent and expertise, generalize to other board games. Moreover, to generalize beyond the class of combinatorial games, we would need to include more realistic models of feature learning, uncertainty, attention and working memory. In other words, by this metric alone, one could argue that the 4-in-a-row task is too simple.

Regardless of the choice of task, this design pressure encourages researchers to explore an ensemble of tasks and not linger too long on any individual one. There is a real danger that, if a particular task and model become a “workhorse” that is treated as synonymous with the cognitive phenomenon it is meant to study (as I argue has happened for random-dot motion, bounded accumulation and evidence integration), researchers will increasingly study idiosyncracies of that task which are unlikely to generalize to the real world.

5.2.1.2 Tasks for which the correct model can be found

To perform a behavioral modeling study, one needs to conduct a behavioral experiment and fit the resulting data with a behavior model. A large component of behavioral modeling consists of searching for better fitting models, and ideally finding the "correct" model; one for which the log-likelihood matches the upper bound set by the entropy of the data. In simple tasks with low-dimensional stimulus and response spaces, the space of possible models (interpreted purely as input-output mappings) is smaller and the chance of finding the optimum increases. Therefore, the desire to model human behavior perfectly pushes towards simpler tasks.

However, we found that even in a task as complex as 4-in-a-row, we could specify a model that passed a number of absolute goodness-of-fit tests (see chapter 2, section 2.16). More importantly, we observed that the scientific conclusion on the nature of talent and expertise was robust to model specifications, including those that are provably incorrect. I suspect this scenario is common when modeling complex tasks: even if the model is incorrect, it may reveal insight into human or animal cognition as long as it captures enough of the variance in behavior.

5.2.1.3 Tasks for which behavioral modeling is tractable and accurate

The validity of an experimental task is intricately linked to the algorithms needed to model human or animal behavior. Human behavior in simple tasks such as the orientation discrimination task used in chapter 4 can be captured with simple models. More complex tasks require more complex models, meaning more model components and therefore more parameters. This increased number of parameters causes statistical problems with parameter and model estimation. Moreover, complex model are less likely to admit a closed-form analytic expression for the likelihood function, and simulating responses from these models often requires more computational time, which is the limiting factor for log-likelihood estimation with inverse binomial sampling. It may therefore seem that task complexity has a strong negative impact on the tractability of behavioral modeling.

One central proposition in this thesis, and the motivation behind its title, is that *behavioral modeling with complex tasks and models is much more tractable than has traditionally been assumed.*

First, consider the statistical properties of parameter and model estimation. Although complex tasks ought to require models with more parameters, we were able to model human behavior in 4-in-a-row with a 10-parameter model, which is still reasonably small. I suspect that in practice, the number of parameters may be less directly related to task complexity, especially if one allows for models that are "good enough". Additionally, the main factor affecting the accuracy of parameter estimation is not the parameter number, but parameter tradeoffs. In simple tasks with low-dimensional stimuli and responses, it is increasingly likely that different parameter choices produce similar input-output mappings, and especially distinguishing sources of noise in tasks such as the orientation discrimination or change localization experiments from chapter 4 is notoriously difficult. On the other hand, in complex tasks with more high-dimensional data, it is more likely for model parameters to be distinguishable.

A similar argument applies to model estimation. More complex tasks generally have higher-dimensional stimuli and responses, and the data will have higher entropy per trial and thus be more informative. Moreover, it is less likely for different algorithms to produce identical input-output mappings, simply because that space is larger. This point bears repeating: *more complex experimental tasks are more likely to provide rich data that can distinguish between different models that instantiate competing scientific hypotheses.*

Therefore, behavioral modeling is more likely to provide statistically accurate results with more complex tasks, in theory. In practice, however, we run into a different constraint: computational time. Imagine conducting a human behavioral experiment with 50 participants and 1000 trials per participant. For parameter optimization, we use BADS with 2000 iterations and for log-likelihood estimation, we use IBS with 100 samples per trial. Finally, we compare about a dozen models by their 10-fold cross-validated log-likelihood. In that case, the total computational cost is approximately 10^{12} times the cost of sampling a model response on a single trial. Given the current cost of computing time[II], this is affordable as long as the number of operations required to sample a model response on a single trial is less than 10^6.

However, the back-of-the-envelope calculation assumed that we compare models with 10-fold cross-validated maximum-likelihood estimation. This is far from the gold standard, and more sophisticated modeling studies may use multi-start methods to increase the accuracy of parameter optimization, MCMC sampling to obtain posterior distributions over parameters, hierarchical models to model the distribution of model parameters across individuals, or parametric bootstrap (Wagenmakers et al., 2004) methods to prevent biases in model estimation. Additionally, some accuracy benchmarks such as parameter and model

[II] For $13 per hour, Amazon Cloud Services provides 128 CPUs that each perform 10^{11} floating point operations per second.

recovery entail parameter fitting on many more fake data sets than participants in the experiment. Therefore, with more complex models that are more computationally expensive to evaluate, one is forced to use less sophisticated statistical tools and the accuracy of parameter and model estimation may suffer.

These considerations imply that tractability and accuracy of model comparison penalizes both too simple and too complex tasks, and that *in practice, parameters and models are most distinguishable when the experimental task is moderately complex.* I regard the results presented in chapters 2&3 as a case study, which demonstrates that the 4-in-a-row task and the heuristic search model are close to the optimum. With the methodological choices we made, we were able to fit 25 models to 330 participants within reasonable computational time[III], and even though we could not use more sophisticated statistical methods, we encountered surprisingly little difficulties in model and parameter estimation. On a personal note, I have always expected that at some point in this research project I would run into severe methodological issues and regret choosing a complex task, but this has never happened.

5.2.1.4 Tasks that are easy to learn

For any experimental task to be useful, subjects have to understand the task and learn strategies within the time allotted for data collection. For this section, it is important to distinguish whether the subjects are human participants or animals. For monkey or especially rodent neuroscience, learnability creates a strong pressure towards simplicity, as animals may simply fail to understand too complex tasks. However, I have been encouraged

[III] The definition of reasonable here is quite relaxed, but both the number of models and participants are relatively high.

by exciting recent studies made possible by gradual advances in animal training methods. For example, Miller et al. (Miller et al., 2016) as well as Akam et al. (Akam et al., 2015) managed to train rodents on a two-step task, Seo & Lee (Seo and Lee, 2008) trained monkeys on a competitive games such as matching pennies or rock-paper-sciccors, and Braden & Purcell (Purcell and Kiani, 2016) trained monkeys on a hierarchical evidence accumulation task.

For human neuroscience, where researchers can provide participants with explicit task instructions and where participants possess rich inductive biases over task structures and strategies, the relation between learnability and complexity is less clear. Some tasks (Cohen and Schneidman, 2013) with elementary instructions can prove exceedingly hard to learn, whereas other tasks people can perform with no instructions at all (a particularly compelling example is the Frostbite game proposed as benchmark for AI algorithms by Lake et al. (Lake et al., 2017) and studied experimentally by Dubey et al. (Dubey et al., 2018)). Instead, people's ability to quickly learn an experimental task seems to me much more clearly related to its naturalness (see section 5.2.1.1). Additionally, I propose that a strong heuristic for designing naturalistic and easy-to-learn tasks is to *choose experimental tasks that participants enjoy.* In my own experience, I have noticed that participants often report cognitive experiments and especially psychophysical tasks to be excruciatingly boring, and I find this concerning.

In summary, I have proposed a framework of different design pressures that together determine the optimal task complexity level. I have described 4 such pressure in detail: naturalness pushes towards complexity, tractability and statistical accuracy which push towards moderate complexity, probability to find "correct" models which pushes towards simplicity, and learnability which pushes towards simplicity in animal experiments and in human behavioral studies is mostly orthogonal to the complexity axis. The weights of these pressures and therefore the specific optimum are subjective, but the standard method of minimizing task complexity given a model distinguishability constraint implies a close-to-infinite weight on pressures towards simplicity, which at least in human behavioral modeling is unjustifiable. The results in this thesis constitute a case study which argues in favor of an optimum complexity level close to that of our 4-in-a-row task. Finally, the main pressures towards simplicity (difficulty in animal training and limited computational time budgets) are likely to grow less severe over time, and therefore the field should increasingly embrace complex experimental tasks.

5.2.2 What are good models?

Given an experimental task, one also has to use the resulting data to inform a choice of model. The primary metric guiding this model choice is its ability to fit the data, quantified by model evidence approximations such as the cross-validated log-likelihood. This is sensible, since model evidence is by definition the model comparison metric that maximizes the chance to infer the correct model when simulating data from a set of candidate model. However, in this section, I will argue for additional metrics to score models on, which may outweigh goodness-of-fit on a given data set.

First and foremost, we have to keep in mind that experimental tasks are useful only insofar as they provide insight into general computational principles underlying real-world decision-making problems. Therefore, a good model needs to be able to make generalized predictions, at least within a domain of tasks. A solid metric to judge *generalization* is log-likelihood, cross-validated across tasks. Additionally, good models capture not just the choices that people or animals make, but also the process by which they arrive at their decisions. It should therefore be possible to fit models on behavioral data, and use it to *predict process data* on single trials. Process data includes response times and eye movements (as in chapter 3), but potentially also galvanic skin response, pupil dilation or neural data. An appropriate metric to test whether a model can capture participants' decision-making process is across-data-type predictive accuracy.

As a corollary, if a model is to be taken seriously as a hypothesis for a strategy by which people or animals make decisions a variety of tasks with varying degrees of complexity, its algorithms need to *efficiently solve the task*. This requirement contains multiple questions. Does the algorithm make efficient use of computational resources? Can the algorithm be flexibly extended or can its parameters be tuned to solve increasingly difficult or complex tasks? Is so, are those extensions also computationally efficient? Does the model allow for an obvious way by which a participant learn the parameters of the algorithm from experience? If so, will that learning algorithm converge as quickly to human-level performance as people do?[IV] Moreover, what trajectory of strategies does the learning algorithm explore on its path to convergence? Note that many of these questions will require additional experiments to investigate and therefore, the quality of a model is not a function of just the experimental data it was initially designed to explain.

Another metric on which I propose to judge models is their number of parameters, or *model dimensionality*. Model evidence approximations such as AIC and BIC commonly contain terms that penalize high-dimensional models, but I will argue that low-dimensional model have advantages beyond such overfitting corrections. I like to think of behavioral modeling as *dimensionality reduction of behavior*. In that picture, models are useful as they allow one to compress all knowledge gained about an individual participant

[IV] This question is elaborated on by Lake et al (Lake et al., 2017) and commentaries by the estimated parameter vector. Hence, the fewer parameters one needs, the more efficient the compression.

Additionally, not all parameters are created equal. In the dimensionality reduction picture, it is natural to rank parameters by their influence on the model's prediction on a participant's choices, which one can quantify by measuring the KL divergence between the model and a model in which that parameter is lesioned. For a linear-Gaussian model, this KL divergence reduces naturally to the trial-to-trial variance explained by that parameter. Similarly, one can measure how much of inter-individual variability a parameter captures, or whether effects of experimental manipulations can be captured by changes in model parameters. In the dimensionality reduction analogy, these ideas correspond roughly to principal components analysis (PCA) or factor analysis.

5.2.2.1 Normative models

One special class are *optimal*, *normative* or *boundedly rational* models, in which the researcher explicitly solves for an algorithm that maximizes task performance given a computational resource constraint. These models can be parametrized by varying the strength of the constraint, or by introducing multiple constraints with weights that may differ across individuals. By construction, each boundedly rational model is efficient for its given cost function, and one can create a class of algorithms by solving the bounded optimality problem for different problems. Additionally, these models are often low-dimensional, since their algorithms are uniquely determined by a resource constraint. It is also likely that experimental manipulations selectively affect individual components of the cost function. For example, cognitive load might affect cost of computation or working memory while stimulus presentation time or visual blurring will affect perceptual precision. Finally, boundedly rational models are exceptionally appealing since all biological processes ultimately have to maximize a cost function, namely the organism's evolutionary fitness.

Unfortunately, although normative models possess many attractive properties, they often struggle to fit behavioral data. In chapter 2, we found that the `Opt-rand` model was among the worst-fitting models for our 4-in-a-row data set. This is unsurprising since we optimized this model for performance without constraints, we only added noise. It is possible a boundedly rational model that maximizes task performance given a limit on depth or amount of search might fit the data better. However, in other research from our lab, we often find that Bayesian models (which provide the optimal performance given perceptual acuity limits) struggle to fit psychophysics data better than models based on simple heuristics.

5.2.2.2 Neural networks

Finally, I would like to discuss one more class of models, namely artificial neural networks (ANNs). A typical ANN consist of an *architecture* which describes how different *units* are connected, and a *learning algorithm* to learn the appropriate *connection weights*, which are the model's parameters. The learning algorithm can optimize for task performance (as in deep Q-learning (Mnih et al., 2013)), goodness-of-fit on human data (in player modeling studies) or some combination thereof (as in AlphaGo). Fundamentally, what these networks have in common is the high number of parameters, and unlimited expressivity. In other words, given enough training, ANNs can fit behavioral data equally well or better than any other model. However, by other metrics they do not fare well. Neural networks are notoriously difficult to generalize across tasks, they are high-dimensional by construction, and the relation between parameters (connection weights) and the training data, task conditions, individuals is often obscure. In my view, these concerns have caused the neuroscience community to largely resist ANNs as models for human cognition or neural processing.

This suggests three possible improvements. First, instead of estimating parameters of the network for each participant, we can improve by estimating hyperparameters of the learning algorithm that determines the parameters. For example, if we find that chess games played by Magnus Carlsen or Fabiano Caruana[V] are well fitted by AlphaZero (Silver et al., 2017), but with different number of rollouts, exploration and evaluation networks (for example, one taken after $200,000$ training steps in AlphaZero's learning algorithm, the other after $300,000$), I would argue that we have improved AlphaZero as a model of chess experts. Another approach is to parametrize the objective that the learning algorithm optimizes and for example find that Magnus and Fabiano possess different subjective values of wins versus draws or losses[VI]. Finally, we may stick to learning connection weights for individual players in individual conditions, but incorporate a term in the objective that encourages individual or across-condition differences to lie in low-dimensional manifolds in parameter space. I suspect that this alone can cause progress towards more *interpretable AI*.

5.2.3 Where does the brain come in?

It has not escaped my attention that I am hoping to obtain a PhD degree in Neural Science whereas this dissertation does not contain any data collected from neurons, circuits,

[V] The current world champion and the challenger in the upcoming world championship match

[VI] Evaluating draws as a negative outcome is known as "contempt" in the computer chess literature.

systems or brains. This primarily reflects a changing personal view of the relation between the study of cognitive and neural science, or the studies of the mind versus the brain.

One may expect that knowledge of the hardware of the nervous system (neurons, axons, circuits) strongly constrains the computations or cognitive processes that these systems can support. On the other hand, one could view the brain as a universal function approximator or Turing machine and consider the space of algorithms that can be implemented in neural hardware unbounded in a formal sense. Over the course of my development as a cognitive neuroscientist, my view has converged somewhere closer to the latter than the former. I have been astounded by the intricacy of biological machinery and the richness of all possible EPSP-to-action-potential mappings that a single neuron possess. My favorite example is post-inhibitory rebound (Sherrington, 1913). Once you combine just a handful of neurons together and add neuromodulation, the resulting circuits can produce surprisingly many different temporal patterns even in the absence of any driving signal. This has been beautifully demonstrated by Eve Marder's work on the lobster stomatogastric nervous system (Marder and Bucher, 2007).

Moreover, some fundamental facts about the structure of our nervous system and nervous systems across the animal kingdom have surprisingly little influence on the models or algorithms proposed in cognitive science. For example, one such fundamental fact is that the cortex is *modular*: organized into distinct brain areas, which differ in their anatomy and subserve different computational functions. However, this modularity seems to me a much less fundamental component of common cognitive models, and in general, I cannot think of a way to determine whether a proposed model for the cognitive processes underlying a particular behavior is compatible with the modularity observed in the brain[VII]. I should

[VII] I'd like to credit Chris Summerfield for inspiring the ideas in this paragraph.

note another fundamental brain fact that constrains cognitive models much more severely, namely that of *latency*, the time required for signals to propagate through axons, for example from the retina through the optic nerve and optic radiations all the way to primary visual cortex. I am reminded of a beautiful set of experiments by Emilio Salinas studying the minimum amount of time that a monkey needs to make a perceptual decision (Stanford et al., 2010).

For the reasons above, I have come to believe that, although there must exist some bounds on the space of algorithms or cognitive processes that our brains can plausibly implement, that space encompasses at least all algorithms that cognitive scientists propose *in practice*. More explicitly, I believe that there is no reason to suggest that the heuristic search models presented in chapter 2 are incompatible with our knowledge of neurobiology.

Given these considerations, I believe the interplay between neuroscience and cognitive behavioral modeling is twofold. First of all, although I don't believe one can discard any cognitive models or theories based as biological implausible just based on general principles, one can record neural data either through animal electrophysiology or human neuroimaging. That data can then serve a similar function as eye movement or response time data presented in chapter 3, to lend credence to a specific cognitive model by correlating its components with neural measurements. Secondly, one can study the brain for its own sake, and use behavioral modeling to determine a set of signals or response characteristics to search for in neural data from various brain areas.

5.2.4 What does victory look like?

The enterprise of modeling human cognition is a gradual process of accumulating understanding by new experiments, theories and models. The ultimate goal is a universal behavioral model: an algorithm that matches all computational processes in the brain, at least up to the level of detail that experimental methods allow to obtain data for, in every conceivable decision-making task. However, this is far too ambitious and unhelpful as a standard of success. Instead, I propose a number of ingredients that are necessary (but not sufficient) before anyone can claim to have understood human cognition and sequential decision-amknig in particular.

First of all, we need to construct a behavioral model that predicts individual people's choices in a variety of complex tasks or domains. The larger the task set, the better, but for understanding model-based planning and sequential reasoning, I would expect at least 4-in-a-row, chess and go to be included, and preferably Atari games and other video games like Starcraft. Crucially, the model has to generalize, both across subtasks within a domain as well as across domains. That is, one should be able to estimate model parameters for an individual player in one game and predict their choice in another.

Additionally the model has to generalize to different data types, including response times, eye movements, pupil dilation, GSR and neural measurement. The model ought to have relatively few parameters, or come with a ranking of parameters in terms of the behavioral variance that they explain. To further increase the model as a plausible mechanism for cognitive processes, it would help to identify experimental manipulations that cause predictable and consistent changes in an individual's parameters, such as to increase or decrease a single parameter while keeping all others constant.

Moreover, the model needs to capture players at different stages of learning a task. Of particular interest are the very first choices that players make when playing a new game, to capture the inductive biases or prior beliefs over strong strategies or task-relevant features that players bring to the game. The model then needs to predict how an individual player's decision-making processes change over time, and how that change depends on the specific episodes that player has experienced while playing the game. Importantly, the learning algorithm thus obtained needs to be formulated in domain-general terms to allow for generalization, and it needs to be sample-efficient. That is, the artificial agent obtained by combining the behavioral model and the learning algorithm needs to match both people's learning curve and the learning trajectory through strategy space. There are two projects I have recently started that fall into this category, which I discussed in section 5.1.1.

5.2.4.1 How behavioral models can be useful

Although the end goal of behavioral modeling as I have defined it is understanding *for its own sake*, we can validate or refute models by testing their *usefulness*. For example, if we believe to have found a model that accurately describes an individual's decision-making strategy for a task as well as their learning algorithm, we can then derive an optimal personal *training curriculum* of episodes to be experienced actively or passively that cause the highest performance improvement in the learning algorithm. That training curriculum ought to then also highly increase the individual player's performance.

Additionally, if our models are domain general, we ought to be able to design games or other tasks that, when an individual players learns them, causes that player to improve on other tasks and provides real-life benefits. This practice known as *brain training* is not commonly considered as part of the repertoire of behavioral modeling, but I argue that cognitive scientists have wrongly interpreted some well-publicized negative results (Owen et al., 2010) as evidence against brain training in general, which instead should be seen as a call to arms to design better games or interventions supported by cognitive neuroscience. If we manage to design a successful brain training game or program, that success will then also offer support for the theories or models that were instrumental to the design. Since there are no compelling reasons to believe that brain training is *a priori* impossible, I find the cognitive neuroscience community's eager dismissal of the concept disturbing.

Finally, throughout this entire thesis I have considered healthy individuals, but our behavioral models ought to also generalize to patient populations and explain the nature of specific *disease states* or provide *behavioral biomarkers* to improve diagnosis criteria. In *cognitive psychiatry*, the field devoted to using behavioral modeling for improved mental healthcare, the emphasis is and ought to be on applications but as with personalized teaching or brain training, successful applications will support the models that inspired them.

I am excited that all the ingredients listed above are currently being researched and the community is slowly progressing towards these goals. I hope that with my research, I have been able to play a small role in accumulating understanding of human cognition and complex decision-making.

References

Acerbi, L. and Ji, W. (2017). Practical bayesian optimization for model fitting with bayesian adaptive direct search. In *Advances in Neural Information Processing Systems*, pages 1836–1846.

Ainge, J. A., van der Meer, M. A., Langston, R. F., and Wood, E. R. (2007). Exploring the role of context-dependent hippocampal activity in spatial alternation behavior. *Hippocampus*, 17(10):988–1002.

Akam, T., Costa, R., and Dayan, P. (2015). Simple plans or sophisticated habits? state, transition and learning interactions in the two-step task. *PLoS computational biology*, 11(12):e1004648.

Alon, U. (2009). How to choose a good scientific problem. *Molecular cell*, 35(6):726–728.

Arad, A. and Rubinstein, A. (2012). The 11–20 money request game: a level-k reasoning study. *The American Economic Review*, 102(7):3561–3573.

Audet, C. and Dennis Jr, J. E. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217.

Bahrami, B., Olsen, K., Latham, P. E., Roepstorff, A., Rees, G., and Frith, C. D. (2010). Optimally interacting minds. *Science*, 329(5995):1081–1085.

Bakkes, S. C., Spronck, P. H., and Van Den Herik, H. J. (2009). Opponent modelling for case-based adaptive game ai. *Entertainment Computing*, 1(1):27–37.

Battaglia, P. W., Hamrick, J. B., and Tenenbaum, J. B. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332.

Beck, J. (2008). *Combinatorial games: tic-tac-toe theory*, volume 114. Cambridge University Press.

Bellman, R. (2013). *Dynamic programming*. Courier Corporation.

Bilalić, M., McLeod, P., and Gobet, F. (2008). Expert and "novice" problem solving strategies in chess: Sixty years of citing de groot (1946). *Thinking & Reasoning*, 14(4):395–408.

Billings, D., Papp, D., Schaeffer, J., and Szafron, D. (1998). Opponent modeling in poker. *Aaai/iaai*, 493:499.

Bizo, L. A., Chu, J. Y., Sanabria, F., and Killeen, P. R. (2006). The failure of weber's law in time perception and production. *Behavioural processes*, 71(2-3):201–210.

Black, J. B. and Bower, G. H. (1979). Episodes as chunks in narrative memory. *Journal of verbal learning and verbal behavior*, 18(3):309–318.

Britten, K. H., Shadlen, M. N., Newsome, W. T., and Movshon, J. A. (1992). The analysis of visual motion: a comparison of neuronal and psychophysical performance. *Journal of Neuroscience*, 12(12):4745–4765.

Brown, N. and Sandholm, T. (2017). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, page eaao1733.

Calderwood, R., Klein, G. A., and Crandall, B. W. (1988). Time pressure, skill, and move quality in chess. *The American journal of psychology*, pages 481–493.

Camerer, C. (2010). *Behavioral game theory*. New Age International Hyderabad.

Camerer, C. F., Ho, T.-H., and Chong, J.-K. (2004). A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 119(3):861–898.

Campitelli, G. and Gobet, F. (2004). Adaptive expert decision making: Skilled chess players search more and deeper.

Chabris, C. F. and Hearst, E. S. (2003). Visualization, pattern recognition, and forward search: Effects of playing speed and sight of the position on grandmaster chess errors. *Cognitive Science*, 27(4):637–648.

Charness, N. (1991). Expertise in chess: The balance between knowledge and search. *Toward a general theory of expertise: Prospects and limits*, pages 39–63.

Charness, N. (2013). Expertise in chess and bridge. In *Complex information processing*, pages 203–228. Psychology Press.

Chase, W. G. and Simon, H. A. (1973). Perception in chess. *Cognitive psychology*, 4(1):55–81.

Chinchalkar, S. (1996). An upper bound for the number of reachable positions. *ICGA Journal*, 19(3):181–183.

Clark, C. and Storkey, A. (2015). Training deep convolutional neural networks to play go. In *International Conference on Machine Learning*, pages 1766–1774.

Cohen, Y. and Schneidman, E. (2013). High-order feature-based mixture models of classification learning predict individual learning curves and enable personalized teaching. *Proceedings of the National Academy of Sciences*, 110(2):684–689.

Cornelissen, F. W., Peters, E. M., and Palmer, J. (2002). The eyelink toolbox: eye tracking with matlab and the psychophysics toolbox. *Behavior Research Methods, Instruments, & Computers*, 34(4):613–617.

Costa-Gomes, M. A. and Crawford, V. P. (2006). Cognition and behavior in two-person guessing games: An experimental study. *American Economic Review*, 96(5):1737–1768.

Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. (2000). Improved opponent modeling in poker. In *International Conference on Artificial Intelligence, ICAI'00*, pages 1467–1473.

Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., and Dolan, R. J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215.

Daw, N. D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704.

Dawson, R. (1953). *Unbiased tests, unbiased estimators, and randomized similar regions.* PhD thesis, Harvard University.

de Groot, A. D. (1946a). *Het denken van den schaker: een experimenteel-psychologische studie.* Noord-Hollandsche Uitgevers Maatschappij.

de Groot, A. D. (1946b). *Het Denken van den sckaken.* Noord-Holland. Uitgev. Maatschappij.

de Groot, M. H. (1959). Unbiased sequential estimation for binomial populations. *The Annals of Mathematical Statistics*, pages 80–101.

Dechter, R. and Pearl, J. (1985). Generalized best-first search strategies and the optimality of a. *Journal of the ACM (JACM)*, 32(3):505–536.

Dowling, W. J. (1973). Rhythmic groups and subjective chunks in memory for melodies. *Perception & Psychophysics*, 14(1):37–40.

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., and Efros, A. A. (2018). Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217.*

Elo, A. E. (1978). *The rating of chessplayers, past and present.* Arco Pub.

Getty, D. J. (1975). Discrimination of short temporal intervals: A comparison of two models. *Perception & Psychophysics*, 18(1):1–8.

Gläscher, J., Daw, N., Dayan, P., and O'Doherty, J. P. (2010). States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron*, 66(4):585–595.

Gobet, F. and Simon, H. A. (1998). Expert chess memory: Revisiting the chunking hypothesis. *Memory*, 6(3):225–255.

Grondin, S. (1992). Production of time intervals from segmented and nonsegmented inputs. *Perception & Psychophysics*, 52(3):345–350.

HALDANE, B. J. (1956). The estimation and significance of the logarithm of a ratio of frequencies. *Annals of human genetics*, 20(4):309–311.

Hamrick, J. B., Battaglia, P. W., Griffiths, T. L., and Tenenbaum, J. B. (2016). Inferring mass in complex scenes by mental simulation. *Cognition*, 157:61–76.

Hamrick, J. B., Smith, K. A., Griffiths, T. L., and Vul, E. (2015). Think again? the amount of mental simulation tracks uncertainty in the outcome. In *CogSci*. Citeseer.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26.

Holding, D. H. (1989a). Counting backward during chess move choice. *Bulletin of the Psychonomic Society*, 27(5):421–424.

Holding, D. H. (1989b). Evaluation factors in human tree search. *The American Journal of Psychology*, 102(1):103–108.

Holding, D. H. (1992). Theories of chess skill. *Psychological Research*, 54(1):10–16.

Hunter, D. R. (2004). Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406.

Huyer, W. and Neumaier, A. (1999). Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14(4):331–355.

Huys, Q. J., Eshel, N., O'Nions, E., Sheridan, L., Dayan, P., and Roiser, J. P. (2012). Bonsai trees in your head: how the pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS computational biology*, 8(3):e1002410.

Huys, Q. J., Lally, N., Faulkner, P., Eshel, N., Seifritz, E., Gershman, S. J., Dayan, P., and Roiser, J. P. (2015). Interplay of approximate planning strategies. *Proceedings of the National Academy of Sciences*, 112(10):3098–3103.

Hyman, R. (1953). Stimulus information as a determinant of reaction time. *Journal of experimental psychology*, 45(3):188.

Jeffreys, H. (1998). *The theory of probability*. OUP Oxford.

Johnson, A. and Redish, A. D. (2007). Neural ensembles in ca3 transiently encode paths forward of the animal at a decision point. *Journal of Neuroscience*, 27(45):12176–12189.

Juavinett, A. L., Erlich, J. C., and Churchland, A. K. (2018). Decision-making behaviors: weighing ethology, complexity, and sensorimotor compatibility. *Current opinion in neurobiology*, 49:42–50.

Kahneman, D. and Tversky, A. (2013). Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the american statistical association*, 90(430):773–795.

Kool, W., Cushman, F. A., and Gershman, S. J. (2016). When does model-based control pay off? *PLoS computational biology*, 12(8):e1005090.

Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., and Poeppel, D. (2017). Neuroscience needs behavior: correcting a reductionist bias. *Neuron*, 93(3):480–490.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

Lee, D. and Seo, H. (2016). Neural basis of strategic decision making. *Trends in neurosciences*, 39(1):40–48.

Lee, S. W., Shimojo, S., and O'Doherty, J. P. (2014). Neural computations underlying arbitration between model-based and model-free learning. *Neuron*, 81(3):687–699.

Lehmann, E. L. and Casella, G. (2006). *Theory of point estimation.* Springer Science & Business Media.

Levy, D. J. and Glimcher, P. W. (2012). The root of all value: a neural common currency for choice. *Current opinion in neurobiology*, 22(6):1027–1038.

Lindner, F. and Sutter, M. (2013). Level-k reasoning and time pressure in the 11–20 money request game. *Economics Letters*, 120(3):542–545.

Linhares, A., Freitas, A. E. T., Mendes, A., and Silva, J. S. (2012). Entanglement of perception and reasoning in the combinatorial game of chess: Differential errors of strategic reconstruction. *Cognitive Systems Research*, 13(1):72–86.

Machinery, C. (1950). Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

Malone, T. (1981). *What makes computer games fun?*, volume 13. ACM.

Marder, E. and Bucher, D. (2007). Understanding circuit dynamics using the stomatogastric nervous system of lobsters and crabs. *Annu. Rev. Physiol.*, 69:291–316.

Maximon, L. C. (2003). The dilogarithm function for complex argument. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 459, pages 2807–2819. The Royal Society.

Miller, K. J., Botvinick, M., and Brody, C. D. (2016). Identifying model-based and model-free patterns in behavior on multi-step tasks. *bioRxiv*, page 096339.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nagel, R. (1995). Unraveling in guessing games: An experimental study. *The American Economic Review*, 85(5):1313–1326.

Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

Newsome, W. T., Britten, K. H., and Movshon, J. A. (1989). Neuronal correlates of a perceptual decision. *Nature*, 341(6237):52.

Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., and Preuss, M. (2013). A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5(4):293–311.

Ortega, J., Shaker, N., Togelius, J., and Yannakakis, G. N. (2013). Imitating human playing styles in super mario bros. *Entertainment Computing*, 4(2):93–104.

Owen, A. M., Hampshire, A., Grahn, J. A., Stenton, R., Dajani, S., Burns, A. S., Howard, R. J., and Ballard, C. G. (2010). Putting brain training to the test. *Nature*, 465(7299):775.

Paninski, L. (2003). Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253.

Pedersen, C., Togelius, J., and Yannakakis, G. N. (2010). Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67.

Pfeiffer, B. E. and Foster, D. J. (2013). Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74.

Purcell, B. A. and Kiani, R. (2016). Hierarchical decision processes that operate over distinct timescales underlie choice and changes in strategy. *Proceedings of the National Academy of Sciences*, 113(31):E4531–E4540.

Runarsson, T. P. and Lucas, S. M. (2014). Preference learning for move prediction and evaluation function approximation in othello. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):300–313.

Saariluoma, P. (1992). Visuospatial and articulatory interference in chess players' information intake. *Applied Cognitive Psychology*, 6(1):77–89.

Seo, H., Cai, X., Donahue, C. H., and Lee, D. (2014). Neural correlates of strategic reasoning during competitive games. *Science*, 346(6207):340–343.

Seo, H. and Lee, D. (2008). Cortical mechanisms for reinforcement learning in competitive games. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1511):3845–3857.

Shaker, N., Yannakakis, G. N., and Togelius, J. (2010). Towards automatic personalized content generation for platform games. In *AIIDE*.

Shannon, C. E. (1950). Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275.

Sherrington, C. S. (1913). Further observations on the production of reflex stepping by combination of reflex excitation with reflex inhibition. *The Journal of physiology*, 47(3):196–214.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Simmons, J. P., Nelson, L. D., and Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological science*, 22(11):1359–1366.

Simons, D. J., Holcombe, A. O., and Spellman, B. A. (2014). An introduction to registered replication reports at perspectives on psychological science. *Perspectives on Psychological Science*, 9(5):552–555.

Smith, A. M., Lewis, C., Hullett, K., Smith, G., and Sullivan, A. (2011). An inclusive taxonomy of player modeling. *University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13*.

Snider, J., Lee, D., Poizner, H., and Gepshtein, S. (2015). Prospective optimization with limited resources. *PLoS computational biology*, 11(9):e1004501.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Solway, A. and Botvinick, M. M. (2015). Evidence integration in model-based tree search. *Proceedings of the National Academy of Sciences*, 112(37):11708–11713.

Stanford, T. R., Shankar, S., Massoglia, D. P., Costello, M. G., and Salinas, E. (2010). Perceptual decision making in less than 30 milliseconds. *Nature neuroscience*, 13(3):379.

Stern, D., Herbrich, R., and Graepel, T. (2006). Bayesian pattern ranking for move prediction in the game of go. In *Proceedings of the 23rd international conference on Machine learning*, pages 873–880. ACM.

Stevens, S. S. (2017). *Psychophysics: Introduction to its perceptual, neural and social prospects*. Routledge.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Thaler, R. H. (2016). Behavioral economics: Past, present, and future. *American Economic Review*, 106(7):1577–1600.

Togelius, J., De Nardi, R., and Lucas, S. M. (2006). Making racing fun through player modeling and track evolution.

Van Harreveld, F., Wagenmakers, E.-J., and Van Der Maas, H. L. (2007). The effects of time pressure on chess skill: an investigation into fast and slow processes underlying expert performance. *Psychological Research*, 71(5):591–597.

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., and Iverson, G. J. (2004). Assessing model mimicry using the parametric bootstrap. *Journal of Mathematical Psychology*, 48(1):28–50.

Wan, X., Nakatani, H., Ueno, K., Asamizuya, T., Cheng, K., and Tanaka, K. (2011). The neural basis of intuitive best next-move generation in board game experts. *Science*, 331(6015):341–346.

Yannakakis, G. N., Spronck, P., Loiacono, D., and André, E. (2013). Player modeling. In *Dagstuhl Follow-Ups*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Yannakakis, G. N. and Togelius, J. (2017). *Artificial Intelligence and Games*. Springer.

Zermelo, E. (1929). Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29(1):436–460.