Do People Think Like Computers?

Bas van Opheusden(✉), Zahy Bnaya, Gianni Galbiati, and Wei Ji Ma

Center for Neural Science and Department of Psychology, New York University, New York City, USA
svo213@nyu.edu

Abstract. Human cognition inspired the earliest algorithms for game-playing computer programs. However, the studies of human and computer game play quickly diverged: the Artificial Intelligence community focused on theory and techniques to solve games, while behavioral scientists empirically examined simple decision-making in humans. In this paper, we combine concepts and methods from the two fields to investigate whether human and AI players take similar approaches in an adversarial combinatorial game. We develop and compare five models that capture human behavior. We then demonstrate that our models can predict behavior in two related tasks. To conclude, we use our models to describe what makes a strong human player.

1 Introduction

Developing a computer program to play a given game as well as the best human players was a significant challenge for early computer scientists, even predating the term *artificial intelligence* [1,2]. Much of the initial progress in game-playing AI was inspired by examining human gameplay and formulating games as *search problems* [3]. Subsequently, the Artificial Intelligence community focused on developing algorithms, approaches and concepts in order to improve computer game play for more games in more domains (Checkers [4], Poker, Chess [5] and Go [6–8]), while generally ignoring potential similarities to human thought processes. Meanwhile, psychologists, neuroscientists and economists have built successful models for human reasoning in simple decision tasks, while ignoring games with large decision spaces [9,10]. Recent approaches have begun using human game play to train stronger AI agents [7].

In this paper, we present AI-based computational models for the behavior of non-expert human players in a simple, adversarial, full-information game. Our models formalize hypotheses for the cognitive processes by which a human player makes a decision on a given task; the models we consider simulate human responses to game positions, making similar decisions to human players. We aim to determine whether modern AI concepts such as heuristic search [3] are useful in explaining human play.

We compare the ability of our models to predict subjects' choices during regular game play. We further show that our main model can predict behavior in two related tasks. Finally, we investigate how strongly the playing strength of our subjects is related to our main model's algorithmic properties, such as search depth, tree size, and the quality of the heuristic function.

A. Plaat et al. (Eds.): CG 2016, LNCS 10068, pp. 212–224, 2016.
DOI: 10.1007/978-3-319-50935-8_20

2 Experimental Methods

We collected data from human subjects playing a simple board game. Two players take turns placing pieces on a 4 by 9 board (Fig. 1A). The black player makes the first move. The goal is to place four consecutive pieces in a row, column, or diagonal. We chose this game because the rules are few and easily learned, it is unfamiliar to our subjects, and it is sufficiently hard to master without being computationally intractable.

We performed two experiments on human subjects with a total of four tasks: (1) playing full games against a human opponent, (2) playing against AI opponents with different playing strengths, (3) deciding between two alternative moves on a given board position (2AFC) (Fig. 1B), and (4) evaluating their winning chances in a given board position (Fig. 1C).

Experiment 1: We recruited 40 subjects and divided them into 20 pairs. Subjects in each pair played multiple games against each other without time constraints, switching colors after every game. The experiment terminated after subjects had played for one hour and finished their last game.

Experiment 2: We recruited 40 additional subjects to perform three tasks. For the first 30 min, subjects played games against AI opponents, switching colors after every game. To make it less likely for subjects to latch onto any particular opponent's idiosyncrasies, and to keep play challenging for all subjects, we selected opponents from a set of 30 AI agents with different playing strengths. We switched to a stronger opponent every time the subject won a game, and to a weaker opponent whenever they lost. In the second task, subjects saw board positions and chose between two marked candidate moves (Fig. 1B). We selected the positions and candidate moves to create difficult choices for subjects. In positions where both candidate moves had the same game-theoretic value, the subject's choice indicates a subjective preference. On trials where one move was strictly better than the alternative, the subject's choice can be used to measure their playing strength. The third and final task, *board evaluation*, required subjects to rate board positions from 1 ('losing') to 7 ('winning') from the perspective of the current player. In the second and third task, each subject completed 84 trials.

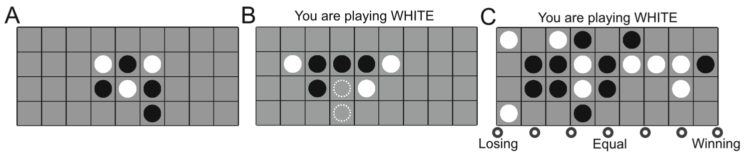

Fig. 1. A: Example of a game position. **B:** On a trial of the 2AFC task, subjects see a board position with two possible moves, and indicate their preference. **C:** On a trial of the evaluation task, subjects see a board position and estimate their winning chances on a 7-point scale.

3 Models of Human Behavior

Our goal is to build a computational model that mimics how human subjects play our game. A model of behavior is an algorithm that, given a board state s, selects a move $a \in A(s)$ from the set of available moves $A(s)$. To account for variability in human choices, our models contain multiple sources of stochasticity. Since players may vary in their decision processes and cognitive abilities, our models have parameters, which we fit to individual subjects. In this section we discuss the following seven items: heuristic function, sources of variability, myopic model, main model, conv-net model, opt-rand model, and fitting the model parameters.

3.1 Heuristic Function

Most of our models rely on a *heuristic function* that assigns a value to each board position. Our heuristic function is a weighted sum of five features. Each feature is counted separately over a player's own pieces and their opponent's pieces. The first feature, which we call the *center* feature and denote by $f_0(s, c)$, measures the number of pieces of color c on the 12 central squares of the board s. The other four features (Fig. 2), denoted by $f_i(s, c)$ with $i = 1, \ldots, 4$, count how often the following patterns occur on the board (horizontally, vertically, or diagonally).

1. *Connected 2-in-a-row:* two adjacent pieces with sufficient 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 terminal boards.

We handpicked these features to reflect heuristics that are intuitive given the goal of the game. We tested additional features, but none of them improved the main model's fit to human play. However, a more systematic approach to select these features is a natural direction that we leave for future work.

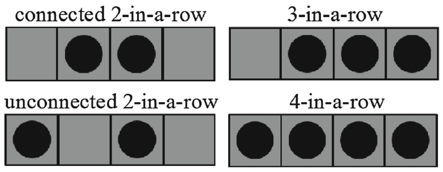

Fig. 2. Patterns in the heuristic function. The four features in our heuristic function. Each feature counts how often one of these patterns occurs on board (horizontally, vertically, or diagonally).

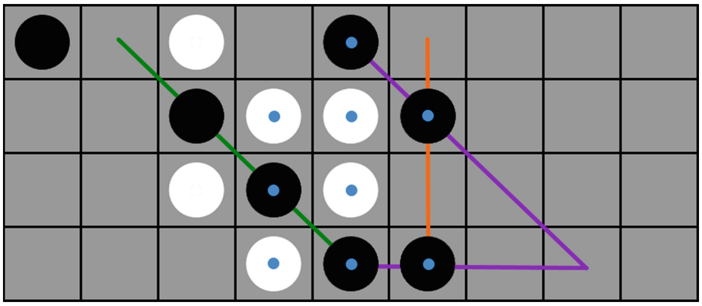

Fig. 3. Heuristic function. In this position, white is to move. Black has 5 pieces on the central squares, white has 4 (marked with blue dots). Black has two connected two-in-a-rows (purple), one unconnected two-in-a-row (orange) and one three-in-a-row (green). White has no instances of any pattern. The value of this board state, from white's perspective, is therefore $H(s) = -w_0 - w_1 - 2w_2 - w_3$. (Colour figure online)

We associate a weight w_i to each of the five features, and write the heuristic function as

$$H(s) = c_{\text{self}} \sum_{i=0}^{4} w_i f_i(s, \text{own color}) - c_{\text{opp}} \sum_{i=0}^{4} w_i f_i(s, \text{opponent color})$$

where $c_{\text{self}} = C$ and $c_{\text{opp}} = 1$ whenever the player is to move in state s, and $c_{\text{self}} = 1$ and $c_{\text{opp}} = C$ when it is the opponent's move. The scaling constant C is a fitting parameter which can vary between subjects. Figure 3 demonstrates a calculation of the heuristic function in an example board state, taken from human play.

The weight parameters $W = \{w_0, w_1, \ldots, w_4\}$ vary between subjects. They encode differences in subjects' preferences, such as their relative inclination to make direct threats (3-in-a-row) over indirect strategic maneuvers (unconnected 2-a-in-row).

3.2 Sources of Variability

Unlike deterministic AI agents, realistic models for human behavior must support variability. Our models are required not only to identify the subject's most likely move given a position, but also to assign some probability to their noisy and inconsistent decisions.

We introduce three sources of variability in our models. (a) *Value noise*: We add Gaussian noise to the heuristic value of each state, reflecting a human tendency to choose almost arbitrarily between two moves of roughly equal value. (b) *Feature dropping*: When counting instances of any one of our patterns, we exclude with probability λ every possible location-orientation combination where that pattern may occur. This mechanism represents lapses of attention, where a subject overlooks a pattern in some region on the board. We denote this

Algorithm 1. Myopic-model(State s, Parameters $\{\lambda, W, lapse\}$):

```
1 if lapse then
2     return random-move
3 else
4     return $\mathrm{argmax}_{a \in A(s)} H_\lambda(T(s,a)) + \mathcal{N}(0,1)$
```

modified heuristic function by $H_\lambda(s)$. (c) *Lapse rate*: On each move, there is some probability that the model makes a completely random move, capturing human moves with no apparent rationale behind them. The lapse rate, feature dropping rate (λ), and feature weights are all model parameters. We now describe the five specific models that we test.

3.3 Myopic Model

After checking for a lapse, the *Myopic* model (shown in Algorithm 1) uses a heuristic function with value noise and feature dropping to evaluate every possible move on a given board position; it then selects the move with the highest value. We use $T(s,a)$ to denote the resulting state by applying action a to state s.

3.4 Main Model

Our main model (described in Algorithm 2) builds a partial game tree similar to algorithms such as Minimax, alpha-beta pruning, and Monte-Carlo Tree Search. Each state is represented as a node in the tree. Each node n has a value estimate $V(n)$ and a set of successors $Succ(n)$.

On each execution, the model initially determines whether a lapse occurs, in which case it makes a random move (lines 1–2). Otherwise, the model builds the root node to represent the current state (line 3) and repeats a procedure to build a partial tree. On each iteration, the algorithm selects a node in the tree for further exploration (line 4). The *selectnode* procedure recursively selects the successor node with the maximal heuristic value until it reaches a leaf node. The selected node is *expanded* (line 5) by the *expand*(n) procedure, which generates successor nodes of the selected node n and assigns each of them a value using the modified heuristic function H_λ. Successor nodes with value less than the best move minus a threshold are pruned from the game tree; the remaining nodes are added to the partial tree.

The *backpropagate* procedure (line 6) recursively updates the values of the predecessor nodes up to the root of the tree. Each node value is assigned the maximum value of its successors. The algorithm iterates for a random number of iterations, with a fixed probability to stop each iteration. Finally, the model returns the move with the highest value (line 7).

Algorithm 2. Main-model(State s, Parameters $\{\lambda, W, \mathit{lapse}, \mathit{stop}\}$):

```
1 if lapse then
2     return random-move
3 root = node(s)
4 while !stop do
5     n=selectnode(root)
6     expand(n)
7     backpropagate()
8 return $\text{argmax}_{n_i \in Succ(root)} V(n_i)$
```

3.5 Conv-net Model

We develop an alternative model based on convolutional neural networks, which have recently been used successfully to play Go [7,8]. Our convolutional neural network (CNN) model treats the game as a classification problem, learning to assign 1 of 36 labels to a board, represented by a $4 \times 9 \times 2$ binary tensor. The network has three layers: an input layer, a hidden convolutional layer, and an output layer. The convolutional layer contains 32 $4 \times 4 \times 2$ filters with rectified linear activation functions. There is no pooling layer between the convolutional layer and the fully output layer. The output layer is a fully connected layer, to which two nonlinearities are applied: the first is a softmax function to convert the output to a probability distribution over the 36 possible labels, the second is a filter that forces zero probability to be assigned to occupied squares.

We fit the CNN model using stochastic gradient descent with Nesterov momentum. To reduce overfitting, we introduce random dropout ($p = 0.75$) between the hidden layer and the output layer and an early stopping condition during training. We use a five-fold cross-validation scheme with the same splits as used for fitting the main model, setting aside 60 % of the data as training data, 20 % as validation data used for the early stopping condition, and 20 % as final test data. Because we did not collect sufficient data to fit the network to each subject individually, we aggregate the data across all subjects for training and report the average log-likelihood per subject. Additionally, we apply reflections to augment the training data to achieve a sufficiently large training set.

3.6 Opt-rand Model

The opt-rand model is a mixture between optimal (i.e., minimax) and random play with only one parameter: the mixture weight. Because human subjects do not have access to the minimax values of each state, we consider the opt-rand model psychologically implausible. However, it still serves as an important control to verify whether our models predict only the subjects' frequency of making mistakes, or more general preferences.

3.7 Fitting the Model Parameters

We use maximum-likelihood estimation to infer the parameter values Θ that maximize the likelihood function $\prod_{(a_t,s_t)\in D} P(a_t|s_t,\Theta)$ where D is the set of all actions performed by a subject in all the states they encountered. Because computing the likelihood analytically or numerically is intractable, we instead estimate the log probability of a subject's move in a given board position using inverse binomial sampling [12]. We use a uniformly unbiased estimator with variance equal to the Cramer-Ráo bound, and optimize the log-likelihood function with *multilevel coordinate search* [13]. We report log-likelihoods for all models with five-fold cross-validation.

4 Results

We compare our models and show which of them best describe subjects' choices. To demonstrate that all parts of our main model are important, we compare our model to *lesion* models generated by removing model components (in Sect. 4.1). Next, we show two specific patterns in human behavior that our model accurately predicts (in Sect. 4.2). We then show that our model is able to predict the subjects' responses in two related tasks (in Sect. 4.3). Finally, we use the model to explain differences in the decision process between stronger and weaker subjects (in Sect. 4.4). We find that the model, fitted to stronger subjects' choices, uses larger trees and has less noise.

4.1 Predicting Human Choices with Our Models

Fig. 4A depicts the cross-validated log-likelihood of our models (*Main*, *Myopic*, *Conv-net* and *Opt-Rand*) for each subject, playing against a human opponent. We also plot the log-likelihood of a completely random model (*chance*). Our models' log-likelihoods are better than chance, demonstrating their ability to predict subjects' responses.

We find that our main model predicts subjects' choices better than the *Myopic* model, suggesting that people indeed build decision trees. The Conv-net model also performs worse than the main model, but this primarily reflects its tendency to overfit training data. All our models perform much better than the *Opt-Rand* mixture model, demonstrating their ability to predict more than only the subjects' error rates.

We next perform a lesioning comparison, examining the relative contribution of different components in our main model by removing them, one at a time. We remove either the pruning rule, the feature-drop procedure, or any of the five features. All of the lesioned models perform worse than the original (Fig. 4B), indicating that these model components are necessary to the main model's ability to predict human behavior. The most and least important features are the *3-in-a-row* and the *center*, respectively. This also demonstrates that the pruning and feature-drop are necessary to capture the subjects' selective attention, either to specific patterns on the board or to a subset of the decision tree.

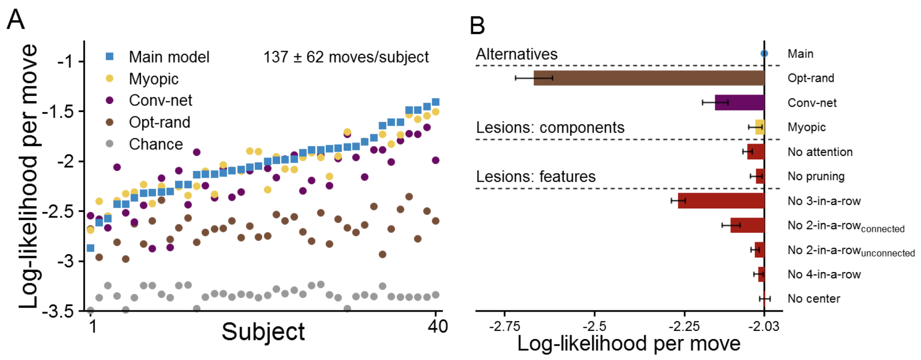

Fig. 4. (A) Log-likelihood of our models for each subject. Our main model performs better than chance, Opt-Rand, Conv-net and the Myopic model. **(B)** Log-likelihood of our models and lesions, averaged across subjects. For each model, the error bars denote the standard error of the mean log-likelihood difference with the main model. The main model performs best, although some lesion models come close.

4.2 Summary Statistics

We have shown that our main model predicts the subjects' choices better than alternative models. Here, we compare the model prediction directly to the subjects' choices, using two *summary statistics*. For each move played by each subject, we measure (1) the distance from the square they moved on to the center of the board, and (2) the number of pieces on the 8 neighboring squares. We plot the average of these statistics as a function of the number of moves played in a game (Fig. 5). We also measure these statistics for moves played by the model in the same positions, as well as random moves. On average, subjects move closer to the center and on squares with more neighboring pieces than random. The model closely matches these two aspects of human play.

4.3 Generalizing Predictions of Our Model

We demonstrate our model's ability to generalize beyond predicting the subjects' choices during full games by inferring parameters for each individual subject from their choices during games, and predicting their 2AFC choices and board evaluations without additional fitting.

To predict a choice on a 2AFC trial, we execute our tree search model as usual, except that we restrict the successor nodes of the root node to the two candidate moves and omit the pruning step. To predict board evaluations, we execute our model and take the value of the root node. If the model lapses, we set this value to 0. Then, we map this value into the subject response interval [1,7] using score $= 3 + 4\tanh(value/20)$.

The average accuracy of the 2AFC prediction across subjects is $56.1 \pm 1.1\,\%$ (Fig. 6A), and the average correlation between the predicted and observed

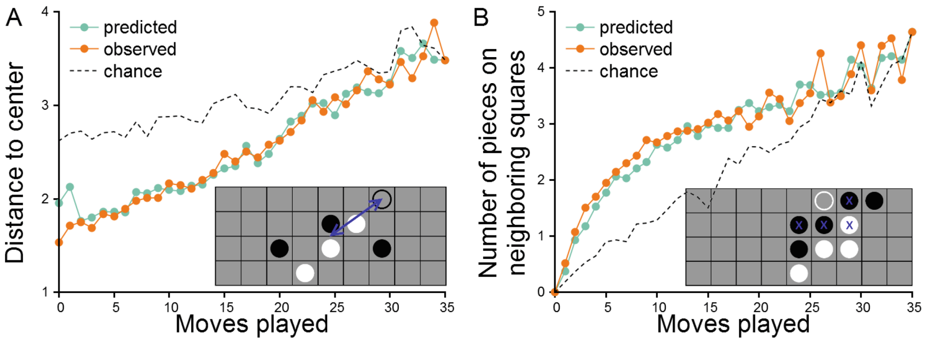

Fig. 5. The predicted and the observed behavior on **(A)** the average distance from the move played by a subject to the center of the board. **(B)** The number of pieces on neighboring squares. Our model reproduces both these patterns. The insets illustrate how these metrics are defined for a given board and a subject's move (open circle).

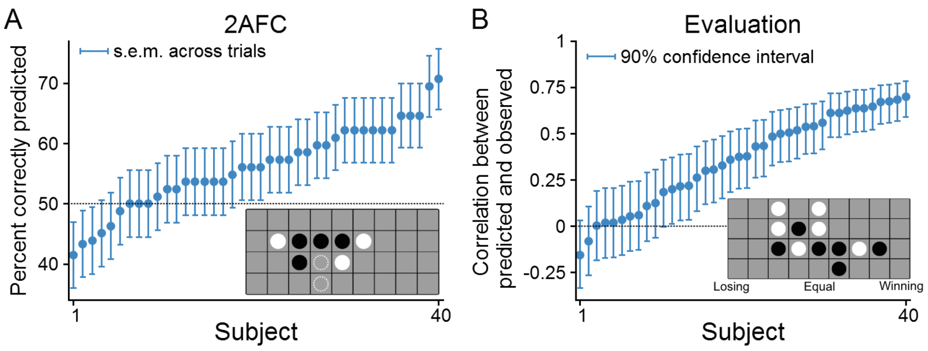

Fig. 6. **(A)** Percentage of correctly predicted choices on the 2AFC task for each subject. **(B)** The correlation coefficients between each subject's board evaluations and evaluations predicted by the model. In both cases, we fitted the model parameters on subjects' choices during games against AI opponents. Both predictions are better than chance for almost all subjects.

evaluations is $\rho = 0.36 \pm 0.04$ (Fig. 6B). The prediction is better than chance for 34 out of the 40 subjects in the 2AFC task and for 38 subjects in the evaluation task.

To put these results into context, we develop an *oracle* model, which selects the optimal move on each 2AFC task (with random tie-breaking). On the board-evaluation task, the oracle responds 1, 4 or 7 for winning, drawn and losing positions, respectively. Overall, the oracle model predicts subjects' choices slightly worse than our main model (percent correct 2AFC: 55.3 $\pm$ 0.6 %, correlation predicted/observed evaluation: $\rho = 0.30 \pm 0.03$, Fig. 7).

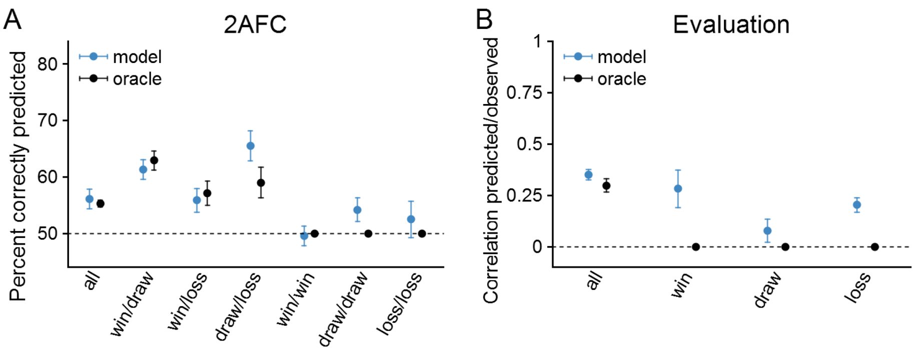

Fig. 7. (A) Performance of our main and oracle models on each category of candidate moves. **(B)** Correlation between predicted and observed evaluations on positions with the same game-theoretic value. In both cases, our model performs on average slightly better than the oracle model. Importantly, our model predicts subjects' preferences when there is no correct decision.

To explain our model's advantage over the oracle model, we compute the percent of correctly predicted 2AFC choices for the main and oracle models for each category of trials (win/win, win/draw, etc.).

For trials where one move is strictly stronger, our model performs comparably to the oracle model, showing that our model does capture the subjects' error rates. For trials where both moves are equally strong, the oracle predicts at chance, but our model performs better, demonstrating that our model predicts the subjective preferences. In the board-evaluation task, we compute the correlation between predicted and observed evaluations across all trials in a category. Again, the oracle model predicts at chance, but our model can predict the subjective evaluations, for either winning or losing positions (but no drawn).

4.4 Playing Strength

The model parameters that we infer for each individual subject reflect how human thought processes differ between subjects, allowing us to examine the differences between strong and weak players. We measure a subject's playing strength by combining 4 metrics: (1) the Elo rating [14] computed from their results in games against AI opponents, (2) the frequency at which they make errors in their games, (3) the percentage of correct choices in the 2AFC task, and (4) the correlation of their board evaluations with the game-theoretic values. All 4 performance metrics correlate with each other across subjects as shown in Table 1.

The playing strength of heuristic search algorithms depends on properties such as the size and depth of the game tree or the 'quality' of the heuristic function. Because our model is stochastic, we can also improve its playing strength by reducing noise. Among these factors, which is responsible for differences in human playing strength?

Table 1. Player strength correlation matrix

	Elo	Success rate	2AFC	Evaluation
Elo	1	0.83	0.61	0.47
Success rate		1	0.47	0.44
2AFC			1	0.43
Evaluation				1

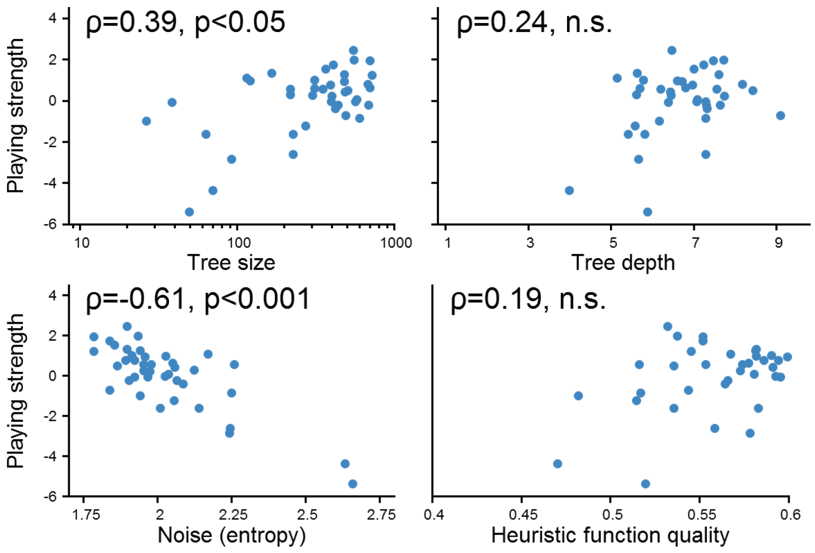

Fig. 8. Correlation between playing strength and size of decision tree, depth of leaf nodes, entropy of the predicted distribution, and heuristic quality. We use *Spearman* correlations to mitigate the effect of outliers. Stronger players build larger trees and have less noise but do not necessarily have better heuristics or search deeper.

For each subject in Experiment 2, we infer model parameters from their choices in games against AI opponents. We let the model with these parameters simulate moves in all positions from the games in Experiment 1. We measure the size of the decision tree built by the model, the average depth of the leaf nodes, the entropy of the model's move distribution, and the correlation between the heuristic function $H(s)$ and the game-theoretic value.

In Fig. 8, we plot these 4 metrics against the playing strength of each subject. The tree size and entropy correlate with playing strength, but the depth of search and heuristic function quality do not; stronger players search more, have more precise board evaluations, and make fewer attentional lapses.

5 Summary and Future Work

We described a model inspired by heuristic search that mimics humans playing a simple combinatorial game. We fitted the model's parameters to individual subjects to capture differences in playing styles. We also suggested alternative models and compared our model to lesions in order to show that the components of our model are necessary to predict human behavior. We then showed that our model predicts subjects' choices in 2AFC tasks and board evaluations. We analyzed player strengths and conclude that stronger players build larger trees and have less noise.

For future work, we plan to investigate whether our models can also describe choices of expert players. We plan to run multiple sessions of Experiment 2 to measure improvements in the subjects' playing strength and investigate which aspects of our model (tree size and depth, noise or heuristic quality) change as a result of experience. We also plan to investigate the encoding of board states in human memory by asking subjects to memorize and then reconstruct board positions, similar to what was done previously in Chess [15]. We are also interested in finding physiological and neural correlates of our model. We plan to record response times, eye movements, and neural activity as measured by an *fMRI* scanner, and use that as further evidence that our model captures the cognitive processes humans use to play games.

References

1. Turing, A.M.: Computing machinery and intelligence. Mind **59**, 433–460 (1950)
2. Shannon, C.E.: XXII. Programming a computer for playing chess. London, Edinb. Dublin Philos. Mag. J. Sci. **41**, 256–275 (1950)
3. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading (1984)
4. Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., Szafron, D.: A world championship caliber checkers program. Artif. Intell. **53**, 273–289 (1992)
5. Campbell, M.A., Joseph, H., Hsu, F.H.: Deep Blue. Artif. Intell. **134**, 57–83 (2002)
6. Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search. In: van den Herik, H.J., Ciancarini, P., Donkers, H.H.L.M.J. (eds.) CG 2006. LNCS, vol. 4630, pp. 72–83. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75538-8_7
7. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484–489 (2016)
8. Clark, C., Storkey, A.: Training Deep Convolutional Neural Networks to Play Go. J. Mach. Learn. Res. **37** (2015)
9. Glimcher, P., Fehr, E.: Neuroeconomics: Decision Making and the Brain. Academic Press, London (2013)
10. Camerer, C., Loewenstein, G.: Behavioral economics: past, present, future. In: Advances in Behavioral Economics. Princeton (2004)
11. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**, 235–256 (2002)

12. de Groot, M.H.: Unbiased sequential estimation for binomial populations. Ann. Math. Stat. **30**, 80–101 (1959)
13. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. J. Global Optim. **14**, 331–355 (1999)
14. Elo, A.E.: The Rating of Chessplayers, Past and Present. Arco Pub. (1978)
15. Chase, W.G., Simon, H.A.: Perception in chess. Cogn. Psychol. **4**, 55–81 (1973)