Comparing Cognitive, Cognitive Instance-Based, and Reinforcement Learning Models in an Interactive Task

Amir Bagherzadehkhorasani (amir.bagherzadeh@psu.edu)
Department of Industrial and Manufacturing Engineering, Penn State, University Park, PA 16802 USA

Farnaz Tehranchi (farnaz.tehranchi@psu.edu)
School of Engineering Design, Technology and Professional Programs, Penn State, University Park, PA 16802 USA

Abstract

This work tries to answer fundamental questions of learning bias in cognitive models, how decision-making strategies in different cognitive models vary and why. Using a biased coin in a coin flip game, we study the number of trials it takes for each cognitive model to learn the asymmetry in the coin. Also, we investigate how the model behaves knowing the asymmetry. A web-based game is designed to simulate coin flipping to collect the models' data. The most common approaches to model the decision-making process are used for this study. Cognitive architectures such as ACT-R and PyIBL with the capability of learning and making decisions are used and compared. Also, we consider Reinforcement Learning with different decision-making strategies such as Thompson Sampling, Boltzmann Exploration, and Epsilon Greedy algorithm. All developed models interact with the task environment and complete the task. To facilitate the interaction between the models and the game's interface, we developed a new tool called VisiTor. VisiTor grants cognitive models the ability to gain information and execute actions in dynamic environments. The results show models are capable of replicating human's main decision-making strategies: matching and maximizing.

Keywords: ACT-R; cognitive modeling; reinforcement learning; instance-based learning; binary choice experiments; decision-making

Introduction

The most commonly used method to study human decision-making procedures consists of observing human performance in a choice task and proceeding with developing a cognitive model. These models emulate human behavior (Cassimatis, Bello, & Langley, 2008); Erev et al. (2005) discussed the learning process with immediate feedback, which consists of different processes such as the tradeoffs of adaptation and maximization in repeated choice tasks. They proposed a Reinforcement Learning model alongside the cognitive strategies to consider the payoff variability and other deviations. Janssen et al. (2012) also utilized ACT-R to study the effect of the reward value. They suggested a new approach to determine the reward that is experienced in the environment. Lebiere et al. (2007) used Instance-Based Learning (IBL) to demonstrate that a binary choice problem with immediate feedback does not always lead to payoff maximization. One of the factors that limits the studies to explore more complicated choice tasks is the restricted cognitive models' capability to interact with task environments. ACT-R is a hybrid cognitive architecture that is consisted of a set of programmable information-processing mechanisms. These mechanisms are used to predict and explain human behavior, including cognition and interaction with the environment (J. R. Anderson et al., 2004; Ritter, Tehranchi, & Oury, 2018; Tehranchi & Ritter, 2018a). Ever since the emergence of ACR-R in 1998 (John R Anderson, Bothell, Lebiere, & Matessa, 1998), several researchers have utilized ACT-R capabilities to simulate human interaction and cognition while performing a specific task (Cao, Ho, & He, 2018; Gray, Schoelles, & Sims, 2005; Hope, Schoelles, & Gray, 2014). ACT-R models typically interact with the world through ACT-R's device interface, an abstract representation of the world based on a simulated Lisp environment provided with ACT-R or by instrumenting interfaces. However, these interactions are limited to being applied to an unmodified ACT-R environment in special windows provided by ACT-R. In other words, if the environment that a model is interacting with is subject to change, the model will not be able to work properly. PyIBL is a Python implementation of a subset of Instance-Based Learning Theory (Gonzalez, Lerch, & Lebiere, 2003). PyIBL does not have a built-in capability to interact with any environment.

Inspired by JSegMan, SegMan, and ACT-CV (Halbrügge, 2013; St. Amant, Riedel, Ritter, & Reifers, 2005; Tehranchi & Ritter, 2018b), we developed VisiTor (Vision+Motor) that generates the required interactions in dynamic task environments. VisiTor simulates users' visual attention (vision) and use of a mouse and keyboard (Motor). This tool allows ACT-R and PyIBL to interact with any environment while keeping the operations similar to users as close as possible and its capabilities are expandable.

Probability Learning and Decision-Making in Psychology Literature

Unknown bias effects on decision-making and prediction of the next outcome using a binary choice prediction task have been studied before. Bilda, Gero, and Sun (2006) conducted a simulation modeling bias for a pitch in baseball. Altmann and Burns (2005) studied the effect of streaks in coin flips on the prediction of the next toss. In binary choice experiments, participants are asked on each trial to predict the outcome of an event such as a coin flip. The outcome is usually biased towards one of the choices, and participants are not informed of the bias. Altmann et al. (2005) claimed that participants tend to adapt their behavior to the relative reward accordingly

instead of maximizing the expected reward. In another word, they try to "match" rather than "maximize." In matching, participants' choices would reflect bias in the coin, while in maximizing, the participants would maximize the reward by choosing the option with a higher probability of success. Assume in a coin flip game that the ratio of head and tail is 3 to 1. While matching, Participants predict heads on roughly 75% of trials by the end of a session. Whereas in the optimal strategy, one should choose head 100 percent of the trials to maximize the number of wins, once they detect the bias. This aligns with (Vulkan, 2000) results. Such behavior is paradoxical because matching results in less reward receipt than maximizing. This is because participants cannot know when a given location or response option will be rewarded, even if they are aware of the overall reward rate.

The effects of age in the strategy taken after learning the bias has been a subject of conflict among Probability Learning studies. The ratio of school-age children demonstrating matching strategy is similar to the ratio of adults using the matching strategy (Brackbill & Bravos, 1962; Derks & Paclisanu, 1967). Also, Younger children (ages 3–5 years) demonstrate maximizing more than older children (Plate, Fulvio, Shutts, Green, & Pollak, 2018). While Moran III and McCullers (1979) have found that adults maximize rewards more effectively than children. Recently, Plate et al. (2018) conducted a comprehensive study on adults and children. They compared their results to 4 different decision-making models: Random model, Matching model, Maximizing model and a combination of the last two (Combination model). Most adults and children's results matched the Combination model based on their study, suggesting participants exhibited matching behavior at the outset of the experiment and then crossed over to maximizing in the experiment. All participants who did not crossover from matching to maximizing were the best fit by the probability Matching model and are sensitive to the underlying probabilities. In summary, all researchers agree that people can identify the bias if the bias is significant enough. However, how they react to the bias is still a subject of discussion.

Probability Learning and Decision-Making in Artificial Intelligence Literature

Probability Learning and decision-making models are not following the same strategy when taking an action. Most of Reinforcement Learning models learn the outcome distribution of each action by using posterior distribution over the outcome of each action. These models seek to find the best possible action for each scenario (Zhu, 2018). On the other hand, cognitive models do not necessarily look for the best action. Instead, they try to simulate human behaviors in the same scenario, regardless of the optimality of choice (Lebiere et al., 2007). In line with psychology literature, cognitive models have different strategies for decision-making. Reinforcement Learning models and cognitive models are capable of imitating both matching and maximizing decision-making strategies.

Agents, developed using Reinforcement Learning (RL), interact with a task environment and generally learn to maximize their rewards (Sutton & Barto, 2018). These agents discover which actions to take to generate the highest possible rewards.

The Reinforcement Learning model discovers the right set of actions to take by trial and error. By observing the result after each instance, the model learns the outcome distribution of the actions. There are two important components in learning the outcome distribution of each choice: (a) how to update the outcome distribution based on the action taken, and (b) what action to take. The reward function is the cornerstone of the learning aspect of RL models. It maps each action to the outcome. The environment's characteristics, such as the delay between taking an action and observing the outcome and the possible outcome distributions, can affect how the reward function is defined (Guo, 2017). Deep Reinforcement Learning models replace the reward function with a Neural Network and let the model determine the best reward function (Li, 2017).

Decision-making strategies such as the Greedy algorithm results in maximizing, Boltzmann Exploration results in matching, and Thompson Sampling (Thompson, 1933) results in the combination of matching and maximizing behavior.

Due to this limitation of greedy algorithms, several methods have been developed to add exploration through randomly perturbing actions that a greedy algorithm would select (Dabney, Ostrovski, & Barreto, 2020; Masadeh, Wang, & Kamal, 2018; Tokic, 2010). These methods are called *Dithering*. The most basic Dithering method is called *Epsilon Greedy Exploration*. This method applies the greedy action with probability $1 - \varepsilon$ and otherwise selects an action uniformly at random. Although this type of exploration improves the performance of the greedy algorithm, it wastes resources by trying all the actions, even those that are unlikely to generate a better reward than what we already have. For example, half of the exploration is wasted by trying action 2. This issue gets worse as the number of actions increases.

Thompson Sampling was introduced more than 80 years ago (Thompson, 1933). This method provides an alternative to dithering that more intelligently allocates the exploration effort. In this method, a Beta distribution with ($\alpha = 1, \beta = 1$) is initially assumed for each action. At each instance, we sample from each action's distribution. Whichever action gives us the largest sample value will be chosen. After the action is taken and the result is observed. If it is a success, α is increased by one. Otherwise, β is increased by one. This process will be repeated each time an action needs to be taken.

Boltzmann Exploration utilizes a similar strategy of decision-making to ACT-R. The actions are taken stochastically. Initially, the reward for all actions is assumed to be equal. At each trial, the probability of taking an action *i*, is calculated as follows:

$$P_i = \frac{e^{U_i/T}}{\sum_{i \in m} e^{U_i/T}}$$

where *m* is the set of all actions. The action is going to be taken based on a discrete distribution with probabilities calculated from this equation. After each trial, the expected rewards for all actions are updated. This equation indicates that as the chance of actions being taken is proportional to the values of U_i/T.

The parameter T is known as temperature. It controls the randomness of the action. The higher the value of T, the more randomness happens in action selection.

Methodology

In this study, we considered a simple coin flip game. Every round, participants and models choose either head or tail. If their choice matches the game's choice, the result is winning the round and a message "Match" will show up. If the choices do not match, the result is losing that round and a message "Wrong" will show up. The probabilities of the computer choosing head or tail are not equal. In 70 percent of the occurrences, head will appear, and the tail will appear in 30 percent of the trials. This game is an online browser game and is written in C# and was first used by (Tehranchi & Ritter, 2020) to study the number of trials needed for ACT-R to match the probability of the biased coin. Figure 1 shows a screenshot of the game when a user starts playing. The models' data of interactions with the game and their final decisions are saved.

Figure 1: The game environment. It consists of a feedback area (where the "Ready" sign is shown) and visual objects (Head and Tail buttons). The feedback area will be changed based on the result of the played round. It shows "Match" if the choice is correct, and it shows "Wrong" if the choice is incorrect.

The Coin flip game is played in an interactive environment. In this environment, the interface is susceptible to change if the user provides input. Every time a model chooses what to play for that round and clicks on the buttons. The feedback area is going to change. Also, the model continues to work even if the location of the environment window is moved. And because of that, the visual module implemented in ACT-R cannot be used for this task.

The ACT-R Model

ACT-R's actions that play the game can be broken down into several subtasks. Each subtask consists of some production rules that ACT-R uses to play the Coin flip game.

Every visual object on display will be represented by a set of unique features for the visual module. Chucks are created by these features that provide declarative memory, the representation of the visual scene by the vision module. These chucks are visual location and visual object types. Production rules' constraints can match the chunks. The model consists of 3 visual objects, "Ready", "Heads", and "Tail", in addition to 15 declarative memory chunks and 12 production rules. Production rules execute shifting attention, finding the ready icon to start, choosing heads or tail, clicking, finding the visual feedback, and assigning reward value to the results. The reward value for matching is 6 and in case of wrong, the reward value of zero is assigned. The Utility Value and the sub-symbolic computations parameter are set to true. The value of temperature varies based on the decision-making strategy we are trying to replicate. All the other parameters are set to default.

Starting the Game

The model looks for the visual object "ready". If it finds it, the model is ready to choose an action. At this point, the reward (Utility) function for all the choices is equal.

Taking an Action

Based on the reward function, the model will take an action, retrieves the visual object corresponding to that action, moves the cursor to the visual location, and clicks. Each action has a probability of being taken. The probability for action *i* is calculated using the Boltzmann Equation:

$$\text{Probability}(i) = \frac{e^{U_i/\sqrt{2}s}}{\sum_{j \in m} e^{U_j/\sqrt{2}s}}$$

Where the summation j is over all the productions which currently meet the conditions required. ACT-R multiplies the temperature value (T in Boltzmann Equation) by the square root of two.

After the model decides what action to take, the model needs to find the visual object corresponding to that action and select (click) it. For this task, VisiTor first finds the location of the visual objects on the screen by Template Matching capability of the OpenCV Python library. The templates are predefined and saved as an image. Then, VisiTor will save that visual object as an image. In order to assure than VisiTor is robust to rescaling and size, different sizes of the template are checked. Then, VisiTor moves the mouse to the location of that object and clicks.

Looking for Feedback

After taking an action, the model expects feedback. The model tries to find which of the feedback visual objects is shown on the screen. The model first retrieves them into the memory module and then utilizes VisiTor. VisiTor search for the feedback visual object that is appearing on the screen.

Updating the Reward Function

Based on the feedback, the model updates the reward function for the action taken in the last step.

$$U_i(n) = U_i(n-1) + \alpha[\, R_i(n) - U_i(n-1)]$$

Where:

- α is the learning parameter
- $R_i(n)$ is the effective reward value given to production i for its n[th] usage
- $U_i(0)$ is the initial utility value for production i

Then the model goes back to the *Taking an action* section and repeats the whole process.

Similar to ACT-R, RL models follow the same set of actions to play the game. The only difference is how the reward function is updated and the decision-making strategy. Here, we tried Epsilon Greedy, Boltzmann Exploration, Thompson Sampling, and PyIBL to analyze the differences. In this section, we will elaborate on the decision-making process of each model and what decision-making strategy they utilize in the coin flip game.

PyIBL Model

PyIBL uses the concept of blending to calculate the utility value of each choice in its decision-making process (Lebiere, 1999). The blending mechanism consists of activation, base level activation, weights, utilities, noise, and temperature.

Activation

A fundamental part of retrieving an instance from the PyIBL model's memory is computing the activation of that instance. The value of the activation is based on (a) how frequently and recently it has been experienced by the model and (b) how well it matches the attributes of what is to be retrieved. The activation is calculated based on the following formula:

$$A_i = B_i + \epsilon_i$$

Where:

- A_i: Activation of chunk *i*. It is also called "match score" M_i
- B_i: This is the base-level activation and reflects the recency and frequency of use of the chunk. We elaborate on this and how to calculate this more
- ϵ_i: A noise value

Base Level

The base-level activation, B_i, describes the frequency and recency of the chunk i. Its value depends upon the decay parameter of Memory, *d*. The base level activation is computed using the amount of time that has elapsed since each of the past appearances of i, which in the following are denoted as the various t_{ij}.

$$B_i = \ln\left(\sum_j t_{ij}^{-d}\right)$$

Activation Noise

The activation noise, ϵ_i, implements the stochasticity of retrievals from Memory. It is sampled from a logistic distribution centered on zero. It is normally resampled each time the activation is computed.

Blending

A weight is calculated for chunks using their corresponding activation values to present the contribution of chunks in the blending value.

$$w_i = e^{\frac{A_i}{\tau}}$$

With the activation values calculated for all the chunks corresponding to an action, the blending value is calculated as follows:

$$BV = \sum_{i \in m} \frac{w_i}{\sum_{j \in m} w_j} u_i$$

Lastly, the action with the largest blending value is taken. If the outcome is already represented by a chunk, the base level activation will be updated. If not, a chunk will be created to represent the outcome in the next blending equation.

Deep Reinforcement Learning

First, a Neural Network predicts the outcome of each action based on the instances the model has seen so far. After each trial, the outcome is observed. Using the observed outcome, the model tries to tune the Neural Network parameters to predict the outcome more accurately. The loss function for the Reinforcement Learning model is defined as follows:

$$\mathrm{L} = \mathrm{E}[(\,\mathrm{U}(\mathrm{s},\mathrm{a};\,\theta_k) - \mathrm{U}(\mathrm{s},\mathrm{a}))^2]$$

Where the first term is the Neural Network predicted reward function and the second term is the actual reward observed by the model. θ represents the Neural Network parameter.

To take an action, the model predicts the reward value for all actions. The reward values are important for all decision-making strategies. Different actions might be taken depending on what decision-making strategy is used. Figures 2 and 3 show the flow chart of how Epsilon Greedy and Thompson Sampling play the coin flip game.

Results

The reward value that is assigned to match or wrong visual objects is an important factor in models' convergence. In case of a small difference between the reward of a match and wrong, all models fail to learn bias and fail to show any decision-making strategy. With a proper choice of reward value, all the models show that they are capable of learning the bias in less than 200 trials. Both ACT-R and PyIBL are capable of implementing matching and maximizing decision-making strategies. Figure 4 shows the effect of temperature on the decision-making strategy of the PyIBL model. For figure 4.a, the temperature value was set to one and for figure 4.b, the temperature was set to 14. For small values of temperature, the PyIBL model will choose the maximizing strategy. As the value of temperature increases, the decision-making strategy move towards matching. If the value of the temperature is set too high, the PyIBL agent will decide completely random (i.e., 50 percent of the time, the PyIBL model chooses head, and 50 percent of the time, it chooses tail).

Figure 5 shows the effect of temperature on the decision-making strategy of the ACT-R model. Figure 5.a displays the proportion of head and tail chosen by ACT-R if the

temperature is set to 3. In Figure 5.b, the temperature was set to 0.5.

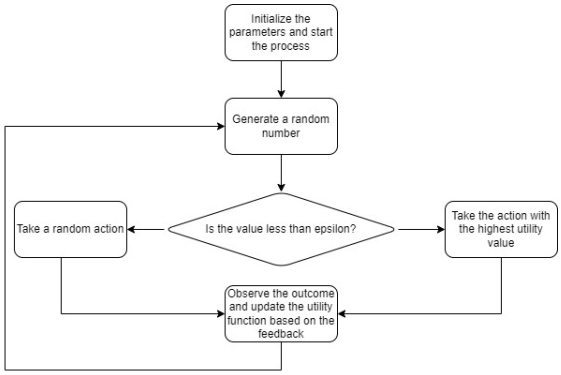

Figure 2: The flowchart for the Epsilon Greedy algorithm contains five processes and one conditional operation. In each step, with the probability of Epsilon, the model takes a random action. Otherwise, it will select the action with the highest predicted utility value.

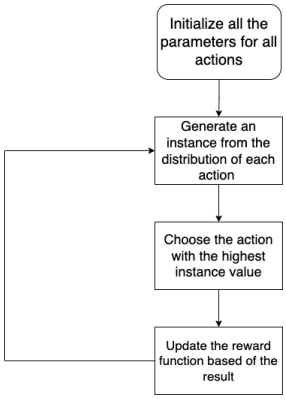

Figure 3: Thompson Sampling Flowchart contains four processes. The beta distribution is assigned to each choice. At the beginning of the training, all parameters are equal to 1. Meaning that the model assumes that parameters are all likely to generate an optimum result. At each step, samples are taken from each action distribution. The biggest sample determines what action should be taken. Then based on the result, the distribution of the action taken is updated.

Similar to PyIBL, smaller values of temperature will result in maximization and as the value of temperature increases, the randomness of choices will increase. ACT-R shows more sensitivity to the value of the temperature in comparison to PyIBL. Meaning smaller changes in the value of temperature in ACT-R will result in more noticeable shifts in decision-making strategy.

Figure 6 shows the Epsilon Greedy maximizes the utility by only taking actions with the highest reward. This is exactly what is expected from this model. The Deep Reinforcement Learning with Epsilon Greedy decision-making strategy is designed to maximize the reward. The experiment shows that the maximizing behavior of the model has started after the fourth trial (where the reward value of the head became larger than the tail). A bad sequence of random occurrences might result in the model taking the wrong action as the maximizing action and may not be able to recover.

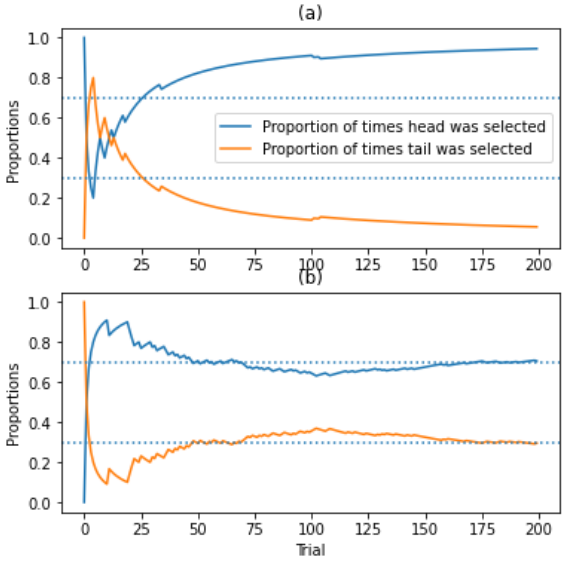

Figure 4: Probability of choosing Head (Blue) and Tail (Orange) over 200 trials by PyIBL in the case of (a) maximizing and (b) matching with different temperatures.

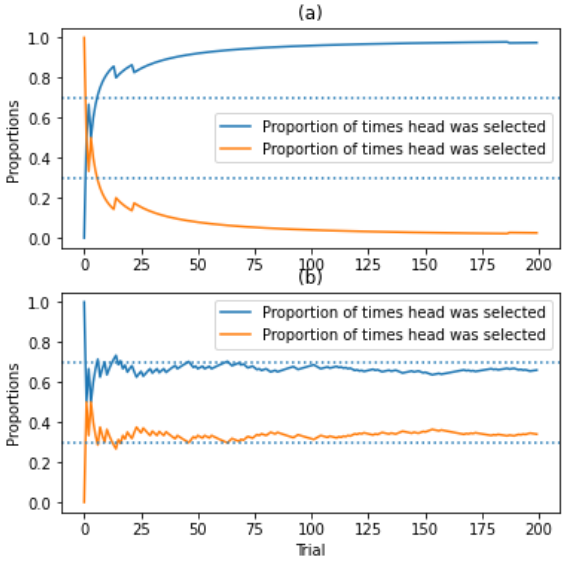

Figure 5: Probability of choosing Head (Blue) and Tail (Orange) over 200 trials by ACT-R in the case of (a) maximizing and matching (b) using different temperatures.

Figure 7 suggests that Thompson Sampling started with matching and then maximized after gaining confidence that the estimated reward value of the head is larger than the tail. Figure 8 shows the decision-making by Deep Reinforcement Learning with Boltzmann Exploration. With the right value of temperature, this model can imitate both matching and

maximizing behaviors. In summary, all the models that utilized the Boltzmann Equation in their action taking (decision-making) strategy (ACT-R, PyIBL and Deep Reinforcement Learning), are capable of both matching and maximizing. Epsilon Greedy decision-making strategy always results in maximizing. Thompson Sampling first matches the probability of the coin and when it is confident in the reward of the head is greater than the tail, it starts to maximize the reward by choosing heads.

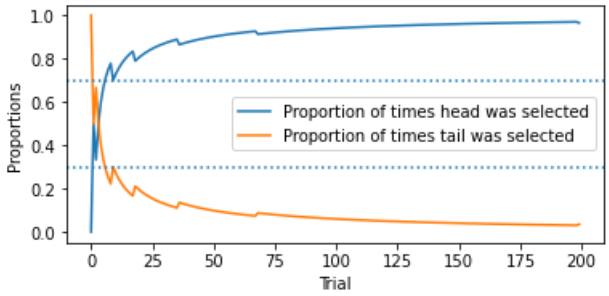

Figure 6: Probability of choosing Head (Blue) and Tail (Orange) over 200 trials by Deep Reinforcement Learning with Epsilon Greedy decision-making strategy.

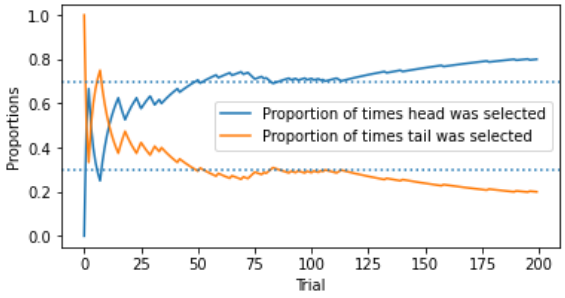

Figure 7. Probability of choosing Head (Blue) and Tail (Orange) over 200 trials by Reinforcement Learning with Thompson Sampling decision-making strategy.

Conclusion

We analyzed different models such as ACT-R, PyIBL, Reinforcement Learning with the Epsilon Greedy algorithm, Boltzmann Exploration, and Thompson Sampling decision-making strategies. We studied the models' capabilities to learn the bias and how they take an action. A web-based biased coin flip game was developed where models can interact and predict the next coin flip's result.

The outcome of the coin flip game (i.e., match or wrong visual objects) will be shown in the game environment. We introduced VisiTor, a Python-based tool that can facilitate the interaction between different models and task environments in any programming languages.

The models utilized two main strategies to choose what action to take. They can "Maximize," meaning they can select the choice they believe has the highest probability of success and maximize their outcome. Or they "Match" the probability of the actions. We showed that among the well-known cognitive architectures and algorithms, ACT-R, PyIBL and Deep Reinforcement Learning with Boltzmann Exploration are capable of imitating all decision-making strategies by setting the right set of parameters.

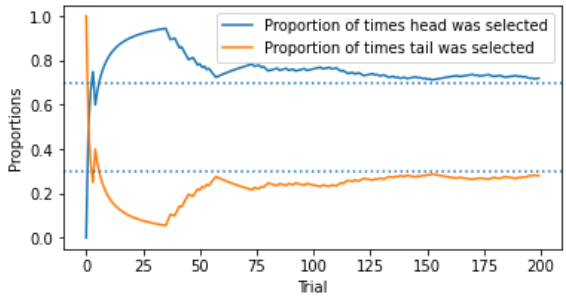

Figure 8: Probability of choosing Head (Blue) and Tail (Orange) over 200 trials by Deep Reinforcement Learning with Boltzmann Exploration.

Epsilon Greedy and Thompson Sampling tend to "Maximize" before the 200th trial. However, Thompson Sampling tends to "Match" at the beginning and then it will "Maximize" the reward. The behavioral studies in this area believe people are using the same set of strategies. However, which strategy is used in what situations is still a topic of conflict. A more systematic study needs to be conducted to show under what circumstances people tend to minimize or maximize. Based on the result, we will be able to see which model can simulate human behavior and under what circumstances.

Future Works

In order to determine which model is behaving closest to humans, a study needs to be conducted to analyze human behavior. Models that utilize the Boltzmann Equation in their decision-making strategy, can be tuned to Match or Maximize. Behavioral studies in this area indicate mixed results and may vary case by case. A systematic review is needed in this area to categorize the results and analyze the reason behind the mixed results that are reported by the studies previously done to analyze human behavior. This experiment needs to be conducted to determine if humans tend to match, maximize, or combination of both.

Also, currently, visual objects need to be predefined for VisiTor. A possible extension for VisiTor is to further extend its capabilities by having the model define the visual objects based on the human eye movement data. Users tend to pay closer attention to the visual objects they utilize to play. In the next version of VisiTor, we plan to have the model detect visual objects based on the eye-tracking data.

Acknowledgments

The Pennsylvania State University supports this work. We would like to thank Dan Bothell, Don Morrison, and Genevieve Gordon for their assistance.

References

Altmann, E. M., & Burns, B. D. (2005). Streak biases in decision making: Data and a memory model. *Cognitive Systems Research, 6*(1), 5-16.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38*(4), 341-380.

Brackbill, Y., & Bravos, A. (1962). Supplementary report: The utility of correctly predicting infrequent events. *Journal of Experimental Psychology, 64*(6), 648.

Cao, S., Ho, A., & He, J. (2018). Modeling and predicting mobile phone touchscreen transcription typing using an integrated cognitive architecture. *International Journal of Human–Computer Interaction, 34*(6), 544-556.

Cassimatis, N. L., Bello, P., & Langley, P. (2008). Ability, breadth, and parsimony in computational models of higher-order cognition. *Cognitive Science, 32*(8), 1304-1322.

Dabney, W., Ostrovski, G., & Barreto, A. (2020). Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*.

Derks, P. L., & Paclisanu, M. I. (1967). Simple strategies in binary prediction by children and adults. *Journal of Experimental Psychology, 73*(2), 278.

Erev, I., & Barron, G. (2005). On adaptation, maximization, and reinforcement learning among cognitive strategies. *Psychological Review, 112*(4), 912.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science, 27*(4), 591-635.

Gray, W. D., Schoelles, M. J., & Sims, C. R. (2005). Adapting to the task environment: Explorations in expected value. *Cognitive Systems Research, 6*(1), 27-40.

Guo, X. (2017). *Deep learning and reward design for reinforcement learning.*

Halbrügge, M. (2013). ACT-CV: Bridging the gap between cognitive models and the outer world. *Grundlagen und Anwendungen der Mensch-Maschine-Interaktion, 10*, 205-210.

Hope, R. M., Schoelles, M. J., & Gray, W. D. (2014). Simplifying the interaction between cognitive models and task environments with the JSON Network Interface. *Behavior Research Methods, 46*(4), 1007-1012.

Janssen, C. P., & Gray, W. D. (2012). When, what, and how much to reward in reinforcement learning-based models of cognition. *Cognitive Science, 36*(2), 333-358.

Lebiere, C. (1999). *Blending: An ACT-R mechanism for aggregate retrievals.* Paper presented at the Proceedings of the Sixth Annual ACT-R Workshop, George Mason University, Fairfax, VA, USA.

Lebiere, C., Gonzalez, C., & Martin, M. (2007). *Instance-based decision making model of repeated binary choice*. Retrieved from

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Masadeh, A. e., Wang, Z., & Kamal, A. E. (2018). Convergence-based exploration algorithm for reinforcement learning. *Electrical and Computer Engineering Technical Reports and White Papers, 1*.

Moran III, J. D., & McCullers, J. C. (1979). Reward and number of choices in children's probability learning: An attempt to reconcile conflicting findings. *Journal of Experimental Child Psychology, 27*(3), 527-532.

Plate, R. C., Fulvio, J. M., Shutts, K., Green, C. S., & Pollak, S. D. (2018). Probability learning: Changes in behavior across time and development. *Child development, 89*(1), 205-218.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2018). ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, e1488.

St. Amant, R., Riedel, M. O., Ritter, F. E., & Reifers, A. (2005). *Image processing in cognitive models with SegMan* (Vol. 4). Mahwah, NJ: Erlbaum.

Tehranchi, F., & Ritter, F. E. (2018a). *Modeling visual search in interactive graphic interfaces: Adding visual pattern matching algorithms to ACT-R.* Paper presented at the Proceedings of ICCM-2018-16th International Conference on Cognitive Modeling.

Tehranchi, F., & Ritter, F. E. (2018b). *Using Java to provide cognitive models with a more universal way to interact with graphical user interfaces.* Paper presented at the 2018 International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction and Behavior Representation in Modeling and Simulation, BRiMS 2018.

Tehranchi, F., & Ritter, F. E. (2020). *Extending JSegMan to Interact with a Biased Coin Task and a Spreadsheet Task.* Paper presented at the 17th International Conference on Cognitive Modeling, ICCM 2019.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika, 25*(3-4), 285-294.

Tokic, M. (2010). *Adaptive ε-greedy exploration in reinforcement learning based on value differences.* Paper presented at the Annual Conference on Artificial Intelligence.

Vulkan, N. (2000). An economist's perspective on probability matching. *Journal of economic surveys, 14*(1), 101-118.

Zhu, J. (2018). *Probabilistic Machine Learning: Models, Algorithms and a Programming Library.* Paper presented at the IJCAI.