Behavioral Analysis of Decision Making in Monkeys Using LSTM

Margaret Kimbis

DATA 612: Deep Learning

Dr. Luntao Liu

University of Maryland College Park

August 13th 2024

Problem Statement

Humans and animals live in a dynamic environment, therefore we are constantly adapting our decision making process based on past experiences as well as the decisions made by others. If we are able to understand the neural and behavioral basis behind normal decision making, it will provide insight into how to better understand disorders such as schizophrenia, autism, and addiction where decision making is impaired. Consequently, researchers will be able to develop better treatment and diagnostic measures for handling these mental and developmental disorders.

Background

Monkeys were trained to play a game against a computer in an oculomotor choice task called Matching pennies. Matching Pennies is a zero-sum game in which two intelligent players are tasked with choosing one of two pennies. If they both choose the same one, player 1 wins, otherwise player 2 wins. The premise for winning is based on the principle of Nash Equilibrium. Nash equilibrium is where no players are able to increase their reward by changing their strategy individually. For example, if player one decides to only select the right target for the remainder of the game, they will be unable to increase their reward because the potential for reward is dependent on both players.

In the context of the data obtained by the Lee Lab, Nash equilibrium is reached when the monkey makes both choices (left target, and right target) with equal probability. The monkey will

play against a computer that has been programmed with three different algorithms (algorithm 0, 1, 2) that exploit different aspects of the animal's choice and reward history. Algorithm 0 uses no previous choice history, while algorithm 1 and 2 do use the animals choice history to try and shape their moves to reach Nash Equilibrium. In order to reach this equilibrium, the monkey must make his successive choices independently from the choices of both players in from previous trials

Motivation for Deep Learning

There are a variety of reasons as to why using a deep learning algorithm is beneficial for exploring this data set. This data set contains large amounts of data. There are 279,669 trials split between the 3 algorithms and 3 monkeys, therefore a deep learning algorithm may be better equipped to handle this extensive data set compared to a simpler model. Using a deep learning model also allows us to predict player moves, identify complex behavioral patterns, and uncover strategies more accurately and efficiently. For example, a deep learning model can determine how the animal's choice varies after winning or losing a trial. RNNs, LSTMs, and Transformers are all examples of deep learning algorithms capable of accomplishing these goals, however LSTMs will be used for this data set due to its capabilities for handling time series data. This comprehensive analysis can provide deeper insights into the player's decision-making process which allows us to better understand normal decision making.

Methods

The data is obtained from the Daeyeol Lee lab at Johns Hopkins with permission from my Principal Investigator. Three male rhesus macaque monkeys were trained to complete this task, and are referred to by an ID letter. Monkeys were seated in a primate chair and faced a

computer monitor where they were presented with the task, and eye calibration was performed to ensure that the monkey was able to make their choice. At the beginning of each trial, the animal was required to fixate on a yellow square, and after 0.5 seconds, two identical green disks were presented. One on the left of the fixation target and one on the right. The left versus right targets serve the same purpose as the typical “heads and tails” in the Matching Pennies task. After another 0.5 seconds the fixation square was extinguished and the animal was required to produce a saccadic eye movement towards one of the targets for 0.5 seconds in order to make their choice selection. After the monkey chose a disk, a red ring was displayed around the target that the computer had selected, the animal wins the trial and is rewarded with juice if it selected the same target as the computer. The data for the monkey's selection, algorithm, ID, trial number, and whether or not reward was received was saved and converted into a data table.

Model Selection

LSTM (Long Short Term Memory) was chosen due to its ability to handle sequence modeling problems. This allows for previous choices made by the monkey to aid in the prediction of future ones.

Preprocessing

The original data is in a txt file, so the first step was to convert it into a csv file and remove any missing values. Next, the data was separated based on the algorithm and only the trials that were conducted under algorithm 0 were kept. The reason being that for algorithm 0, the monkey plays against the computer who chooses its target with a probability of $p = 0.5$. This means that this algorithm provides insight into the original decision making process of the

monkey, before their choices are altered due to prior game history. For example, algorithm 2 is based on the monkey's performance in algorithm 0 and 1. In these games, if the monkey chooses the left target in algorithm 1 with $p = 0.8$, in algorithm 2, the computer will choose the left target with $p = 0.2$.

After specifying the algorithm, the monkey of interest was then selected. The model was trained using Monkey ID: 13, and tested for generalization on Monkey ID: 112, and 18. Once data filtering was complete, the Monkey ID and Algorithm column were excluded due to its interference with the features' shape later on during model training.

Various packages were utilized during this project to facilitate the training, testing and validation of the model. Tensorflow and Keras were imported to develop and train the LSTM model. Pandas, Numpy and Matplotlib were implemented for data manipulation and visualization. Finally, Sci-kit learn was used for model evaluation including accuracy, precision, recall, F1-score, and time series cross validation.

Move Prediction

The data for this task was collected over numerous days, consequently it is important to group the data by session. This ensures that a decision that the monkey made on Friday will not be used to predict a choice the monkey made on Monday. This was achieved by using the groupby function from pandas. Next, the feature and target selection occurred using indexing. Features included the "Reward", "Monkey Choice", and "Computer Choice" columns since all were relevant for prediction. The target was specified as "Monkey Choice", since the aim of this model is to predict future moves based on prior ones. The shape of the features were printed, yielding a vector of (5326, 3). This ensured that the features had successfully been selected.

Next, a sequence function was developed that grouped the data by session, extracts features and targets, and returns a numpy array of X and y. After the sequence was developed, the length was specified. The number of prior moves to be used for prediction was set to 5. The sequences for X and y were created using the custom function, then using the train_test_split function, X_train, X_test, y_train, and y_test were created. The function specified an 80:20 training and testing split, and random_state was used to randomly separate the data into training and testing.

Model parameters were defined, putting timesteps equal to the number of prior moves, and the number of output dimensions were set to 200. X_train and y_train were reshaped to size (None, 1) to allow the model to run. Following this, the LSTM model was defined. First, the input layer was added, and the shape was specified since X_train and y_train were reshaped. An LSTM layer was added using ReLu activation, followed by a dropout layer. I tested different values for this dropout layer, and after hyperparameter tuning, I set the value back to zero. Two dense layers using ReLu activation were added, alongside two more dropout layers with a probability of 0.5. The final output layer used a sigmoid activation function due to its usefulness for RNN models. An adam optimizer was used with a value of 0.007, and the model was created mapping the inputs to output. The model was compiled using a binary cross entropy loss function, and accuracy was specified as the metric of choice. Finally the model was trained using the .fit function, over 50 epochs with a validation split of 0.2. After training was complete, the model resulted in accuracy = 0.76, loss =0.46, validation accuracy = 0.70, and validation loss: 0.65. The model was then tested on Monkey ID: 112 (accuracy = 0.77, loss =0.46, validation accuracy = 0.71, and validation loss: 0.60) , and Monkey ID: 18 (accuracy = 0.99, loss =0.05, validation accuracy = 0.99, and validation loss: 0.08) for generalization. This implies that The

model generalizes decently well, for other monkeys, but performs extremely well for Monkey 18. However this may be indicative of overfitting for Monkey 18.

Behavior After Reward

Using the model I created above. I can look into different aspects of the monkey's behavior during the game. One area of interest is how reward affects move choice. First, I defined a custom function that calculates the likelihood that the monkey will repeat the same move after reward. This function extracts data from the reward column index and finds instances where the monkey received reward. The function loops through indices where the monkey receives reward, and then checks the next two choices to see if they had been repeated. Numpy was used to calculate the total instances of repeated choices after reward, as well as the total number of rewards the monkey received. This was then calculated as a percentage for Monkey 13 (Total rewards: 2646, Count of repeated choices after rewards: 1656, Percentage of repeated choices: 62.59%), Monkey 18 (Total rewards: 832, Count of repeated choices after rewards: 822, Percentage of repeated choices: 98.80%) and Monkey 112 (Total rewards: 2159, Count of repeated choices after rewards: 1514, Percentage of repeated choices: 70.13%). These percentages were then plotted as a bar plot using matplotlib in *Figure 1.1, Figure 1.2, and Figure 1.3*.

These results indicate that the monkeys tend to repeat choices after receiving reward. Despite these results, it's important to consider biases towards specific targets. For example, Monkey 13 chose the right target 70% of the time, and Monkey 112 chose the left target 70% of the time. Monkey 18 however, chose the right target 90% o the time, indicating a significant bias

towards the right target. While decision making may be influenced by reward, a more comprehensive analysis would need to be conducted in order to remove the confounding variable of the monkey's bias towards a specific target.

Figure 1.1

Percentage of Repeated Choices for Monkey 13

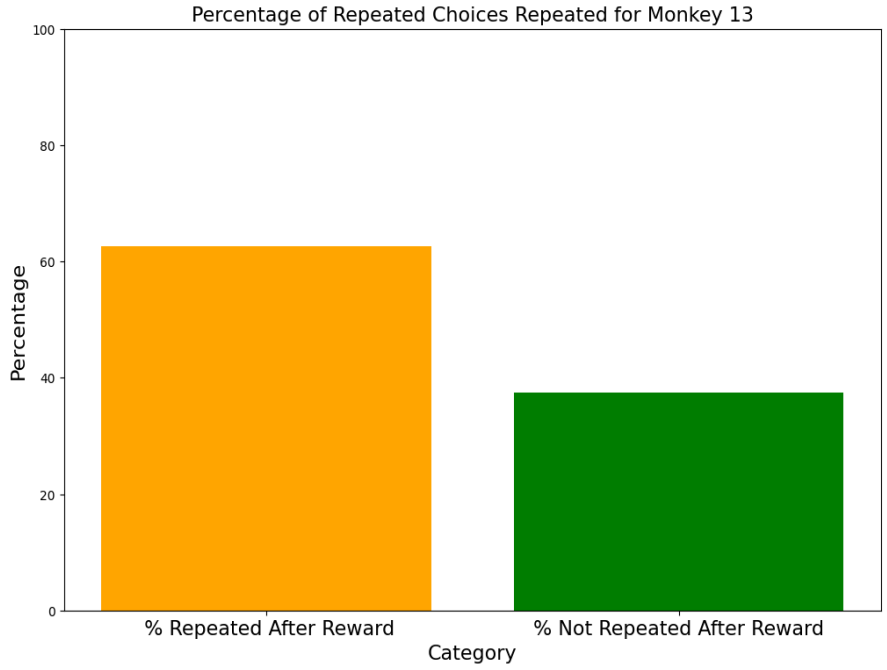


Figure 1.2

Percentage of Repeated Choices Repeated for Monkey 112

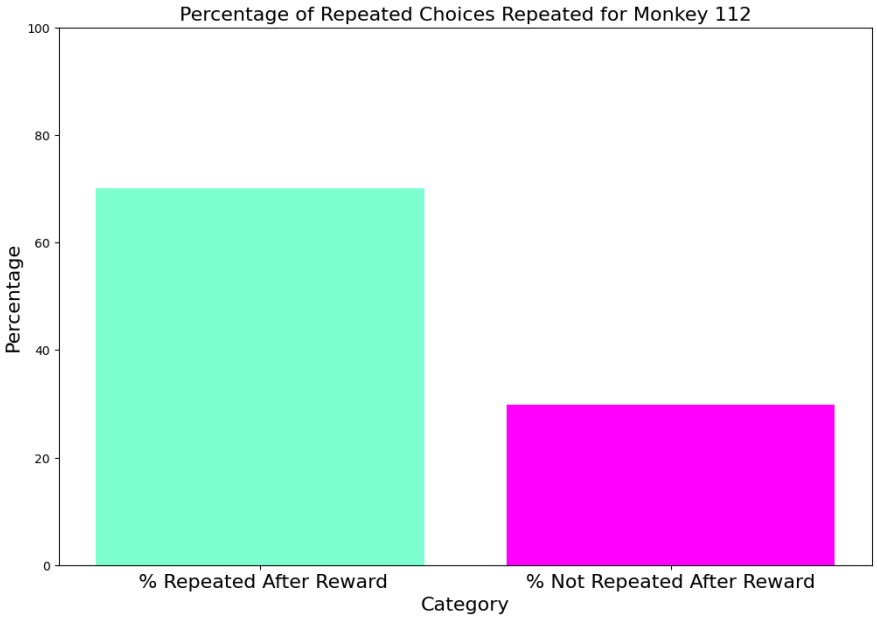
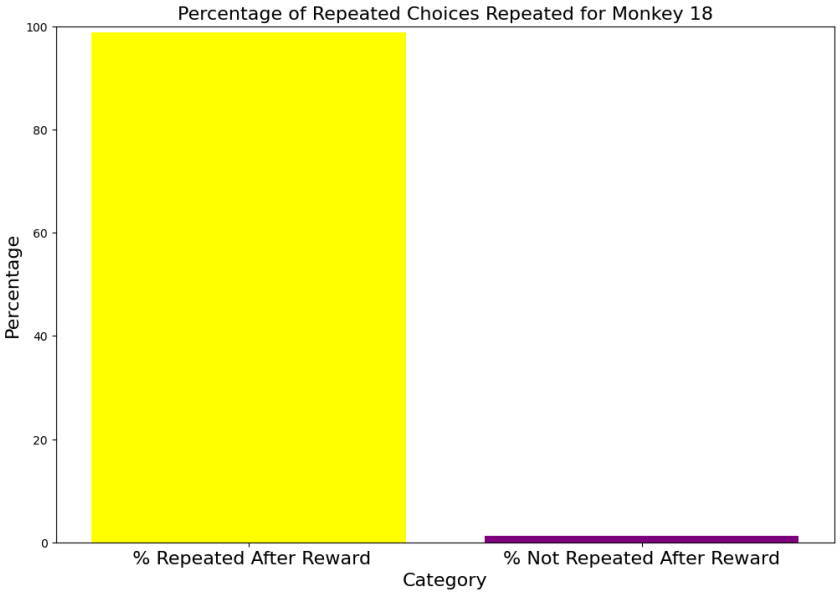


Figure 1.3

Percentage of Repeated Choices Repeated for Monkey 18



Model Evaluation

Time Series Cross Validation was chosen to evaluate the model. The choice for this validation technique is due to its ability to preserve the temporal order of the data, which is essential for game play prediction. Tensorflow, Keras, and Sci-kit learn were used for this technique. The data was split into 3 folds using the TimeSeriesSplit function. I Reinitialized the model before each fold of Time Series Cross-Validation using the same hyperparameters as the original model. The data was re-initialized before each fold to avoid errors. Errors were stored for each split, and then averaged to give the mean squared error (Average MSE = 0.18). MSE calculates the squared difference of the actual versus predicted player moves, so an MSE of .18 indicates that the model is performing well. However since this model is essentially a binary classification (predicting either the left or right target) there is still room for improvement to obtain a lower MSE.

Aside from the Time Series Cross Validation, accuracy, precision, recall, and F1- score were also used to evaluate the model's performance. The accuracy obtained during the model evaluation is different from the accuracy that was received during training. This score reflects the model's accuracy on never before seen data. The accuracy (accuracy = 0.71) score obtained is decent, however there is still room for improvement. Ideally the model would have obtained an accuracy of 0.80. The precision (precision = 0.75) value, while relatively high, indicates that numerous false positives are still being produced by the model. This model is producing a high enough recall (recall = 0.83) implying that the model is capturing most of the positives. Finally, the F1-score (F1-score = 0.79) shows that there is balance between precision and recall. Overall the models performance is satisfactory however there is still room to improve the models performance.

Challenges and Future Directions

During the development of the model, I encountered challenges that required troubleshooting. Hyperparameter tuning was particularly difficult, as some changes led to high accuracy but low validation accuracy, which was a sign for overfitting. Additionally, adjusting output dimensions had caused validation accuracy to exceed training accuracy, signaling that the model was not fitting well to the data. Determining the most effective activation functions was another challenge, and I struggled with regularization techniques. Another aspect I struggled with was determining the optimal number of moves to use for prediction. Although I initially experimented with a large number, a lower number was deemed more logical, as it is unlikely that the monkey remembered a sequence of prior moves up to 15 decisions long.

Despite hyperparameter tuning extensively, I still was receiving a very low accuracy during training. I realized that I had not been grouping by session number. I also discovered that my model included two unnecessary features—Monkey ID and Algorithm number—which were significantly reducing the model's accuracy. After correcting these two errors, model accuracy significantly increased.

In the future I would like to use a similar approach on another task. One of the other projects in our lab involves training monkeys to play four in a row with a computer while we collect neural and behavioral data. I would like to deploy a similar model to help predict player moves, detect biases, and uncover player strategies in this more complex game. Additionally, I would like to work towards tuning the hyperparameters more to receive greater accuracy, precision, recall, F-1 score, and a lower MSE. Finally, it could be interesting to try using another approach like GRU to see if it yields better results.

Overall, this model utilized deep learning to predict decision making in monkeys, as well as provided insight into biases, and determine how reward influences behavior. Deep learning is a powerful tool that can be used to help expand the field of psychology and neuroscience. If we are able to understand normal decision making in primates, we will in turn be able to better understand abnormal decision making. Using deep learning algorithms for this purpose has the ability to not only help improve our current diagnostic and treatment methods, but also to make diagnostic tools more reliable.

References

Orabi, A. H., Buddhitha, P., Orabi, M. H., & Inkpen, D. (2018, June). Deep learning for depression detection of twitter users. In Proceedings of the fifth workshop on computational linguistics and clinical psychology: from keyboard to clinic (pp. 88-97).