

Behavioral Analysis of Matching Pennies Task in Rhesus Macaques using LSTM

Maggie Kimbis
DATA 612



Table of Contents



01 Problem Statement

02 Model Selection and Model
Evaluation

03 Code Demo

04 Results

05 Challenges

Problem Statement

We live in a dynamic environment, therefore we are constantly adapting our decision making process based on past experiences and the decisions of others

If we can 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.

Background

- Monkeys were trained to play a game against a computer in an oculomotor free 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. In this case the monkey will choose left target or right target
- The monkey will play against a computer that has been programmed with three different algorithms (0,1,2) that exploited different aspects of the animals choice and reward history
 - My model utilizes algorithm 0
- I want to develop a model to predict player moves as well as to gain insight into behavioral patterns.
 - For example: how does reward affect choice?

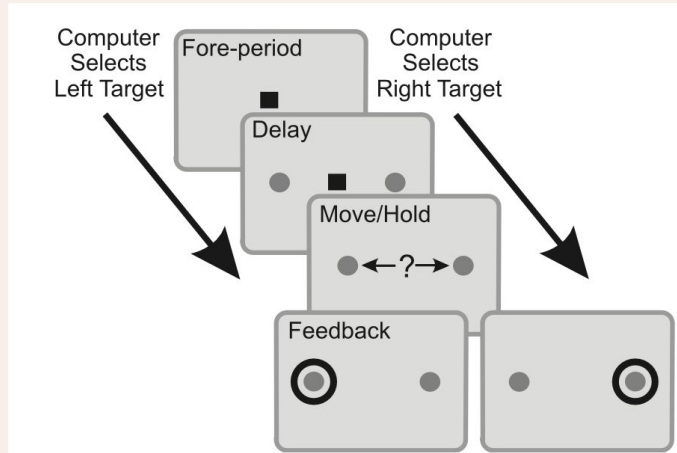


Fig. 1. Spatial layout and temporal sequence of the free-choice task.

Model Selection Overview

- **LSTM (Long Short Term Memory)** will be used due to its ability to handle sequence modeling problems. I used this model to predict player moves and assess behavioral changes after reward.
- Steps:
 - **Preprocessing:**
 - Convert txt file to csv
 - Handle missing values
 - Separate the data based on the algorithm, Monkey ID, and keep relevant columns
 - **Move Prediction:**
 - Grouped data by sessions
 - Extract features and columns
 - Specify 5 previous moves to be used for prediction
 - Create Sequences
 - Define the model
 - Train the model
 - Compare predictions to real data
 - Look at the predictions
 - Generalization using the two other monkeys
 - **Behavior After Reward**
 - Develop function that extracts data from the model to determine the frequency of repeated choices after receiving reward
 - Plot the percentage using matplotlib

Model

What does the data look like?

| | Session | Computer Choice | Monkey Choice | Reward |
|--------|---------|-----------------|---------------|--------|
| 201362 | 1 | 1 | 0 | 0 |
| 201363 | 1 | 1 | 0 | 0 |
| 201364 | 1 | 0 | 1 | 0 |
| 201365 | 1 | 1 | 1 | 1 |
| 201366 | 1 | 0 | 1 | 0 |

What parameters are used?

Model: "functional"

| Layer (type) | Output Shape | Param # |
|--------------------------|--------------|---------|
| input_layer (InputLayer) | (None, 5, 3) | 0 |
| lstm (LSTM) | (None, 200) | 163,200 |
| dropout (Dropout) | (None, 200) | 0 |
| dense_2 (Dense) | (None, 1) | 201 |

Total params: 163,401 (638.29 KB)
Trainable params: 163,401 (638.29 KB)
Non-trainable params: 0 (0.00 B)

LSTM Model Hyperparameters:


- Output Dimensions: 200 neurons
- Input: ReLu activation
- Dense: ReLu activation
- Output: Sigmoid activation (better for RNN models, and binary classification)
- Learning Rate: Adam optimizer 0.007
- Compiled the model using binary cross entropy for the loss function which is typical for binary classification, and accuracy was used for the ,metrics
- Trained with 50 epochs, batch size of 32, and validation split of 20%




Model Evaluation Overview

- **Time Series Cross Validation:** This validation technique was chosen due to its ability to honor the temporal order of the data, which is essential for game play

Steps:

- Split the data using Time Series Split function
 - Check the shape
 - Re-initialize the model
 - Train the model
 - Evaluate and store errors
 - Calculate the average mean squared error
 - **Accuracy**
 - Calculate the accuracy of the model using the accuracy_score function from sklearn
 - **Precision, Recall, F-1 Score**
 - Ensure the predicted versus actual values are the same size
 - Calculate Precision, Recall, and F-1 Score using the precision_recall_fscore_support function from sklearn
- 



Code Demo



Results

Project Training Performance

Monkey ID: 112

- Model accuracy: 0.7727
- Loss: 0.4647
- Validation accuracy: 0.7098
- Validation loss: 0.5977

Monkey ID: 18

- Model accuracy: 0.99
- Loss: 0.05
- Validation accuracy: 0.99
- Validation loss: 0.08

Monkey ID: 13

- Model accuracy: 0.7661
- Loss: 0.4606
- Validation accuracy: 0.6968
- Validation loss: 0.6494

What does this say about Generalization ability?

- The model generalizes decently well, for other monkeys, but performs extremely well for Monkey 18. This can indicate overfitting for Monkey 18

Moves

First 10 moves Monkey ID: 112

| Actual Choice | Predicted Choice |
|---------------|------------------|
| Left Target | Left Target |
| Left Target | Left Target |
| Right Target | Right Target |
| Right Target | Left Target |
| Left Target | Left Target |
| Right Target | Right Target |
| Left Target | Left Target |
| Left Target | Right Target |
| Left Target | Left Target |
| Right Target | Right Target |

Moves

First 10 moves Monkey ID: 18

| Actual Choice | Predicted Choice |
|---------------|------------------|
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Left Target | Left Target |
| Right Target | Right Target |

Moves

First 10 moves Monkey ID: 13

| Actual Choice | Predicted Choice |
|---------------|------------------|
| Right Target | Right Target |
| Left Target | Right Target |
| Right Target | Left Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Right Target | Right Target |
| Left Target | Left Target |
| Right Target | Left Target |
| Right Target | Right Target |
| Left Target | Right Target |

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. These results indicate that the Monkey's tend to repeat choices after receiving a reward. However, It is important to consider potential biases towards a certain target before drawing conclusions.

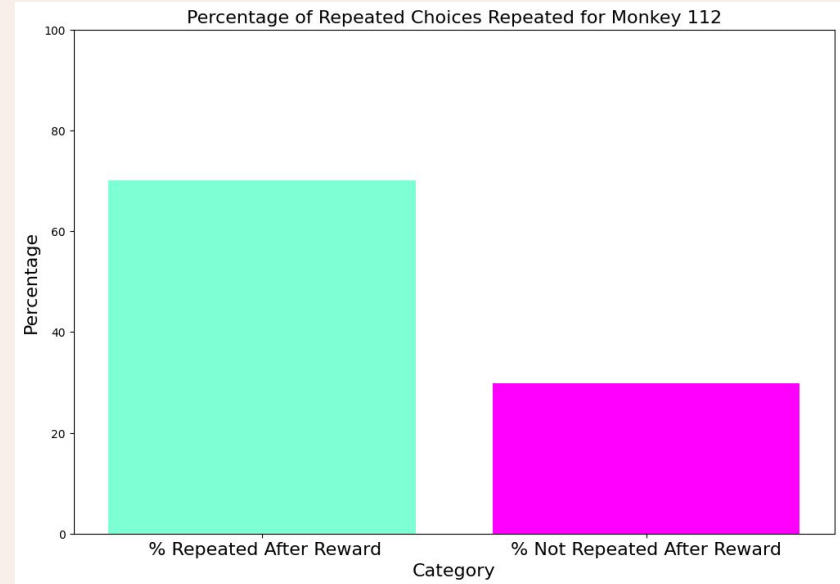
Monkey ID: 112:

Total rewards: 2159

Count of repeated choices after rewards: 1514

Percentage of repeated choices: 70.13%

Monkey 112 chooses the Left target 70% of the time.



Behavior After Reward

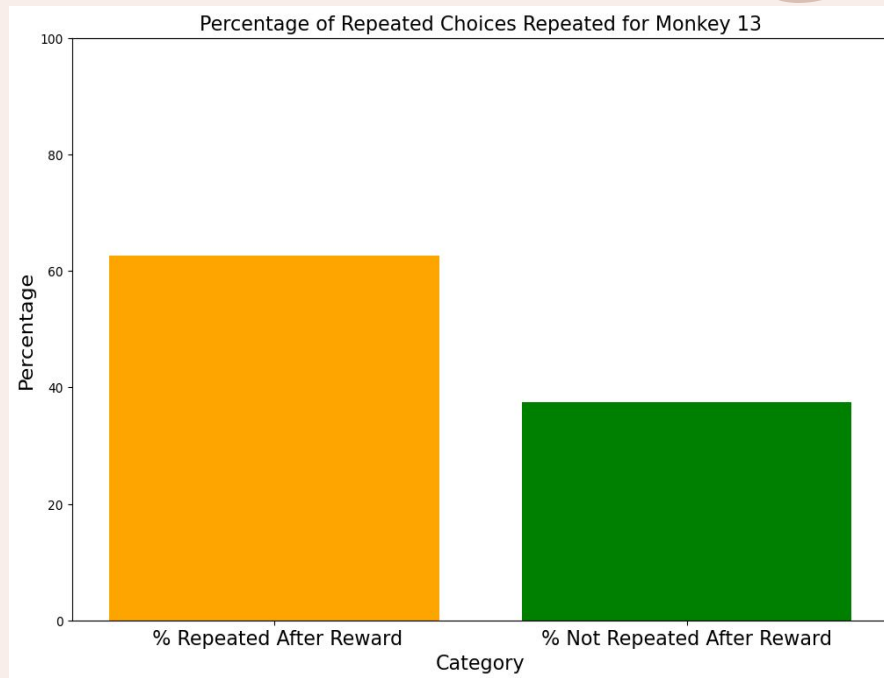
Monkey ID: 13:

Total rewards: 2646

Count of repeated choices after rewards: 1656

Percentage of repeated choices: 62.59%

The monkey chose the right target 70% of the time.



Behavior After Reward

Monkey ID: 18:

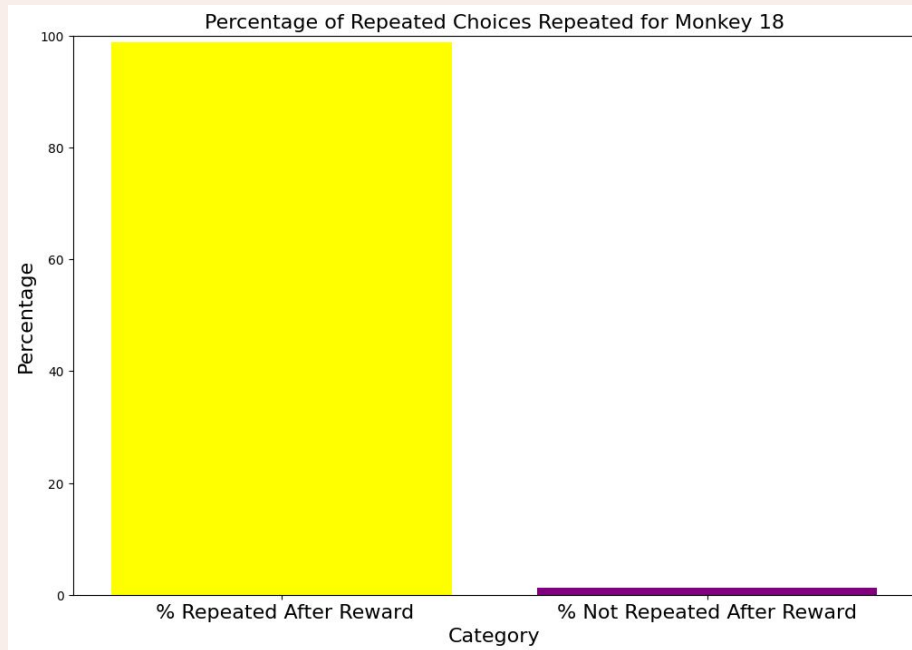
Total rewards: 832

Count of repeated choices after rewards: 822

Percentage of repeated choices: 98.80%

As you can see, this monkey has a very high Percentage of repeated rewards, but if you recall the table of predicted versus actual moves, the monkey has an extreme bias towards the right target. This monkey chose the right target 90% of the time.

This likely means that despite this high number, the monkey likely does not change his behavior depending on reward. It is also interesting to note that for this monkey's first trial he chose the right target and received reward.



Model Evaluation

Time Series Cross Validation:

- The data was split into 3 folds
- Cross Validation used the same hyperparameters as the original model
- Errors were stored for each split, and then averaged to give the mean squared error. I found an average mean squared error of **0.18** across all folds.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

The equation is annotated with blue boxes and labels: 'Mean' above the fraction, 'Error' above the difference term, and 'Squared' above the exponent.

- **MSE calculates the squared difference of the actual versus predicted player moves**

- An MSE of .18 indicates that the model is performing well, however since it is essentially a binary classification (left or right target) there is still room for improvement

Model Evaluation

Accuracy, Precision, Recall, and F1-Score:

Accuracy: 0.71: 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. An accuracy of 0.71 is decent, however there is still room for improvement. Ideally the model would have obtained an accuracy of 0.80

Model Evaluation

Accuracy, Precision, Recall, and F1-Score:

- **Precision: 0.75**
 - This is a decent precision, however we are still getting numerous false positives
- **Recall: 0.83**
 - This is a good recall, indicating that the model is capturing most of the positives
- **F1-Score: 0.79**
 - There is balance between precision and recall. This score indicates overall the models performance is satisfactory however there is still room for improvement

Challenges and Future Directions

Challenges

- I faced many challenges during model development and evaluation
- Trouble tuning hyperparameters
 - Some changes caused a good accuracy, with a very low validation accuracy
 - Originally, changing output dimensions resulted in no change, or caused the validation accuracy to become better than the training accuracy indicating that the model was not fitting well to the data
 - Trouble determining which activation functions were best
 - Adding and removing L2 regularization
 - Determining the number of moves to use for prediction, at first I tried a high number, but logically a lower number made more sense because it is unlikely that the monkey remembered a large sequence of player moves
- I realized I was not grouping by session number
- I realized that my model included two extra features: Monkey ID, and Algorithm number which was reducing my accuracy significantly
- Error messages, and figuring out where in the code there was a problem.
- Constantly restarting run time, due to code not running properly

Future Directions

- In the future I would like to use a similar approach on another task. One of the other projects in our lab involves training monkey's 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
- Work towards tuning the hyperparameters more to receive greater accuracy
- Try using another approach like GRU to see if it yields better results