Dopamine Simulation System by Akhil Kaza

This report aims to dissect the production of the Dopamine Simulation System. The fusion of my deep passion for Artificial Intelligence and my profound curiosity for Psychology and Neuroscience has entailed me into the intersection of computer science and biology, inspiring the development of this model.

The program incorporates the hypothesis that the brain is of a video game addict while leveraging a reinforcement learning strategy. It measures the brain's craving for dopamine when engaging with specific stimuli. Given the hypothesis, reward probability counters vary for each stimulus, heavily influenced by video game addiction. All probabilities are assumed based on the hypothesis; there has been no data or research to support such probabilities being derived. Positive feedback is akin to a reward prediction error, whereby the reward is greater than expected, likewise does negative feedback, whereby the reward is lower than expected.

Probabilities and expected rewards dynamically change, learning from previous feedback. Positive feedback influences an increase in the expected reward variable, as does negative feedback cause a decrease in expected reward.

The ultimate objective of the program is to implement Thompson Sampling and the Rescorla-Wagner model to simulate how the brain prioritizes maximum release of dopamine in decision-making through the lens of a video game addict’s brain. It demonstrates the pleasure of instant dopamine actions such as video gaming and provides reasoning for why these activities are so addictive.

I begin by importing NumPy.

An integral part of my program is the NumPy module. This package on Python has been extremely effective in mitigating Mathematical logic in the program, initialising matrices and establishing reward counters, expected rewards and aiding the execution of beta distribution.

After this, I name three different actions: socialising, gaming, and exercise and initialise each a reward probability, and predefine the number of iterations the program shall execute. This was set at 50,000 to eradicate erroneous results from determining incorrect outcomes.

Leveraging the NumPy module, I produced a 50,000 × 3 zero matrix, corresponding with the number of iterations and stimuli. Since this a zero matrix, all values are predefined to 0.

Visual guide:

A screenshot of a computer code

Description automatically generated

(50,000 repeated rows)

Using the NumPy module, a random number between 0 and 1 is generated. If the random value is lower than the probability value for a respective stimulus (e.g. gaming – 0.43 probability) then the element representing this iteration becomes 1, representing positive feedback. This strategy, also known as random threshold comparison, is an effective way to leverage probabilities when allocating positive reward feedback. Since there can only be a set number of values lower than or equal to respective stimulus probability between 0 and 1, this leverages the probability values when allocating positive feedback. For example, there can only be 43 values smaller than or equal 0.43 between 0 and 1, therefore a 43% chance of positive feedback.

Then, I establish the reward counters for each stimulus – alpha and beta. Alpha represents positive rewards, and beta represents negative rewards. Alongside this I initialise expected rewards (V) for each stimulus, initially setting all three at 0.5. Additionally, I predefine a learning rate for the Rescorla-Wagner model of 0.4. It is important that the learning rate of any AI model is set at the optimal amount – an abundance could entail into overlearning, which includes erroneous data, however a learning rate set at a minimum makes a stubborn model which won’t learn anything. Thus, I regard 0.4 amongst the optimal learning rates.

To define the best model per iteration I applied the Thompson Sampling algorithm. I use beta distribution to collect a sample for each stimulus, incrementing each positive and negative count by 1 to avoid any errors. On the condition the sample (random\_beta) is greater than the current largest reward (max\_random), the sample stored as random\_beta becomes the new max\_random value, and the stimulus of such is selected to undergo Rescorla-Wagner calculations.

Introducing the Rescorla-Wagner model, I derive the actual reward of the selected stimulus, before calculating the prediction error. Simply, I subtract the expected reward from the actual reward. Now, further leveraging the RW model, I update the expected reward, which is the product of actual reward and the previously calculated prediction error.

Rescorla-Wagner model equation:

ΔV=α⋅β⋅(λ−V)

WHERE:

* ΔV: The change in the associative strength between a conditioned stimulus (CS) and an unconditioned stimulus (US) during a single trial.
* α - Alpha: The strength of the conditioned stimulus (CS). It is a constant between 0 and 1.
* β - Beta: The strength of the unconditioned stimulus (US). It is also a constant between 0 and 1.
* λ - Lambda: The maximum possible associative strength that the unconditioned stimulus (US) can support. It represents the magnitude of the unconditioned response (UR) that can be elicited by the US.
* VVV: The current associative strength between the conditioned stimulus (CS) and the unconditioned stimulus (US).

In simple terms:

* λ−V: This term represents the prediction error.
* α⋅β: This is a constant governing how much learning happens in each trial, depending on the characteristics of the stimuli.

Dotting back to the alpha and beta reward counts, I conduct a Bayesian update, whereby the alpha reward count is incremented by 1 if the selected element becomes 1, symbolising a reward being reached. Otherwise, the beta reward count is incremented by 1 to represent that a reward has not been reached.

Upon completion of 50,000 iterations, the program outputs the positive feedback per stimulus. Additionally, leveraging the powerful module NumPy further, the best stimulus is derived, and is outputted to the user.

Despite the promise of my program, there are a numerous optimisation I should consider to implement, regarding accuracy and validity.

A key optimisation to consider is the depth of scientific research. Despite the rigour of my model, the probabilities are assumed on a hypothesis. These probabilities are foundational mechanisms of the model, thus making it weak. This can be strengthened through incorporating scientific research to help determine probabilities, providing stronger underpinning of the model. Additionally, leveraging scientific research would increase the validity of the model significantly, in comparison to estimated probabilities based on context instead of research.

An observation of my model was the

To prove this, I conducted 5 test runs and recorded the results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test NO: | Socialising reward | Gaming reward | Exercise reward | Reward ratio |
| 1 | 0 | 1976 | 8 | 0: 271 :1 |
| 2 | 5 | 2127 | 1 | 5: 2127 :1 |
| 3 | 3 | 2154 | 0 | 3: 2154 :0 |
| 4 | 3 | 2154 | 2 | 3: 2154 :2 |
| 5 | 7 | 2141 | 1 | 7: 2141 :1 |

Average reward ratio – 3.6 : 2150.4 : 2.4

Based on this observation, it can be concluded that my model effectively simulates the brain of a video game addict and demonstrates how rewarding gaming can be to the brain. The degree of positive reward reaped by gaming, compared to other activities like exercise and reading, illustrates the power of instant gratification that gaming provides. This continuous positive feedback loop creates a reinforcement mechanism, making it difficult for individuals to break away from such activities.

The tests show that frequent dopamine rewards lead to stronger associations between the stimulus (gaming) and the reward, reinforcing the behaviour over time. This insight aligns with real-world addiction, where the brain prioritizes behaviours that trigger higher dopamine release, making it harder to shift focus toward less immediately rewarding activities like exercise or reading. Ultimately, the simulation provides a clear understanding of how addiction to highly rewarding stimuli, such as gaming, can become deeply ingrained in the brain, explaining why overcoming such addictions poses significant challenges.

Key terms:

Conditioned Stimulus (CS) – Stimulus whereby rewards, and expectation are influenced by previous experiences.

Unconditioned Stimulus (US) – Stimulus whereby rewards, and expectation are not influenced by previous experiences.

Associative Strength – Learning power

Rescorla-Wagner model – Mathematical model leveraged for reinforcement learning strategy

Bayesian update – Facilitates incremental changes in reward counters - alpha and beta.

Alpha – Positive reward counter

Beta – Negative reward counter

Beta distribution – Models probabilities and proportions, useful in Bayesian tasks.

Thompson sampling – Reinforcement algorithm implemented to mitigate decision making through balancing exploitation and exploration.