Homework #4

Daniel Dager

Introduction

In this report, we will be examining the application of the drift-diffusion model and recurrent neural networks to two-alternative forced-choice tasks.

The drift-diffusion model is a mathematical model that describes the decision-making process of an agent. The model is based on the idea that the agent accumulates evidence over time until a decision threshold is reached. Since it is a model that can be simulated over time, it can be used to generate both the decision and the reaction time of the agent, which allows for a further dimension of comparison with behavioral data.

Recurrent neural networks are a class of artificial neural networks that have connections between nodes that form a directed cycle. As opposed to feedforward neural networks, which have connections that only go in one direction, recurrent neural networks have connections that allow information to be passed from one node to another and back again. These networks are used to model the dynamics of a system and can also be used to predict the behavior of the agent over time.

Finally, the two-alternative forced-choice task is a task in which the agent is presented with two options and must choose one. This task is commonly used in psychology to study decision-making and reaction times. The task can be manipulated by altering the degree of evidence in favor of one option over the other, or by increasing the overall difficulty of the task by introducing stochastic noise.

1.1

We begin by giving equation for the drift-diffusion model:

$$\frac{dx(t)}{dt} = (I_A - I_B) + \sigma\eta(t) \tag{1}$$

Here, x is the decision variable, which accumulates in the direction of options A or B. I_A and I_B are the decision inputs, which can be interpreted as the evidence for options A and B, respectively. Finally, $\eta(t)$ is the noise term sampled from a Gaussian distribution with zero mean and unit variance, and σ is a scalar applied to the noise term to control its magnitude.

The decision is made when the decision variable x reaches the threshold μ for A or $-\mu$ for B, and the reaction time is the time it takes for the decision variable to reach the threshold.

We now simulate the model 10 times using the Euler method, for 10000 time steps at $dt = 0.1$, with the following parameters: $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, and $\mu = 2$. We will simulate the model for 100 trials and plot the decision variable $x(t)$ for each trial:

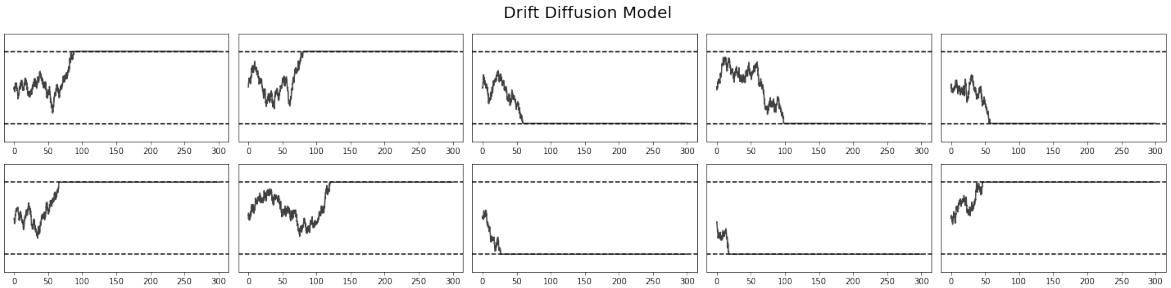


Figure 1: The dotted lines represent the decision thresholds μ and $-\mu$.

1.2

Now we will run the same simulation 1000 times and keep track of the decision counts for each trial, including the number of times that the model makes "no decision" (i.e., the decision variable does not reach the threshold within the simulation time). We will then plot the histogram of the decision counts for each trial:

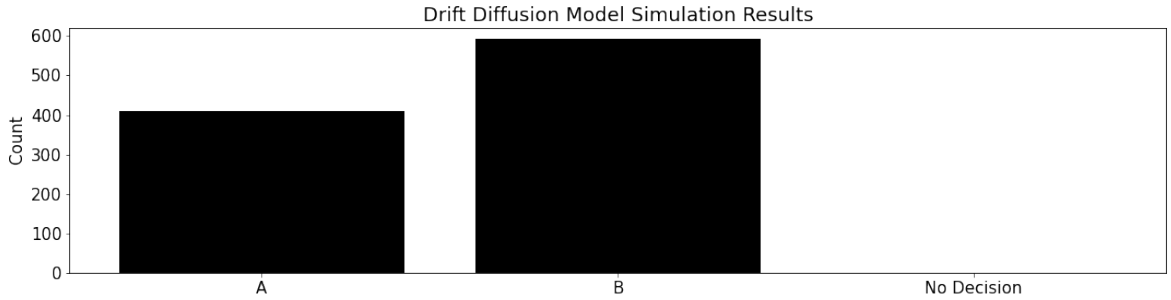
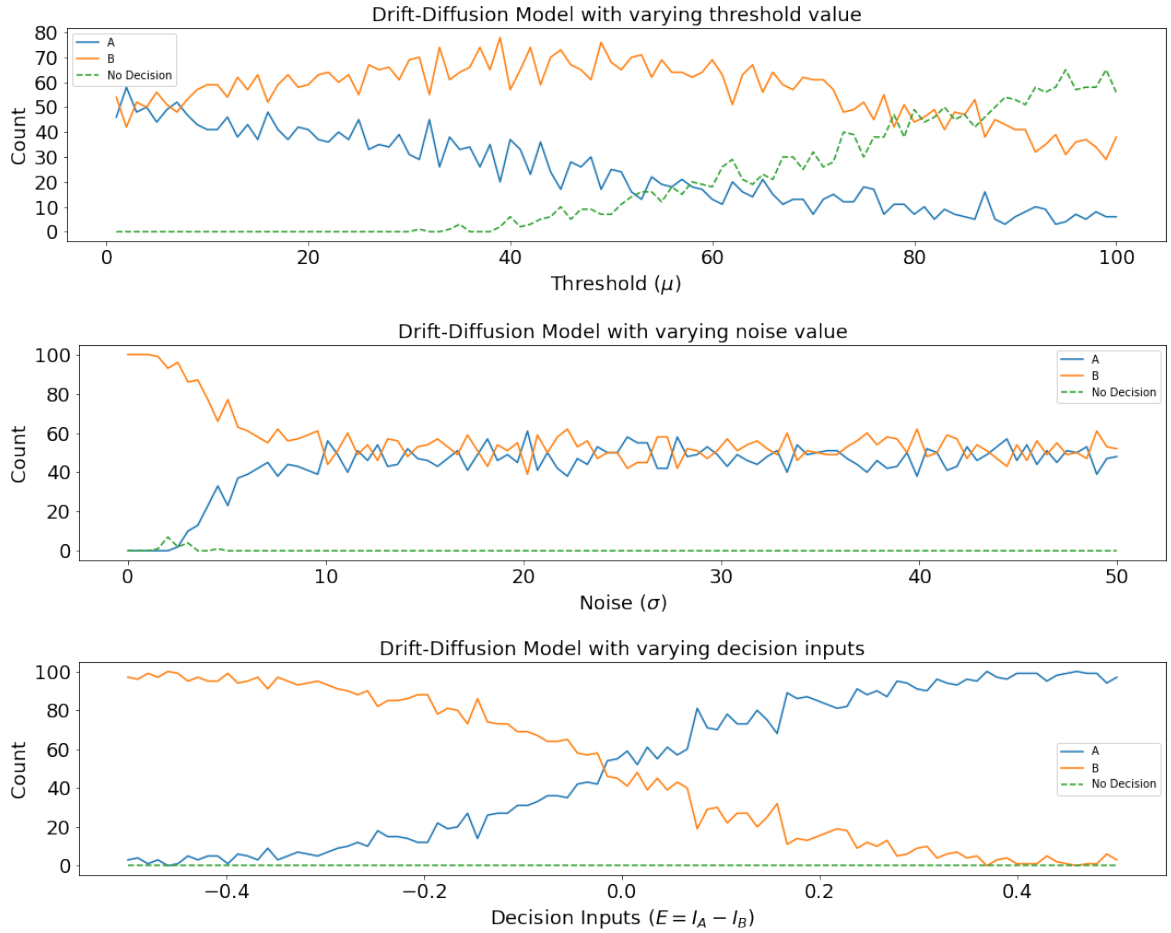


Figure 2: The parameters of the model favor option B over option A, and the model never fails to make a decision.

1.3

In this section, we will perform three sets of simulations of the drift-diffusion model, each time varying one of the parameters while fixing the others. This will allow us to observe the effect of each parameter on the decision-making process.



When we vary the threshold value μ, we can observe two main effects. When μ is small, the decision variable reaches the threshold more quickly, which makes the model more succeptible to noise, reducing the effect in the difference between the decision inputs, and the model is also much less likely to fail to make a decision.

On the other hand, when μ is large, the decision variable takes longer to reach the threshold, which makes the model less succeptible to noise, increasing the effect of the difference between the decision inputs, and the model also becomes much more likely to fail to make a decision.

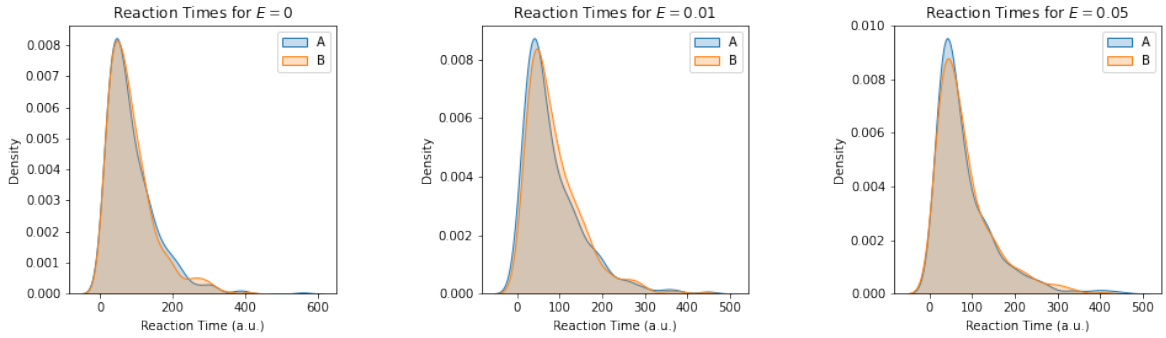
Now, when we vary the noise term σ, we can observe that the model becomes less responsive to the difference in the decision inputs as σ increases. It quickly approaches a 50-50 chance of choosing either option, thus becoming purely stochastic. The model also becomes extremely unlikely to fail to make a decision as σ increases.

Finally, when we vary the difference between the decision inputs $E = I_A - I_B$, we observe that the model goes from being very likely to choose option B to being very likely to choose option A as E increases. All the while, the model is extremely unlikely to fail to make a decision.

Overall, we can characterize the effects of the parameters on the model as follows. The threshold value μ is negatively correlated with the speed of the decision-making process, the likelihood of making a decision, and the influence of the noise term, and is positively correlated with the influence of the decision inputs. The noise term σ is negatively correlated with the influence of the decision inputs, and is positively correlated likelihood of making a decision. The difference between the decision inputs E controls the bias of the model.

1.4

In our last exercise regarding the drift-diffusion model, we plot the reaction time distributions for both decisions given varying differences in decision inputs:



What we observe from these plots is that as the difference between the decision inputs increases, the reaction time distributions for both decisions become more separated. This is because the decision variable accumulates evidence in favor of one option over the other, and the larger the difference between the decision inputs, the faster the decision variable reaches the threshold for that option. This results in shorter reaction times for the option with the larger decision input and longer reaction times for the option with the smaller decision input.

This is reflected in the fact that the distribution of reaction times for option A have a higher peak, indicating that there is a concentration of reaction times around a certain value. We also see a slightly longer tail for the reaction times of option B, indicating that there are some trials in which the decision variable takes longer to reach the threshold for option B than for option A.

2.1

In this part of the report, we will focus on the application of recurrent neural networks to two-alternative forced-choice tasks. Our first order of business is to generate a dataset of 1000 trials, 500 for training, and 500 for testing.

We again employ a noise term σ that scales a sampling from a Gaussian distribution with zero mean and unit variance. We also introduce a new notion, velocities, which have a similar purpose to the drift-diffusion model's decision inputs. These velocities indicate the direction and speed of the evidence in the stimulus presented to the agent.

In this 2AFC task, the agent can decide between -1 and 1, and their decision is correct if the sign of the decision matches the sign of the stimulus, which is determined by the velocity. The possible velocity values for the training set and the test set are $[-5, -4, -3, -2, -1, 1, 2, 3, 4, 5]$. The possible values for the noise term in the training set are $[0, 0.1, 0.05]$. For the test set they are $[0.5, 1]$.

To generate the input data for the dataset, we sample the velocities and noise terms for each trial, and then generate the input data by adding the noise term to the velocity at each time step. We then generate the target data by setting the target to 1 if the sign of the input data is positive, and -1 if the sign is negative. Below we plot the inputs and targets for a few trials in both sets:

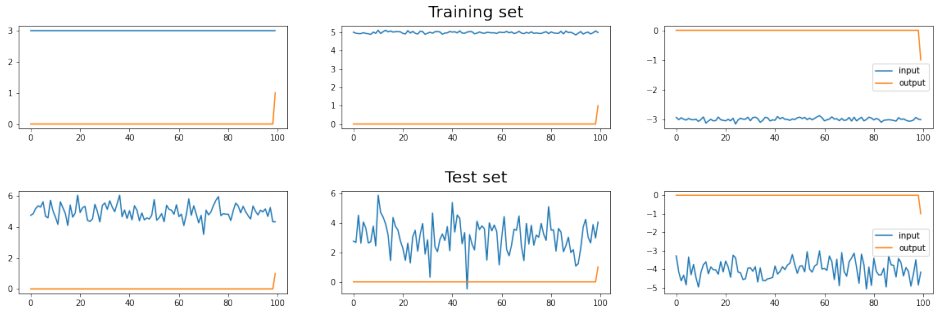


Figure 3: Note that the target data only assumes values of 1 and -1 at the last time step.

We can cleary see the influence of the noise term on the test data, as the input data is much more noisy than the training data. In effect, this serves as a means of preventing us from presuming that are model is better than it actually is due to overfitting, because the test data is inherently more difficult to predict than the training data.

2.2

In order to train our RNN on this dataset, we need to define a loss function that measures the difference between the predicted output and the target output. This loss value will then be used to update the weights of the network in the direction that minimizes the loss:

$$\mathcal{L} = \frac{1}{P} \sum_{p,t,l} M_{p,t,l} \left(z_{p,t,l} - z^*_{p,t,l} \right)^2 \tag{2}$$

This may look complicated, but when we break it down, we see that the loss function is simply the mean squared error between the predicted output $z_{p,t,l}$ and the target output $z^*_{p,t,l}$, where p is the trial index, t is the time step index, and l is the output index.

Most importantly, the $M_{p,t,l}$ term is a mask that is 1 if the time step is the last time step of the trial, and 0 otherwise. This is because we only care about the output at the last time step of the trial, as this is the decision that the agent makes. The loss is then averaged over P many trials.

2.3

We now train the RNN on the training set using the Adam optimizer with a learning rate of 0.0001 for 30 epochs. We plot the loss over time below, observing that the loss is below 0.1 after 27 epochs:

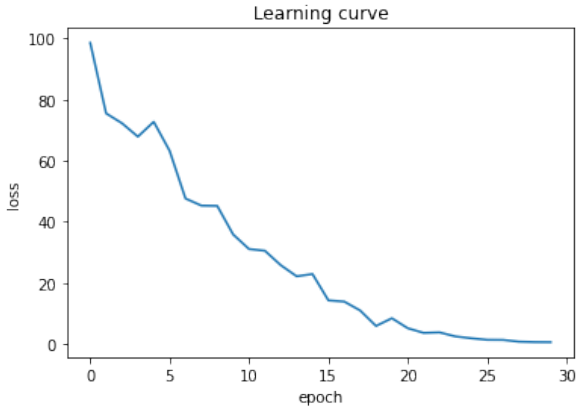


Figure 4: In this case, the loss decreases steadily over the entire training loop.

2.4

Next, we define our accuracy metric as the percentage of trials in which the agent makes the correct decision. To facilitate this calculation, use the sign function to convert the output of the RNN to a decision:

$$\text{Accuracy} = \frac{1}{P}\sum_{p}\left(sign(z_p) - z_p^*\right) \tag{3}$$

We then calculate the accuracy of the RNN on both the training and test sets, and find that the accuracy on the training set is 100%, while the accuracy on the test set is 91.4%. As predicted, this indicates that the model is probably overfitting to the training data, but the difficulty of the test data prevents the model from achieving perfect accuracy.

2.5

We have access to the trajectories of the outputs of the RNN for each trial, which describes how the decision variable of the agent evolves over time. We plot the trajectories for all of the trials in the dataset below, color coding them according to the true direction of the stimulus:

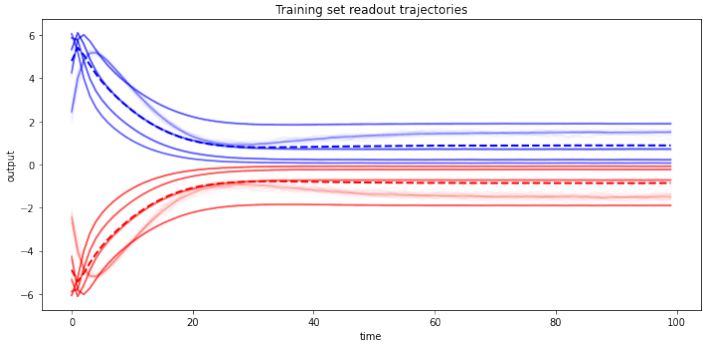


Figure 5: The dotted lines represent the mean trajectories of the trials for each stimulus direction.

Figure 6: Note that the trajectories are much more noisy for the test data than for the training data.

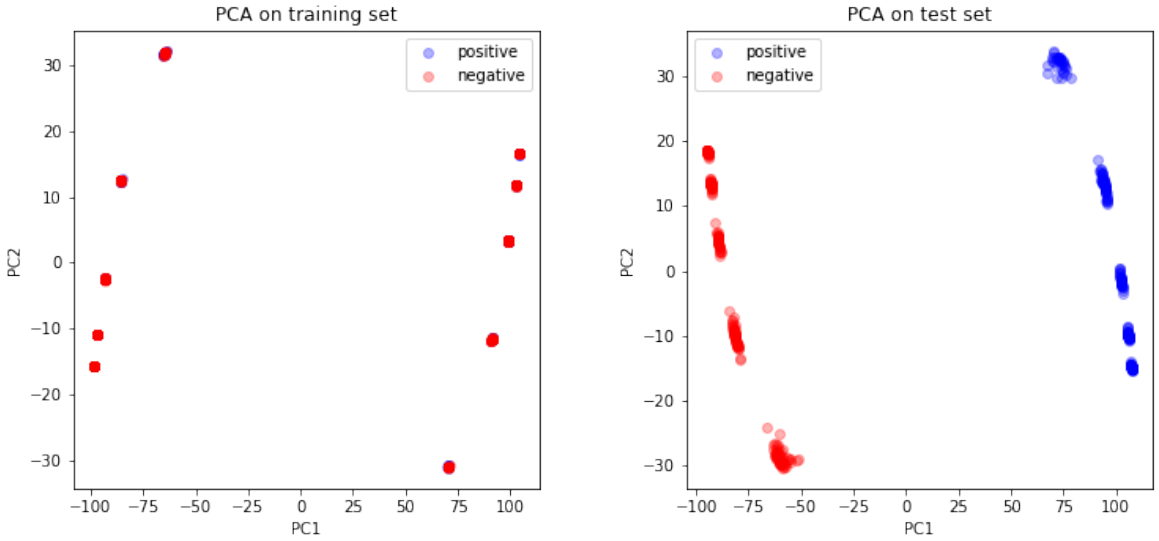
In the training data, which we know was generated using lower values for the noise term, we can see that the trajectories are much smoother and more consistent. They converge upon the correct velocity values, and the mean trajectories converge on the correct decision values.

The trajectories for the test data, on the other hand, are much more noisy and less consistent. This is consistent with our expectations, as the test data was generated using higher values for the noise term. However, we observe that the mean trajectories still converge on the correct decision values, indicating that the model is able to generalize to the test data despite the increased noise.

2.6

Finally, we conduct PCA analysis on the population activity of the "neurons" of the RNN. This may sound strange, but the idea is that each node of the RNN represents a neuron in the agent. In our definition of the RNN, we apply `torch.tanh()` to our input vector $\boldsymbol{x}$ to produce a "firing rate". The input vector $\boldsymbol{x}$ is imagined as the membrane voltage of the neuron.

The idea behind applying PCA to the population activity of the neurons is this the activity actually exists on a lower dimensional space than that which the shape of the data suggests. By applying PCA, we can reduce the dimensionality of the data and visualize the population activity in a more interpretable way:



Before performing PCA, we first had to collect the population activity of the neurons. First we took the array of all of the outputs of the RNN for all of the trials in the dataset, which is equivalent to an array of the membrane voltages of each neuron at each time step for each trial. We then reshaped this array to $(P, T \times N)$, where P is the number of trials, T is the number of time steps, and N is the number of neurons in the RNN. We then applied the `torch.tanh()` to turn the membrane voltages into firing rates, and then finally applied PCA to the population activity.

The PCA analysis reveals that the population activity of the neurons is in fact concentrated in a lower dimensional space than the shape of the data suggests. This is because the neurons are correlated with each other, and the activity of one neuron can be predicted from the activity of the other neurons. Furthermore, we observe that how the neurons are organized in the space most likely corresponds to their response to different stimuli. Looking particularly at the test data, we see that the neurons are organized in a way that separates the different stimulus directions, and the different velocities, indicating that the neurons are encoding the stimulus information in their activity.

Conclusion

In this report, we explored the application of the drift-diffusion model and recurrent neural networks to two-alternative forced-choice tasks. We saw how the both models could be used to simulate the decision-making process of an agent over time. The drift-diffusion model can be used to generate the decision and reaction times of an agent, giving us a valuable tool for comparing cognitive models to behavior data. It also captures a foundational notion of decision-making, that of accumulating evidence over time until a decision threshold is reached. This theory underpins many other models of decision-making, specifically those that conceive of cognitive systems as fundamentally probabilistic.

The recurrent neural network, rather than being just a way of modeling the outputs of a decision process, also models, albeit in an extremely simple way, the actual neuronal dynamics that may be present in the brain of a subject faced with a decision task. The PCA analysis of the population activity of the neurons of the RNN revealed that the neurons organize their activity in a way that separates the different stimulus directions, serving a model for how neurons may encode information about stimuli in their firing rates and mutual activity. This is a powerful demonstration of the potential of RNNs to both model the dynamics of a system and to predict the behavior.