MOD201: Decision Making

Ciprian Bangu

May 19 2024

Contents

1 Introduction

Introduce the 2AFC task and the idea of decision making in the brain.

2 The Drift Diffusion Model

The decision making process in the 2AFC task described above can be approximated by 'drift-diffusion' model. This model is given by the following equation:

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

where x is a decision variable, I_A and I_B are the neural signals to make decision A and B respectively, σ is the noise level, and $\eta(t)$ is a Gaussian white noise process with zero mean and unit variance. The decision of the subject, either A or B is represented by the crossing of the decision variable of some threshold: μ in the case the subject decides for outcome A, or $-\mu$ in the case the subject decides for outcome B. The differnce $I_A - I_B$ can be thought of as the evidence, E, the subject has for deciding in eithe direction. If this difference is positive, the subject is more likely to decide in favor of A, and if it is negative, the subject is more likely to decide in favor of B.

The idea is that this model captures the process of the integration by neurons in the Lateral Intraparietal area (LIP) of signals of motion direction in either the A or B direction as reported by direction sensitive neurons in the Medio-Temporal visual cortex.

2.1 Simulating the DDM

To get a sense of the dynamics of the stochasitc process produced by the model, we can simulate the model and plot the decision variable $x(t)$ as a function of time. Figure 1 below illustrates 10 trials of the model, given the initial conditions: $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, and possible timesteps = 10000.

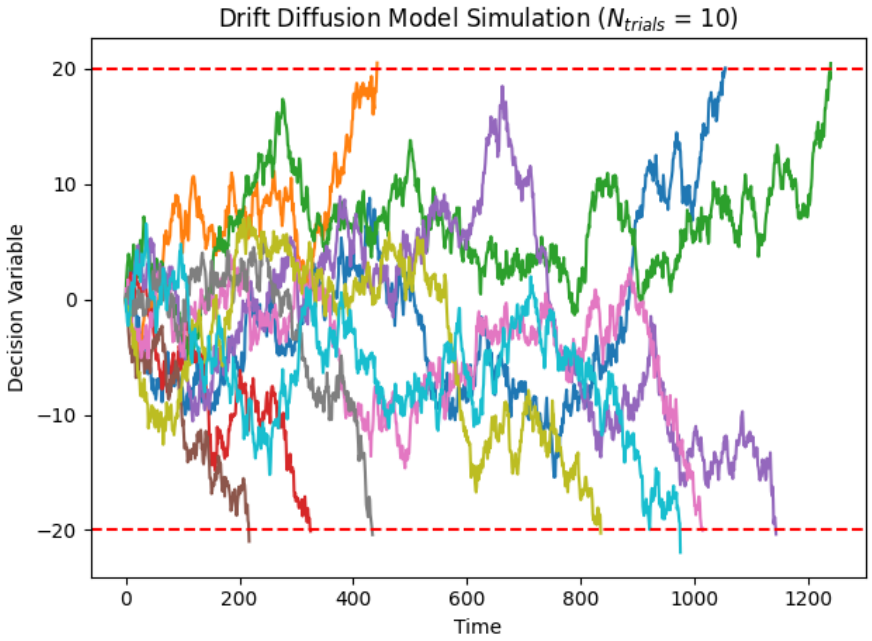

Figure 1: Simulation of the Drift Diffusion Model with $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000. Different trials are indicated by different colors. The dashed red lines indicate the decision variable thresholds.

As we can see from figure 1, the decision variable evolves randomly over time thanks to the gaussian noise in the model. Moreover, while it does not evolve in the same way every time, the decision variable eventually meets or crosses one of the thresholds, ending the trial, for every trial. **IS THERE ANYTHING ELSE TO SAY HERE?**

Recall that we introduced the notion of evidence E as the difference between the signals I_A and I_B. In Figure 1, we see that there are 3 trials where the decision variable crosses the threshold for A and 7 trials where

the decision variable crosses the threshold for B. This is consistent with the fact that $I_B > I_A$. We can see the impact of this difference in evidence for one decision over another by looking at the number of trials that end in favor of one decision over the other, over more simulations of the model.

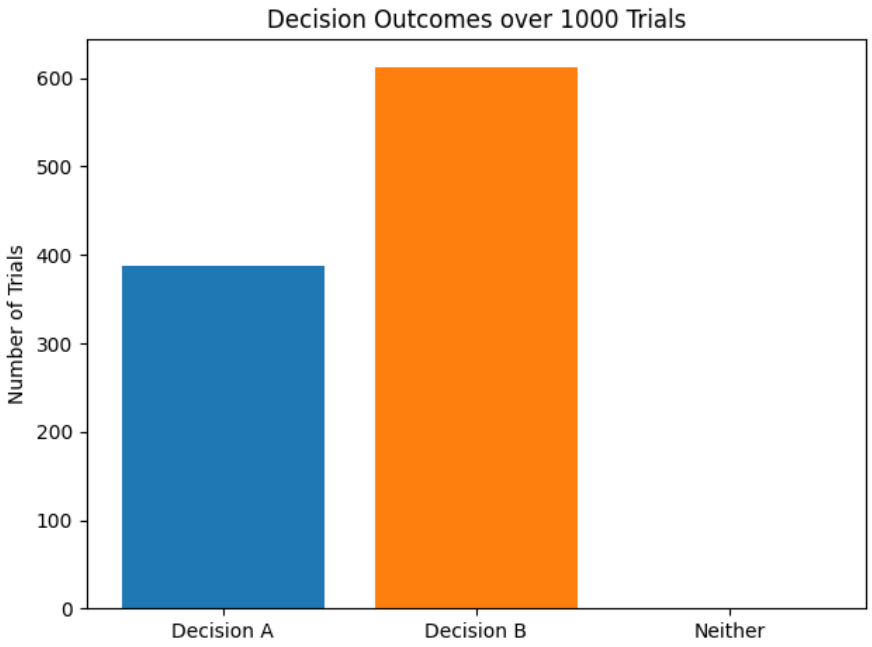

Figure 2: Outcomes of the Drift diffusion model over 1000 simulations. $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$. The number of trials that end in favor of decision A is plotted in blue, and the number of trials that end in favor of decision B is plotted in orange.

Figure 2 above illustrates the proportion of outcomes for each decision after 1000 simulations of the model. From Figure 2, we can see that the split between decisions A and B is about 60:40 in favor of decision B. Thus, for this model, even a small difference in the evidence in favor of one decision over another can lead to a significant difference in the number of trials that end in favor of one decision over the other. In this case a 0.05 evidence advantage for B led to a 50% increase in the number of trials that ended in favor of B over A.

2.2 Varying other parameters

We have seen how a small differential in the evidence can impact the distribution of outcomes for the model. However, there are more parameters we can vary to get differential behavior. In this section, we will vary μ σ and E to observe how the distribution of outcomes changes.

ADD THE THING ABOUT SUBJECT INTERNAL AND EXTERNAL PARAMETERS

2.2.1 Varying μ

First, let us vary the threshold parameter of the model, μ. Recall, the threshold parameter, and its negative, are the values at which, once x reaches (or crosses) either of them, the simulation stops and the subject is said to have decided in favor of the corresponding decision. In Figure 3 below, we keep the same intial conditions as before, but vary μ from 1 to 100. Given the small amount of simulations, the difference in the number of trials that end in favor of B rather than A is initially small, and in some cases negative. However, as μ increases, there begins to be a clear advantage for decision B over A. This is because the evidence imablance combined with the higher threshold makes it less likely that x will reach or cross μ on any given trial. From the figure, we see that this decrease in A decisions is linear as μ increases. However, the relationship between increases in μ (or, really, increases in the *magnitude* of μ, since it is negative for decision B) is parabolic. Initially, increases in μ have a positive imapct on the number of trials that end in favor of B. However, this effect flattens out around $\mu = 50$, after which the number of trials that end in favor of B begins to decrease (although it remains higher than the number in favor of A). This is because the model is less likely to cross any threshold as μ increases, since it only has a finite amount of time to reach μ, which will take longer as μ increases.

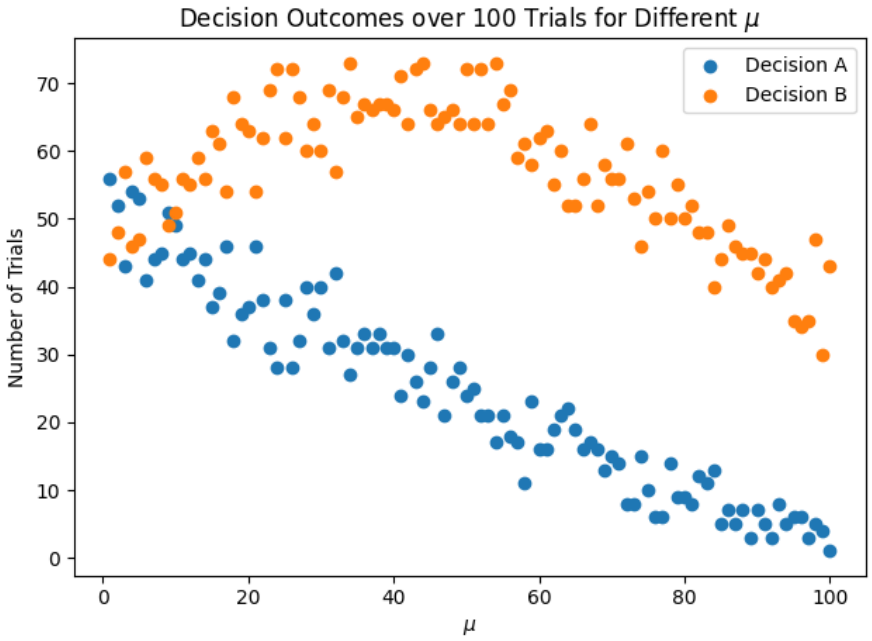

Figure 3: Number of trials that end in favor of decision A and B over 100 simulations of the DDM, varying μ from 1 to 100. $I_A = 0.95$, $I_B = 1$, $\sigma = 7$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000.

2.2.2 Varying σ

Now, let us vary σ, or the magnitude of the noise in the model. In Figure 4 below, we keep the same initial conditions as before, but vary σ from 0 to 50, simulating the model for 100 trials.

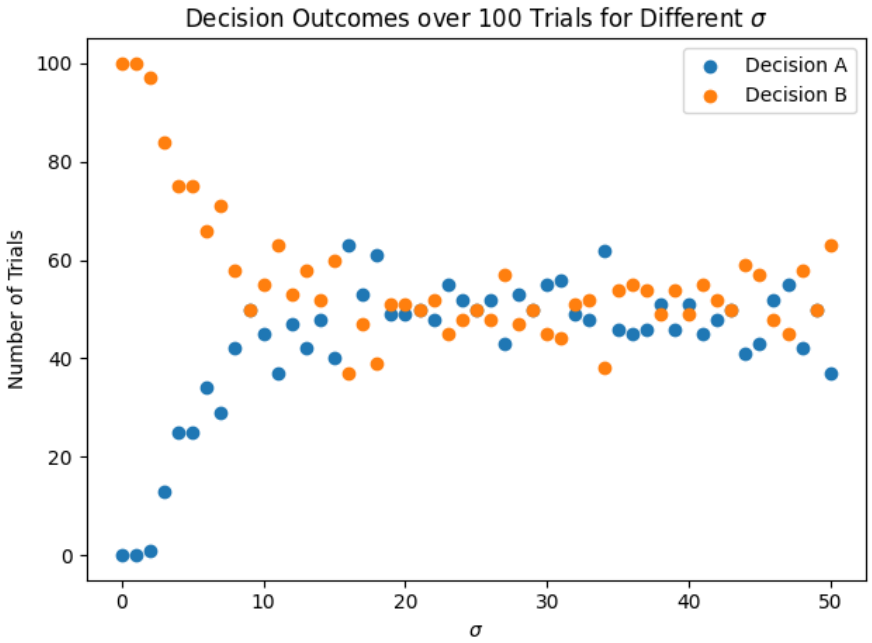

Figure 4: Decision outcomes over 100 simulations of the ddm for varying values of σ. $I_A = 0.95$, $I_B = 1$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps = 10000. σ varies from 0 to 50.

In figure 4, we can observe the effect the magnitude of the Gaussian noise has on the subject's decision outcomes. Recall that, given our intial conditions, there is a slight evidential advantage for Decision B. When the $\sigma = 0$, this advantage is fully borne out, in that the model always decides in favor of B. However, as the noise level increases, the number of trials ending in B decreases, and the number of trials ending in A increases. As the σ the number of trials for each decision becomes about equal. This is because for high levels of σ, the

noise term in the model dominates the signal term, so x's crossing of a threshold is governed by chance rather than evidence.

2.2.3 Varying E

Finally, let us vary the evidence term E in the model. Previously, we noted how an imbalance in E lead to a slight preference for one outcome over another. In Figure 5 below, we keep the same initial conditions as before, but vary E from -0.5 to 0.5. This larger magnitude of values allows us to see this effect more clearly.

Figure 5: Decision outcomes over 100 simulations of the DDM for varying values of E. $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps $= 10000$. E varies from -0.5 to 0.5.

From Figure 5, we can see that, The decision outcomes follow a sigmoidal pattern, and are symmetrical around $E = 0$. When E is negative and large, the model nearly always decides in favor of B. As E goes to 0 from the left, the model begins to decide for A more often, eventually becoming random. Conversely, when E is positive and large, the model nearly always decides in favor of A. Similarly, as E approaches 0 from the right, the model begins to decide for B more often, eventually becoming random.

Do I need this? This makes sense, as the more compelling the evidence for one decision over another, the more likely the model is to decide in favor of that decision. However, as the evidence becomes less compelling, the model is more likely to decide randomly.

2.3 Reaction Time Distributions

In the previous sections, we have examined at the distribution of the decisions made by the model. In this section, we will examine the dsitribution of the reaction times of the model, or how long it takes the model to make a decision given some initial conditions. Specifically, we will investigate how the distribution of reaction times changes as we vary E in the model.

Keeping the initial conditions the same as before, but modulating E, Figure 6 below illustrates the differences in reaction time distributions. From Figure 6 we can see that the reacion times are normally distributed with a rightward skew, indicating that for each condition, most of the trials end before 100 seconds. Moreover, we can see that when the evidence is equal, or just slightly greater, for either option, the distributions overlap: it takes the subject about the same amount of time to make a decision either way. However, when the evidence is skewed towards one outcome, in this case A, a greater proportion of trials where A is the outcome end quickly than those where B is the outcome. The area of the B distribution is also smaller in this case, indicating less trials end in favor of B when the evidence is skewed towards A. This is because $\frac{dx(t)}{dt}$ is larger and positive when E is larger and positive. Thus, the pressure towards the A threshold is greater, leading to quick decisions in favor of A, and slower decisions in favor of B: it will only go towards the B threshold if it is forced to by the noise term.

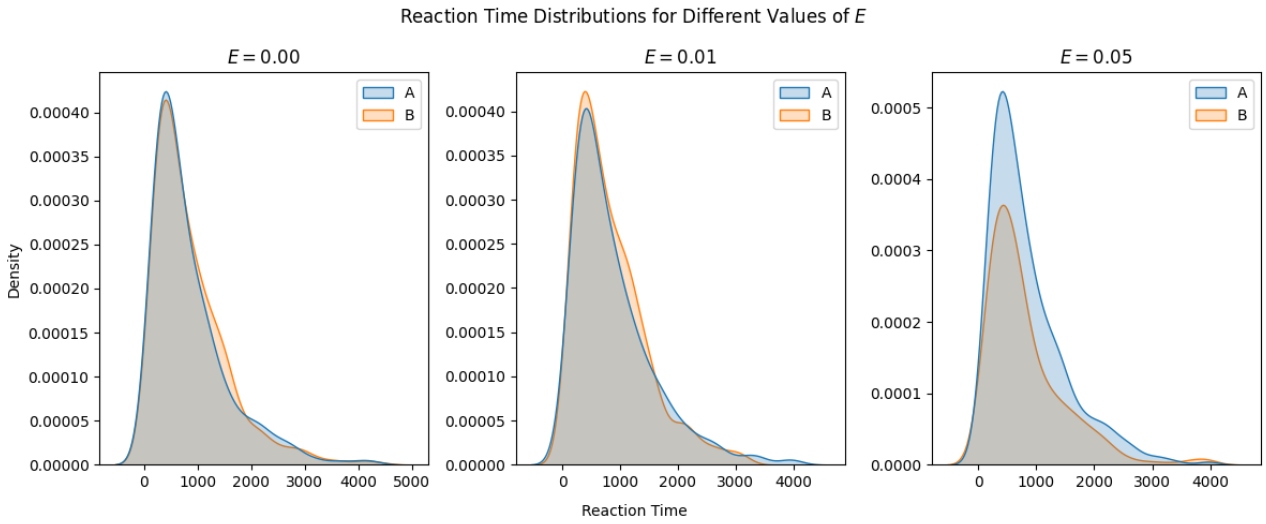

Figure 6: Reaction time distributions for the DDM for varying values of E. $\sigma = 7$, $\mu = 20$, $x(0) = 0$, $dt = 0.1$, possible timesteps $= 10000$. E varies from 0.00 to 0.05.

2.4 Summary

In this section, we have examined the Drift Diffusion Model as a way of modeling the decision making process undergone by a subject's brain in a 2AFC task. We have seen how the model can be used to approximate the integration of informtation over by time by different brain regions, and lead to decisions in favor of one outcome over another based on both subject-internal parameters (μ and σ), and external parameters (E). Moreover, we have seen how variation in these parameters can lead to different patterns of decision making for subjects, both in terms of number of A vs B decisions, and the amount of time taken to make these deciisons.

In the next section, we will look at the same task through the lens of a different model of decision making, namely a Recurrent Neural Network.

3 The Reccurent Neural Network Model

In the previous section, we examined the Drift Diffusion Model as a way of modeling the neural decision making process in a 2AFC task. However, the DDM we described did not actually *perform* the task in any meaningful sense. In this section, we will train a Recurrent Neural Network (RNN) to perform the 2AFC task itself, and then analyze its performance and dynamics.

3.1 The Representing the 2AFC Task

In order to train the network, we first need to simulate the task itself. We will represent the task stimulus as a vector $\mathbf{u}$ of length T, where T is the length of the trial, and each element of $\mathbf{u}$ is a scaler u_t representing the type of stimulus: positive for motion rightward, or negative for motion leftward; if $u_t = 0$ there is no motion. u_t's magnitude represents the *coherence* of the stimulus. Formally, u_t is defined as:

$$u_t = v + \xi_t \, \mathcal{N}(0, \sigma) \tag{2}$$

where v is velocity of the stimulus (coherence and direction), and ξ_t is some gaussian noise with mean 0 and standard deviation σ, representing the noise in the stimulus.

We will also define an intended output, which represents the true direction of the stiumulus, and is the value we want our RNN to be able to predict. It is defined as a vector $\mathbf{z}^*$ of length T, where:

$$\begin{cases} \mathbf{z^*}_x = 0 & x \in [0, T-1] \\ \mathbf{z^*}_T = \text{sign}(\bar{\mathbf{u}}) \end{cases} \tag{3}$$

where $\bar{v}$ is the average velocity of the stimulus over the trial. Thus, the network will need learn to output 1 when the average velocity is negative, and -1 when the average velocity is positive.

To train our network, we need to provide it a mask to indicate we only want it to learn the final decision. This is defined as a vector $\mathbf{m}$ such that:

$$\mathbf{m} = \begin{cases} 0 & x \in [0, T-1] \\ 1 & x = T \end{cases} \tag{4}$$

Finally, we need to create a training set for the network, i.e., a collection of simulated stimuli, intended outputs, and mask values that we will train the model on, as well as a test set, i.e., a seperate set of stimuli on which to test the model's performance. We will do this by perfroming the above mentioned steps for 500 simulated trials. The simulations for the training set will randomly choose an initial v between (integers) -5, 5 (exlcuding 0), and a random level of noise $\sigma = [0, 0.1, 0.05]$. The test set will have a random initial v between -5, 5 (excluding 0), and a random level of noise $\sigma = [0.5, 1]$.

Figure 7 below illustrates a sample of the input vectors in the training set and, and their corresponding true output vectors.

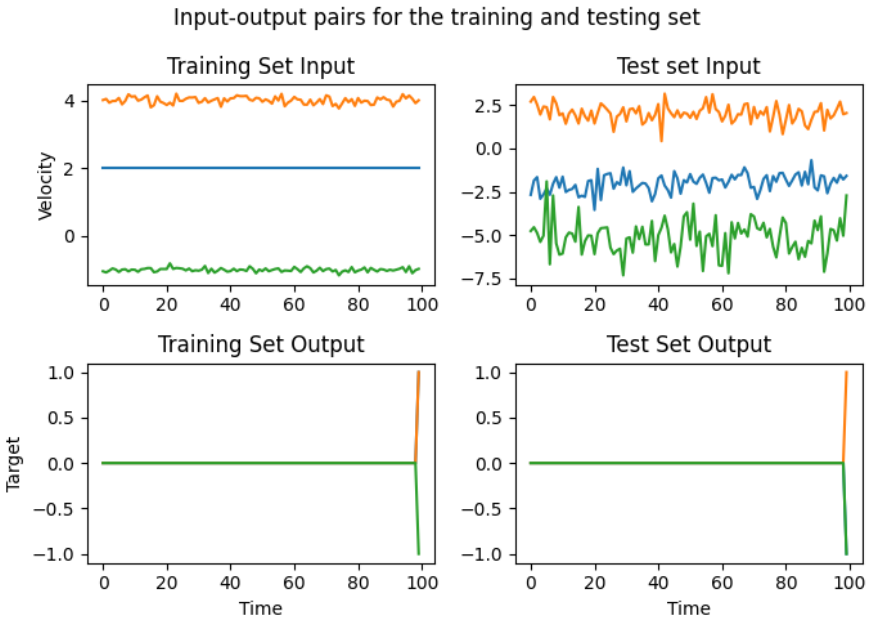

Figure 7: Sample of input output pairs for the 2AFC task. The top row represents the input vectors, and the bottom row represents the true output vectors.

From Figure 7 we see that when the initial input vector is positive, the velocirty stays positive, and the tru output is 1. Conversely, when the initial input vector is negative, the velocity stays negative, and the true output is -1.

3.2 Designing an RNN for the 2AFC Task

Now that we created our task, we must design the RNN that will learn from the training set to predict the output of the test set.

A Recurrent Neural network is a set of inter-and-self connected nodes that can be used to model the input output pairs of a sequence. In our model, each neuron x_i will have a firing rate given by $\phi(x_i(t))$, where $\phi = \tanh$. The parameters of the model are J_{ij}, the synaptic weights between neurons, B_{ik} which are the weights between the input to the netowrk and the neurons, and W_{kj} which are the weights of the output of the neurons.

Taken together, we have that the dynamics of a neuron, given P inputs of signal u_k are given by:

$$\dot{x}_i = -x_i(t) + \sum_{j=1}^{N} J_{ij}\phi(x_j(t)) + \sum_{k=1}^{P} B_{ik}u_k(t) \tag{5}$$

and the output of the network is given by:

$$z_k(t) = \sum_{j=1}^{N} W_{kj}\phi(x_j(t)) \tag{6}$$

Training this network amounts to finding the optimal values of J_{ij}, B_{ik}, and W_{kj} that minimize the error between the output of the network and the desired output. This is done through backpropgation, where the error from a predicition is shared throughout the network and the weights are updated accordingly.

While the above is the general makeup of an RNN, we will have to do some additional work to adapt it to the 2AFC task. Firstly, we will need to discretize the neuronal dynamic for simulation. This can be done using Euler's method resulting in the following:

$$x_{i,t+1} = (1 - \Delta t)x_{i,t} + \Delta t(\sum_{j=1}^{N} J_{ij}\phi(x_{j,t}) + \sum_{k=1}^{P} B_{ik}u_{k,t}) \tag{7}$$

which can be vectorized as:

$$\mathbf{x}_{t+1} = (1 - \Delta t)\mathbf{x}_t + \Delta t(\mathbf{J}\phi(\mathbf{x}_t) + \mathbf{B}\mathbf{u}_t) \tag{8}$$

Finally, we need to define the loss function the network will use to learn the task. For this network, we will use the mean squared error loss function, which is defined as:

$$\mathcal{L} = \frac{1}{P}\sum_{t=1}^{T} m_{p,t,k}(z_{p,t,k} - z^*_{p,t,k})^2 \tag{9}$$

where P is the number of nuerons in the network, p is the trial number, t is the timestep, k is the input dimension, and $m_{p,t,k}$ is the mask for the network. The network will learn to minimize this function by adjusting its parameters via gradient descent, i.e., to make its output $z_{p,t,k}$ as close to $z^*_{p,t,k}$ as possible.

3.3 Training and assessing the RNN

Having generated our data, and designed our network, we can now feed it the training set to allow the network to learn the task. Figure 8 illustrates the decrease in the learning loss over training epochs.

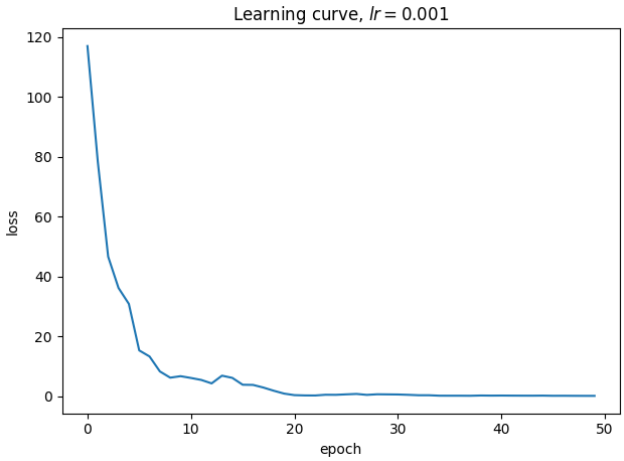

Figure 8: The learning curve of the RNN over 50 epochs, minimizing the Mean Squared Error with a learning rate of 0.01

We can assess the performance of our network by inspecitng it's accuracy, i.e., how well it did on categorizing the input vectors in each set. We define accuracy as

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{the total number total predictions}}. \tag{10}$$

Assessing our network, we find that it has a 100% Accuracy on the training set, and a 94.44% accuracy on the test set.

4 Conclusion