Uncovering Neural States in Rodents

Alexander J Ohrt Sander Ruud Simon Liabø Jesper Bengston

November 2022

1 Introduction

The objective of this work is to understand and implement a simplified version of the method introduced by Linderman et. al [1]. This version of the method will then be applied to data collected from mouse brains. We are looking for a relationship between the direction the head of a mouse is facing and how the neurons in its brain are firing. Such a relationship can give insight into the spatial representation of the environment that the rat observes in the hippocampus. In the article, the authors follow a Bayesian approach, utilizing Bayesian hiearchical models with several levels. We want to simplify this and test a simple Poisson HMM on a similar problem. Our hope is that the simple model will uncover a connection between the states of the HMM and head direction data. Prior to the investigation, we believe that the simpler model is sufficient for finding a reasonable solution to this task.

2 Data

As noted, we are given a data set collected from mouse brains. The file `Mouse28-140313_BS0150_HMMready.mat` contains two objects:

`resampledAwakeHeadAngleData` : $(\theta_t)_{t=1,\dots,T} \in \mathbb{R}^{T\times 1}$, θ_t = angle at time t.

`celldata`: $(y_t)_{t=1,\dots,T} \in \mathbb{R}^{T\times N}$, y_t = #firings of neuron $1, ..., N$ in $(t-1, t]$.

The `celldata` contains neural firings for 71 neurons in the brain. These firings have been recorded during 15243 time bins, where the number of firings are recorded in each bin. The degree to which the neurons fired, varies greatly. Some neurons did not even fire at all. In our analysis, we removed we neurons which did not fire more than 100 times during the recordings, reducing the number of neurons to 59.

3 Hidden Markov Model

In order to find the spacial representation of the environment we decided to use a Hidden Markov Model (HMM). In this model, we assume a latent Markov process, denoted by S_t, where $t \in \{0, 1, 2, \dots, T\}$ denotes the time index up to some finite time T. This sequence will be called the *state* process. The hidden state process is only accessible through an observable sequence, $y_t|S_t$, which is used for fitting the model and for inference concerning the hidden states. This can be treated as a generative model, meaning that we may assume that the `celldata` that has been collected has been generated from a HMM. The following model has been used

$$p(y_{1:T}, S_1|\boldsymbol{\pi}, P, \boldsymbol{\Lambda}) = p(S_1|\boldsymbol{\pi}) \prod_{t=2}^{T} p(S_t|S_{t-1}, P) \prod_{t=1}^{T} p(y_t|S_t, \boldsymbol{\Lambda}),$$

where $\boldsymbol{\pi}$ denotes the starting probabilities of the state process, P denotes a transition probability matrix for the state process and $\boldsymbol{\Lambda}$ are parameters in the conditional distribution given each state. We assume that the

observable process given the state sequence is a Poisson process, parameterized by Λ. The state process (in a discrete time, countable state space Markov process) transitions between a given set of states, according to the transition matrix P. In Linderman et. al [1], the authors develop an algorithm that finds an optimal number of states given an observable sequence. In our case, we will simply assume a given fixed amount of states.

Let S_t denote the state at time t and let

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_{1,S_t} \\ \lambda_{2,S_t} \\ \vdots \\ \lambda_{N,S_t} \end{bmatrix}, \qquad \boldsymbol{y}_t \mid S_t = \begin{bmatrix} y_{1,t} \\ \vdots \\ y_{N,t} \end{bmatrix}, \qquad y_{c,t} \mid S_t \sim \text{Pois}\left(\lambda_{c,S_t}\right).$$

The probability of each state in the first step is

$$p\left(S_1 \mid \boldsymbol{\pi}\right) = \text{Multinomial}\left(S_1 \mid \boldsymbol{\pi}\right), \tag{1}$$

where $\boldsymbol{\pi} = \{\pi_i\}$ is assumed to be a uniform prior, $\pi_i = 1/N$, making all states equally likely to begin with. The probability of each subsequent state, S_t, is then multinomially distributed with transition probabilities according to the transition matrix, P,

$$p(S_t|S_{t-1}, P) = \text{Multinomial}(S_t|P_{S_{t-1},:}),$$

and the firing of a neuron can be described as

$$p(y_t|S_t, \Lambda) = \prod_{c=1}^{C} \text{Pois}(y_{c,t}|\lambda_{c,S_t}).$$

Thus we can see that the rate $\lambda_{c,St}$ is specific to both the state and neuron, which in turn entails that the neuron activations also are specific to each neuron and state.

4 Methods

We fit a HMM using the package `depmixS4` [2] in R. Post-processing of the results are done in a mix of R and Python, simply because we are not equally familiar with both languages. We try to fit the HMM with the entire (remaining) data set, which contains 59 neurons, that are recorded in the specified time bins. The method `depmix` is fitted with a formula as specified for multivariate data in the documentation (code is attached in Appendix A). The hyperparameters that were used for the method were `nstates = 35` and `instart = pies`, where `pies` is a vector of equal probabilities for each state $i \in \{1, \ldots, 35\}$. Thus, we assume a fixed number of 35 states with equal prior probabilities in the state process. As a sidenote, we ran into some convergence issues when trying to fit the model with all 59 neurons. However, the algorithms converged after removing the four last neurons, which is why we ended up fitting the model on the first 55 neurons. The HMM then fits 55×35 parameters λ_{il}, i.e. there is one Poisson rate parameter per state $i \in \{1, \ldots, 35\}$, per neuron $l \in \{1, \ldots, 55\}$.

5 Results

After fitting the HMM, the transition matrix between the hidden states is post-processed in order to uncover a relationship between the `celldata`, the head angle data and the hidden states. We also want to see if our simple implementation can capture the underlying features of the data. We are told that the states should lie on a circle in a low dimensional manifold, which is what we are looking for. The transition matrix as returned from the model is shown in Figure 1. This does not give any immediate hints in the direction of a circular shape.

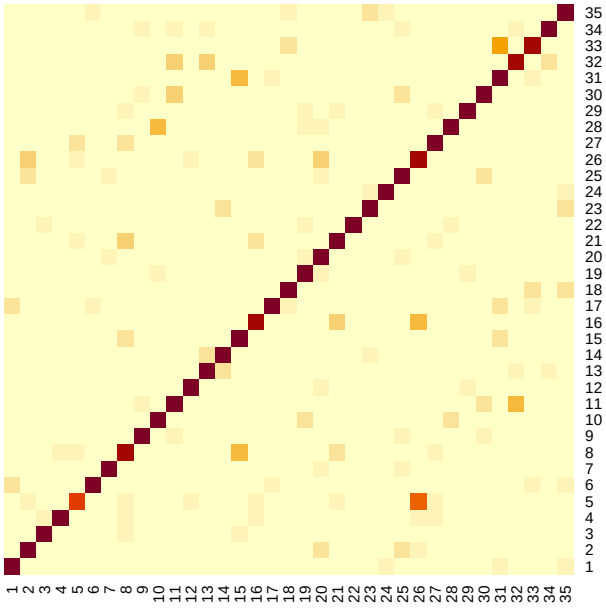

Figure 1: Transition matrix

Moving on, we do some clustering on the transition matrix in order to group similar states based on Euclidean distance. This should give some more insights into how the states are connected to each other. Figure 2 shows the recorded head angle as a function of time, color-labelled according to a K-means clustering into 10 clusters of the corresponding hidden states inferred by the HMM. The patterns are very clear here: each cluster contains states where the head angles of the mouse are almost constant in time. That is, when the mouse has been looking in a specific direction, it turns out that the generative model predicts the state as being the same in time. In addition, the HMM respects the cyclic behaviour of the directions, since we can see that the top and bottom angles (at radians 0 and 2π) have the same label. This makes sense because the head will point in the same direction after moving a full circle or 2π radians. This is a further indication of the circular nature of the data.

Next we employ hierarchical clustering on the transition matrix. Figure 3 shows an ordered version of the transition matrix shown in Figure 1, according to this clustering. Despite the fact that the ordering method has removed some states from the plot, it looks cyclic as well.

Figure 4 shows a plot of the recorded head angles on the unit circle, labelled by their corresponding states clustered into groups calculated by the K-means clustering that was described earlier. In addition, we have plotted the average head angle corresponding to each group of inferred states in the generative model. As discussed when looking at Figure 2, it is apparent that the groups are well divided around the unit circle. Please notice that the color schemes are not the same in Figure 4 and Figure 2 unfortunately, which should have been done for clarity.

Finally, different methods for dimensionality reduction on the transition matrix are attempted. First we employ PCA, which is a linear method. The scores of the first two principal components are plotted in Figure 5. Note that only $\sim 13\%$ of the variance in the transition matrix is captured. Despite this, the plot indicates that the inferred states might lie on a circular two-dimensional manifold. Because of the low amount of variability captured by the first two principal components, we employ a nonlinear dimensionality reduction method called t-SNE. After reducing the dimension of the transition matrix to two using t-SNE, the results are shown in Figure 6. Oddly enough, this dimensionality reduction technique has not found an equally clear circular manifold from the data. We would have to dive a bit deeper into the algorithms to explain why this is the case, something that we have not done. Perhaps the nonlinear relationship between

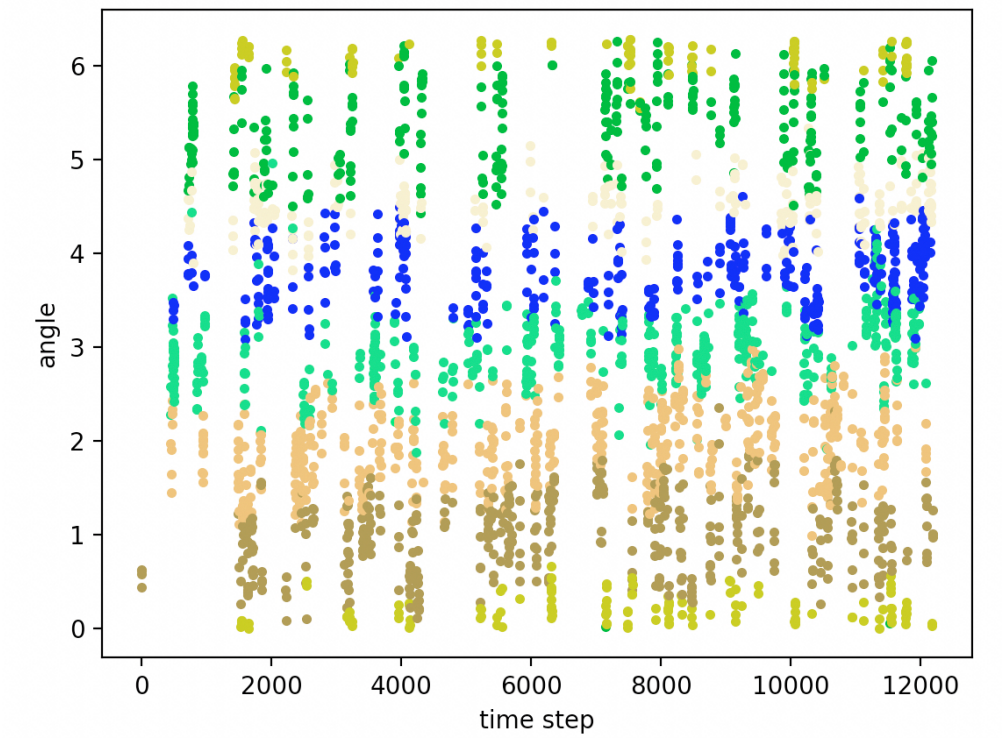

Figure 2: Angle as a function of time. The colors are based on a K-means clustering on the transition matrix, into 10 clusters.

the inferred states is different?

6 Conclusion

After fitting our simple HMM to the mouse data and performing some post-processing, we reach the conclusion that the expected cyclic nature of the neurons has been found by the model. The observed sequence (neuron firings) corresponding to the inferred hidden states appear to be cyclical after dimensionality reduction, signalling that the model has been able to model the generative relationship between the true angle of the head of the mouse and the neuron activations. All in all, our results show a clear relationship between the head angle of the mouse, and the corresponding neuro activations.

Notice that we could also have considered using model selection methods like AIC, BIC or cross-validation in order to find a "close to optimal" number of states, instead of simply setting the number of states to an arbitrary amount as done here. This would perhaps have lead to a better generative model but would have added complexity to the project.

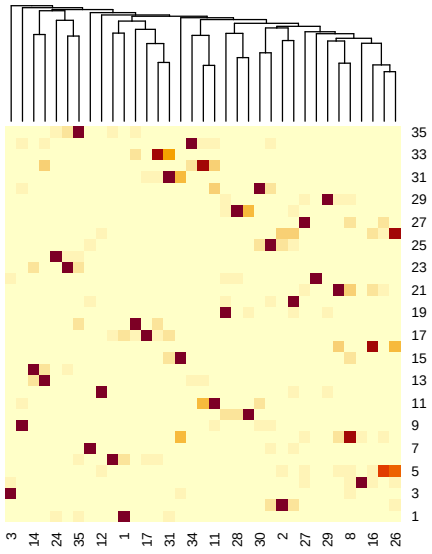

Figure 3: Hierarchically clustered transition matrix. It looks kind of cyclic.

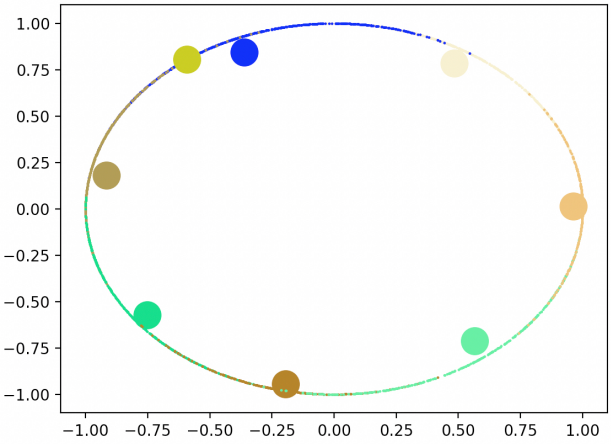

Figure 4: Plot of $\cos\theta$ vs $\sin\theta$, for each recorded head angle θ. The dots show the average value of each of the points in each cluster.

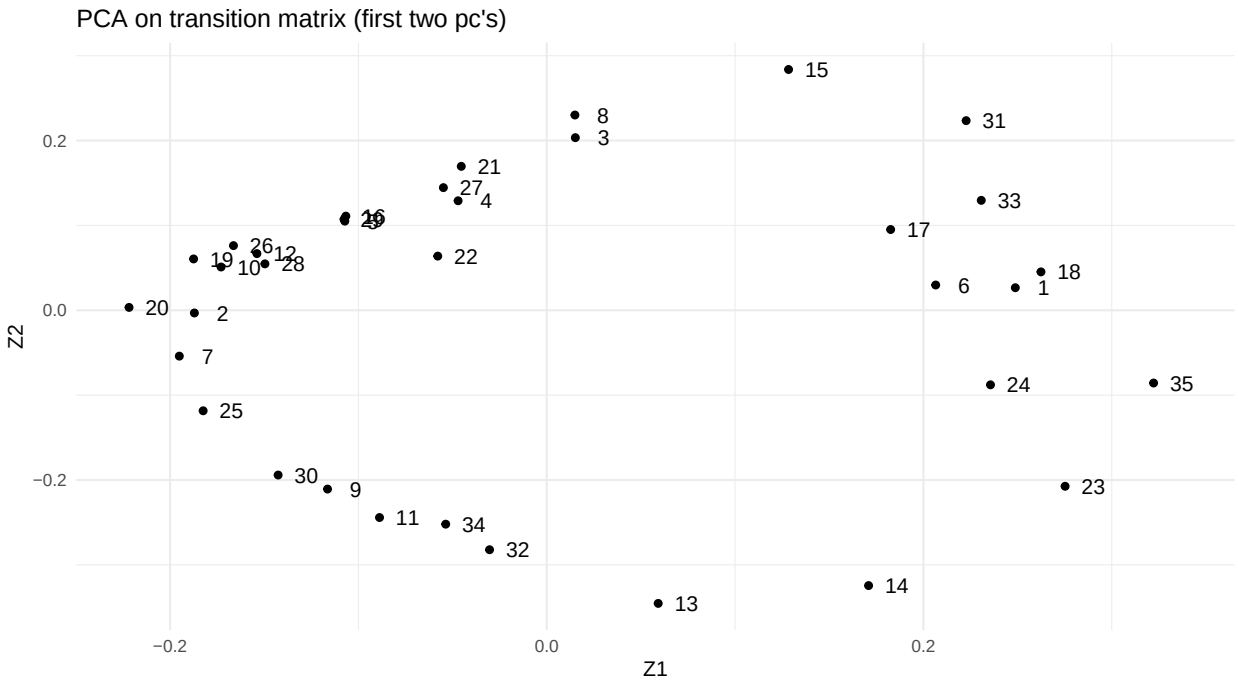

Figure 5: Scores of the first two components of PCA are shown.

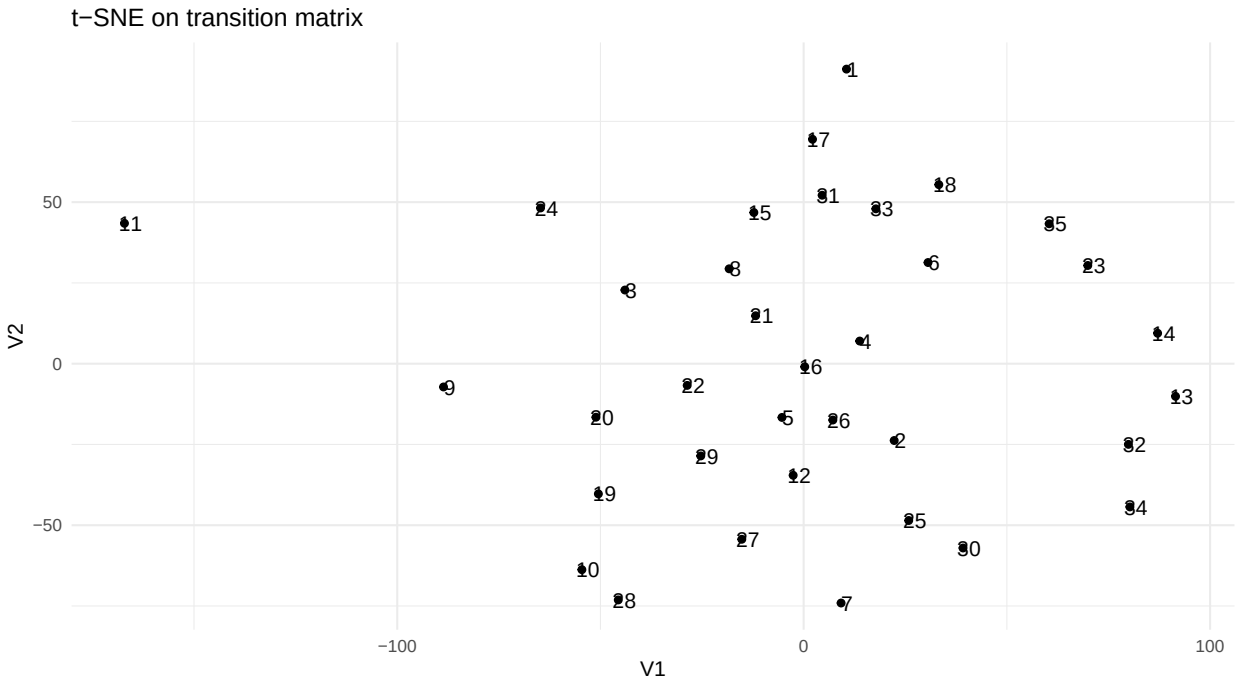

Figure 6: Results from t-SNE, reduced to two dimensions.

Appendix A R code

```r
rm(list=ls())
set.seed(1234)
library(R.matlab)
library(ggplot2)
library(dplyr)
library(Rtsne)

data <- readMat("../Mouse28-140313_BS0150_HMMready.mat")
celldata <- data$celldata
celldata_df <- as.data.frame(t(celldata))
angdata <- data$resampledAwakeHeadAngleData
thresh <- rowSums(celldata)
THR <- 100 # I think this is a good threshold
celldata_active <- celldata_active[thresh > THR, ]
celldata_active_df <- as.data.frame(t(celldata_active))
nstates <- 35 # Fixed number of states.
pies <- rep(1, nstates)/nstates # Uniform prior state probabilities.


formel <- list()
families <- list()
n_neurons = 55 # Reduce this
for (i in 1:n_neurons){
  formel[[i]] <- as.formula(paste0(paste0("V",i), "~1"))
  families[[i]] <- poisson()
}

# Part 1:
model <- depmix(formel, family = families, data = celldata_df,
                nstates = nstates, instart = pies)
fm <- fit(model)

write.csv((fm@posterior), "../posterior.csv")
nstates = length(fm@posterior)-1
transition_matrix <- matrix(getpars(fm)[(nstates+1):(nstates^2+nstates)],
                            byrow=TRUE,nrow=nstates,ncol=nstates)
write.csv(transition_matrix, "../trans_matrix.csv")



# Part 2: Only considering active neurons

model2 <- depmix(formel, family = families, data = celldata_active_df,
                nstates = nstates, instart = pies)
fm2 <- fit(model)

write.csv((fm@posterior), "../posterior2.csv")
nstates = length(fm@posterior)-1
transition_matrix2 <- matrix(getpars(fm)[(nstates+1):(nstates^2+nstates)],
                            byrow=TRUE,nrow=nstates,ncol=nstates)
write.csv(transition_matrix2, "../trans_matrix2.csv")


# Generating plots

inferred_states <- fm@posterior[,1]
length(inferred_states)
table(inferred_states)

# Play with the transition matrix.

# Plot the transition matrix.
pdf(file = "trans_matrix.pdf", width = 9, height = 5)
transition_matrix %>% heatmap(Rowv = NA)
dev.off()
```

```r
pdf(file = "trans_matrix2.pdf", width = 9, height = 5)
# Heatmap without the rearrangement because of the dendrogram.
transition_matrix %>% heatmap(Colv = NA, Rowv = NA)
dev.off()

# Try to do PCA on transition matrix.
pca <- princomp(transition_matrix)
summary(pca)
scores <- pca$scores
labels <- 1:nstates
df <- data.frame(cbind(scores[,1], scores[,2], labels))
df$labels<- as.factor(df$labels)
colnames(df) <- c("Z1", "Z2", "St")
plt <- tibble(df) %>%
  ggplot(aes(x = Z1, y = Z2)) +
  geom_point() +
  geom_text(aes(label = St), nudge_x = 0.015) +
  ggtitle(paste0("PCA on transition matrix (first two pc's)")) +
  theme_minimal()

pdf(file = "PCA.pdf", width = 9, height = 5)
print(plt)
dev.off()

# Try with a non-linear dim. red. method also, since PCA does not capture a lot of the
    variability in the first components.
tsne_out <- Rtsne(transition_matrix, pca = F, perplexity = as.integer(30/3)-1, theta = 0.8)
t_sne <- data.frame(cbind(tsne_out$Y,"state"=1:15)) %>%
  ggplot(aes(x = V1, y = V2)) +
  geom_point() +
  geom_text(aes(label = state), nudge_x = 1.5) +
  ggtitle(paste0("t-SNE on transition matrix")) +
  theme_minimal() # Not really a circle tbh.

pdf(file = "t-SNE.pdf", width = 9, height = 5)
print(t_sne)
dev.off()

# Hierarchical clusterings
hcl <- hclust(dist(transition_matrix), method = "ward.D")
cut <- cutree(hcl, k = 4) # We cut it into 4 groups.

pdf(file = "hclust.pdf", width = 9, height = 5)
plot(hcl) # Produce the same dendrogram as in the heatmap above.
rect.hclust(hcl, k = 4, border = 2:6)
dev.off()

# We group the inferred states according to the cut above.
grouped_inferred_states <- cut[inferred_states]
```

Appendix B Python code

```
#matplotlib.use('Agg')
import scipy.io
import matplotlib.pyplot as plt
import scipy.stats
import numpy as np
import pandas as pd
from collections import defaultdict
from sklearn.cluster import AgglomerativeClustering
from sklearn import cluster
import random

def create_plots(trans_matrix_csv, posterior_csv):
    matrix = pd.read_csv(trans_matrix_csv, index_col=None)
    transition_matrix = matrix[["V"+str(i) for i in range(1, len(matrix)+1)]].to_numpy()
    posterior = pd.read_csv(posterior_csv)
    fname = '../Mouse28-140313_BS0150_HMMready.mat'
    mat = scipy.io.loadmat(fname)
    angdata = np.ravel(np.array(mat['resampledAwakeHeadAngleData']))
    celldata = np.array(mat['celldata'])
    celldata = celldata.astype(int)
    ## toss out neurons that are not very active (because they boring)
    thrfrs = np.sum(celldata, 1)
    THR = 100 # I think this is a good threshold
    celldata = celldata[thrfrs>THR,:]
    Tfit = int(round(0.8*len(celldata[0,:])))
    Sfit = np.transpose(celldata[:,:Tfit])
    Stest = np.transpose(celldata[:,Tfit:])
    Sfit = Sfit.copy(order='C')
    Stest = Stest.copy(order='C')
    angfit = angdata[:Tfit]
    angtest = angdata[Tfit:]
    Sfit = (posterior[["S"+str(i) for i in range(1, len(matrix))]].to_numpy())
    state_sequence = posterior["state"].values
    used_states = posterior["state"].unique()
    k_means = cluster.KMeans(n_clusters=10, algorithm = "full")
    k_means.fit(transition_matrix)
    values = k_means.cluster_centers_
    f = defaultdict(list)

    for i, elem in enumerate(state_sequence):
        if i > 1200: break
        if not np.isnan(angfit[i]):
            elem = k_means.labels_[elem-1]
            f[elem].append(angfit[i])
    number_of_colors = len(f.keys())
    color = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
                for i in range(number_of_colors+100)]
    for i, elem in enumerate(state_sequence):
        if i > 1200: break
        if not np.isnan(angfit[i]):
            elem = k_means.labels_[elem-1]
            plt.plot(i, angfit[i], ".", color=color[elem])
    plt.xlabel("time step")
    plt.ylabel("angle")
    plt.show()
    i=0
    for key, value in f.items():
        value = np.array(value)
        value = value[~np.isnan(value)]
        x = np.cos(value)
        y = np.sin(value)
        plt.scatter(np.average(x), np.average(y), label = key, s = 300, c=color[i]) # Plot
    average
        plt.scatter(x, y, label = key, c=color[i], s=.5) # Plot each point
        i+=1
```

```
65      plt.show()
66
67  if __name__ == "__main__":
68      create_plots("trans_matrix.csv", "posterior.csv") # All considered neurons
69      create_plots("trans_matrix2.csv", "posterio2r.csv") # Only active neurons
```

References

[1] S. W. Linderman et al. "A Bayesian nonparametric approach for uncovering rat hippocampal population codes during spatial navigation". In: *Journal of Neuroscience Methods* 263 (2016), pp. 36–47. ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2016.01.022. URL: https://www.sciencedirect.com/science/article/pii/S0165027016000418.

[2] I. Visser and M. Speekenbrink. "depmixS4: An R Package for Hidden Markov Models". In: *Journal of Statistical Software* 36.7 (2010), pp. 1–21. URL: https://www.jstatsoft.org/v36/i07/.