Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Ulrik Bernhardt Danielsen

A step toward model selection in unsupervised clustering of animal behavior

Master's thesis in Applied Physics and Mathematics
Supervisor: Benjamin Adric Dunn
Co-supervisor: Ingeborg Gullikstad Hem
June 2023

NTNU
Norwegian University of
Science and Technology

Ulrik Bernhardt Danielsen

A step toward model selection in unsupervised clustering of animal behavior

Master's thesis in Applied Physics and Mathematics
Supervisor: Benjamin Adric Dunn
Co-supervisor: Ingeborg Gullikstad Hem
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Contents

Abstract

Animal movements are commonly divided into distinct behaviors characterized by a set of body movements. Using time-frequency analysis to extract information about repeating movement patterns, followed by dimensionality reduction techniques such as principal component analysis and t-distributed stochastic neighbor embedding, we can separate and visualize such behaviors in a two-dimensional space. Applying this methodology to sensor data, we cluster the behavior of freely behaving rats. We investigate ways of performing model selection and use the results to show how including facial tracking recordings in addition to postural sensor data changes behavioral segmentation. Additionally, we investigate the use of topological data analysis as a tool for comparing behavioral mappings, by viewing transitions between behaviors as edges in a weighted directed graph. In doing so, we aim at an increasingly quantitative model selection phase.

Sammendrag

Dyrs bevegelsesmønster deles ofte inn i distinkte atferder, karakterisert av et sett kroppsbevegelser. Ved bruk av tidsfrekvensanalyse til å nyttegjøre informasjon om gjentakende bevegelsesmønstre, etterfulgt av dimensjonreduksjonmetoder som ”principal component analysis” og ”t-stochastic neighbor embedding”, kan vi separere og visualisere ulike atferder i et todimensjonalt rom. Denne metoden bruker vi på ekte sensordata, og grupperer atferden til fritt romsterende rotter. Vi undersøker måter å gjennomføre modellselektering og bruker resultatene til å vise hvordan å inkludere sensordata fra værhårene i tillegg til sensordata fra kroppsdeler endrer atferdgrupperingen. Videre undersøker vi bruken av topologisk dataanalyse som et verktøy for å sammenligne modeller, ved å se på overganger mellom atferder som rettede kanter i en vektet graf. Slik sikter vi på en mer kvantitativ modellselekteringsfase.

Acknowledgements

Many thanks to Associate Professor Benjamin A. Dunn for supervising this thesis, and introducing me to the group of people I am about to mention. His academic curiosity and eagerness to investigate new ideas were motivating and inspirational. My co-supervisor Postdoctoral Fellow Ingeborg G. Hem deserves special thanks. Our weekly conversations have been invaluable, and I am very grateful for the time she invested in guiding me through this process. I also wish to thank the Whitlock Group at Kavli Institute for Systems Neuroscience, in particular Associate Professor Jonathan Whitlock, Postdoctoral Fellow Jerneja Rudolf, and the outstanding lab rat Petulia. Always inviting, helpful, and interested in my progress, they made this thesis feel like a team effort. I'd also like to thank PhD Candidate Melvin Vaupel for a crash course in persistent homology, it is difficult stuff.

Lastly, I need to thank my daughter Ebba Margrethe and her amazing mother Vilde. Having you both in my life have brought calm and reassurance to a sometimes stressful and difficult process. Thank you for a special and memorable year.

Chapter 1

Introduction

1.1 Motivation

Studying behavior is the aim of ethology, and has been researched for many decades. Describing animal behavior is useful in many research situations. Among early studies, ornithologists documented behavior in great detail, comparing tame and wild birds [1]. Medical researchers could be interested in observing change in behavior as an animal is introduced to a specific drug or stimuli [2]. In animal welfare science, we may wish to detect abnormal behavior as a way of diagnosing diseases. Behaviorism is concerned with how behaviors are learned by environmental stimuli. In every case, the ability to link behavior to other factors is based on information about the behaviors themselves.

Unfortunately, this information is very inaccessible. Looking through animal video recordings with the aim of determining the animals behavior, has two severe problems. First, there does not exist a set of known behaviors which the animal selects from. The results would therefore be restricted to the predefined behaviors chosen by the researcher. Second, it is very time consuming. Given a substantial amount of data material, manual labeling is not feasible. These issues create the need to describe the behavior automatically and unsupervised.

A clustering algorithm, taking recorded motion data as input, and mapping each time point to an enumerated behavior, is from this point on referred to as a behavioral model. It is uniquely defined by the steps themselves, along with any chosen hyperparameters. Training the model then refers to performing all the steps in the clustering algorithm, resulting in an ethogram that describes how the animal moves between the detected clusters. If we do not specify the behaviors we look for, prior to training the behavioral model, the model is unsupervised.

One immediate question becomes: How to determine the performance of such a behavioral model? When presented with several ethograms, it is hard to choose between them. Again, the easy answer, to manually look back at the recordings, remains very inefficient and time-consuming. Depending on the nature of the clustering algorithm, some theoretical results can guide us toward a good model. However, these results are often only that, guidelines, providing a starting point. In this thesis, we explore the available model selection toolbox, i.e., several methods aiding in selecting the best model are tested on real data samples. Furthermore, we aim to extend the toolbox, exploring how we could provide some quantitative measurements to the model selection phase, using topological data analysis.

1.2 Previous work

A clustering pipeline, using time-frequency analysis, followed by clustering mapped time points in two-dimensional space, was first developed by researchers at Princeton University [3], mapping the actions of fruit flies and comparing results between sexes. There, theoretical guidelines were used to choose hyperparameters. The same team of researchers later extended the analysis of this data set, exploring the hierarchical structure in the clustered behaviors, using the information bottleneck algorithm [4]. This technique has also been applied to data from freely behaving rats [5]-[6], using postural tracking data from 3D motion capture, linking behavior to disease and neural recordings. In the mentioned studies, the specific parameters are chosen based on theoretical guidelines. The choices were evaluated, based on a posteriori evaluation of the clustered time points in the raw data material. Other approaches to measuring behavior exists, fitting models directly on the raw data [7]. An overview of approaches and challenges is given in a paper by Gordon J. Berman [8].

The mentioned research supports the idea that behavior is structured hierarchically. If a behavior is defined as an action sequence, we can always divide it into smaller segments. In other words, how we define behavior is dependent on the choice of time scale. Based on this assumption, it is reasonable to assume the existence of an underlying structure in the transitions between behaviors. Although this has been studied as a means to understanding the nature of behavior, it has not been utilized in testing the robustness of clustering algorithms. If we can show such an invariant structure when varying parameters in the model, we are able to make more informed parameter choices.

1.3 Overview

This thesis builds on a specialization project, which was aimed at developing the necessary code and methodological understanding [9]. The clustering algorithm was developed step by step, discussing how each choice affects the behavioral clustering. Sections 2.1 to 2.8 and Sections 3.1 to 3.3 are restated from the specialization project to illustrate how the clustering algorithm works, and the foundational theory underlying it.

Chapter 2 states the mathematical theory on which the clustering algorithm is based. The necessary foundation for time-frequency analysis is included, viewing the wavelet transform as an extension of the ideas behind the Fourier transform. Dimensionality reduction through principal component analysis and t-stochastic neighbor embedding is explained. Kernel density estimation and watershed segmentation—necessary for clustering in the two-dimensional plane—are also included. Lastly, some key concepts in topological data analysis is developed.

Chapter 3 explains the clustering algorithm. Each step is motivated in detail, indicating where we make parameter choices. A section on model selection is included, stating some tools available to us.

Chapter 4 gives a description of the data used in the analysis in this thesis. It includes details about the data collection, data quality, sample size and structure. A brief description of a visualization tool used for visualizing clustered behaviors is also given.

In Chapter 5, we explore how the procedure changes when introducing data from facial tracking as well as postural tracking. Several models, aimed at dealing with two different data characteristics, are trained and tested against each other. Using the tools available to us, two models are chosen, one trained on only the postural data, and the other trained on all available data.

Chapter 6 investigates the behaviors detected by the models found in Chapter 5. Through investigating the raw data, we show how the detected behaviors differ, and how some behaviors become divided into smaller action sequences with the addition of facial tracking data.We also use topological data analysis in this framework, both on a toy example and on the real data.

Finally, chapter 7 discusses our findings, and summarizes the results.

Chapter 2

Theory

The clustering algorithm consists of many sequential steps, combining elements from different fields of mathematics. Sections 2.1 to 2.4.2 motivates the wavelet transform as an extension of the Fourier transform on time series data, commonly used in signal processing. Dimensionality reduction, probability density estimation, and watershed segmentation are explained in Sections 2.6 to 2.8. These methods serve as the building blocks for the clustering procedure.

Sections 2.1 to 2.8 are restated from the specialization project [9].

2.1 Time series analysis

We define time series as a realization $y_t = \{y_{t_1}, y_{t_2}, \dots, y_{t_n}\}$ of a stochastic process $Y(\omega, t)$, where $\omega \in \Omega$, Ω being the sample space, and $t \in \mathbb{Z}$, $\mathbb{Z}$ being the chosen index set [10]. It is an ordered series of random variables which can be described completely by its joint probability function

$$F_{t_1,\dots,t_n}(x_1, \dots, x_n) = \Pr\{y_{t_1} \leq x_1, \dots, y_{t_n} \leq x_n\}.$$

The mean and variance functions of a time series y are defined as

$$\mu_t = E(y_t) \tag{2.1}$$

and

$$\sigma_t^2 = E(y_t - \mu_t)^2.$$

Given two random variables in a series, y_{t_1} and y_{t_2}, we define the covariance function and correlation function as

$$\gamma(t_1, t_2) = E[(y_{t_1} - \mu_{t_1})(y_{t_2} - \mu_{t_2})] \tag{2.2}$$

and

$$\rho(t_1, t_2) = \frac{\gamma(t_1, t_2)}{\sqrt{\sigma_{t_1}^2}\sqrt{\sigma_{t_2}^2}}. \tag{2.3}$$

2.1.1 Stationarity

A time series y_t is nth-order stationary if for any shift h and indexes $t_1, t_2, \ldots, t_n$ if

$$F_{y_{t_1}, \ldots, y_{t_n}}(x_1, \ldots, x_n) = F_{y_{t_1+h}, \ldots, y_{t_n+h}}(x_1, \ldots, x_n). \tag{2.4}$$

If (2.4) holds for all n, the time series is called *strictly* stationary. We also define a nth-order *weakly* stationary time series y_t if the first n joint moments are finite and time-invariant. Specifically, we define the second-order weakly stationary, i.e. with constant and time-invariant mean function (2.1), and where the covariance function (2.2) is solely a function of the time difference, as *covariance* stationary. When the covariance function between t_1, t_2 can be written as a function of the time difference $h = |t_1 - t_2|$, i.e. $\gamma(t_1, t_2) = \gamma(h) = \gamma_h$, we call it an *autocovariance* function. The same is true for the correlation function (2.3), which when is a function of the time difference is called an *autocorrelation* function (ACF). Figure 2.1 shows an example of a time series. As the mean seems to increase with t it is non-stationary.

We also define the common estimates for the mean and covariance functions. They are called the sample mean and sample covariance and are written as

$$\bar{y_t} = \frac{1}{n}\sum_{i=1}^{n} y_{t_i}, \tag{2.5}$$

$$\hat{\gamma}_h = \frac{1}{n}\sum_{i=1}^{n-h}(y_{t_i} - \bar{y_t})(y_{t_i+h} - \bar{y_t}), \tag{2.6}$$

respectively.

2.1.2 Detrending

Many methods for analyzing and processing time series require stationarity [11]. If the series is non-stationary, we can split it into a stationary and a non-stationary part, the latter is called the *trend*. Mathematically we write it as

$$y_t = \mu_t + x_t,$$

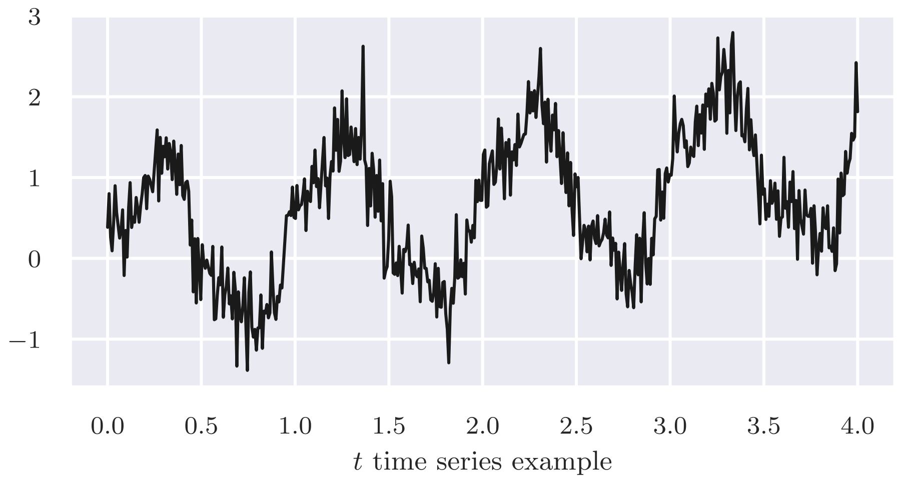


Figure 2.1: Example of a non-stationary time series with $t_0 = 0, t_n = 4$. The discrete data points are connected to better visualize the movement through time.

where x_t denotes the stationary part and μ_t the trend. The process of finding μ_t and then computing $x_t = y_t - \mu_t$ is called *detrending*. Detecting the trend can be done in many ways, for instance using regression techniques or smoothing. The simplest way is to assume a linear trend, $\mu_t = \beta_0 + \beta_1 t$ and estimate the parameters using least squares. In Figure 2.2 the linear regression fit is shown, showing an upward trend in the time series.

2.2 Fourier analysis

Let $Z_1, Z_2, \ldots, Z_n$ be a sequence of numbers. For simplicity in notation we assume n to be an odd number. It can be shown that the sequence can be represented as a linear combination of complex exponentials

$$Z_t = \sum_{k=-\frac{n-1}{2}}^{\frac{n-1}{2}} c_k e^{\frac{i2\pi kt}{n}}. \tag{2.7}$$

This comes from the fact that the set

$$\left\{ e^{\frac{i2\pi kt}{n}} \middle| k \in \left[-\frac{n-1}{2}, \frac{n-1}{2} \right] \right\}$$

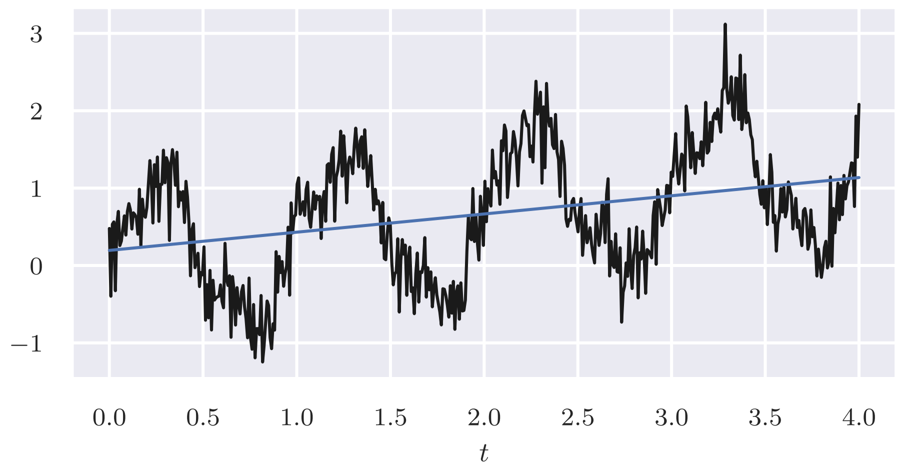


Figure 2.2: Time series from Figure 2.1 with an estimated linear trend shown in blue. Clearly, there exists an upward trend in the data.

consists of n orthogonal functions [10]. I.e., that

$$\sum_{i=1}^{n} e^{\frac{i2\pi kt}{n}} e^{-\frac{i2\pi jt}{n}} = \begin{cases} n, & k = j \\ 0, & k \neq j \end{cases}.$$

The coefficients c_k are given by

$$c_k = \frac{1}{n}\sum_{t=1}^{n} Z_t e^{-\frac{i2\pi kt}{n}}.$$

It is clear that (2.7) is periodic with period n, meaning $Z_{t+jn} = Z_t, j = 0, \pm 1, \pm 2, \ldots$. Thus the Fourier series can capture periodic sequences. The smallest positive integer n for which $Z_{t+n} = Z_t$ is called the fundamental period, with corresponding fundamental frequency $2\pi/n$. For the components $k = \pm j, j = 1, 2, \ldots, (n-1)/2$ the frequencies are multiples of the fundamental frequency, $\omega_k = k(2\pi/n)$. The set of frequencies making up the series is called the *spectrum*. As a consequence, the coefficients c_k can be viewed as weighting the importance of the contributions for the different frequencies making up the full sequence. This is formalized by the definitions of energy and power,

$$\text{energy} = \sum_{t=2}^{n} Z_t^2 = n \sum_{k=-\frac{n-1}{2}}^{\frac{n-1}{2}} |c_k|^2, \tag{2.8}$$

$$\text{power} = \frac{\text{energy}}{n} = \sum_{k=-\frac{n-1}{2}}^{\frac{n-1}{2}} |c_k|^2. \tag{2.9}$$

Let p_k be the contribution to the power from frequency $k = 0, 1, \ldots, (n-1)/2$. As ω_k and ω_{-k} corresponds to the same frequency the contribution is given as $p_0 = c_0^2, p_k = 2|c_k|^2, k = 1, \ldots, (n-1)/2$. The values p_k are called the power spectrum of the series.

2.2.1 Discrete-time Fourier transform

We have seen that all sequences of length n can be viewed and parameterized as Fourier series with period n. Moving to non-periodic sequences essentially amounts to taking the limit of the series as n approaches infinity. Formally we now let Z_t be a finite discrete function of t, where $Z_t = 0$ when $|t| > M$ for some integer M. Choosing $n = 2M + 1$ the function

$$Y_{t+jn} = Z_t,\ t \in \left[-\frac{n-1}{2}, \frac{n-1}{2}\right],\ j \in \mathbb{Z}$$

is periodic with period n. Its Fourier series is

$$Y_t = \sum_{k=-\frac{n-1}{2}}^{\frac{n-1}{2}} c_k e^{\frac{i2\pi kt}{n}}.$$

As $Y_t = Z_t$ when $t \in [-(n-1)/2, (n-1)/2]$, and $Z_t = 0$ when $|t| > (n-1)/2$, the coefficients c_k can be written as the infinite sum

$$\begin{aligned} c_k &= \frac{1}{n} \sum_{t=-\infty}^{\infty} Z_t e^{\frac{-i2\pi kt}{n}} \\ &= \frac{2\pi}{n} f\left(\frac{2\pi k}{n}\right), \end{aligned}$$

where

$$f(\omega) = \frac{1}{2\pi} \sum_{t=-\infty}^{\infty} Z_t e^{-i\omega t}.$$

If we now take the limit $Z_t = \lim_{n\to\infty} Y_t$ the summation becomes an integral over the length 2π [10]. This gives the relation

$$Z_t = \int_{-\pi}^{\pi} f(\omega) e^{i\omega t} d\omega, \quad t \in \mathbb{Z} \tag{2.10}$$

$$f(\omega) = \frac{1}{2\pi} \sum_{t=-\infty}^{\infty} Z_t e^{-i\omega t}, \quad -\pi \leq \omega \leq \pi, \tag{2.11}$$

where $f(\omega)$ in (2.11) is called the discrete-time Fourier transform of Z_t. As opposed to the periodic case (2.7) where the periodic sequence was made up of a finite number of frequencies, the non-periodic sequence is an integral over a continuum of frequencies ω. We call $|f(\omega)|$ the spectrum of the sequence, and the function $g(\omega) = 2\pi|f(\omega)|^2$ the energy spectrum. The energy spectrum definition comes from Parseval's relation

$$\sum_{t=-\infty}^{\infty} |Z_t|^2 = 2\pi \int_{-\pi}^{\pi} |f(\omega)|^2 d\omega. \tag{2.12}$$

It is worth noting that the relation in (2.12) only holds when the sequence Z_t is absolutely summable, i.e., that

$$\sum_{t=-\infty}^{\infty} |Z_t| < \infty. \tag{2.13}$$

2.3 Spectral density estimation

Let y_t be a stationary time series where the autocovariance function γ_h from (2.2) is absolutely summable. Then we can write γ_h as a Fourier transform pair

$$f(\omega) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} \gamma_h e^{-i\omega h}, \tag{2.14}$$

$$\gamma_h = \int_{-\pi}^{\pi} f(\omega) e^{i\omega h} d\omega. \tag{2.15}$$

It can be shown [11] that the spectrum $f(\omega)$ in (2.14) is real-valued and non-negative. Furthermore, as $\mathrm{Var}(y_t) = \gamma_0$, we get the interpretation

$$\mathrm{Var}(y_t) = \int_{-\pi}^{\pi} f(\omega) d\omega,$$

i.e., that $f(\omega)$ is the contribution to the variance for frequency ω. We often want to locate these important frequencies, and thus an important task is to estimate this spectrum.

2.3.1 Periodogram

Again we consider a time series sample $y_1, y_2, \ldots, y_n$ where n is chosen to be odd for simplicity. It can be written as a real Fourier representation

$$y_t = a_0 + \sum_{k=1}^{\frac{n-1}{2}} (a_k \cos(\omega_k t) + b_k \sin(\omega_k t)),$$

where $\omega_k = 2\pi k/n, k = 0, 1, \ldots, (n-1)/2$ and the coefficients are given as

$$a_0 = \bar{y_t}, \quad a_k = \frac{2}{n}\sum_{t=1}^{n} y_t \cos(\omega_k t), \quad b_k = \frac{2}{n}\sum_{t=1}^{n} y_t \sin(\omega_k t).$$

We then define the periodogram as

$$I(\omega_k) = \begin{cases} n a_0^2, & k = 0 \\ \frac{n}{2}(a_k^2 + b_k^2), & k = 1, \ldots, (n-1)/2. \end{cases} \tag{2.16}$$

The periodogram is of interest as it has a large value if the frequency ω_k is of importance in the series. A scaled periodogram $\frac{2}{n} I(\omega_k)$ estimates the sample variance of the sinusoid component at frequency ω_k [11]. A periodogram for the time series example in Figure 2.1 is shown in Figure 2.3. Note that the time series is linearly detrended for the periodogram is computed. The two highest peaks match the underlying generating function which can be revealed to be $\sin(2\pi t) + \frac{1}{4}\cos(6\pi t) + t/3$.

2.3.2 Sample spectrum

One intuitive way of estimating the spectrum is to replace the autocovariance with the sample autocovariance from Equation (2.6). I.e., we define the sample spectrum for a realization $y_1, y_2, \ldots, y_n$

$$\hat{f}(\omega) = \frac{1}{2\pi} \sum_{k=-(n-1)}^{n-1} \hat{\gamma}_k e^{-i\omega k}. \tag{2.17}$$

At the Fourier frequencies ω_k, it is related to the periodogram through [10]

$$\hat{f}(\omega_k) = \frac{I(\omega_k)}{4\pi}.$$

Although $\hat{f}(\omega_k)$ is asymptotically unbiased, meaning $\lim_{n\to\infty} E(\hat{f}(\omega)) = f(\omega)$, it lacks consistency in the variance as n tends to infinity, i.e.,

$$\lim_{n\to\infty} \mathrm{Var}(\hat{f}(\omega_k)) \neq 0.$$

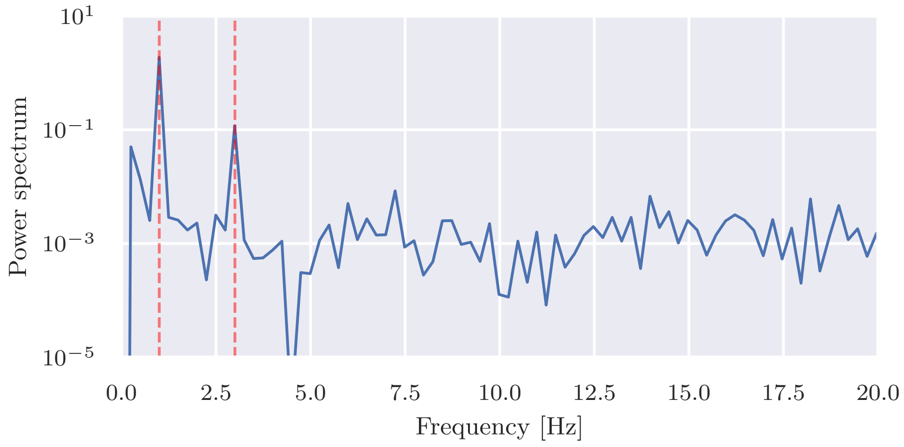


Figure 2.3: Periodogram for the time series shown in Figure 2.1. The time series is detrended assuming a linear trend. Observe that the two highest peaks in the periodogram are at the underlying frequencies 1Hz and 3Hz shown with dashed red lines.

2.3.3 Spectral window

The fact that the variance does not decrease with the sample size produces quite jagged and noisy spectrum estimations [11]. To account for this we introduce the spectral window which smooths the spectrum. Mathematically the smoothed spectrum is written as

$$\hat{f}_{\mathcal{W}}(\omega_k) = \sum_{j=-m}^{m} \mathcal{W}_n(\omega_j)\hat{f}(\omega_k - \omega_j), \tag{2.18}$$

where $\omega_k = 2\pi k/n,\ k = 0, 1, \ldots, (n-1)/2$ and m is a function of n, typically $m \ll n$. The value of m decides how many points in the neighborhood of ω_k should be included in the smoothing. Furthermore, the function $\mathcal{W}_n(\omega_j)$ is chosen to have the following properties,

$$\begin{aligned} &\sum_{j=-m}^{m} \mathcal{W}_n(\omega_j) = 1, \\ &\mathcal{W}_n(\omega_j) = \mathcal{W}_n(-\omega_j), \\ &\lim_{n\to\infty} \sum_{j=-m}^{m} \mathcal{W}_n^2(\omega_j) = 0. \end{aligned} \tag{2.19}$$

We view the smoothed spectrum as a weighted average of the sample spectrum (2.17) in a window around the target frequency ω_k. How the weights are distributed in the window is governed by $\mathcal{W}_n(\omega_j)$, giving it the name spectral window. Because of the property in (2.19) we have

$$\begin{aligned}\operatorname{Var}(\hat{f}_{\mathcal{W}}(\omega_k)) &\approx \sum_{j=-m}^{m} \mathcal{W}_n^2(\omega_j)(f(\omega_k))^2 \\ &= (f(\omega_k))^2 \sum_{j=-m}^{m} \mathcal{W}_n^2(\omega_j) \overset{n\to\infty}{=} 0,\end{aligned}$$

assuming $f(\omega)$ is approximately constant in the window. We can thus reduce the variance by increasing the points included in the window. However, when doing so we introduce bias.

2.3.4 Lag window

It can be shown [10] that the spectral window forms a Fourier transform pair with a lag window $W_n(k)$, i.e,

$$W_n(k) = \int_{-\pi}^{\pi} \mathcal{W}_n(\omega) e^{i\omega k} d\omega, \quad k = 0, \pm 1, \dots, \pm M.$$

The lag window is a weighting function applied to the sample autocovariance

$$\hat{f}_{\mathcal{W}} = \frac{1}{2\pi} \sum_{k=-M}^{M} W_n(k)\hat{\gamma}_k e^{-i\omega k},$$

with $W_n = W(k/M)$, W being a bounded even continuous function

$$\begin{aligned}&|W(t)| \le 1,\\ &W(0) = 1,\\ &W(t) = W(-t),\\ &W(t) = 0, \quad |t| > 1.\end{aligned}$$

Figure 2.4 shows the popular *hanning* window given by

$$W_n^H = \begin{cases} \frac{1}{2}(1 + \cos(\frac{\pi k}{M})), & |k| \le M \\ 0, & |k| > M, \end{cases} \tag{2.20}$$

with $M = 5$.

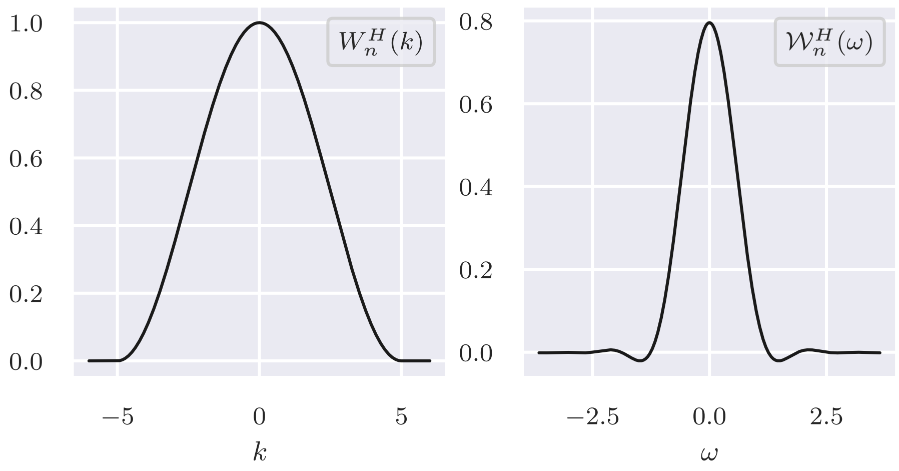


Figure 2.4: *Left*: The hanning window given by Equation (2.20). *Right* The corresponding spectral window, i.e., its Fourier transform.

2.4 Time-frequency analysis

The methods mentioned so far are only suitable for extracting information about the frequencies of a signal. If the process is stationary this is often all we need. However, if the process is non-stationary and the frequencies change over time, the periodogram is unable to capture this. The original time series contains all the information in the time scale, and the Fourier transform contains all the information in the frequency scale. We need something in-between.

2.4.1 Short-time Fourier transform

A simple and intuitive way to gain information about the changes in frequency is to divide the time interval into sections and compute the Fourier transform on each section. Then we can plot how the periodogram changes over time. If we divide the time interval using a window function this method is called the short-time Fourier transform (STFT). Mathematically we define it for a discrete sequence $y = y_1, y_2, \ldots, y_n$, which is windowed by $W_n(t)$ around time τ, as

$$S_y(\omega, \tau) = \frac{1}{2\pi} \sum_{t=-\infty}^{\infty} y_t W_n(t-\tau) e^{-i\omega t}. \tag{2.21}$$

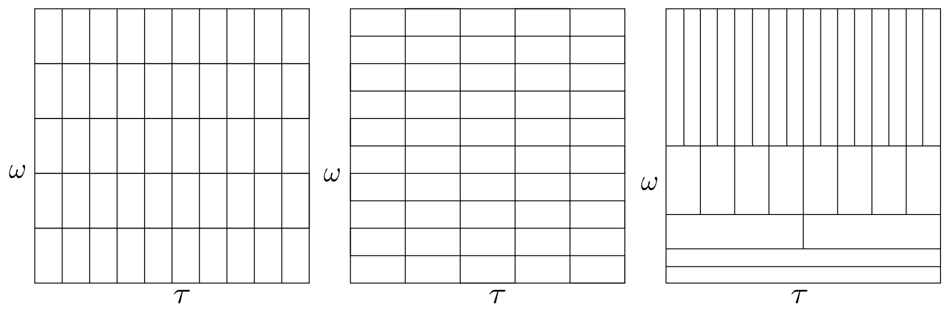


Figure 2.5: *Left/middle:* The figures illustrate the tradeoff in resolution for time and frequency for the short-time Fourier transform from Equation (2.21), increasing the resolution in time decreases the resolution in frequency and vice versa. *Right:* Illustrates the corresponding resolution for the continuous wavelet transform from Equation (2.22).

The estimated spectrum $|S_y(\omega,\tau)|^2$ is now a function of both time and frequency and is commonly called the spectrogram [12], which can be plotted to show the changes in frequencies over time.

Unfortunately, the STFT has some severe drawbacks. It can be shown that there exists a limit to the possible precision achieved when measuring both frequency and time simultaneously [13]. There exists a tradeoff in time and frequency resolution, as shown in Figure 2.5. This is very intuitive as frequency has to be measured over a certain time period. Decreasing the width of the window function makes detecting smaller frequencies harder, and vice versa. Thus we need to know the approximate frequency scales in the data before performing the STFT, or we can try several window widths. In the case where frequencies exist in the data on several different scales, the STFT in Equation (2.21) becomes unsuitable.

2.4.2 Wavelet transform

The wavelet transform works in a very similar way to the STFT, but circumvents the problem of choosing a fixed width of the window by introducing scaling. Let $\psi(t)$ be a window function (possibly complex-valued) which we will call the mother wavelet. One way of scaling the mother wavelet by a factor s is

$$\psi_s(t) = \frac{1}{s^{1/2}}\psi\left(\frac{t}{s}\right).$$

If we also allow a time shift around τ we arrive at the wavelets

$$\psi_{s,\tau}(t) = \frac{1}{s^{1/2}} \left(\frac{t-\tau}{s} \right).$$

This culminates in the definition of the continuous wavelet transform (CWT) of a function f,

$$\tilde{f}(s,\tau) = \int_{-\infty}^{\infty} \psi^*_{s,\tau}(t) f(t) dt, \tag{2.22}$$

where $\psi^*_{s,\tau}$ is the complex conjugate of the function $\psi_{s,\tau}$ [13].

Again considering the discrete sample $y_1, y_2, \dots, y_n$ with $\delta t = y_{t+1} - y_t,\ t = 1, \dots, n$, its continuous wavelet transform is given by [14]

$$\tilde{f}_n(s,\tau) = \sum_{t=1}^{n} y_t \psi^* \left(\frac{\delta t}{s} (t-\tau) \right), \tag{2.23}$$

where τ now is a time index $\tau \in \{1, 2, \dots, n\}$. It is computationally efficient [14] to represent the CWT as the inverse Fourier transform

$$\begin{aligned} \tilde{f}_n(s,\tau) &\sum_{t=1}^{n} \hat{y}_k \hat{\psi}^*(s\omega_k) e^{i\omega_k n \delta t}, \\ \hat{y}_k &= \frac{1}{n} \sum_{t=1}^{n} y_t e^{-\frac{i2\pi k t}{n}}, \\ \omega_k &= \begin{cases} \frac{2\pi k}{n\delta t}, & k \le \frac{n}{2} \\ -\frac{2\pi k}{n\delta t}, & k > \frac{n}{2} \end{cases}. \end{aligned}$$

As always we are interested in the power spectrum, which for the CWT is defined as $|\tilde{f}_n(s,\tau)|^2$.

There exist many choices for the mother wavelet, and certain important criteria it must satisfy [13]. One popular wavelet is the Morlet wavelet defined as

$$\psi_0(\eta) = \pi^{-1/4} e^{i\omega_0 \eta} e^{-\eta^2/2}, \tag{2.24}$$

where ω_0 is a parameter chosen to fit the criteria.

In Figure 2.6 a scaleogram is shown for the function

$$f(t) = \begin{cases} 3\sin(2.4 \cdot 2\pi t) + 2\sin(4.7 \cdot 2\pi t) + \sin(17 \cdot 2\pi t), & t < 30 \\ \sin(0.7 \cdot 2\pi t) + 2\sin(1.4 \cdot 2\pi t) + 3\sin(6.4 \cdot 2\pi t), & t \ge 30, \end{cases} \tag{2.25}$$

with added Gaussian noise. The continuous wavelet transform is computed at 18 frequencies between 0.5Hz and 20 Hz.

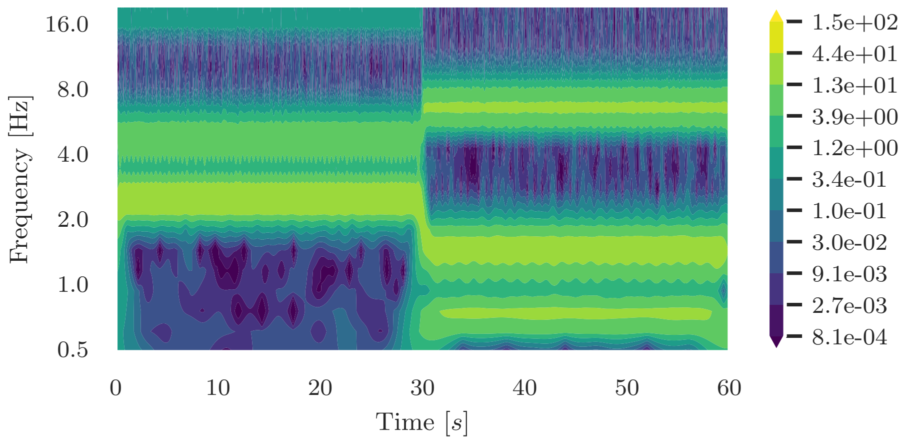


Figure 2.6: An example of a scaleogram of the continuous wavelet transform of function (2.25) with added Gaussian noise.

2.5 Piecewise polynomials

Suppose we have an interval $[a, b]$ divided into M contiguous subintervals. The connecting edges of the subintervals $a = \xi_0, \xi_1, \dots, \xi_{M-1}, \xi_M = b$ are called knots. On each of the intervals $[\xi_i, \xi_{i+1}], i = 0, \dots, M-1$ we define a polynomial $p_i(t)$. The function

$$f(t) = \begin{cases} p_0(t), & t \in [\xi_0, \xi_1) \\ p_1(t), & t \in [\xi_1, \xi_2) \\ & \vdots \\ p_{M-1}(t), & t \in [\xi_{M-1}, \xi_M] \end{cases}$$

is called a *piecewise polynomial.*

2.5.1 Splines

In the definition of piecewise polynomials no restrictions are made on the polynomials, they are allowed to take any form. As in [15] we define a *spline* $s_k(t)$ of order k on the interval $[a, b]$ as a piecewise polynomial where

$$s_k(t) \in \mathcal{P}^k, \quad t \in [\xi_i, \xi_{i+1}], \quad i = 0, 1, \dots, M-1$$
$$s_k(t) \in \mathcal{C}^{k-1}[a, b].$$

The spline consists of piecewise polynomials of order k and has continuous derivatives up to order $k-1$. A common choice is letting $k=3$, providing continuous second derivatives over the interval. These are called *cubic* splines and are often considered sufficiently smooth for function approximations. It is also common to add curvature constraints at the endpoints, $s_3''(a) = s_3''(b)$, arriving at the *natural* cubic splines.

2.5.2 Regression splines

Suppose now we have data points $y_{t_1}, y_{t_2}, \dots, y_{t_n}$ on $[a = t_1, b = t_n]$. A spline of order k with chosen knots at $a = t_1 = \xi_0, \xi_1, \dots, \xi_M = t_n = b$ can be parameterized as

$$s_k(t) = \sum_{i=1}^{M+k} \beta_i h_i(t), \tag{2.26}$$

where the functions h_i are the truncated-power basis set

$$\begin{aligned} h_j(t) &= t^{j-1},\ j = 1, \dots, k+1, \\ h_{k+1+l}(t) &= (t - \xi_l)_+^k,\ l = 1, \dots, M-1, \end{aligned}$$

with $(t)_+ = \max\{t, 0\}$ [16]. The parameters β_i can be found using least squares. An example of cubic spline regression is shown in Figure 2.7.

2.6 Dimensionality reduction

Suppose we have n data points, each having p numerical features. Mathematically we represent them as the vectors $x_i = (x_{i1}, x_{i2}, \dots, x_{in}) \in \mathbb{R}^p,\ i = 1, 2, \dots, n$, often represented in the $n \times p$ data matrix $X,\ X_{ij} = x_{ij},\ i = 1, 2, \dots, n,\ j = 1, 2, \dots, p$. Dimensionality reduction is about finding a low-dimensional representation of the data containing most of its properties. There can be many reasons for performing dimensionality reduction, and often it is performed as a form of feature extraction used before predictive modeling or clustering [16]. If the data is sparse or the features highly correlated, reducing the dimensionality could improve computation speed and remove noise. Reducing the data to two dimensions often allows for easier visualization and interpretability of the data.

2.6.1 Principal component analysis

One of the most commonly used dimensionality reduction methods is called principal component analysis (PCA). There are multiple ways to both derive and interpret the method. Given the $n \times p$ data matrix X we find a sequence of $q \leq p$ orthogonal $p \times 1$ unit vectors $v_1, v_2, \dots, v_q$, each one chosen such that $Z_j = Xv_j$

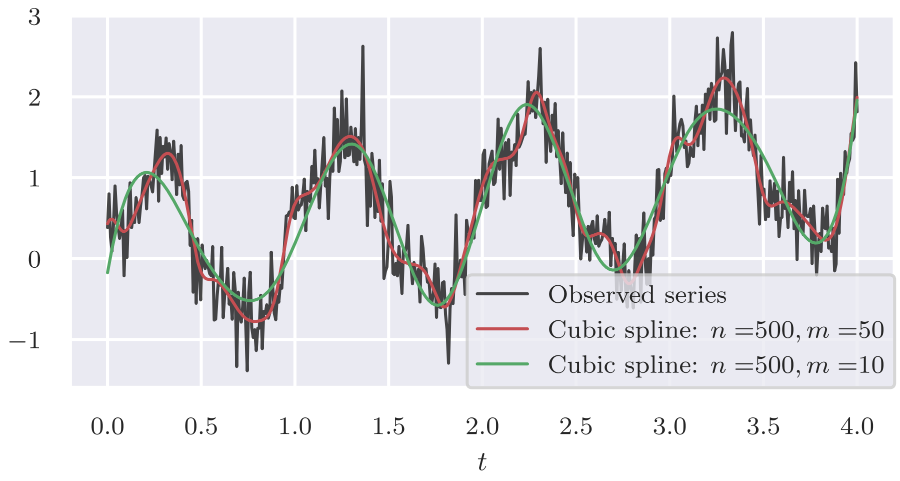


Figure 2.7: Two cubic splines fitted using least square regression on the time series from Figure 2.1. Observe that the red spline with 50 knots (including endpoints) fits the data closer than the green spline with 10 knots.

has maximum variance. In matrix form, we write it as $Z = XV_q$, where the columns of V_q are $v_j, j = 1, 2, \ldots, q$. The principal components scores—which represent our new features when applying dimensionality reduction—is given by the q columns in Z. Let S be the sample covariance matrix of X. It can be shown [17] that the vectors v_j that maximizes the variance while subject to the mentioned constraints are the eigenvectors of S, with the corresponding eigenvalue λ_j. Note that $\lambda_i > \lambda_{i+1},\ i = 1, 2, \ldots, p-1$. For this reason, we need to standardize the columns of X to have mean zero and variance one before applying PCA. The proportion of the total variance explained by the first q principal components is given by

$$\frac{\sum_{i=1}^{q} \lambda_i}{\sum_{j=1}^{p} \lambda_j}.$$

When using PCA for dimensionality reduction we set a variance threshold, and keep the number of principal components maintaining sufficient variance.

2.6.2 t-Stochastic Neighbor Embedding

Another dimensionality reduction method especially suited for visualizing data in two dimensions is called t-Stochastic Neighbor Embedding (t-SNE). Again, we start with the $n \times p$ data matrix and wish to embed it into a low-dimensional space preserving as much of the original structure in the data as possible. Instead of trying to emulate the Euclidian distances between the data points $x_1, x_2, \ldots, x_n$, we convert them into conditional probabilities

$$p_{j|i} = \frac{\exp\left\{\frac{-||x_i - x_j||^2}{2\sigma_i}\right\}}{\sum_{k \neq i} \exp\left\{\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right\}}, \quad i \neq j,\ p_{i|i} = 0. \tag{2.27}$$

For a point x_i, $p_{j|i}$ is large for nearby points x_j, with magnitude proportional to a Gaussian distribution centered at x_i with variance σ_i^2 [18]. The individual variance σ_i^2 is a tuning parameter to be chosen by the user.

In the low-dimensional representation, we model the similarities as joint probabilities based on the Student t-distribution with one degree of freedom

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}, \quad i \neq j,\ q_{ii} = 0. \tag{2.28}$$

It is beneficial to define a joint probability distribution in the high dimensional space as with $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$ [18]. To find the low dimensional embedding Y, t-SNE minimizes the total Kullback-Leibler divergence between the discrete joint probability distributions

$$KL(P||Q) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right). \tag{2.29}$$

The choice of using the Student-t distribution in Equation (2.28) as opposed to a Gaussian (as in the high dimensional space (2.27)), stems from its heavier tails. This allows data points that are relatively dissimilar in the original data, but still close enough to make a difference in the cost function, to be modeled far apart in the low dimensional embedding. I.e., t-SNE focuses on the local structures in the data and allows clusters to be well separated in the embedding.

Furthermore, we need to choose the tuning parameters σ_i. They relate to the Gaussian distribution modeling the neighbors of x_i. For points with many close-by neighbors it makes sense to keep σ_i small, while increasing it is beneficial when there are few neighbors. The algorithm solves this by letting the user choose a perplexity parameter

$$\text{Perp}(P_i) = 2^{H(P_i)}, \tag{2.30}$$

where

$$H(P_i) = -\sum_{j=1}^{n} p_{j|i} \log_2(p_{j|i})$$

is the Shannon entropy. Then it finds σ_i such that the perplexity satisfies our choice. Van der Maaten and Hinton [18] suggest values between 5 and 50, roughly reflecting the number of neighbors in the algorithm.

2.7 Kernel density estimation

Let $x_1, x_2, \ldots, x_n,\ x_i \in \mathbb{R}^p$ be a random sample from a random variable, with an underlying unknown density function $f(x)$ which we want to estimate. The general expression for the multivariate kernel density estimator is [19]

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_p\left[H^{-1}(x - x_i)\right], \tag{2.31}$$

where $|H| = |\det(H)|$, H being the $p \times p$ non-singular bandwidth matrix. The function $K_p : \mathbb{R}^p \to \mathbb{R}$ in (2.31) is called the kernel function, and must satisfy

$$\int_{\mathbb{R}^p} K_p[u]du = 1, \quad \int_{\mathbb{R}^p} uK_p[u]du = 0 \in \mathbb{R}^p, \quad \int_{\mathbb{R}^p} uu^T K_p[u]du = I_p,$$

I_p being the $p \times p$ identity matrix [19]. We can view $\hat{f}(x)$ as a mixture of densities centered at the observations $x_1, x_2, \ldots, x_n$. The bandwidth matrix H needs to be selected by the user. Changing the bandwidth matrix has a large impact on the resulting estimated density, and is often chosen to suit the problem at hand. An example of kernel density estimation is shown in Figure 2.8.

One common way to select the bandwidth matrix is to minimize the asymptotic mean integrated squared error

$$\text{AMISS} = \frac{1}{nh^p}\int K(u)^2 du + \frac{h^4}{4}\int \{\text{tr}[AA^T\nabla^2]f(u)\}du, \tag{2.32}$$

where $H = hA$, with A having a unit determinant [19]. There is no closed-form solution, but (2.32) shows that the bandwidth h should be $\mathcal{O}(n^{-4/(p+4)})$. When using a Gaussian kernel on a multivariate normal distribution, the optimal bandwidth matrix is diagonal with elements

$$H = \left(\frac{4}{p+2}\right)^{1/(p+4)} \Sigma^{\frac{1}{2}} n^{-1/(p+4)}.$$

For $p = 2$, the estimated bandwidth matrix

$$\hat{H} = \hat{\Sigma}^{\frac{1}{2}} n^{-1/6} \tag{2.33}$$

is known as Scott's rule of thumb [20].

2.8 Watershed segmentation

Watershed segmentation is an image processing technique commonly used to separate objects in an image. Assuming a greyscale image it treats it like a

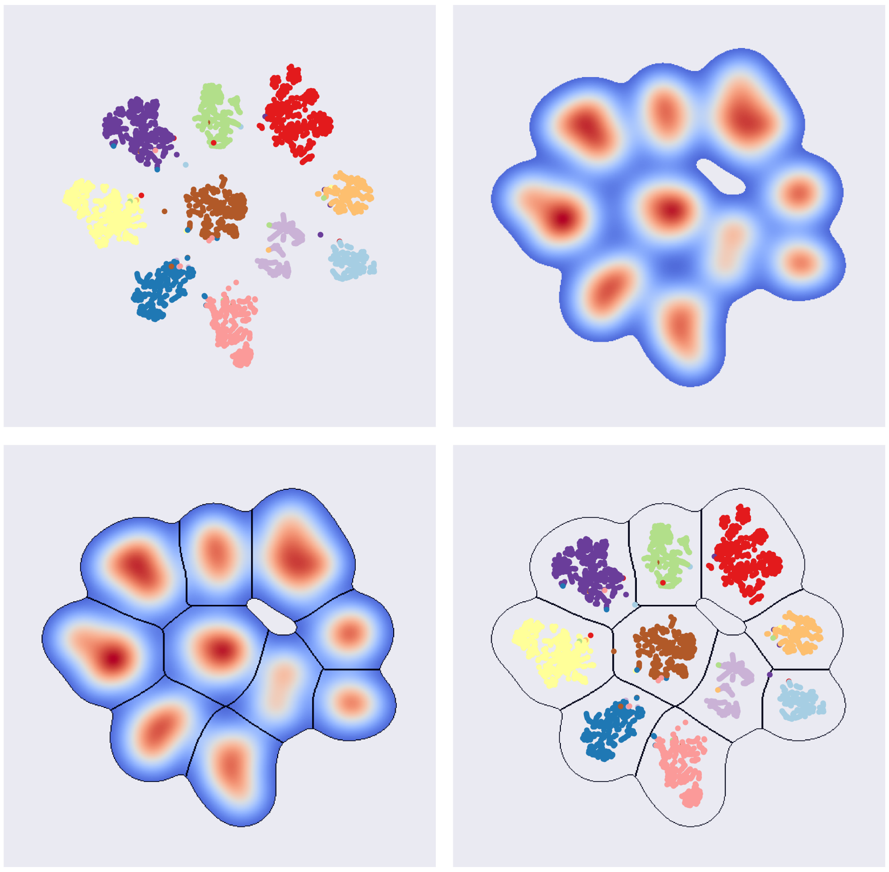


Figure 2.8: Clustering the two-dimensional t-SNE embedding. *Top left*: t-SNE embedding of simulated data, color-coded by actual behavior. *Top right*: Kernel density estimation of the t-SNE embedding. *Bottom left*: Watershed segmentation applied to the kernel density estimation. *Bottom right*: Contours of the watershed segmentation with the t-SNE clusters shown inside.

topological map where intense areas are considered to have large altitude [21]. Using an estimated gradient the algorithm finds the local minima in the image, gradually ”floods” the topography with water, and marks the lines where the filled basins meet. The watershed segmentation applied to an estimated probability density function is shown in Figure 2.8.

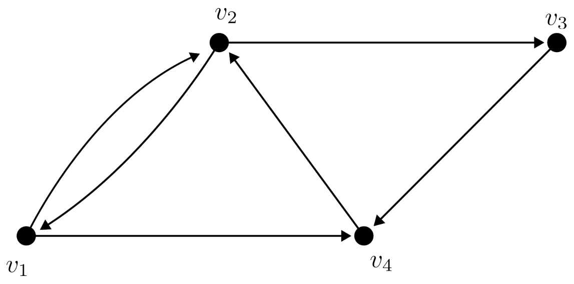


Figure 2.9: Example of a directed graph with four vertices and six edges.

2.9 Applied topology

2.9.1 Directed graph

A graph $\mathcal{G} = (V, E)$ is a collection of vertices V and edges E, where both V and E are finite non-empty sets [22]. When E is ordered pairs of vertices we call $\mathcal{G}$ a directed graph. Figure 2.9 shows a directed graph with vertices $\{v_1, v_2, v_3, v_4\}$ and edges $\{(v_1, v_2), (v_2, v_1), (v_2, v_3), (v_4, v_2), (v_1, v_4)\}$.

2.9.2 Simplicial complexes

Let $\mathcal{S}$ be a discrete set. A collection of finite subsets of $\mathcal{S}$, X, is called an abstract simplicial complex if for all finite subsets $\sigma \in X$, its subsets τ are also subsets of X [23]. We call σ k-simplices, where $|\sigma| = k + 1$, and the simplices $\tau \subset \sigma$ its corresponding faces. If X and σ are ordered sets, we call call X an abstract directed simplicial complex. X_k is defined as the set of all k-simplices of X.

The simplicial complex X is turned into a topological space by what is called a geometric realization [23]. For a given k, we find all k-simplices of and *attach* them along their common faces. The k-skeleton is defined as $X^{(k)} = X_k \cup X_{k-1} \cup \ldots \cup X_0$, with the dimension of X being defined as the smallest k such that $X = X^{(k)}$ [24].

2.9.3 Flag complexes

We now wish to construct an abstract simplicial complex from a graph. To achieve this, we define the flag complex of a graph as the maximal simplicial complex having the graph as its 1-skeleton [23]. In the specific case of a directed graph $\mathcal{G} = (V, E)$, the directed flag complex $F(\mathcal{G})$ has 0-simplices $F_0 = V$, and all other k-simplices $F_k, k \geq 1$ are $k + 1$ tuples $(v_0, v_1, \ldots, v_k)$ such that $v_i \in V$ and $(v_i, v_j) \in E, 0 \leq i < j \leq k$. For such a simplex $(v_0, \ldots, v_k)$, v_0 is called the source, and v_k the sink. The flag complex of the graph shown in Figure 2.9 has 0-simplices $\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}$, 1-simplices $\{v_1, v_2\}, \{v_2, v_1\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_2\}$ and one 2-simplex $\{v_1, v_4, v_2\}$.

2.10 Homology

To describe high-level features of simplicial complexes we use homology. Again, let X be an ordered simplicial complex and $\mathbb{F}$ a field. Then, for $k \geq 0$ we let $C_k(X)$ be a vector space with basis elements being the k-simplices of X, X_k. If $\sigma = \{v_0, v_1, \ldots, v_k\}$ is a k-simplex in X, we denote the face where v_i is removed $\sigma^i(\sigma) = \{v_0, \ldots, \hat{v}_i, \ldots, v_k\}$. We extend this notion in the k-th boundary map $\partial_k : C_k(X) \to C_{k-1}(X)$

$$\partial_k(\sigma) \sum_{i=0}^{k} (-1)^i \partial_i(\sigma),$$

for $k \geq 1$ [25]. It can easily be checked that $\partial_k \circ \partial_{k+1} = 0, k \geq 1$. Using this fact we can define the important k-th homology group as

$$H_k(X) = H_k(X; \mathbb{F}) = Z_k / B_k,$$

where $Z_k = \mathrm{Ker}(\partial_k) \subseteq C_k$ is called the k-cycles and $B_k = \mathrm{Im}(\partial_{k+1}) \subseteq C_k$ the k-boundaries. The rank of $H_k(X)$ is called the k-th Betti number and represents the number of holes in X.

2.10.1 Persistent homology

Where homology deals with finding topological features in structures, persistent homology aims at finding the important features by varying the spacial scale. To achieve this we need some sort of distance metric on the simplicial complexes. First we define a filter function $f : X \to \mathbb{R}$ which has the property that if τ is a face of a simplex σ in X, then $f(\tau) \leq f(\sigma)$ holds. We call X together with f a filtered simplicial complex. A sublevel complex is then defined as $X[b] = f^{-1}((-\infty, b])$ for a given $b \in \mathbb{R} \cup \{\infty\}$. For a simplicial complex X we can obtain a nested sequence of subcomplexes

$$X[1] \subseteq X[2] \subseteq \cdots \subseteq X[n] = X,$$

which is also known as a filtration. For this filtration we have a homomorphism

$$h_k^{i,j} : H_k(X[i]) \to H_k(X[j]),\ i \leq j,\ k \geq 0,$$

induced by the inclusion map $X[i] \to X[j]$ [25]. The image $\mathrm{Im}(h_k^{i,j})$ is the k-th persistent homology group corresponding to the given filtration. We call its rank a persistent Betti number, which represents the number of homology classes that exist in $X[i]$ and still exists in $X[j]$. Let α be a homology class. We say that it is born at $X[i]$ if it is not in the image of the map induced by the inclusion $X[i-1] \subset X[i]$ [26]. Similarly, we say α dies at $X[j]$ if the image of the map induced by $X[i-1] \subset X[j]$ contains α, while the image of the map induced by $X[i-1] \subset X[j-1]$ does not, given that α is born at $X[i]$. The persistence $j-i$ is a measure of how long the homology class lives within the filtration.

We can visualize the persistence in a filtered simplicial complex X with its persistence diagram, $\mathrm{Dgm}_p(f)$. The persistence diagram is a multiset of points in $\bar{\mathbb{R}}^2$, where a point $u = (i, j)$ is included if there exists a homology which is born at $X[i]$, and dies at $X[j]$.

It is also possible to compare such persistence diagrams. First, we state the L_∞ norm, $||p-q||_\infty = \max\{|p_1 - q_1|, |p_2 - q_2|\}$. Let $\mathrm{Dgm}_p(f)$ and $\mathrm{Dgm}_p(g)$ be two persistence diagrams. They are defined by their multisets of points P and Q, respectively. The bottleneck distance between $\mathrm{Dgm}_p(f)$ and $\mathrm{Dgm}_p(g)$ is defined as

$$d_{\mathrm{B}}(P, Q) = \inf_{\gamma} \sup_{p} ||p - \gamma(p)||_\infty, \tag{2.34}$$

where $p \in P$ and γ is a bijection from P to Q [27]. An example of two persistence diagrams plotted together with their bottleneck distance marked is shown in Figure 2.10. The bottleneck distance has nice stability properties, as shown in [27]. Summarized, small perturbations in the input space give small changes in the bottleneck distance between the persistence diagrams.

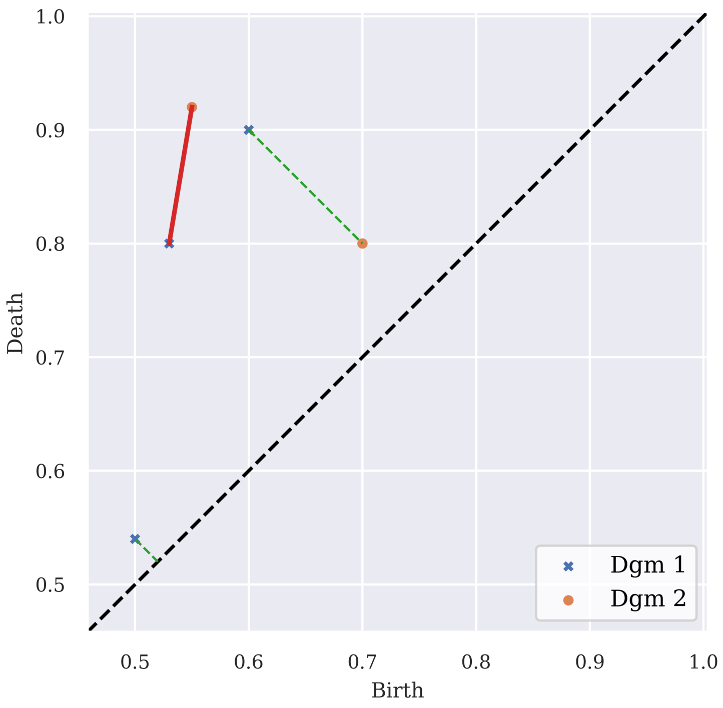


Figure 2.10: Example of two persistence diagrams, representing the multisets of points $\{(0.6, 0.9), (0.53, 0.8), (0.5, 0.54)\}$ and $\{(0.55, 0.92), (0.7, 0.8)\}$. The red line is the bottleneck distance, and the green stapled lines illustrates the corresponding optimal bijection.

Chapter 3

Methodology

The starting point of the analysis is a set $\mathbf{Y} = \{Y_d : d = 1, 2, \ldots, D\}$, where Y_d is again a set of independent time series, $Y_d = \{y_1^d(t), y_2^d(t), \ldots, y_n^d(t) : t \in \{t_1, t_2, \ldots, t_{m_d}\}\}$. Each Y_d represents an animal for which n time series are collected tracking parts of its movements. Recording frequency is the same across animals. The goal of the analysis is to cluster these time points into distinct distinguishable actions. Looking at detrended motion recordings as periodic signals, we extract information about the periodicity in the movements at a range of frequencies through time-frequency analysis. These extracted features are embedded into two dimensions, where we can use image segmentation techniques on an estimated two-dimensional probability distribution.

Sections 3.1 to 3.3 are restated from the specialization project [9].

3.1 Feature extraction

3.1.1 Detrending

The first step is to analyze each time series separately, modifying them and thus creating $D \times n$ features which we will extract the behavioral information from. Let $y_t = \{y_1, y_2, \ldots, y_m\} = y_i^d(t)$ for some $d \in \{1, 2, \ldots, D\}$ and $i \in \{1, 2, \ldots, n\}$ be one such series.

We detrend the data as $y_t = s_3(t) + x_t$, $s_3(t)$ being the non-linear trend and x_t the detrended time series. The cubic spline $s_3(t)$ is found using least square regression as in Equation (2.26) with equally spaced internal knots $\xi_1, \xi_2, \ldots, \xi_M$.

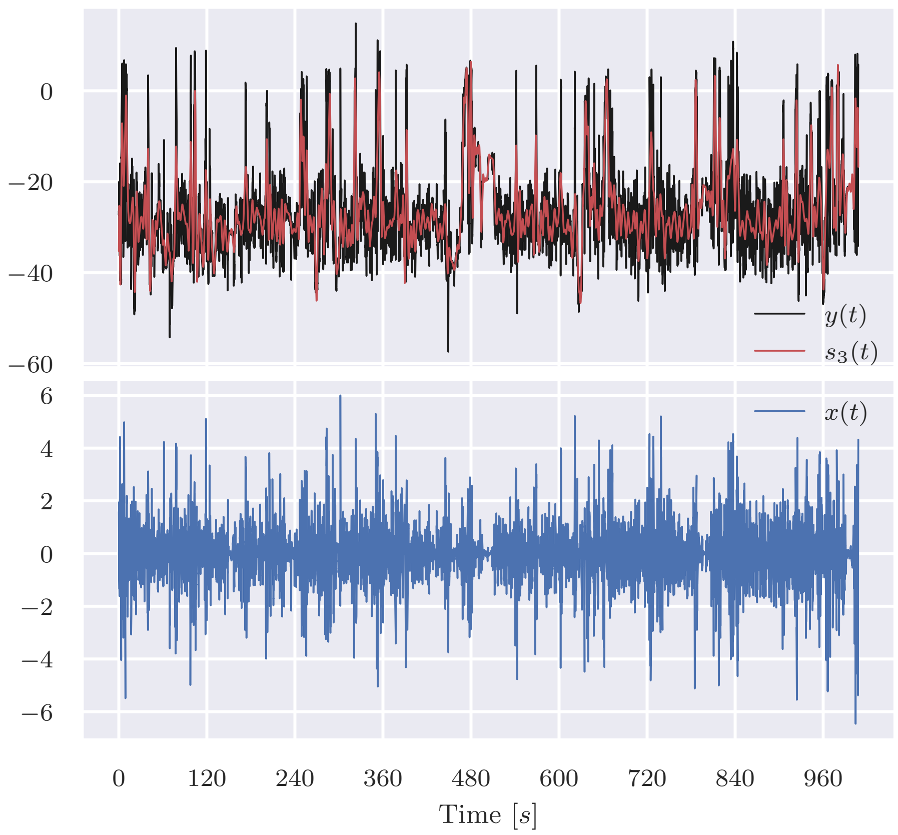


Figure 3.1: Detrended time series using cubic spline regression. The top figure shows the original time series in black with the fitted trend in red. The bottom figure shows the detrended time series.

We choose the knots to have a fixed frequency, i.e., we choose $\Delta\xi = \xi_{i+1} - \xi_i, i = 2, \dots, M$. Figure 3.1 shows an example of a detrended time series using cubic splines with interior knots every 240th-time point.

3.1.2 Time-frequency analysis

To make the extracted features comparable, we normalize the time series by dividing it by its standard deviation

$$\sigma_x = \frac{1}{m}\sum_{t=1}^{m}\left(x_t - \frac{1}{m}\sum_{i=1}^{m} x_i\right)^2 .$$

The detrended time series contain information about how the movements vary around the trend. We assume this movement is repetitive in nature and can be interpreted as a signal. By applying time-frequency analysis to the detrended time series, we obtain additional information about how the frequencies of these movements vary over time. In turn, this can contribute to detecting behavioral clusters.

As we are unaware of at which scales these movements occur beforehand, we opt for a continuous wavelet transform as opposed to the short-time Fourier transform. Let J be the number of predefined frequency scales, ranging from $\omega_{\min}$ to $\omega_{\max}$. The scales $s_1, s_2, \dots, s_J$ in Equation (2.23) are chosen as fractional powers of two—as suggested in [14]—e.g.,

$$s_j = \frac{2^{(j-1)\delta j}}{\omega_{\max}}, \quad j = 1, 2, \dots, J, \tag{3.1}$$

$$\delta j = \frac{1}{J-1}\log_2\left(\frac{\omega_{\max}}{\omega_{\min}}\right). \tag{3.2}$$

The choice of mother wavelet can for instance be based on expected features in the time series [14]. Using Equation (2.23) we compute the discrete continuous wavelet transform $\tilde{f}_m(s,\tau)$, and in turn the scaled scaleogram

$$\frac{1}{s_j}\left|\tilde{f}_m(s_j,t)\right|^2, \quad j = 1, 2, \dots, J, \; t = 1, 2, \dots, m. \tag{3.3}$$

The scaling is done to keep the power comparable across the varying scales [28]. An example plot of such a scaleogram is shown in Figure 3.2. To make the estimated power comparable we take the square root before proceeding

3.2 Manifold embedding

After performing the time-frequency analysis on every time series for all animals, we concatenate the results into a new feature vector. For animal d, we have m_d time points, each containing information about the n trend values, together with the $n \cdot J$ frequency powers. The concatenated feature vector thus has dimension

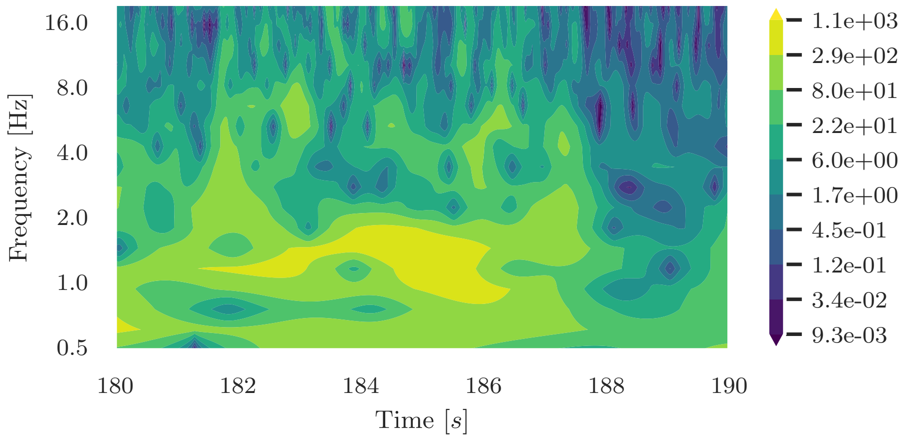


Figure 3.2: Scaleogram of the continuous wavelet transform performed on the detrended time series shown in Figure 3.1. The figure shows how the power at different frequency scales changes over time.

$(m_1 + m_2 + \cdots + m_D) \times (n \cdot (J + 1))$. To cluster these time points into behaviors, we first reduce the feature dimension. Logically, there should exist strong correlations between the various frequencies.

As a first step in dimensionality reduction, we use principal component analysis, keeping features explaining at least 95% of the variance. This both reduces further computational complexity and removes noise. To embed the data into two dimensions, we apply t-distributed stochastic neighbor embedding. It has one hyperparameter, the perplexity (2.30), which is chosen by inspecting some embeddings with values between 10 and 200. As t-SNE has quadratic memory complexity, we only embed a subset of the data. The subset is chosen through downsampling at a predefined frequency such that the number of training points is below 50000.

3.3 Clustering

We cluster the time points by dividing up the two-dimensional embedding into regions. First, we embed all points by giving each point the two-dimensional coordinates to the training point (used for t-SNE) with the smallest euclidean distance in the principal component space. Aiming to find regions with large clusters of points, we compute an estimated probability density through Gaussian kernel density estimation (2.31). Finally, we apply watershed segmentation

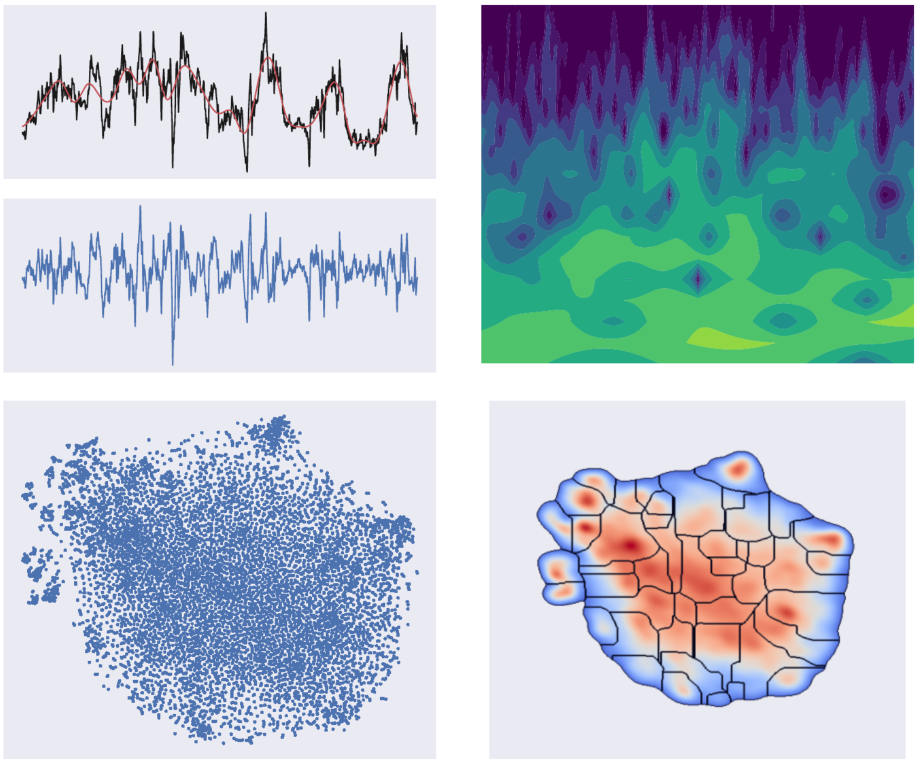


Figure 3.3: Methodology structure. *Top left:* Raw time series input data, detrended using spline regression. *Top right:* Scaleogram showing the power at various frequencies found via wavelet transformation. *Bottom left:* t-stochastic neighbor embedding of the principal components explaining 95% of the variance in the extracted features. *Bottom right:* Watershed segmentation of the kernel density estimation of the t-SNE embedding.

on the kernel density estimation, dividing the t-SNE plane into regions. The methodology structure is shown in Figure 3.3.

3.4 Feature frequency

Suppose that some of the feature recordings were captured at different frame rates while covering the same time interval. As PCA needs all input features to have the same dimension, the features need to be manipulated prior to the dimensionality reduction. A simple solution is to downsample all features onto the one with the lowest frequency. We can achieve this in multiple ways depending on the situation. One general approach is to interpolate the high-frequency features and compute the values at the low-frequency time points. This approach is also quite robust to missing values, given that the frequencies are sufficiently large.

3.5 Model selection

When performing the behavioral clustering we make a lot of choices affecting the resulting t-SNE embedding. Already when performing the wavelet transform, we choose both the number of frequency scales and the range. Then we choose the number of principal components used, how to downsample the features, the perplexity parameter for t-SNE, etc. Each selection produces a number of clusters representing animal behavior. How to choose between them is unfortunately not straightforward. The obvious metric would be how well the clusters each represent an actual behavior (if that even makes sense), and how well these behaviors are separated in the t-SNE plane. To do this, we would have to look at video recordings for every possible model, and manually decide the best one. In addition to being extremely time consuming, it defeats the purpose of this automatic clustering algorithm—that it is unsupervised. For this reason, it is common to choose sensible parameters, then be content if the clustering makes sense by inspecting the video material. If any of the behaviors are similar, we can then manually group them together into larger behavioral regions where the included behaviors share characteristic traits [5].

3.5.1 Partial labeling

Although we are not able to manually label the entire dataset, it is possible to label some easily identifiable assumed behaviors. After training the models we then observe how well they cluster these behaviors together. While still a manual task, it can give some insight into how the models differ. Using partial labels as an initial evaluation of a model, we can often rule out poorly performing models, before moving on with hyperparameter tuning on the more promising ones.

3.5.2 Spectral power averaging

Looking through the important features of each detected cluster is a common way to explore their meaning and properties in unsupervised clustering. As we expand the feature space with time-frequency analysis, there will easily be too many features for this to be a feasible approach. However, if we average the spectral power found by wavelet analysis over frequency ranges—either all of them or

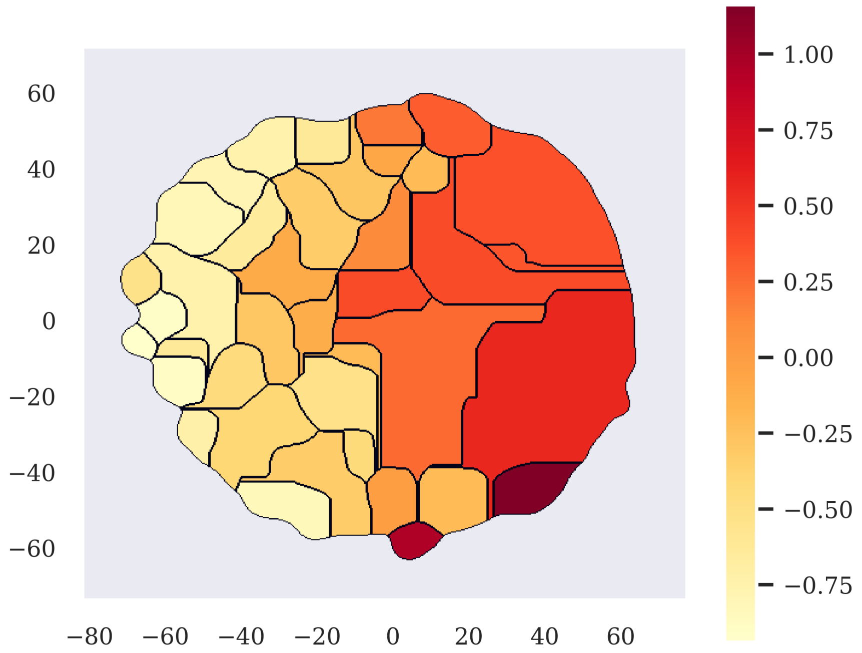


Figure 3.4: Averaged spectral power for a specific feature, computed for each detected behavioral cluster.

a subset—we obtain some indication of which features play an important role in the various detected behaviors. The averaged spectral power is plotted over the t-SNE plane like in Figure 3.4. In this example the plotted feature is more active in the behaviors found on the right side of the t-SNE plane. Together with partial labeling, this could provide stronger evidence that the model performing well. This is because some features are known to be active in the predefined behaviors. For instance, if an animal is looking around while standing still, the features measuring the neck movement should be active.

3.5.3 Comparing probability transition matrices

When tuning the hyperparameters using the partially labeled data, it is hard to compare the results just by looking at the plots. How can we determine the best one when we do not know what the behaviors represent, and if they make sense? This is commonly solved by selecting a set of parameters that seem reasonable, and checking a posteriori by looking at the behaviors in the recordings. If we had a way of comparing two models mathematically, we could justify selecting a model in a range of hyperparameters where the models are similar.

Two immediate difficulties when considering similarity metrics between clusterings are that the number of clusters differs between models, and the behavioral labels $1, 2, 3, \ldots$ varies. Behavior 23 in one model is not the same behavior as behavior 23 in another. We thus need to convert the models into structures that can be compared despite these differences.

The probability transition matrix describes contains the probabilities for moving from one behavior to another. It is estimated by taking the number of moves between two behaviors and dividing by the total number of time points. These matrices contain information about how the different behaviors act together. Shuffling the behavioral labels correspond to shuffling rows in the matrix. However, there is no apparent way of comparing such matrices of different sizes and order.

The next step is to view the probability transition matrices as weighed directed graphs. Each behavior corresponds to a node in the graph, and the transition probabilities to the weighted edges. We can then compare two such graphs using persistent homology. Either we compare their persistence diagrams visually, or by computing similarity metrics, e.g., the bottleneck distance (2.34).

Chapter 4

Data

To investigate the inclusion of facial tracking recordings as input features, we use data collected by the Whitlock group at Kavli Institute for Systems Neuroscience at NTNU. A Long Evans rat was recorded in 20 separate sessions during six days, each session lasting approximately ten minutes. In eleven of the sessions, two objects were present in the arena for the rat to interact with. The light conditions also varied across sessions, where 13 of them being light and seven dark. Seven postural features were recorded during each session, with a capture frame rate of 120 Hz. Together they record the head and back positioning, and the neck angle and forward speed. A more detailed description of the 3D postural motion recording can be found in [6].

Facial tracking has not been implemented in behavioral models by the group previously. To obtain the data, an open-source pose estimation tool on high-frequency facial video recordings is used to extract a series of individual whisker angles. The group provided this concise summary of the data collection procedure:

> The animal's whisker and eye movements are tracked using miniature cameras. A head-mounted set-up was custom-designed in the lab in collaboration with a local company (Inventas AS, Trondheim, NO) to combine the reflective markers for posture tracking with the facial tracking cameras and housing for a neural probe. The two facial tracking cameras (CMT, China; model: CMT-1MP-OV9281-C520 with 8464B-C Lens) record at 210 frames/s and each capture one eye and one whisker pad in their field of view.

> The recorded videos are subsequently analyzed using the DeepLabCut platform [29] to extract the position of points of interest from all video frames. Those are used to finally extract the angles of deflection of whiskers (10 whiskers on each side, from the 2nd and 3rd row in the whisker pad).
>
> [. . .] To allow for precise alignment of data from all three recording systems (Neuropixels neural recordings, 3D posture recording using the Optitrack system and the two facial tracking cameras) unique sequences of TTL pulses (250ms pulse duration, with inter-pulse intervals in the range of 250 - 1500ms) were produced using an Arduino Microcontroller and sent to all system to be recorded as a continuous synchronization signal (as a digital channel trace on the Nero pixels recording, and as a IR LED light recorded in the facial and posture tracking systems).

Of the 20 tracked whiskers, four of them are of poor quality due to the positioning of the camera and are excluded from the analysis. In total, this leaves us with seven postural features and 16 facial tracking features, recorded at 120 Hz and 210 Hz, respectively.

4.1 Data

The 20 recording sessions provide us with 1399722 postural samples recorded at 120 Hz, each containing seven features, and 2460083 facial tracking samples recorded at 210 HZ, each containing 16 features. Table 4.1 describes how the samples are distributed across the sessions, along with information about the light conditions and whether or not objects were present in the arena. Each of the 16 facial tracking features represents a unique whisker at either the left or right side of the rat's face. For various reasons, the postural tracking can at times be quite poor, resulting in one or more of the features containing missing values. Such samples, containing one or more missing values, were removed prior to the analysis. After removing the poor samples, we are left with 957941 postural samples. That the number of missing values is this high is not particularly concerning, as we know in which sessions the tracking is bad, and since the missing values are grouped together on intervals, leaving them out does not affect the time-frequency analysis negatively.

4.2 Partial labels

In addition to the data, the Whitlock group also provided partial labels to some sessions. Six easily identifiable rat behaviors were chosen and labeled across eight

Session	Postural samples	Facial samples	Lighting	Objects present
C4_1	71462	125597	Light	No
C4_2	71220	125137	Dark	No
C4_3	71002	124789	Dark	Yes
C5_1	71767	126134	Light	No
C6_1	71709	126033	Light	No
C6_2	61676	108398	Dark	No
C7_1	71182	125107	Light	No
C7_2	67453	118553	Dark	Yes
C8_2	71715	126043	Light	No
C8_3	71328	125361	Dark	Yes
C8_4	57756	101509	Light	Yes
C9_2	71712	126038	Light	Yes
C9_3	71701	126017	Light	Yes
C10_1	71708	126032	Light	No
C10_2	71509	125680	Dark	No
C10_3	71806	126203	Light	Yes
C11_1	71101	124964	Light	Yes
C11_2	71001	124788	Light	Yes
C11_3	71628	125891	Dark	Yes
C12_1	69286	121773	Light	Yes

Table 4.1: Recording session information for the 20 sessions used in the analysis. The number of recorded samples is stated, both from the postural and facial tracking sensors, along with the conditions in the arena.

of the sessions. Then the behaviors were divided into subcategories, described in Table 4.2.

4.3 Graphical user interface

An important tool for post hoc labeling of the behaviors is a graphical user interface (GUI) developed by the group. It is a custom plugin for the open-source image processing software Fiji [30], and has been employed in previous research [6] [31]. We use it as a visualization tool, where an animation of the rat is created using the features from OptiTrack. An example of the animation is shown in Figure 4.1. After training the model, and classifying the recording time points into a set of unknown behaviors, we can use the GUI to visualize the behaviors, and in turn label them a posteriori. Using csv-files containing start and end frames for the behaviors, the GUI can jump to the corresponding

Name	Description
Rearing wall 1	Rearing (standing on the hind legs) against the wall with simple whisking.
Rearing wall 2	Rearing against the wall with more complex whisking.
Rearing open	Rearing in the arena (not against the wall).
Eating	Eating snacks with the front paws.
Grooming face	Either scratching the face with font paws, or belly with front or back paws.
Freeze 1	Short freeze with whiskers being still.
Freeze 2	Longer freeze including some whisker movement.
Touching object 1	Interacting with objects with left whisker engaged.
Touching object 2	Interacting with objects with both whiskers engaged.
Touching object 3	Interacting with objects with both whiskers engaged, and the rat is climbing on the objects.
Running across 1	Casually paced trot across the arena.
Running across 2	Faster jumpy run across the arena.

Table 4.2: Partial labeling behavior descriptions.

parts of the recordings automatically. It is also possible to include synced video recordings in the analysis, side by side with the animation.

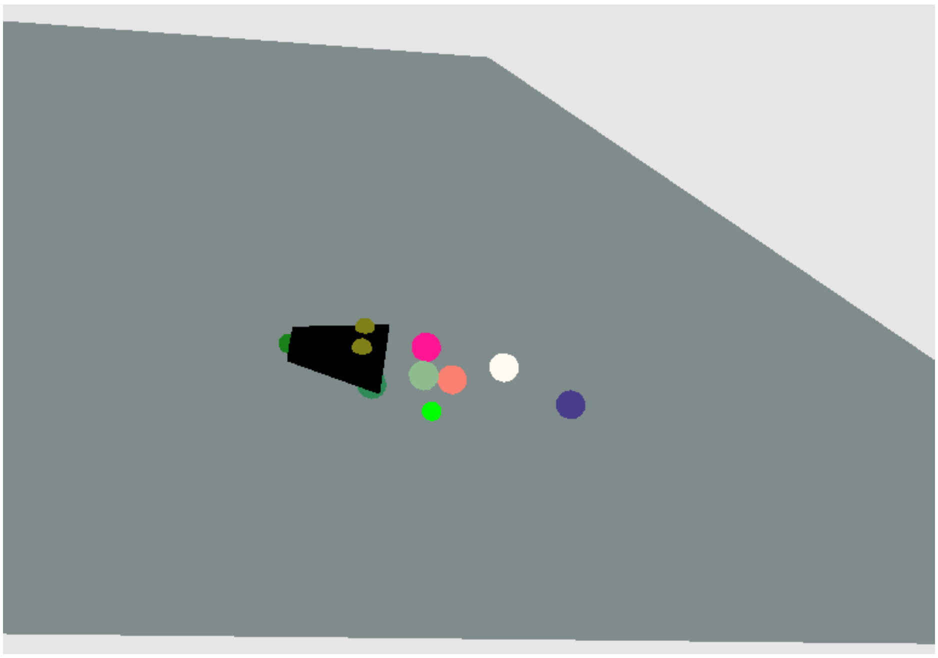


Figure 4.1: An example screenshot of the animated rat visualization obtained using the open-source image processing software Fiji [30].

Chapter 5

Exploratory data analysis

The clustering algorithm involves several hyperparameters—perplexity, bandwidth, etc—while also being open to tweaking several steps. Hence, we need to explore some options and do model selection, both with and without including the facial tracking features.

5.1 Postural features only

Our main goal is to show how including facial features changes the behavioral clustering output. To compare results, we need a benchmark using only the 7 postural features, recorded at 120 Hz. After detrending using cubic spline regression and normalizing across samples, we apply a continuous wavelet transform on each of the seven time series. We use a Morlet mother wavelet on 18 frequencies between 0.5 Hz and 20 Hz, chosen as in Equations (3.1) and (3.2). Adding each power spectrum to the feature space leaves us with a total of 133 features before performing the dimensionality reduction. First, these features are reduced to 66 dimensions by principal component analysis, the threshold chosen as the minimum number of components explaining at least 95% of the variance. The samples were subsampled at 3.758 Hz, resulting in 30902 training samples to be used in the t-SNE embedding. Finally, all samples are embedded in two dimensions using t-SNE with perplexity parameter 30. To estimate a probability density in the t-SNE plane, we apply kernel density estimation with the bandwidth chosen automatically by Scott's rule (2.33). Finally, we cluster the t-SNE embeddings using

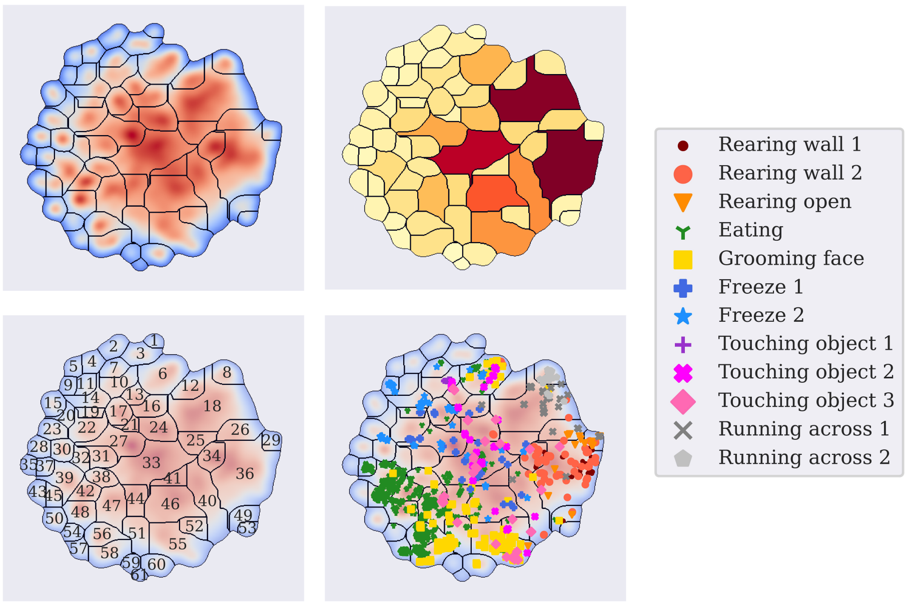


Figure 5.1: Visualization of trained model using only postural features. *Top left:* Heat map showing the estimated probability density in the t-SNE plane, divided into 61 behavioral regions. *Top right:* Intensity plot showing relative time spent in each of the behaviors. More intense color corresponds to large amount of time spent in behavior. *Bottom left:* Behavioral labels given to the various regions. *Bottom right:* Partial labels assigned to parts of the training data, based on visual inspection of video recordings.

a watershed segmentation on a 500 × 500 grid, yielding 61 distinct behaviors. A visualization of the model is shown in Figure 5.1.

Looking at the partial label plot at the bottom right, we observe that the model performs quite well in terms of clustering several of the predefined behaviors. It is able to distinguish between the two running behaviors in behavior 8 and 18. Furthermore, rearing behavior, eating, and grooming are all somewhat separated in the t-SNE plane, to the right, left, and bottom, respectively. Freezing and interacting with the objects seem to be more spread out. An interesting observation is that while it seems to separate rearing against the wall from rearing in the open arena, it does not differentiate between the two subcategories for rearing against the wall. What separated these two behaviors in the labeling was the whisker movement. Showing a better separation of these behaviors when including the facial tracking features could suggest an improvement to the model.

5.2 Facial tracking features only

Next, we try to train a model using only the whisker angles from the facial tracking as input features. Intuitively, this should not make for a great model. It seems ambitious to think that this alone is enough to cluster complex behaviors. However, it may help to understand the input data, and for which behaviors we can hope to get better segmentation when we use all available features.

The hyperparameters are chosen as before. Now we have 16 original features, which are expanded through the time-frequency analysis to 304. As the facial tracking is recorded in 210 Hz, with no missing values, we have a 2460083×304 sized input matrix, which we reduce to 145 dimensions through the initial PCA. The reduced data where downsampled at 2.56 Hz, producing 30002 training samples for t-SNE. Ultimately, the data where clustered into 62 distinct behaviors.

Figure 5.2 shows a visualization of the model trained only on facial tracking features, i.e., 16 whisker angles. Although the partial labels generally seem more spread out compared to the model trained on postural features only, we observe some segmentation, especially on the rearing, grooming, and eating behaviors. As expected, running across the arena is hard to distinguish using only whisker movements. At least, this suggests that the features obtained through facial tracking contain information to aid the separation of behaviors.

5.3 Initial combined model

Using all available features, i.e. combining postural and facial tracking data, to train the model creates some immediate challenges. As the postural recordings and facial tracking samples data at different frequencies, we need to modify the features prior to the principal component step. Different ways of doing this are discussed in Section 3.4. To utilize all samples to the best effect, we perform the time-frequency analysis prior to downsampling. Using the same parameter choices as in previous sections, we are left with 957941 samples containing 133 frequencies, and 2460083 samples containing 304 features. Each of the 304 facial tracking features is linearly interpolated, and computed at the 957941 time points containing postural features without missing values. This is thought to be sufficient, as both 120 Hz and 210 Hz are high frequencies compared to actual postural and whisker movements. Principal component analysis reduces the feature space from 437 to 212 dimensions. Again, the data is further subsampled to 30902 samples and reduced to two dimensions using t-SNE with perplexity parameter 30. Watershed segmentation on the estimated probability density clusters the data into 39 behaviors.

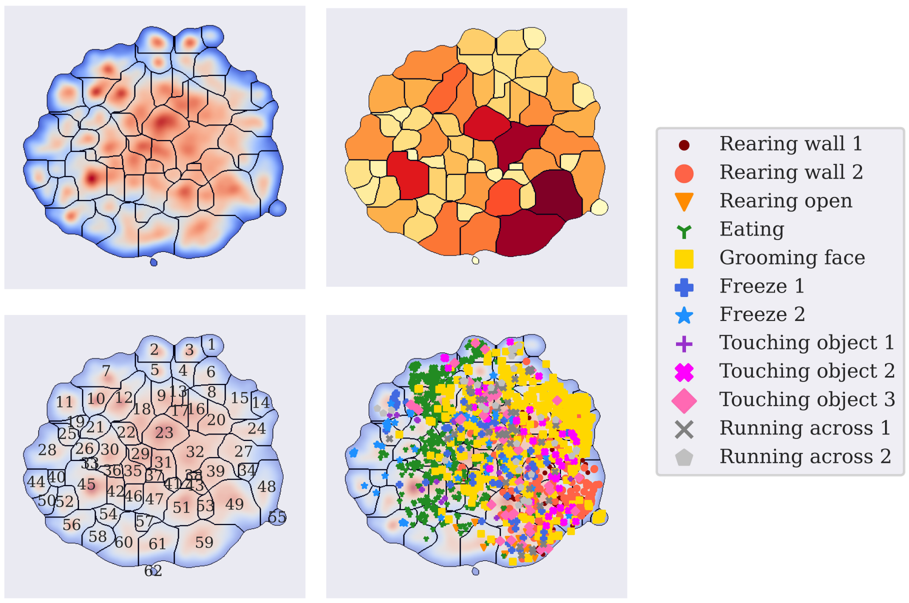


Figure 5.2: Visualization of trained model using only features from the facial tracking *Top left:* Heat map showing the estimated probability density in the t-SNE plane, divided into 62 behavioral regions. *Top right:* Intensity plot showing relative time spent in each of the behaviors. More intense color corresponds to large amount of time spent in behavior. *Bottom left:* Behavioral labels given to the various regions. *Bottom right:* Partial labels assigned to parts of the training data, based on visual inspection of video recordings.

A visualization is shown in Figure 5.3. Comparing this partial label plot to the one from the model using only postural features in Figure 5.1, there does not seem to be an immediate improvement. You could instead argue that some of the partially labeled behaviors, such as the rat grooming its face and running across the arena, now are more poorly separated in the t-SNE plane. The poor separation of grooming behaviors is especially surprising, as the facial tracking data alone seemed to contain information about these behaviors, as shown in Figure 5.2.

Continuing from here, we note that the number of principal components explaining at least 95% of the variance in the data is high relative to the number of features. A natural assumption about the whiskers is that they often move together. If this is true, then PCA should be an effective tool for reducing the dimensionality substantially. When this does not happen, and we go from 437 to

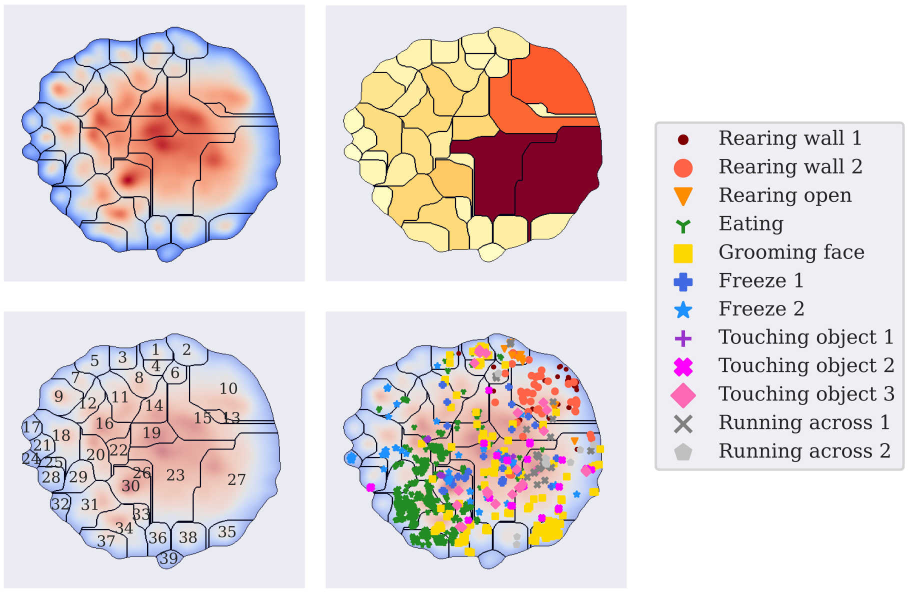


Figure 5.3: Visualization of the initial trained model using both postural features and features from the facial tracking *Top left:* Heat map showing the estimated probability density in the t-SNE plane, divided into 39 behavioral regions. *Top right:* Intensity plot showing relative time spent in each of the behaviors. More intense color corresponds to large amount of time spent in behavior. *Bottom left:* Behavioral labels given to the various regions. *Bottom right:* Partial labels assigned to parts of the training data, based on visual inspection of video recordings.

212 features, it could suggest that the facial tracking data is very noisy. Another theory is that the frequency range in which we look for spectral power should be different for the facial tracking data. Before going further in-depth on the specific models, doing hyperparameter tuning, and looking into the behaviors in detail, we want to test whether or not some more substantial changes to the model could improve the results.

5.4 Model variations

Figure 5.4a shows the partial labels for the model trained on postural features only, while Figure 5.4b shows the partial labels for the initial combined model.

5.4.1 Changing the frequency range

The first instinct when discussing adjusting the frequencies, is to increase the frequencies for which we compute the continuous wavelet transform. For the postural features, we still use 18 frequencies between 0.5 Hz and 20 Hz, while for the facial tracking features, we increase the range to 1-30 Hz. Keeping the other hyperparameters as before, the model finds 48 distinct behaviors. The partial labels plotted over the segmented density estimation are shown at the top right in Figure 5.4c.

If we assume whisker movements in general to have a higher frequency than other body parts, such as the neck and back, we would need the frequencies to cover the expected range better. In other words, we make the frequency range narrower to better differentiate between whisker movements. Changing the minimum and maximum frequency to 1 Hz and 10 Hz, respectively, results in the model shown at the bottom left in Figure 5.4d, leaving all other hyperparameter choices as before.

5.4.2 Manually selecting principal components

Again, supposing the facial tracking data contains a lot of noise, a way of removing this is to include fewer principal components as t-SNE input. We test two approaches. Firstly, we simply use the 50 first principal components. Figure 5.5 shows how the explained variance increases as a function of included principal components for both the model trained on postural features only, and including facial tracking features. It shows that including 50 principal components explained somewhere between 70% and 80% of the variance in the data. This model is shown in Figure 5.4e.

Secondly, we try computing the principal components separately for the postural features and facial tracking features. Then we add together the 50 first components from the postural data, and ten first from the facial tracking data. Since the model trained on postural features only showed very promising initial results, we want to keep much of this information. The facial tracking data should give a slight edge in detecting certain behaviors, but must not introduce too much new noise to the training data. This model is shown in Figure 5.4f.

5.4.3 Choosing between the models

With the exception of the model trained only on postural features, the plots in Figure 5.4 are very similar. The partial labels are grouped very similar but with some differences in how much some behaviors spread out. E.g., the rearing behaviors are more spread out in 5.4c, 5.4e, and 5.4f, than in 5.4a, 5.4b and 5.4d. Of the three models which separate rearing best, the model computing principal

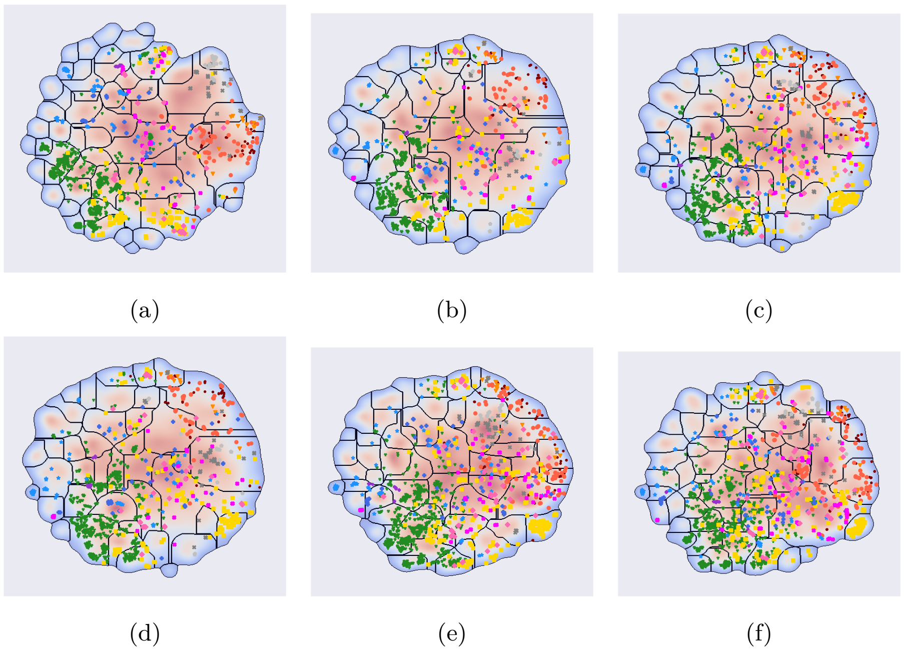


Figure 5.4: Partial labels assigned to parts of the training data, based on visual inspection of video recordings. (a): Model trained only on the postural data. (b): Model trained on all data. Equal frequency range for all features. (c): Model trained on all data. Frequency range where CWT is computed set higher for the facial tracking features. (d): Model trained on all data. Frequency range where CWT is computed set narrower for the facial tracking features. (e): Model trained on all data. Only the 50 first principal components are used as training data for the t-SNE embedding. (f): Model trained on all data. Principal component analysis performed separately on the postural data and the facial tracking data. The 50 first principal components from the postural data, together with the ten first principal components from the facial tracking data, are used as training data in the t-SNE embedding.

components separately also separate the two ways of running across the arena quite well. Based on this, we move forward with hyperparameter tuning on this one.

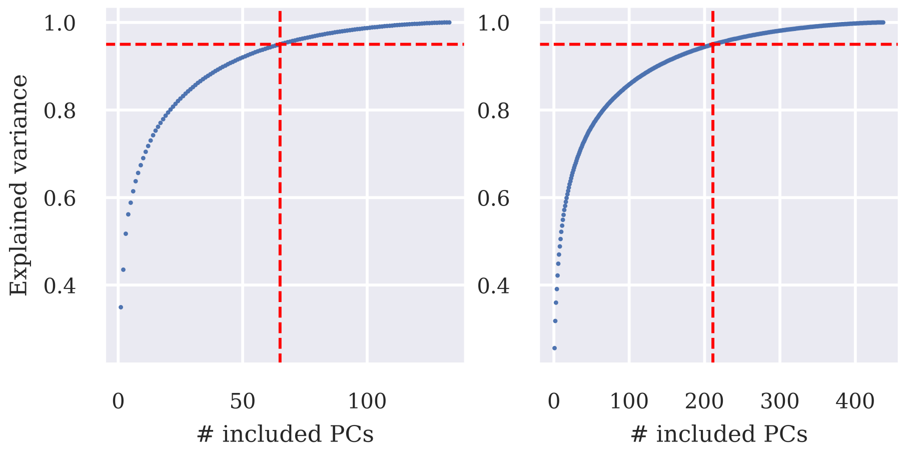


Figure 5.5: Explained variance in the data as a function of principal components included in the dimensionality reduction. Stapled red line indicates number of principal components needed to explain at least 95% of the variance in the data. *Left:* Model trained using only the postural data. *Right:* Model trained using all the data.

5.5 Hyperparameter tuning

In all the behavioral models we have discussed, there are two hyperparameters that greatly affect the final behavioral clustering. The perplexity parameter (2.30) which was originally set to 30, and the bandwidth parameter h used in the kernel density estimation (2.31), which was computed using Scott's rule (2.33). Instead of using Scott's rule, we can manually select the bandwidth, controlling the smoothness of the density estimation. A lower bandwidth results in many segmented behaviors, while a large bandwidth results in fewer segmented behaviors.

Training the models is computationally intensive. For this reason, we first tune the parameters separately to find good intervals, before we do a grid search. The results suggest a range between 20 and 80 for the perplexity parameter, and between 0.1 and 0.15 for the bandwidth. We train the chosen models—one using only postural features, and one with facial tracking features and separate principal component computations—using seven equidistant values for both the perplexity and bandwidth in selected regions. For each model we manually look through the 49 partial label plots, selecting the ones which seem to best segregate the partially labeled data. This is not an exact science, and we need to prioritize between the behaviors. Furthermore, there could be equally good behavioral representations that do not coincide with our partially labeled data. For the model with postural features only, we choose perplexity 50 and bandwidth 0.117. For the model with facial tracking features included, we choose perplexity 50 and bandwidth 0.108. All the figures used in the hyperparameter tuning can be found at `http://www.folk.ntnu.no/ulrikbd` [32].

Chapter 6

Results

Now that we have chosen one model trained on postural data and one trained on postural and facial tracking data combined, we are ready to compare them. We are interested in how behavioral segmentation changes when introducing facial tracking data. Thus, we need to investigate the detected behaviors in the raw data, using the GUI described in Section 4.3. Additionally, topological data analysis is tested on these models, varying the bandwidth hyperparameter in the kernel density estimation.

6.1 Models

6.1.1 Postural features

In Section 5.5 we finished the model selection by selecting the hyperparameters through a grid search. For the model trained only on postural features, the optimal perplexity parameter used in t-SNE was found to be 50. The optimal bandwidth parameter, used in the Gaussian kernel density estimation, was found to be 0.117. A visualization of this model is shown in Figure 6.1. Looking at the partial labels shown at the bottom right, some of the manually labeled behaviors are well separated. The two ways of running across the arena are grouped in behaviors 6 and 12. All of the rearing behaviors are clustered in behavior 23, in which the animal spends a lot of time judging by the relative time plot on the top right. Eating and Grooming face are slightly overlapping in the bottom left corner of the t-SNE mapping. Freeze behaviors seem clustered to the left, but

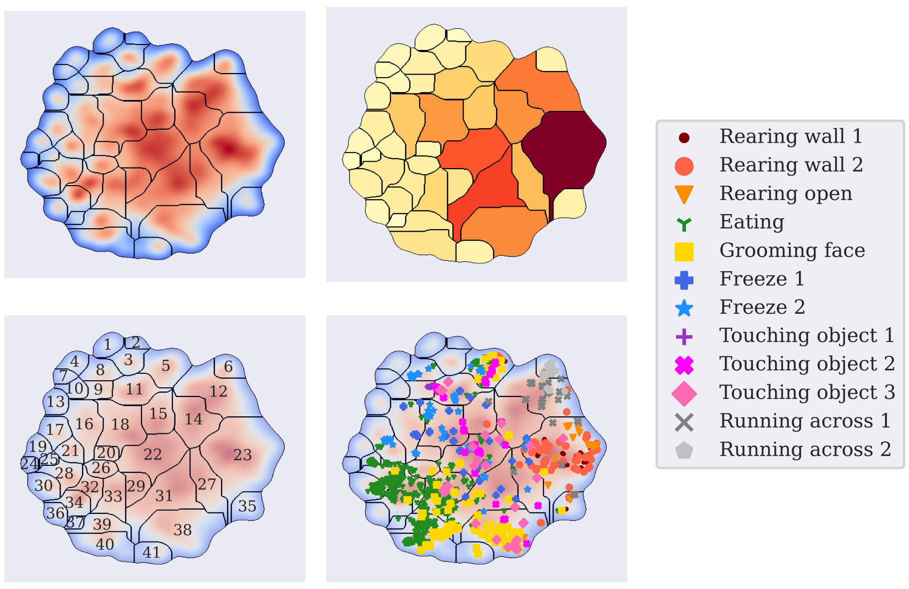


Figure 6.1: Visualization of the best model using only postural features. *Top left:* Heat map showing the estimated probability density in the t-SNE plane, divided into 41 behavioral regions. *Top right:* Intensity plot showing relative time spent in each of the behaviors. More intense color corresponds to a large amount of time spent in behavior. *Bottom left:* Behavioral labels given to the various regions. *Bottom right:* Partial labels assigned to parts of the training data, based on visual inspection of video recordings.

overlapping quite a bit with the touching object labels. Behavior 5 seems to be a mix of everything.

As an aid in determining the nature of the behavioral clusters, we look at the averaged spectral power for the various features, as discussed in Section 3.5.2. Figure 6.2 shows how the behaviors clustered to the right in the t-SNE plane have above-average power in the speed feature. Similar arguments can be made for every feature, both at high and low frequencies. In total, this gives some indication of the movements adding up to the detected behaviors. Spectral averaging plots for all the features can be found at `http://www.folk.ntnu.no/ulrikbd` [32].

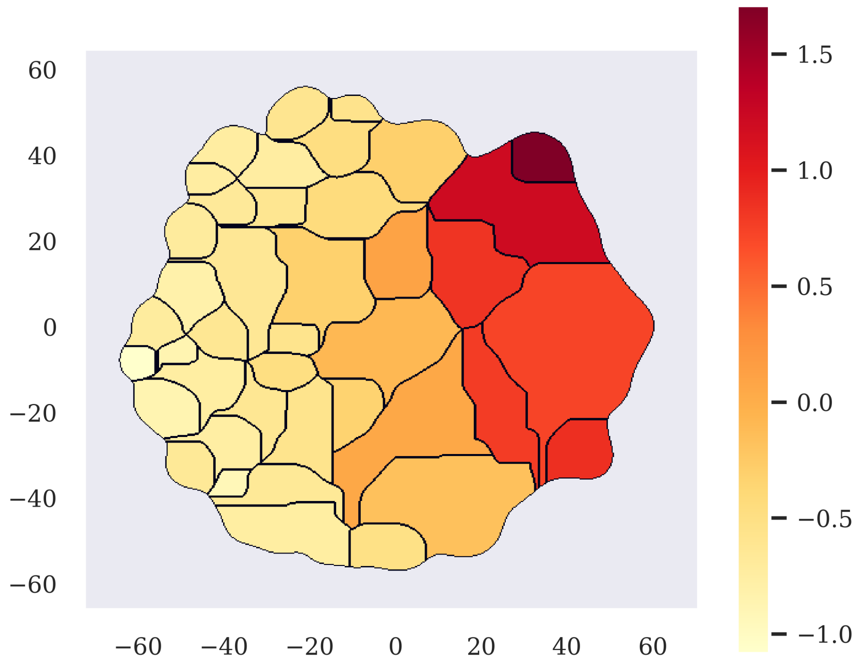


Figure 6.2: Averaged spectral power for high frequencies. Computed for the speed feature in the model trained on postural features, visualized in Figure 6.1.

6.1.2 Including facial tracking features

The model trained on all the data best segmenting the partial labels was found in Section 5.5. It computes the principal components separately for the postural and facial tracking features, including the 50 and 10 features explaining the most variance, respectively. These are concatenated and reduced to two dimensions using t-SNE with perplexity parameter 80. When estimating the probability density over the t-SNE plane, we use a Gaussian kernel density estimator with bandwidth parameter 0.117. A visualization of the model is shown in Figure 6.3. The immediate impression from the partial label plot is that this model divides the various forms of pre-labeled rearing behaviors into multiple behaviors, where the clustering based only on postural features groups them into one.

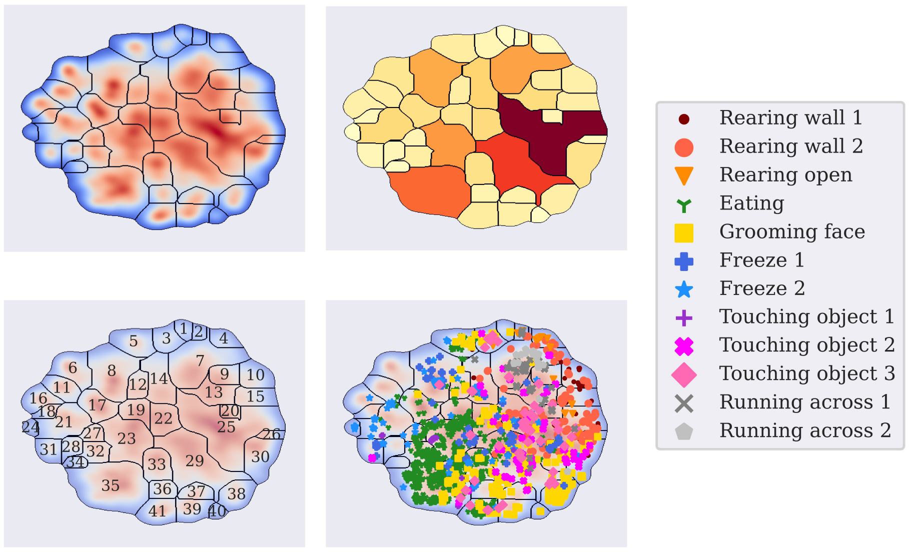


Figure 6.3: Visualization of the best model using all available features. *Top left:* Heat map showing the estimated probability density in the t-SNE plane, divided into 41 behavioral regions. *Top right:* Intensity plot showing relative time spent in each of the behaviors. More intense color corresponds to a large amount of time spent in the region. *Bottom left:* Behavioral labels given to the various regions. *Bottom right:* Partial labels assigned to parts of the training data, based on visual inspection of video recordings.

6.2 Clustering rearing behavior

When clustering using only postural features, all partially labeled rearing behaviors are mapped to behavior 23. However, when using the facial tracking data, the behaviors are spread out across several behaviors to the right in the t-SNE plane 6.3. This could provide insight into how the inclusion of facial tracking data affects the clustering.

Using the graphical user interface discussed in Section 4.3, we can look back at the time intervals mapped to each behavior. This provides a lot of information about which movements the clusters represent, which in turn is useful when labeling the behaviors a posteriori.

Such an analysis is performed on the behaviors which are indicated as rearing behaviors by the partially labeled data. The results are summarised in Table 6.1. Using the facial tracking data, the model maps the partially labeled data into multiple clusters to the right in the t-SNE plane. Some of these clusters—specifically behavior 4, 10, and 15—are specific forms of rearing. These are not detected when only using the postural data. The partially labeled samples are also mapped to other behaviors, which are harder to label using the graphical user interface. Examples of such behaviors are 2, 7, 25, and 26.

Model	Behavior	Description
PF	23	Very clearly a rearing behavior, where the rat is stretching its body upwards, both against the wall and towards the open arena. Also includes the time leading up to the rearing—and sometimes also after—when the rat is walking.
FTF	10	Touching the walls with its head before or during rearing. The head is moving back and forth at a high frequency.
FTF	4	Rat is stretched out upwards. Mostly towards a wall, but could also be in the open arena. Jumps upwards from this stretched-out position.
FTF	15	Rat is stretched out, rearing against a wall. Moving its head looking around to the sides.
FTF	7	Behavior varies. Some rearing, but also some quick dashing.
FTF	2	Behavior varies. The head is moving, sometimes while rearing.
FTF	25	Rat is rearing or sitting still next to the wall.

Table 6.1: Clustered behaviors labeled a posteriori using the graphical user interface discussed in Section 4.3. The PF model refers to the model trained using only postural features, visualized in Figure 6.1. The FTF model refers to the model trained on all the data, including the facial tracking, visualized in Figure 6.3.

6.3 Persistent homology

We also want to see whether computing the persistent homology of the transition probability matrices, as described in Section 3.5.3, can tell us something about the clustering. To exemplify the reasoning behind the procedure, we create a toy example where the same situation is described with two different sets of behaviors.

6.3.1 Toy example

We imagine a situation where someone is sitting at their desk, writing their thesis. Then they stand up and walk to the kitchen to get some coffee, before going back. Dividing this scenario into consecutive behaviors, we get a transition probability matrix which we can represent as a graph, and compute its persistent homology. Though both the number of behaviors and what they represent differ, we hope that the underlying cyclic structure will be picked up.

Simulation

In the Scenario 1, we divide the act of getting coffee into four smaller behaviors. In the second, we divide the act into seven smaller behaviors. They are shown in Table 6.2. Then, we simulate 170 seconds, where each second is labeled as one

Behavior	Scenario 1	Scenario 2
1	Write thesis	Write thesis
2	Walk through the hallway	Open the door
3	Get coffee	Walk through the hallway
4	Speak to family	Get a cup from the cupboard
5		Pour coffee
6		Speak to family
7		Stare out the window

Table 6.2: Two different behavioral segmentations, corresponding to the same act of getting coffee.

of the behaviors. Noise is added by selecting a random behavior at every 20th second.

Computation

The transition probability matrix for Scenario 1 and 2 are

$$\begin{bmatrix} 0.95 & 0.02 & 0.03 & 0 \\ 0.25 & 0.58 & 0.17 & 0 \\ 0.03 & 0.09 & 0.83 & 0.06 \\ 0 & 0 & 0.33 & 0.67 \end{bmatrix} \text{ and } \begin{bmatrix} 0.95 & 0.02 & 0.03 & 0 & 0 & 0 & 0 \\ 0.5 & 0.17 & 0.17 & 0 & 0.17 & 0 & 0 \\ 0.12 & 0.25 & 0.5 & 0.12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.25 & 0 & 0.25 \\ 0 & 0.06 & 0.06 & 0 & 0.82 & 0 & 0.06 \\ 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ 0 & 0 & 0 & 0.08 & 0 & 0.08 & 0.85 \end{bmatrix},$$

respectively, rounded to the second decimal point. As we only care about the transitions from one behavior to another, we do not need to consider the diagonal. Additionally, when considering real data, the diagonal will often contain values close to 1. Thus, we set all values on the diagonal to 0, and standardize the rows.

When including the transition probabilities p as weighted edges in the graph, it is useful to instead take $1 - p$. The algorithm starts with no edges included, and will then include the edges with the highest original transition probability first. After doing the adjustments, the transition matrices for Scenario 1 and 2 becomes

$$\begin{bmatrix} 0 & 0.67 & 0.33 & 1 \\ 0.4 & 0 & 0.6 & 1 \\ 0.83 & 0.5 & 0 & 0.67 \\ 1 & 1 & 0 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 0.67 & 0.33 & 1 & 1 & 1 & 1 \\ 0.4 & 0 & 0.8 & 1 & 0.8 & 1 & 1 \\ 0.75 & 0.5 & 0 & 0.75 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0.5 & 1 & 0.5 \\ 1 & 0.67 & 0.67 & 1 & 0 & 1 & 0.67 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0.5 & 1 & 0.5 & 0 \end{bmatrix},$$

respectively, rounded to the second decimal point. The resulting graph for Scenario 1 is shown in Figure 6.4. Notice how when a behavior always moves to the same behavior, as behavior four does in Scenario 1 and behavior six in Scenario 2, the edge between the corresponding nodes in the graph will always be included.

Results

By viewing the modified transition matrices as weighted graphs, we now compute the persistence pairs for each scenario. The persistence diagram comparing these persistence pairs is shown in Figure 6.5. To the left, with birth at 0, are the H_0

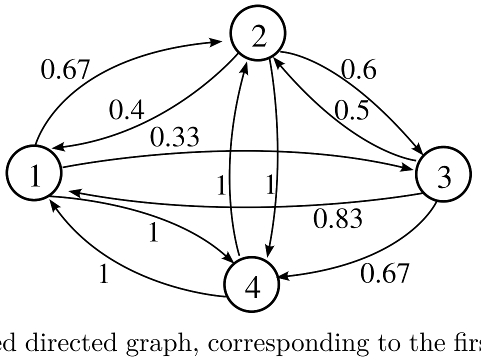


Figure 6.4: Weighted directed graph, corresponding to the first example scenario discussed in Section 6.3.1. Each node represents a behavior, and the edges represent a modified transition probability between the behaviors.

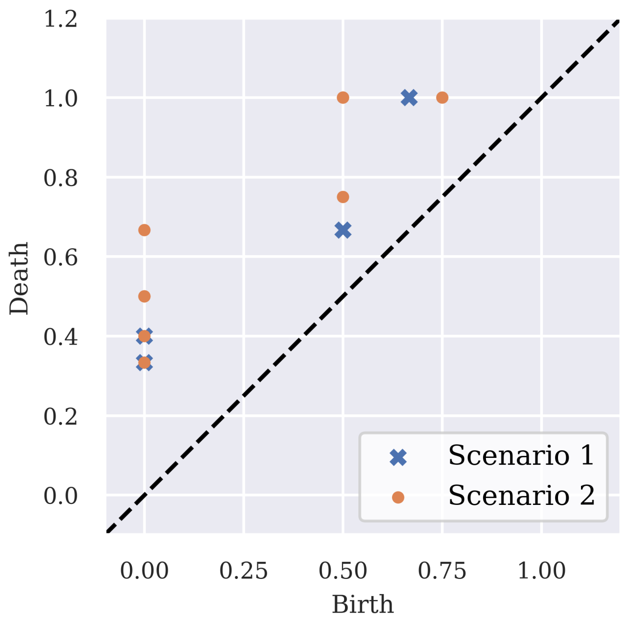


Figure 6.5: Persistence diagrams for each of the two behavioral scenarios compared in the toy example discussed in Section 6.3.1.

homology classes. Some of them are overlapping in the figure. Scenario 1 also has zero-dimensional homology classes dying at 0.33 and 0.4.

It is more interesting to look at the H_1 homology classes. For Scenario 1, two such topological features appear. The first one appears at 0.5 and dies at 0.67. If we look at the transition matrix, this feature represents the cycle between behaviors 1, 2, and 3. At 0.67, all edges between these behaviors are filled in, and the cycle disappears. Another cycle appears at 0.67 between behaviors 3 and 4, which does not die before all edges are filled in.

The bottleneck distance between the two persistence diagrams (2.34) is 0.267. The bijection is represented by matching pairs of points. As the number of points is different, some points are matched with the diagonal. The optimal matching is shown in Figure 6.6.

Based on the example scenarios alone, it is hard to determine the benefit of this approach. The interpretations of the topological features in this space are still quite abstract. By comparing persistence diagrams and bottleneck distances on actual transition probability matrices with a larger number of behaviors, we will better know if this approach serves a purpose in model selection.

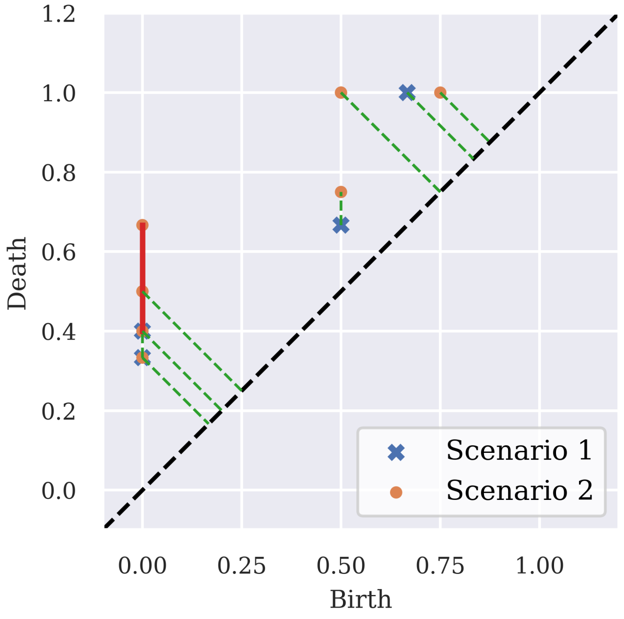


Figure 6.6: Persistence diagrams for each of the two behavioral scenarios compared in the toy example discussed in Section 6.3.1. The bottleneck distance is shown as a red line, with dashed green lines indicating the matched points in the chosen bijection.

6.3.2 Application on real data

To show how the persistence diagrams compare between models change with the different hyperparameters, we use the two selected models found in Section 6.1. For both the model trained on postural features only and the one using all available data, we retrain the models with 50 equidistant bandwidth parameter values ranging from 0.1 to 0.15. The bandwidth parameter is a natural first choice. It does not change the t-SNE mapping itself, only the segmentation in the t-SNE plane. When increasing the bandwidth parameter we effectively increase the size of the regions, and in turn, decrease the number of behaviors. Smaller regions will gradually combine into larger ones, hopefully resulting in a hierarchical structure we can detect.

Some of the persistence diagrams for the models trained on postural features are shown in Figure 6.7. Similar persistence diagrams for the models trained on the facial tracing data included are shown in Figure 6.8. The rest can be

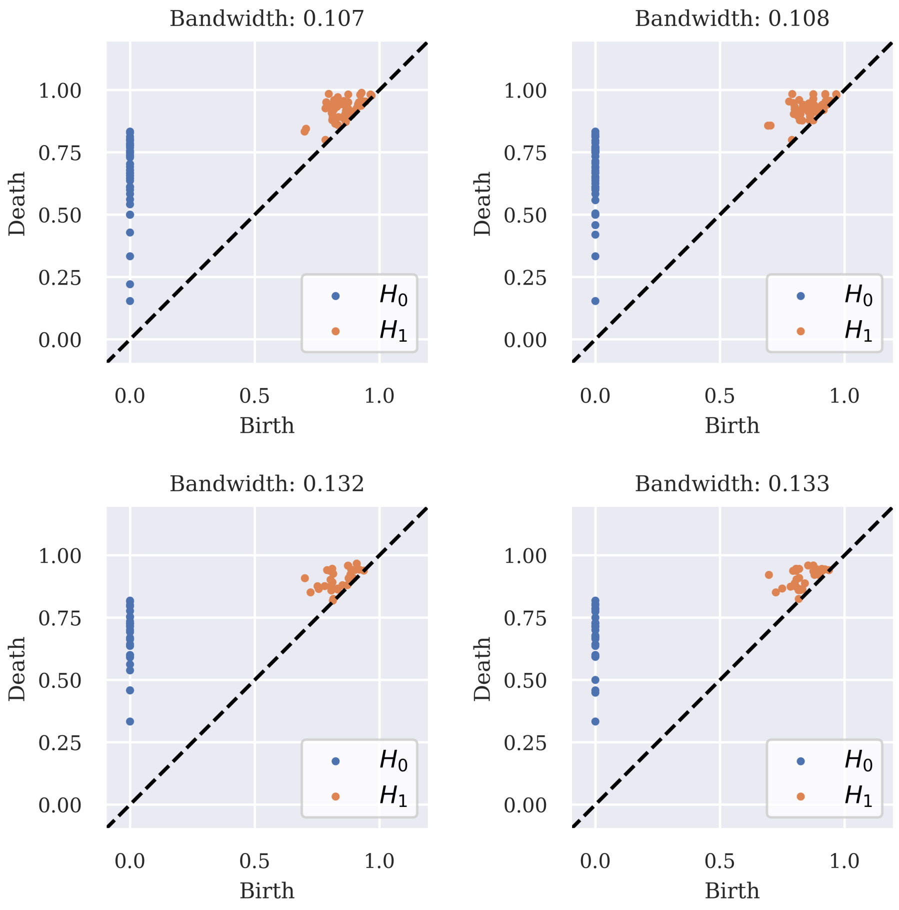


Figure 6.7: Persistence diagrams computed by transforming the transition probability matrices to directed graphs, with edges weighted by the transition probabilities. The diagrams shown are computed on the model trained on postural features only, discussed in Section 6.1, varying the bandwidth parameter in the kernel density estimation.

found at `http://www.folk.ntnu.no/ulrikbd` [32]. As the bandwidth parameter moves from 0.1 to 0.15, some topological features appear and disappear from the persistence diagrams. By visual inspection, it seems like persistence diagrams for models trained with similar bandwidth parameter, also appear more similar.

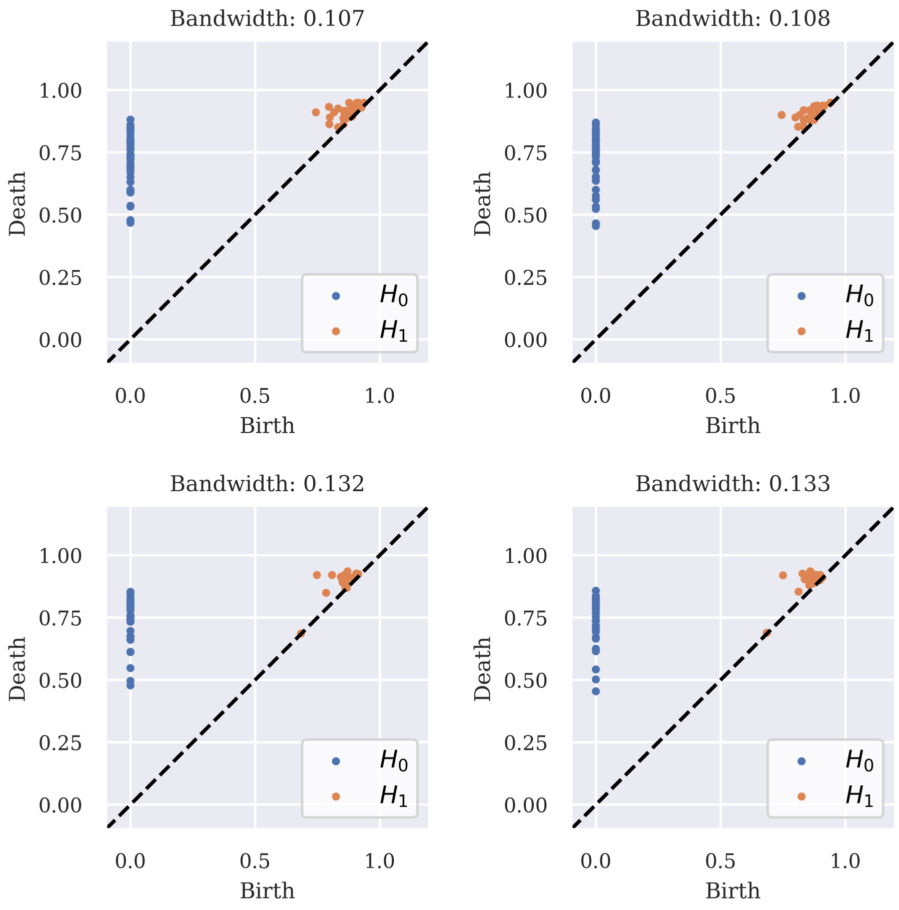
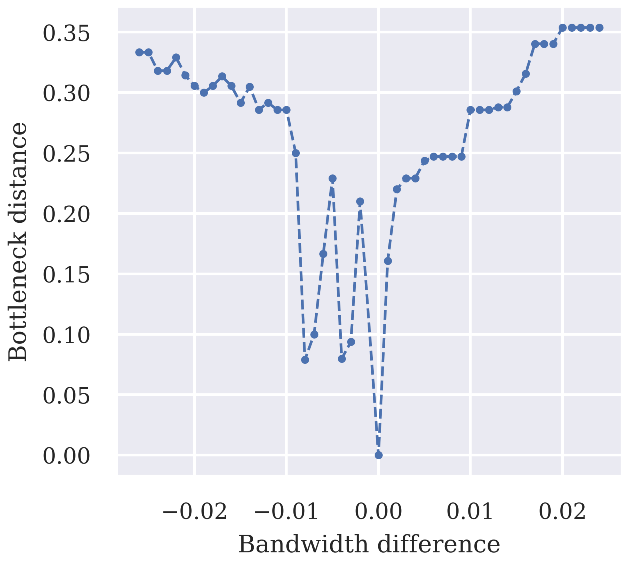


Figure 6.8: Persistence diagrams computed by transforming the transition probability matrices to directed graphs, with edges weighted by the transition probabilities. The diagrams shown are computed on the model trained on all the data, discussed in Section 6.1, varying the bandwidth parameter in the kernel density estimation.

Computing bottleneck distances

To compare models using the bottleneck distance (2.34), we choose a reference model with bandwidth parameter in the middle of the interval. We then compute the bottleneck distance to each of the other 49 models, allowing us to see how it evolves as a function of the bandwidth difference. The results are visualised in Figure 6.9 As expected, when the models have similar bandwidth parameter, the bottleneck distance is small.



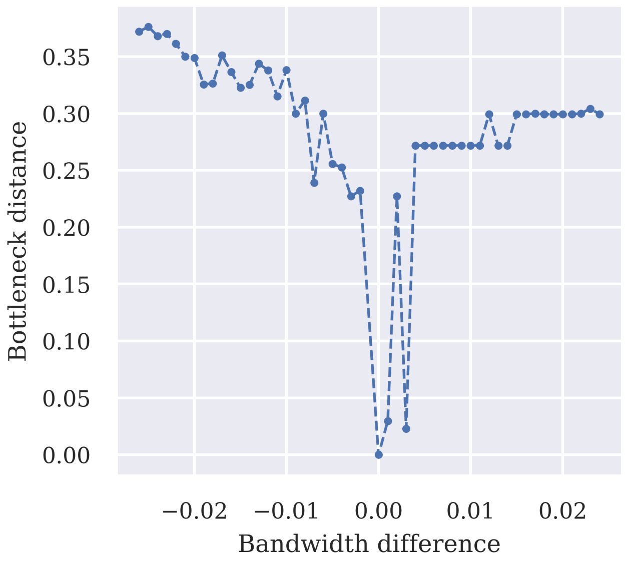


Figure 6.9: Bottleneck distance as a function of bandwidth difference. The bottleneck distance is computed relative to a model trained using bandwidth 0.126, close to the middle of the parameter tuning interval. *Top:* Model trained only on the postural data. *Bottom:* Model trained on the combined postural and facial tracking data.

Chapter 7

Discussion and conclusions

The original aim of the thesis was to investigate topological data analysis as a tool for detecting structural patterns in animal behavior, invariant over a range of hyperparameters. Additionally, as I had implemented a behavioral clustering algorithm during my specialization project [9], the Whitlock group at Kavli Institute for Systems Neuroscience asked if I could assist them with testing the clustering on newly available data. This data included recorded angles from several of the rat's whiskers. Exploring new models and implementing tests for these was, although time-consuming, very useful and instructive. It illustrated the challenges related to model selection in behavioral clustering, and why quantitative measures comparing models are useful.

7.1 Model selection

Model selection in behavioral clustering is difficult. It is hard to tell by the two-dimensional mappings how models differ, and which one to prefer. Partial labeling has proved a very useful tool, both for tuning the hyperparameters and for understanding how the models work. Regions in the two-dimensional t-SNE plane correspond to different body movements and action sequences. The partial labels provide insight about these regions, and work as a starting point for further analysis. They can also help to rule out poor-performing models quickly.

Investigating clusters through video recordings a posteriori remains inefficient but necessary to confirm beliefs about the final clusters. Therefore, a way of doing this as efficiently as possible is necessary. In this thesis, we used a graphical user interface developed in-house at Kavli Institute of Systems Neuroscience 4.3. Being able to look through the raw material corresponding to individual behaviors one at a time, proved invaluable. It eases the workload and speeds up the process.

We investigated the use of topological data analysis as a model selection tool. By transforming the estimated transition probability matrix into a weighted directed graph, we can use homology to find high-level topological features in the behavioral structure, such as cycles. Persistent homology measures how long such topological features exist in the graphs as edges are included gradually according to the corresponding transition probability. The resulting persistence diagrams can then be used to visualize similarities between two models, possibly with different numbers of clustered behaviors. The bottleneck distance, a similarity metric between two such persistence diagrams, is also available in this theoretical framework. It is useful for visualizing how clusterings result from models trained with hyperparameters close to each other, all represent a similar structure when looking at the transitions between them.

7.2 Facial tracking features

In this thesis, we clustered real data from a Long-Evans rat roaming around in an arena, with and without including facial tracking recordings. For both data sets, several models were tested, and corresponding hyperparameters were tuned. As the facial tracking was performed at a higher capture frequency rate, the features from the facial tracking were downsampled after computing the wavelet transform. The facial tracking data contained a lot of information, and we were able to separate some behaviors from this data alone. Several changes to the algorithm were explored, aiming to include the useful information contained in the facial tracking data, but without much of the added noise. Finally, we selected a model computing the principal components separately, before concatenating them when performing t-stochastic neighbor embedding.

Certain differences between the models stood out. Partial labeling indicated that the rearing behaviors, which were clustered into one single behavior when only using postural features, spread out into several behaviors when including the facial tracking data. This was confirmed by looking back at the behaviors in the raw data, using the graphical user interface.

7.3 Further work

Not all of the behaviors found using the facial tracking data were investigated in the raw material. There is more information to be extracted and analyzed, which proved too time-consuming for this thesis. Making use of actual footage of the whiskers for the individual behaviors could also provide more insight into how the rearing behaviors are separated.

Although we investigated several models combining the postural and facial tracking data, we likely did not find the best one. A lot of choices are made during the model training phase, and the best one might not be the one we landed on. Having an open mind to new alternatives in the clustering algorithm is important.

The idea of using topological data analysis to compare ethograms from trained models still requires more investigation. We showed that the computation is possible using existing software for Python, and explored some simple cases. More testing should however be done. Further work includes visualizing the bottleneck distance as a function of other parameters, e.g., the perplexity parameter. Visualizing the bottleneck distance as a two-dimensional heat map would be useful for detecting intervals for hyperparameters resulting in a similar clustering structure. A way to extract information about what the topological features we compare represent in terms of behavioral structures would be a great step forward. Other ways of visualizing the persistence diagrams could also be tested, as more ways of visualizing persistence pairs exist. In which dimensions to search for and compare topological features could also be explored in more detail.

Appendix A

Code

All data analysis were performed using the Python programming language [33]. General scientific programming was aided by NumPy [34] and Scipy [35]. Figures was created using Matplotlib [36] and seaborn [37]. For computing the wavelet transform we used PyCWT [14]. The dimensionality reduction was performed using modules from scikit-learn [38]. Both Giotto-tda [39] and Scikit-TDA [40] was used to compute and visualize the persistent homology.

Due to high-dimensional input data, the analysis in this thesis would not be possible without using the high-performance computing research infrastructure IDUN [41].

All the code for training and analyzing the models used in this thesis can be found at `https://github.com/ulrikbd/master_thesis_code` [42].

Bibliography

[1] K. Schulze-Hagen and T. R. Birkhead, “The ethology and life history of birds: The forgotten contributions of oskar, magdalena and katharina heinroth,” *Journal of Ornithology*, vol. 156, no. 1, pp. 9–18, 2015. DOI: `10.1007/s10336-014-1091-3`.

[2] J. Klaminder, G. Hellström, J. Fahlman, *et al.*, “Drug-induced behavioral changes: Using laboratory observations to predict field observations,” *Frontiers in Environmental Science*, vol. 4, 2016. DOI: `10.3389/fenvs.2016.00081`.

[3] G. J. Berman, D. M. Choi, W. Bialek, and J. W. Shaevitz, “Mapping the stereotyped behaviour of freely moving fruit flies,” *Journal of The Royal Society Interface*, vol. 11, no. 99, p. 20 140 672, 2014. DOI: `10.1098/rsif.2014.0672`.

[4] G. J. Berman, W. Bialek, and J. W. Shaevitz, *Predictability and hierarchy in drosophila behavior*, 2016. DOI: `10.1101/052928`.

[5] J. D. Marshall, D. E. Aldarondo, T. W. Dunn, W. L. Wang, G. J. Berman, and B. P. Ölveczky, “Continuous whole-body 3d kinematic recordings across the rodent behavioral repertoire,” *Neuron*, vol. 109, no. 3, 2021. DOI: `10.1016/j.neuron.2020.11.016`.

[6] “Behavioral decomposition reveals rich encoding structure employed across neocortex,” *bioRxiv*, 2022. DOI: `10.1101/2022.02.08.479515`. eprint: `https://www.biorxiv.org/content/early/2022/02/10/2022.02.08.479515.full.pdf`. [Online]. Available: `https://www.biorxiv.org/content/early/2022/02/10/2022.02.08.479515`.

[7] A. Wiltschko, M. Johnson, G. Iurilli, *et al.*, "Mapping sub-second structure in mouse behavior," *Neuron*, vol. 88, no. 6, pp. 1121–1135, 2015. DOI: `10.1016/j.neuron.2015.11.031`.

[8] G. J. Berman, "Measuring behavior across scales," *BMC Biology*, vol. 16, no. 1, 2018. DOI: `10.1186/s12915-018-0494-7`.

[9] U. B. Danielsen, "Clustering animal behavior," 2022. [Online]. Available: `https://github.com/ulrikbd/specialization_project`.

[10] W. W. Wei, *Time series analysis univariate and multivariate methods*, 2nd ed. Pearson, 2006.

[11] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: With R examples*, 4th ed. Springer International Publishing, 2017.

[12] L. Cohen, "Time-frequency distributions-a review," *Proceedings of the IEEE*, vol. 77, no. 7, pp. 941–981, 1989. DOI: `10.1109/5.30749`.

[13] G. Kaiser, *A friendly guide to wavelets.* Birkhäuser, 1994.

[14] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bulletin of the American Meteorological Society*, vol. 79, no. 1, pp. 61–78, 1998. DOI: `10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2`.

[15] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics.* Springer, 2010.

[16] T. Hastie, J. Friedman, and R. Tisbshirani, *The elements of Statistical Learning: Data Mining, Inference, and prediction*, 2nd ed. Springer, 2017.

[17] I. T. Jolliffe, *Principal component analysis. 2nd ed.* Springer-Verlag, 2002.

[18] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[19] J. S. Simonoff, *Smoothing methods in statistics.* Springer, 1998.

[20] D. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization.* Mar. 2015, ISBN: 9781118575536.

[21] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1163–1173, 1996. DOI: `10.1109/34.546254`.

[22] J. Bang-Jensen and G. Z. Gutin, *Digraphs theory, algorithms and applications.* Springer, 2002.

[23] R. W. Ghrist, *Elementary applied topology.* Createspace, 2014.

[24] M. W. Reimann, M. Nolte, M. Scolamiero, *et al.*, “Cliques of neurons bound into cavities provide a missing link between structure and function,” *Frontiers in Computational Neuroscience*, vol. 11, 2017. DOI: `10.3389/fncom.2017.00048`.

[25] D. Lütgehetmann, D. Govc, J. P. Smith, and R. Levi, “Computing persistent homology of directed flag complexes,” *Algorithms*, vol. 13, no. 1, p. 19, 2020. DOI: `10.3390/a13010019`.

[26] H. Edelsbrunner and J. Harer, “Persistent homology—a survey,” *Surveys on Discrete and Computational Geometry*, pp. 257–282, 2008. DOI: `10.1090/conm/453/08802`.

[27] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, “Stability of persistence diagrams,” *Proceedings of the twenty-first annual symposium on Computational geometry*, 2005. DOI: `10.1145/1064092.1064133`.

[28] Y. Liu, X. San Liang, and R. H. Weisberg, “Rectification of the bias in the wavelet power spectrum,” *Journal of Atmospheric and Oceanic Technology*, vol. 24, no. 12, pp. 2093–2102, 2007. DOI: `10.1175/2007jtecho511.1`.

[29] A. Mathis, P. Mamidanna, K. M. Cury, *et al.*, “Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning,” *Nature Neuroscience*, vol. 21, no. 9, pp. 1281–1289, 2018. DOI: `10.1038/s41593-018-0209-y`.

[30] J. Schindelin, I. Arganda-Carreras, E. Frise, *et al.*, “Fiji: An open-source platform for biological-image analysis,” *Nature Methods*, vol. 9, no. 7, pp. 676–682, 2012. DOI: `10.1038/nmeth.2019`.

[31] B. Mimica, B. A. Dunn, T. Tombaz, V. S. Bojja, and J. R. Whitlock, “Efficient cortical coding of 3d posture in freely behaving rats,” 2018. DOI: `10.1101/307785`.

[32] U. B. Danielsen. (2023), [Online]. Available: `http://www.folk.ntnu.no/ulrikbd` (visited on 05/24/2023).

[33] G. van Rossum, “Python tutorial,” Centrum voor Wiskunde en Informatica (CWI), Amsterdam, Tech. Rep. CS-R9526, May 1995.

[34] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: `10.1038/s41586-020-2649-2`. [Online]. Available: `https://doi.org/10.1038/s41586-020-2649-2`.

[35] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: `10.1038/s41592-019-0686-2`.

[36] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[37] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: 10.21105/joss.03021. [Online]. Available: https://doi.org/10.21105/joss.03021.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] G. Tauzin, U. Lupo, L. Tunstall, *et al.*, "Giotto-tda: A topological data analysis toolkit for machine learning and data exploration," *Journal of Machine Learning Research*, vol. 22, no. 39, pp. 1–6, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-325.html.

[40] N. Saul and C. Tralie, *Scikit-tda: Topological data analysis for python*, 2019. DOI: 10.5281/zenodo.2533369. [Online]. Available: https://doi.org/10.5281/zenodo.2533369.

[41] M. Själander, M. Jahre, G. Tufte, and N. Reissmann, *Epic: An energy-efficient, high-performance gpgpu computing research infrastructure*, 2022. arXiv: 1912.05848 [cs.DC].

[42] U. Danielsen, *Master thesis code*, https://github.com/ulrikbd/master_thesis_code, 2023.