NTNU
Norwegian University of
Science and Technology

Modelling Neuronal Activity using Lasso Regularized Logistic Regression

Haris Fawad

Master of Science in Physics and Mathematics
Submission date: June 2017
Supervisor: Mette Langaas, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Summary

Studies have shown that neurons in a mouse's anterior lateral motor cortex (ALM), which is a part of the brain related to the planning and the execution of tongue movements, predict certain movements seconds before they occur (Li et al., 2015). This raises the question: how does this so-called preparatory activity in the ALM translate into commands in other motor-related parts of the brain that eventually trigger a body movement? To investigate this, we fit lasso penalized logistic regression models that relate the activity of a neuron to the activity of all other neurons (in the data set). In addition, the regression models include the effect of a certain type of stimulus that is given to the mouse, as part of the experiment in which the data is collected. The estimated regression parameters are then used to estimate a network of neurons that shows how the neurons are connected to each other, and hence in effect, visualize the underlying information flow in the ALM. Additionally, the estimated parameters also show the so-called tuning of a neuron, that is, the relation between the activity of a neuron and the given stimulus.

The analyses are based on the so-called `alm-1` data set, which contains extracellular recordings of neurons of 19 adult mice. The lasso penalized logistic regression models are fit to data from a single mouse, which revealed that each neuron is seemingly tuned to a different stimuli. Additionally, in the estimated network of neurons, constructed by evaluating (family-wise error rate) adjusted p-values controlled at significance level 5%, the so-called fast spiking (FS) neurons seem to be a central part of the underlying network, as these neurons have the highest amount of connections compared to so-called pyramidial neurons.

Preface

This thesis concludes the course TMA4905: Master's in Statistics, offered at the Department of Mathematical Sciences at the Norwegian University of Science and Technology (NTNU). This completes my degree from the study program Master of Science in Applied Physics and Mathematics, where my main profile was Industrial Mathematics. The work in this thesis was carried out in the spring semester of 2017. However, a related introductory level project was completed during the previous fall semester, which provided a so-called warm start to the work in this thesis. The topic of this thesis is modelling neuronal activity using lasso penalized regression models.

I would like to thank my supervisor Prof. Mette Langaas for her clear guidance, constant availability and honest feedback throughout the past two semesters. I would also like to thank my co-supervisor Benjamin Adric Dunn for his patience and his tremendous knowledge in systems neuroscience and everything related. I appreciate both of you for making this past year a great learning experience.

Finally I would like to thank my family and friends, who helped me keep my mind off my studies when I needed it the most. I dedicate this thesis to my mother, who's hard work and determination put me in a position to pursue my interests.

Haris Fawad
Trondheim, Norway
June, 2017

Contents

Glossary

`GoodTrials`	A set of trials where the mouse was performing. According to the documentation (`http://crcns.org/files/data/alm-1/crcns_alm-1_data_description.pdf`) of the `alm-1` data set, these trials are fit to be analyzed.
ALM	Anterior lateral motor cortex. The ALM in a mouse brain is related to planning directional tongue movements, that is, left or right licks.
correct lick left trials	A subset of `GoodTrials`, where the mouse correctly licked left.
correct lick right trials	A subset of `GoodTrials`, where the mouse correctly licked right.
delay epoch	The interval in trial time, following the sample epoch, where the mouse retains/remembers its decision to lick left or right. A delay epoch lasts on average about 1.5 s.
excitatory connection	A signal from neuron a to neuron b that increases the chance of neuron b to fire.
inhibitory connection	A signal from neuron a to neuron b that decreases the chance of neuron b to fire.
photostimulation	A method used to inactive targeted types of neurons in the brain. See Li et al. (2015, Methods).
response epoch	The interval in trial time, following the delay epoch, where the mouse licks left/right. A response epoch lasts on average a little over 1 s.
sample epoch	The interval in trial time where a stimulus is provided to the mouse. A sample epoch lasts on average a little over 2 s.
session time	The time interval in which the experiment with a single mouse is conducted, lasting between 1 to 2 h.

spike train	A sequence of event times where a neuron fired during a finite time interval $(0, T]$, denoted $s_1, s_2, ..., s_m$.
trial time	The time interval in which the mouse is provided a stimulus, and its decision to lick left/right is monitored. There are hundreds of trials in a session, each lasting 5 s.

Chapter 1

Introduction

We begin this chapter with a problem description, explaining why the work in this thesis is interesting, both from a neuroscientific and a statistical point of view. The remaining parts of this chapter elaborate on the neuroscientific context relevant for the analyses in this thesis.

1.1 Problem Description

This thesis focuses on analyzing neural activity in a specific part of the brain in mice, called the anterior lateral motor cortex (ALM), which is related to the planning and the execution of movements. Studies have shown that the activity of neurons in the ALM predicts certain movements seconds before they occur (Li et al., 2015). A central research question in neuroscience is then how this so-called preparatory activity in the ALM, translates into movement commands. That is, we can imagine there is a network of neurons in the ALM were the activity originates, and then travels to other parts of the brain.

The main goal of our analyses is to estimate this underlying network of neurons. Simply put, this can be done by modelling the activity of neuron j by the activity of all the other neurons $k \neq j$ (in the data set). Additionally, we aim to analyze the activity of neuron j by a certain type of stimulus that is related to the mouse's tongue movements. This will reveal how the neurons in the ALM relate to this motor-related stimulus. And hopefully, this activity-stimulus analysis will also show the aforementioned preparatory activity.

Influenced by the work of Stevenson et al. (2012) and Pillow et al. (2008), we employ regression models to relate the activity of neuron j to the activity of all the other neurons $k \neq j$, in addition to the effects of the stimulus. Furthermore, it's known that the activity of neuron j at time t_2 is dependent on neuron j's own past activity at time t_1, where $t_1 < t_2$. This is not unique to neurons in the ALM, but rather, a consequence of a neurons anatomy. Hence, the regression models considered in this thesis relate the activity of neuron j by accounting for the history of neuron j, the connectivity of neurons $k \neq j$, and the motor-related

stimulus. Specifically, we intend to fit lasso penalized logistic regression models.

Contrary to the work of Stevenson et al. (2012) and Pillow et al. (2008), we aim to estimate the underlying network of neurons by finding significant connections between neurons j and k, by evaluating familiy-wise error rate (FWER) adjusted p-values. From a statistical point of view, this is an interesting endeavour, since it is difficult to find the distribution (and hence p-values) of the estimated parameters from a lasso model. Hence, we look towards a pragmatic solution, called multi-sample splitting, which is proven by Buhlmann and van de Geer (2011) to control the FWER at a pre-specified level.

1.2 Background in Neuroscience

Neuroscience focuses on studying the nervous system. In this section we give a brief introduction to some aspects of the nervous system, of which the brain is a central part. In Section 1.2.1 we focus on a specific area of the brain related to movements of the body. In Sections 1.2.2, 1.2.3, and 1.2.4 we define the *neuron*, what it means for a neuron to be active, and how to record that activity, respectively.

1.2.1 Motor cortex

The majority of the processing in the brain takes place on its outermost layer called the *cerebral cortex*, which is the familiar wrinkly folded surface. The cerebral cortex can be divided into roughly four major areas, called *lobes*. In our case, we focus on the frontal lobe, which includes (among others) the *motor cortex*. The motor cortex is involved in planning and executing voluntary movements (Li et al., 2015, p. 51). Figure 1.1 shows where the motor cortex is located in the brain, and some of its important components.

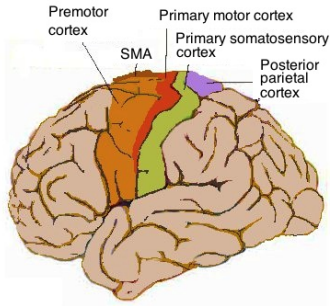

Figure 1.1: The sensorimotor cortex (a combination of motor cortex and somatosensory cortex) in the human brain. The motor cortex is made up of the primary motor cortex (red), the premotor cortex (orange) and the supplementary motor area (SMA) (brown). The remaining (green and purple) areas are parts of the somatosensory cortex. This sketch shows the left hemisphere of the brain.

Source: `https://en.wikipedia.org/wiki/Motor_cortex`

1.2.2 Neurons

The nervous system is a vast communication network found throughout the whole body in humans and other vertebrates. The main functional units of the nervous system are the *neurons*. These are cells that propagate information by generating electrical signals, depending on what input signals they receive. That is, the functionality of a neuron is to (i) receive signals, (ii) process these signals, and decide whether or not this information should be passed along, and (iii) communicate the resulting signal (if any) to target cells, which could be neighbouring neurons or some other type of cells such as muscle cells. This functionality is reflected in the anatomy of a neuron, which is illustrated in Figure 1.2. The dendrites are the many branches where a neuron receives input signals. The soma is the body of the neuron, and this is where the input signals from the dendrites are processed. The axon is a cable-like structure that carries the signal away from the soma, and towards the axon terminal, where the signal is eventually communicated to other cells. Whenever a neuron transmits a signal through its axon, we say that the neuron *fires*.

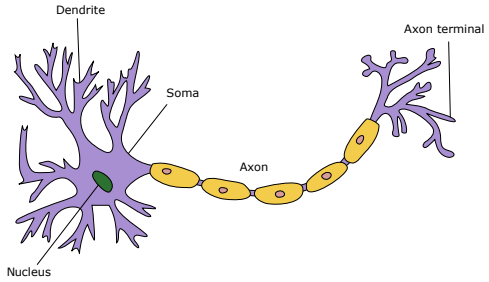

Figure 1.2: Anatomy of a neuron. The information flow travels from the dendrites to the axon terminals.

Source: `https://en.wikipedia.org/wiki/Neuron`

1.2.3 Action Potential and Neural Connections

In animal cells, there's usually a surplus of negative electrical charge inside the cell, and a surplus of positive electrical charge outside the cell. This voltage difference across the cell membrane is called a *membrane potential* (and has a typical value of $-70\,\mathrm{mV}$). For electrically charged cells, such as neurons, this membrane potential fluctuates over time, as shown in Figure 1.3. First off, this figure shows two important levels of the membrane potential: a *resting potential* (shown at $-70\,\mathrm{mV}$), which is a baseline potential, and a higher *threshold potential* (at $-55\,\mathrm{mV}$). If enough stimulus is provided to the neuron (at its dendrites), the membrane potential rises above the threshold potential. This causes a chemical chain reaction, which further raises the membrane potential (depolarization), eventually leading to an *action potential* (peak). Overfly simplified, an action potential in the soma of a neuron causes it to generate and transmit an electrical signal through its

axon. That is, a neuron is said to fire whenever an action potential (in the soma) is reached. Right after the action potential is reached, the membrane potential quickly drops (repolarization), and often undershoots to a value below the resting potential. During this period, known as the *refractory period* (which generally lasts 1 ms), the proteins in the neuron actively try to recover the membrane potential back to its resting state. Any stimulus provided to the neuron during this period will generally not lead to a new action potential.

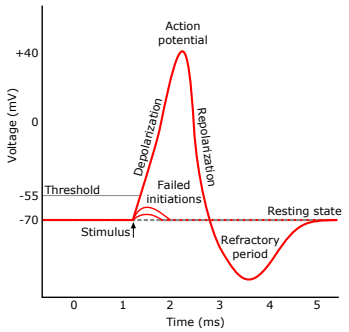

Figure 1.3: An illustration of the various stages of the membrane potential, leading to an action potential (peak). The vertical axis shows the value of the membrane potential.

Source: `https://en.wikipedia.org/wiki/Action_potential`

The stimulus provided to a neuron can be signals from other neurons. This type of signal can either be *excitatory* or *inhibitory*. Given that the membrane potential is at its resting state, an excitatory signal can raise the membrane potential. That is, if the excitatory signal is strong enough, the membrane potential will rise over the threshold potential, eventually resulting in the neuron (that received the signal) to fire. On the other hand, an inhibitory signal will further decrease the membrane potential (from its resting state), which will lower the chance of the neuron to fire.

Neurons that provide excitatory signals are called *principle neurons*, while those that provide inhibitory signals are called *interneurons*. There are studies that have tried to estimate the number of connections one would expect between principle neurons in a given part of the brain (Schroter et al., 2017). (The same is difficult to study for interneurons, which are smaller in size than principle neurons). An operating estimate is 5% of the total number of possible (directed) connections. That is, say there are 10 (principle) neurons in a network. Then, there are $10^2 - 10 = 90$ possible directed connections among these neurons, and we can expect there to be about $90 \cdot 0.05 = 4.5$ directed connections in the underlying network.

Connections between neurons can reveal how the information flows in the underlying physical network of neurons. Consider neurons a and b, where neuron a transmits signals towards neuron b. However, the signal from neuron a does not necessary travel straight from neuron a's axon terminal to neuron b's dendrites. It is possible that the signal from a might travel through one or several other neurons

before it reaches neuron b. Imagine there was a way of monitoring the activity of both neurons a and b simultaneously. Then, for example, we could observe that when neuron a fires, so does neuron b, and it does so rather immediately. That is, neuron a seems to provide an excitatory signal to neuron b, since neuron b fires almost immediately after neuron a has fired. Figure 1.4 shows conceptually how the activity of neuron a can affect the activity of neuron b, where the x-axis shows the time since neuron a fired, called lag. This figure illustrates an example where the most recent activity of neuron a excites neuron b, while earlier activity of neuron a inhibits neuron b. Moreover, Figure 1.4 shows that the lag can be divided into three intervals, where the numbers on the lag-axis are based on a consensus in the neuroscience community. These intervals describe the underlying physical pathway on which the signal from neuron a travels to reach neuron b.

common connection The effect in the interval from 0 to 3 ms can be attributed to a third (unobserved) neuron, say neuron c, which may have a physical connection to both neurons a and b. That is, the so-called connectivity effect seen in this interval may actually originate from neuron c, which excites both neurons a and b simultaneously.

direct connection The effect from 3 to 15 ms can be understood as a direct path between neurons a and b in the traditional sense. That is, the effect seen in this interval actually originated in neuron a and reach neuron b directly.

indirect connection The effect from 15 to 100 ms is usually understood as activity that originated at neuron a, that traveled through one (or several) other neuron(s) before reaching neuron b.

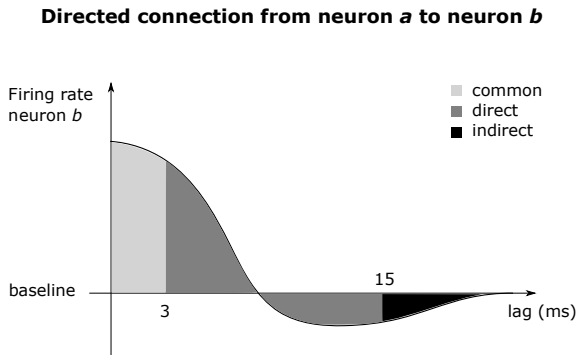

Figure 1.4: A conceptual plot of a so-called connectivity effect. The effect in the three intervals 0 to 3 ms, 3 to 15 ms and 15 to 100 ms is attributed to a common, direct and indirect connection from neuron a to neuron b, respectively. The y-axis is the so-called firing rate of neuron b, which represents how often neuron b fires. The baseline represents neurons b's average background firing rate. The x-axis is the time since neuron a fired.

1.2.4 Recording Neural Activity

In general, the activity of neurons can be recorded by so-called multielectrode arrays (MEAs), which are devices that contain multiple electrodes/channels from which neural activity can be detected. Essentially, whenever a neuron transmits an electrical signal through its axon, there is a change in voltage in its extracellular (meaning "outside the cell") environment. An MEA is able to detect this voltage change, and hence, making it possible to detect when a neuron fires.

There are two classes of MAEs: implantable and non-implantable. Using implantable MAEs, it is possible to conduct so-called *in vivo* experiments, that is, experiments on living animals. An example of such an implantable MAE is shown in Figure 1.5. This MAE has dimension 8×4, that is, there are 32 electrodes/channels that can record the neural activity in the area where this MAE is implanted.

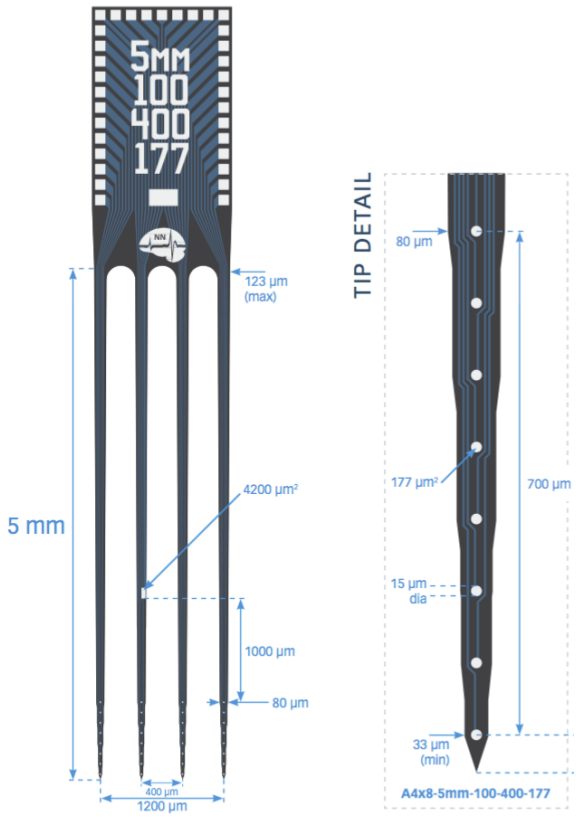

Figure 1.5: An implantable MAE. This is a silicon probe of type `A4x8-5mm-100-200-177` manufactured by NeuroNexus. The white dots are the electrodes/channels that record the neural activity.

Source: NeuroNexus' 2016 product catalog page 106. `http://neuronexus.com/images/Catalogs/2016_NeuroNexusCatalog_Web_20160115.pdf`

1.3 The `alm-1` Data Set

In this section we describe the so-called `alm-1` data set[1] underlying the analyses in this thesis. This data set contains extracellular recordings from anterior lateral motor cortex (ALM) neurons of adult mice performing a tactile decision behaviour. There are two dimensions to the `alm-1` data set, a *behavioural* dimension and a *neural* dimension, both of which are described in Section 1.3.3. Section 1.3.2 gives a quick overview of the data set. However, we begin with Section 1.3.1 with some background on why this type of data is interesting. For more details than provided in this section, see Li et al. (2015, Methods).

1.3.1 Motivation

As mentioned in Section 1.2.1, the motor cortex is crucial for planning and executing movements. Li et al. (2015) report that neurons in the premotor cortex display preparatory activity that predict specific movements. That is, neurons in the premotor cortex are seen to be active long before a specific movement actually occurs. A central research question is then how this preparatory activity translates into commands in the motor centers that eventually trigger a movement. Furthermore, the authors mention that studies have shown that damage to the premotor cortex on one side of the brain causes a disruption in movements in the contra-lateral space. That is, there is some relation between which hemisphere of the brain displays activity in the premotor cortex, and which side of the body the movement occurs.

The ALM in a mouse brain is related to planning directional tongue movements, that is, left or right *licks*. Similar to the premotor cortex, a large proportion of neurons in the ALM display preparatory activity predicting movements. Furthermore, inactivating the ALM (through so-called photostimulation) on one hemisphere during movement planning, causes disruption in upcoming events in the contra-lateral space. Hence, the ALM in mouse brains can roughly be understood as the premotor cortex in human brains.

Focusing on how the information flows in the ALM, two types of neurons were recorded in these experiments: *intratelencephalic* neurons and *pyramidial tract* neurons. The former type of neuron projects to (that is, sends signals to) other cortical areas in the brain, while the latter projects out of the cortex and into motor-related areas in the brainstem. Thus, pyramidial tract neurons are at the output end of the network of neurons in the ALM.

1.3.2 Data Structure

The complete data set contains behavioural and neural data from 19 mice. Experiments on each of these 19 mice are done independently. However, each mouse undergoes the same experimental procedure over 3-8 consecutive days, called *sessions*. Each session lasts between 1 to 2 hours, and consists of several *trials*. In each

[1] The `alm-1` data set is provided by Nuo Li, from the lab of Karel Svaboda at Jalia Farm, and is available for download at `http://crcns.org/data-sets/motor-cortex/alm-1`.

trial, the mouse is instructed to carry out some tactile decision (explained in more detail in Section 1.3.3). These trial-wise decisions are recorded, along with the mouse's neural activity over the entire session. This neural activity is represented as time stamps (in session time) whenever each identified neuron in the ALM is active (that is, whenever it fires).

This data structure is conceptualized in Figure 1.6. This figure shows that the trials are of fixed length (each trial lasts 5 s, which we've seen in the data set), and that between successive trials, there is a gap of several seconds (perhaps used by the experimenter to initialize any experimental conditions). Since the activity of the neurons is recorded during the whole session time, some of the activity happens during a trial, while some happens in-between trials. We've chosen to ignore the activity that happens in-between trials. That is, our analyses are based on activity that falls within the 5 s long trials, which correspond to the time-stamps illustrated by the thick vertical lines in Figure 1.6.

See Appendix A.1 for a summary of the number of sessions, trials and neurons for each of the 19 mice in the `alm-1` data set.

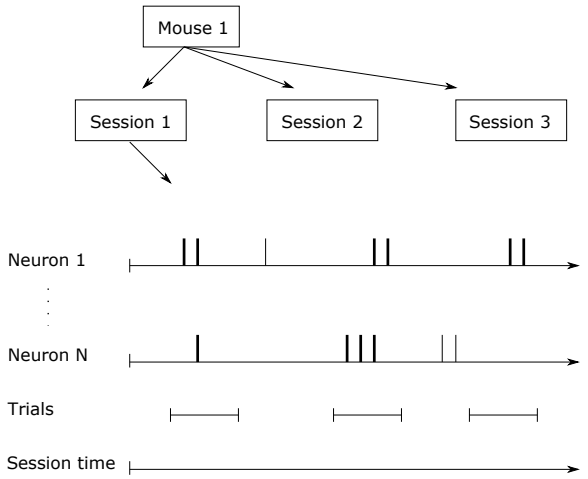

Figure 1.6: Structure of the `alm-1` data set. In this scenario, `Mouse 1` has three sessions, in which there are recordings of N neurons (a different number in each session). The vertical lines for each neuron represent the time-stamp (in session time) when that neuron was active. The thicker time-stamps represents activity that happens during trials (included in our analyses), while the thinner time-stamps represents activity outside trials (ignored in our analyses).

1.3.3 Experimental Methods

Behavioural Dimension Each trial has a fixed structure, composed of a so-called *sample epoch*, a *delay epoch* and a *response epoch*. A stimulus is provided to the mouse in the sample epoch, in the form of a metal pole, which the mouse

can feel with its whiskers. This pole can appear at two locations, and the mouse is trained (prior to the experiment) to give a physical response depending on the pole's location. This physical response is in the form of two distinct tongue movements: a left lick and a right lick, each corresponding to a specific pole location. That is, the mouse receives a distinct stimulus in the sample epoch, and makes a decision whether to lick left or to lick right. Right before the delay epoch, the pole is retracted, and the mouse is supposed to remember its decision (to lick left or right). An auditory cue marks the start of the response epoch, which informs the mouse that it's time to give a response based on the stimulus provided earlier. That is, the left or right lick happens during the response epoch. This trial structure is shown in Figure 1.7. In addition, this figure shows that the sample and the delay epochs last 1.3 s each. We've observed that this varies somewhat across trials, however, the total length of the trial is exactly 5 s (for all trials).

Trials are distinguished based on the mouse's performance. According to the documentation[2] accompanying the data set, only trials where the mouse is performing should be analyzed. These trials are termed `GoodTrials`. In these so-called `GoodTrials`, there is a subset of trials where the mouse gives a correct response. That is, this subset of correct trials are trials where the pole was at a location corresponding to, say, a left lick, and the mouse in fact responded with a left lick. Hence, these correct trials can be divided into two disjoint sets, called *correct lick left trials* and *correct lick right trials*.

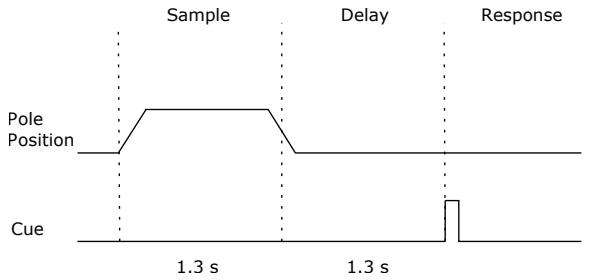

Figure 1.7: Trial structure. This figure is copied from Li et al. (2015, p. 52).

Neural Dimension The neural activity is recorded using the 8×4 silicon probe shown earlier in Figure 1.5. This probe was implanted in the ALM by performing a craniotomy (a surgical operation that removes a part of the scull to access the brain) on the mouse one day before the first session. An actual example of the 8×4 grid of electrodes/channels on which the neurons are recorded is shown in Figure 1.8. Whenever a neuron fires, the change in electrical charge in its extracellular environment is measured at the blue dots of this figure, as described in Section 1.2.4. This activity is then stored as a time-stamp (illustrated as vertical lines in Figure 1.6).

Furthermore, the plots in Figure 1.8 show how the neurons are located relative to each other in two-dimensional space in the ALM, which might be of importance

[2] Available at `http://crcns.org/files/data/alm-1/crcns_alm-1_data_description.pdf`.

when we aim to estimate the underlying network of neurons in Chapter 7. Note that several neurons have been recorded on the same electrode/channel. The numbers displayed in these plots are arbitrary labels used to distinguish between neurons within a session. That is, neuron j in session 1 is not the same as neuron j in session 2. Also, it is not a given that an electrode/channel records the same neuron over several sessions. For example, although neurons 12, 3 and 10 in sessions 1, 2 and 3 appear to be at the same location, they might not represent the same neuron. This is because the implanted probe might have moved between sessions. Finally, we note that neuron 15 in session 2 has been recorded at two different locations (both at top row at third column, and at third row at second column). It is unclear why this is the case.

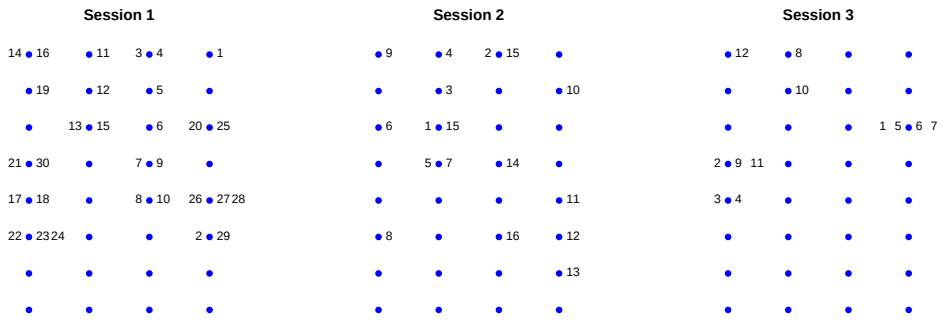

Figure 1.8: The recorded neurons on the 8×4 silicon probe. These are sessions `20130701`, `20130702` and `20130703` from mouse `ANM210861`, where there were identified a total of 30, 16 and 12 neurons, respectively. Each blue dot represents an electrode/channel on the probe where neural activity can be detected. The numbers are neuron labels, chosen arbitrary to distinguish neurons within a session.

1.4 Outline

In Chapter 2 we present the generalized linear model (GLM) framework, which lays the foundation for the regularized regression models, specifically the lasso, presented in Chapter 4. Chapter 3 discusses challenges related to multiple hypothesis testing, which is relevant for the multi-sample splitting procedure explained in Section 4.6, which is the main method of inference considered for the parameters of the lasso penalized regression model. In Chapter 5 we present additive models, which enables us to model non-linear effects in the predictors, central to our analyses. Chapters 2, 3, 4 and 5 complete Part I of this thesis.

Part II of this thesis begins with Chapter 6, where we specify the regression models considered in our analyses. The main results are presented in Chapter 7. Chapter 8 concludes this thesis with a discussion and a conclusion.

Part I

Statistical Models and Methods

Chapter 2

The Generalized Linear Model

In this chapter we introduce the generalized linear model (GLM), of which logistic regression is a special case. We also present the iteratively re-weighted least squares (IWLS) algorithm, which is a method to find the maximum likelihood estimates (MLEs) of a GLM. Furthermore, we briefly discuss aspects of model selection such as the deviance, the Akaike information criterion (AIC) and deviance residuals. We also consider hypothesis tests for the regression parameters such as the Wald test and the likelihood ratio (LR) test.

2.1 The GLM-framework

In short, the GLM consists of three elements (i) a probability distribution from the exponential family (ii) a linear predictor η (iii) and a link function g that connects the mean of the probability distribution to the linear predictor. In the following we elaborate on each of these elements.

Consider n independent observations $y_1, ..., y_n$, where y_i is treated as a realisation of a random variable Y_i. Assume that Y_i has a probability distribution from the exponential family, that is

$$Y_i \sim f_{Y_i}(y_i; \theta_i, \phi),$$

where θ_i is the (one dimensional) parameter of the family, and ϕ is called the dispersion parameter. These parameters are essentially location and scale parameters, respectively. The probability distribution function can be expressed as (McCullagh and Nelder, 1989, p. 28)

$$f_{Y_i}(y_i; \theta_i, \phi) = \exp\Big(\frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\Big), \tag{2.1}$$

where $a(\phi)$, $b(\theta_i)$ and $c(y_i, \phi)$ are known functions. If ϕ is known, θ_i is called the canonical parameter. Furthermore, θ_i is related to the mean and the variance of

the distribution through

$$\mu_i = \mathrm{E}(Y_i) = b'(\theta_i) \tag{2.2}$$
$$\mathrm{Var}(Y_i) = b''(\theta_i)a(\phi) \tag{2.3}$$

The linear predictor can be written as

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}, \tag{2.4}$$

where $\boldsymbol{\eta} = (\eta_1, ..., \eta_n)^T$, $\mathbf{X}$ is an $n \times p$ design matrix and $\boldsymbol{\beta}$ is a $p \times 1$ vector of the unknown parameters. These unknown parameters can be estimated by maximizing the likelihood function, as will be discussed in Section 2.2. For models with an intercept term, the first column of the design matrix $\mathbf{X}$ is a column of ones.

The link function g connects the mean of the distribution to the linear predictor by

$$\eta_i = g(\mu_i). \tag{2.5}$$

Whenever θ_i is a canonical parameter in (2.1), the link function g is the function that expresses θ_i in terms of μ_i, that is

$$\theta_i = g(\mu_i). \tag{2.6}$$

In this setting g is referred to as the canonical link function. It is possible to use non-canonical link functions, but then consideration should be made such that the domain of the link function matches the range of the mean of the probability distribution.

2.1.1 Linear Model for Continuous Data

Consider n independent observations $y_1, ..., y_n$ where $y_i \in \mathbb{R}$ is treated as a realization of a normally distributed random variable Y_i. That is

$$\begin{aligned} Y_i &\sim N(\mu_i, \sigma^2) \\ f_{Y_i}(y_i; \theta_i, \phi) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{1}{2}\frac{(y_i - \mu_i)^2}{\sigma^2}\Big) \\ &= \exp\Big(\frac{\mu_i y_i - \frac{1}{2}\mu_i^2}{\sigma^2} - \frac{y_i^2}{2\sigma^2} - \frac{1}{2}\log\big(2\pi\sigma^2\big)\Big). \end{aligned} \tag{2.7}$$

Comparing (2.7) to (2.1), we see that $a(\phi) = \phi = \sigma^2$, that is, θ_i is a canonical parameter. We also note that $b(\theta_i) = \frac{1}{2}\mu_i^2$ which, according to (2.2) and (2.3), verifies that μ_i and σ^2 are the mean and the variance of Y_i, respectively. Furthermore, we identify θ_i as μ_i which, according to (2.6), leads to the canonical link function

$$g(\mu_i) = \mu_i,$$

i.e. the identity function. Hence, by combining (2.4) and (2.5), we get the familiar linear regression model

$$\mu_i = \mathbf{x}_i\boldsymbol{\beta}, \tag{2.8}$$

where $\mathbf{x}_i$ is the ith row of the design matrix $\mathbf{X}$.

2.1.2 Binomial Model for Binary Data

Consider that the n independent observations $y_1, ..., y_n$ are such that each y_i is the number of *successes* in n_i independent Bernoulli trials. That is, $y_i \in \{0, ..., n_i\}$. The result of one single Bernoulli trial is a random variable Z such that (Dobson and Barnett, 2008, p. 123)

$$Z = \begin{cases} 1, & \text{with} \quad P(Z=1) = p \\ 0, & \text{with} \quad P(Z=0) = 1-p, \end{cases} \tag{2.9}$$

where $z = 1$ is considered a success. That is, each y_i is a realization of a random variable

$$Y_i = \sum_j^{n_i} Z_{i,j},$$

where $P(Z_{i,j} = 1) = p_i$ for all j's. The distribution of Y_i is then given by

$$\begin{aligned} Y_i &\sim \text{binomial}(n_i, p_i) \\ f_{Y_i}(y_i; \theta_i, \phi) &= \binom{n_i}{y_i} p_i^{y_i} (1-p_i)^{n_i - y_i} \\ &= \exp\left(\log\left(\frac{p_i}{1-p_i} \right) y_i + n_i \log(1-p_i) + \log\binom{n_i}{y_i} \right). \end{aligned} \tag{2.10}$$

When $n_i = 1$, the distribution of Y_i reduces to the Bernoulli distribution of Z. Comparing (2.10) to (2.1), we see that $a(\phi) = \phi = 1$, that is, θ_i is a canonical parameter. Furthermore, we have that

$$\theta_i = \log\left(\frac{p_i}{1-p_i} \right)$$

and

$$b(\theta_i) = n_i(\theta_i + \log(1 + \exp(-\theta_i))),$$

where we've used

$$p_i = (1 + \exp(-\theta_i))^{-1} \tag{2.11}$$

in the expression for $b(\theta_i)$. Hence, from (2.2) and (2.3) we get

$$\begin{aligned} \mu_i = \text{E}(Y_i) &= \frac{n_i}{1 + \exp(-\theta_i)} = n_i p_i \\ \text{Var}(Y_i) &= \frac{n_i}{(1 + \exp(-\theta_i))^2} = n_i p_i (1-p_i). \end{aligned} \tag{2.12}$$

And according to (2.6), the canonical link function is

$$\begin{aligned} g(\mu_i) &= \log\left(\frac{p_i}{1-p_i} \right) \\ &= \text{logit}(p_i) \end{aligned} \tag{2.13}$$

where the last step is the definition of the logit-function. Hence, combining (2.4) and (2.5) gives the binomial regression model

$$\text{logit}(p_i) = \mathbf{x}_i \boldsymbol{\beta}. \tag{2.14}$$

2.2 Parameter Estimation

As mentioned, the unknown parameters $\boldsymbol{\beta}$ in (2.4) can be estimated by maximizing the likelihood function. These estimates are called maximum likelihood estimates (MLEs), and are denoted $\hat{\boldsymbol{\beta}}$. Except in the case of linear regression, calculation of the MLEs require iterative methods. In our analysis we've used the built-in function `glm` in `R` to obtain an estimate for $\hat{\boldsymbol{\beta}}$. The `glm` function uses a method called iteratively re-weighted least squares (IWLS). The following is an overview of the IWLS algorithm.

2.2.1 The Log-Likelihood Function

The log-likelihood function for a single observation y_i is given by

$$\begin{aligned} \log L_i(\theta_i, \phi; y_i) &= \log f_{Y_i}(y_i; \theta_i, \phi) \\ &= \frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi), \end{aligned} \tag{2.15}$$

where we've used (2.1). And the log-likelihood for the set of independent observations $\mathbf{y} = (y_1, ..., y_n)^T$ is simply

$$\log L(\boldsymbol{\theta}, \phi; \mathbf{y}) = \sum_{i=1}^{n} \log L_i(\theta_i, \phi; y_i). \tag{2.16}$$

Note that θ_i is related to the mean of the distribution μ_i in (2.2), which itself is related to the linear predictor η_i in (2.5). And η_i is again related to the unknown parameters $\boldsymbol{\beta}$ through (2.4). This is relevant because it shows that the log-likelihood function in (2.15), and hence in (2.16), are functions of $\boldsymbol{\beta}$.

A formal definition of the MLE $\hat{\boldsymbol{\beta}}$ is given by (Rodríguez, 2007, Appendix A, p. 1)

$$\log L(\hat{\boldsymbol{\beta}}; \mathbf{y}) \geq \log L(\boldsymbol{\beta}; \mathbf{y}) \quad \text{for all} \quad \boldsymbol{\beta}. \tag{2.17}$$

2.2.2 Fisher's Score Function and the Information Matrix

The first derivative of the log-likelihood is called Fisher's score function, and is given by (Rodríguez, 2007, Appendix A, p. 3)

$$\mathbf{u}(\boldsymbol{\beta}) = \frac{\partial \log L(\boldsymbol{\beta}; \mathbf{y})}{\partial \boldsymbol{\beta}}, \tag{2.18}$$

where $\mathbf{u}$ is a $p \times 1$ vector whenever $\boldsymbol{\beta}$ is a $p \times 1$ vector, as in (2.4). In general the score function can be written as a function of both the parameters $\boldsymbol{\beta}$ and the random vector $\mathbf{Y} = (Y_1, ..., Y_n)^T$, where observation y_i is considered a realisation of the random variable Y_i. Hence $\mathbf{u}(\boldsymbol{\beta}, \mathbf{Y})$ is a random vector. In that case, its mean and covariance are given by (Rodríguez, 2007, Appendix A, p. 3)

$$\begin{aligned} \mathrm{E}(\mathbf{u}(\boldsymbol{\beta}, \mathbf{Y})) &= 0 \\ \mathrm{Var}(\mathbf{u}(\boldsymbol{\beta}, \mathbf{Y})) &= \mathrm{E}(\mathbf{u}(\boldsymbol{\beta}, \mathbf{Y})\mathbf{u}(\boldsymbol{\beta}, \mathbf{Y})^T) = \mathbf{I}(\boldsymbol{\beta}), \end{aligned} \tag{2.19}$$

where $\mathbf{I}$ is a $p \times p$ matrix, called the information matrix.

If the log-likelihood is a concave function, we can find the MLE by setting the first derivative of the log-likelihood equal to zero (Rodríguez, 2007, Appendix A, p. 3). That is, we need to solve the system of equations

$$\mathbf{u}(\hat{\boldsymbol{\beta}}) = \mathbf{0}. \tag{2.20}$$

Using a first order Taylor series, we can expand the score function evaluated at the MLE $\hat{\boldsymbol{\beta}}$ around an arbitrary value $\boldsymbol{\beta}_0$

$$\mathbf{u}(\hat{\boldsymbol{\beta}}) \approx \mathbf{u}(\boldsymbol{\beta}_0) + \frac{\partial \mathbf{u}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}\bigg|_{\boldsymbol{\beta}=\boldsymbol{\beta}_0} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0),$$

and using (2.20), we can solve for $\hat{\boldsymbol{\beta}}$, such that

$$\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}_0 - \mathbf{H}^{-1}(\boldsymbol{\beta}_0)\mathbf{u}(\boldsymbol{\beta}_0), \tag{2.21}$$

where $\mathbf{H}(\boldsymbol{\beta}) = \partial \mathbf{u}(\boldsymbol{\beta})/\partial \boldsymbol{\beta}$ is a $p \times p$ matrix called the Hessian. This expression forms the basis of an iterative technique called the Newton-Raphson method (Rodríguez, 2007, Appendix A, p. 5), where a given trial value is updated using (2.21) until convergence. An alternative method, known as Fisher scoring, is given by (Rodríguez, 2007, Appendix A, p. 5)

$$\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}_0 + \mathbf{I}^{-1}(\boldsymbol{\beta}_0)\mathbf{u}(\boldsymbol{\beta}_0), \tag{2.22}$$

where the Hessian in (2.21) is approximated by the information matrix $\mathbf{I}$ using (Rodríguez, 2007, Appendix A, p. 4)

$$\mathbf{I}(\boldsymbol{\beta}) = -\mathrm{E}\Big(\frac{\partial^2 \log L(\boldsymbol{\beta}; \mathbf{y})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}\Big) = -\mathrm{E}(\mathbf{H}(\boldsymbol{\beta})). \tag{2.23}$$

To find expressions for the score function $\mathbf{u}(\boldsymbol{\beta})$ and the information matrix $\mathbf{I}(\boldsymbol{\beta})$, we differentiate the log-likelihood in (2.15) using the chain rule (McCullagh and Nelder, 1989, p. 41)

$$\frac{\partial \log L_i}{\partial \beta_j} = \frac{\partial \log L_i}{\partial \theta_i} \frac{\mathrm{d}\theta_i}{\mathrm{d}\mu_i} \frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} \frac{\partial \eta_i}{\partial \beta_j}.$$

Using (2.2) and (2.3) we derive that $\mathrm{d}\mu_i/\mathrm{d}\theta_i = \mathrm{Var}(Y_i)/a(\phi)$, and from (2.4) we get that $\partial \eta_i/\partial \beta_j = x_{ij}$, such that

$$\frac{\partial \log L_i}{\partial \beta_j} = \frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} x_{ij}.$$

This gives that each component of the score function in (2.18) is given by (Dobson and Barnett, 2008, p. 65)

$$u_j = \sum_{i=1}^{n} \frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} x_{ij}. \tag{2.24}$$

And using (2.19), we get that the elements of the information matrix $\mathbf{I}(\boldsymbol{\beta})$ are given by (Dobson and Barnett, 2008, p. 65)

$$\begin{aligned} I_{jk} &= \mathrm{E}(u_j u_k) \\ &= \mathrm{E}\Big(\sum_{i=1}^{n} \frac{(Y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} x_{ij} \sum_{l=1}^{n} \frac{(Y_l - \mu_l)}{\mathrm{Var}(Y_l)} \frac{\mathrm{d}\mu_l}{\mathrm{d}\eta_l} x_{lk} \Big) \\ &= \sum_{i=1}^{n} \frac{\mathrm{E}(Y_i - \mu_i)^2 x_{ij} x_{ik}}{(\mathrm{Var}(Y_i))^2} \Big(\frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} \Big)^2, \end{aligned}$$

since $\mathrm{E}((Y_i - \mu_i)(Y_l - \mu_l)) = 0$ for $i \neq l$ because the random variables Y_i are independent, since the observations themselves y_i are assumed independent (Dobson and Barnett, 2008, p. 65). And by using $\mathrm{E}(Y_i - \mu_i)^2 = \mathrm{Var}(Y_i)$ we get that

$$I_{jk} = \sum_{i=1}^{n} \frac{x_{ij} x_{ik}}{\mathrm{Var}(Y_i)} \Big(\frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} \Big)^2. \tag{2.25}$$

Finally, we can write the information matrix as (Dobson and Barnett, 2008, p. 66)

$$\mathbf{I}(\boldsymbol{\beta}) = \mathbf{X}^T \mathbf{W}(\boldsymbol{\beta}) \mathbf{X}, \tag{2.26}$$

where we define $\mathbf{W}(\boldsymbol{\beta})$ as an $n \times n$ diagonal matrix with elements

$$w_{ii} = \frac{1}{\mathrm{Var}(Y_i)} \Big(\frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} \Big)^2, \tag{2.27}$$

where $\mathrm{d}\mu_i / \mathrm{d}\eta_i$ is evaluated at $\boldsymbol{\beta}$.

2.2.3 The Iteratively Re-Weighted Least Squares (IWLS) Algorithm.

Consider now the expression in (2.22) as an iterative algorithm (Dobson and Barnett, 2008, p.65) such that

$$\mathbf{b}^m = \mathbf{b}^{m-1} + \mathbf{I}^{-1}(\mathbf{b}^{m-1}) \mathbf{u}(\mathbf{b}^{m-1}),$$

where $\mathbf{b}^m$ is a vector of estimates of the parameters $\boldsymbol{\beta}$ at the mth iteration. Multiplying both sides of this expression with the information matrix $\mathbf{I}$, we get

$$\mathbf{I}(\mathbf{b}^{m-1}) \mathbf{b}^m = \mathbf{I}(\mathbf{b}^{m-1}) \mathbf{b}^{m-1} + \mathbf{u}(\mathbf{b}^{m-1}), \tag{2.28}$$

Using (2.24) and (2.25), we see that the right-hand side of this expression is

$$\sum_{k=1}^{p} \sum_{i=1}^{n} \frac{x_{ij} x_{ik}}{\mathrm{Var}(Y_i)} \Big(\frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} \Big)^2 b_k^{m-1} + \sum_{i=1}^{n} \frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{\mathrm{d}\mu_i}{\mathrm{d}\eta_i} x_{ij},$$

which can be written as

$$\mathbf{X}^T \mathbf{W}(\mathbf{b}^{m-1}) \mathbf{z}(\mathbf{b}^{m-1}), \tag{2.29}$$

where we've defined $\mathbf{z}(\mathbf{b}^{m-1})$ with components

$$z_i = \sum_{k=1}^{p} x_{ik} b_k^{(m-1)} + (y_i - \mu_i)\frac{\mathrm{d}\eta_i}{\mathrm{d}\mu_i}. \tag{2.30}$$

Using (2.26) and (2.29), we can rewrite (2.28) as

$$\mathbf{X}^T\mathbf{W}(\mathbf{b}^{m-1})\mathbf{X}\mathbf{b}^m = \mathbf{X}^T\mathbf{W}(\mathbf{b}^{m-1})\mathbf{z}(\mathbf{b}^{m-1}). \tag{2.31}$$

Note that for a normal linear model, the so called normal equations have the form (Dobson and Barnett, 2008, p. 89)

$$\mathbf{A}^T\mathbf{A}\mathbf{b} = \mathbf{A}^T\mathbf{y}, \tag{2.32}$$

where $\mathbf{A}$ is the design matrix, $\mathbf{b}$ is the estimate of the MLE (called the least square estimator) and $\mathbf{y}$ are the data. Comparing (2.31) and (2.32), we see that they have the same form, except for the weights $\mathbf{W}$ in (2.31). Furthermore, the equations in (2.31) need to be solved iteratively, contrary to (2.32), since both $\mathbf{z}$ and $\mathbf{W}$ are dependent on $\mathbf{b}^{(m-1)}$. Hence, the iterative method of (2.31) is called the iteratively re-weighted least squares (IWLS) algorithm. This algorithm is said to converge whenever the difference between $\mathbf{b}^{(m)}$ and $\mathbf{b}^{(m-1)}$ becomes small, relative to a tolerance. Then, $\mathbf{b}^{(m)}$ is taken to be an estimate of the MLE $\hat{\boldsymbol{\beta}}$ (Dobson and Barnett, 2008, p. 66).

2.3 Model Adequacy

In this section we discuss model adequacy, and define the deviance and the deviance residuals. The deviance is later used in Section 2.4.2 to construct likelihood ratio tests. Furthermore, we discuss Akaike's information criterion (AIC) as an approximation to the Kullback-Leibler (K-L) information.

2.3.1 Log-Likelihood Ratio Statistic

To assess the adequacy of a model, say ω, we compare it with the so called saturated model Ω. The two models have the same distribution and the same link function, but the saturated model Ω is more general than ω, since it has the maximum number of parameters, say m, usually one for each observation. We now define the likelihood ratio (Dobson and Barnett, 2008, p. 79)

$$\lambda = \frac{L(\hat{\boldsymbol{\beta}}_\Omega; \mathbf{y})}{L(\hat{\boldsymbol{\beta}}_\omega; \mathbf{y})},$$

where $\hat{\boldsymbol{\beta}}_\Omega$ is the MLE of $\boldsymbol{\beta}$ under the saturated model Ω, and similarly $\hat{\boldsymbol{\beta}}_\omega$ is the MLE of $\boldsymbol{\beta}$ under the model of interest ω. Generally, the saturated model Ω is considered a perfect fit to a given set of data $\mathbf{y}$. That is, under the same assumed distribution and link function, no other model will fit the data better than model Ω, such that $\lambda \geq 1$ (Dobson and Barnett, 2008, p. 79).

The log-likelihood ratio is simply

$$\log \lambda = \log L(\hat{\boldsymbol{\beta}}_\Omega) - \log L(\hat{\boldsymbol{\beta}}_\omega) \tag{2.33}$$

2.3.2 The Deviance

The deviance is defined as (Dobson and Barnett, 2008, p. 80)

$$D = 2 \log \lambda, \tag{2.34}$$

and in the following we derive its sampling distribution.

The log-likelihood function evaluated at $\boldsymbol{\beta}$ can be approximated by a Taylor series expansion around an arbitrary value $\boldsymbol{\beta}_0$ as (Dobson and Barnett, 2008, p. 77)

$$\log L(\boldsymbol{\beta}) = \log L(\boldsymbol{\beta}_0) + (\boldsymbol{\beta} - \boldsymbol{\beta}_0)^T \mathbf{u}(\boldsymbol{\beta}_0) - \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^T \mathbf{I}(\boldsymbol{\beta}_0)(\boldsymbol{\beta} - \boldsymbol{\beta}_0), \tag{2.35}$$

where we've approximated the Hessian by its expected value $-\mathbf{I}(\boldsymbol{\beta}_0)$, as defined in (2.23). If we move the first term on the right-hand side of this equation to the left-hand side, and set $\boldsymbol{\beta}_0$ equal to the MLE of $\boldsymbol{\beta}$, we get

$$\log L(\boldsymbol{\beta}) - \log L(\hat{\boldsymbol{\beta}}) = -\frac{1}{2}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T \mathbf{I}(\hat{\boldsymbol{\beta}})(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}),$$

where $\mathbf{u}(\hat{\boldsymbol{\beta}}) = 0$ according to (2.20). This gives that

$$2(\log L(\hat{\boldsymbol{\beta}}) - \log L(\boldsymbol{\beta})) = (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})^T \mathbf{I}(\hat{\boldsymbol{\beta}})(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}).$$

The expression on the right-hand side is the Wald statistic, which is shown to have an asymptotic chi-square sampling distribution in Section 2.4.1, with degrees of freedom equal to the number of parameters. The distribution of the deviance D can now be derived by considering (Dobson and Barnett, 2008, p. 80)

$$\begin{aligned} D &= 2(\log L(\hat{\boldsymbol{\beta}}_\Omega) - \log L(\hat{\boldsymbol{\beta}}_\omega)) \\ &= 2(\log L(\hat{\boldsymbol{\beta}}_\Omega) - \log L(\boldsymbol{\beta}_\Omega)) - 2(\log L(\hat{\boldsymbol{\beta}}_\omega) - \log L(\boldsymbol{\beta}_\omega)) \\ &+ 2(\log L(\boldsymbol{\beta}_\Omega) - \log L(\boldsymbol{\beta}_\omega)), \end{aligned} \tag{2.36}$$

where we've simply subtracted and added the terms involving log-likelihoods of $\boldsymbol{\beta}_\Omega$ and $\boldsymbol{\beta}_\omega$. The first two terms in (2.36) have the distributions $\chi^2(m)$ and $\chi^2(p)$, respectively (see Section 2.4.1), where m is the number of parameters in the saturated model Ω and p is the number of parameters in the model of interest ω. The third term in (2.36), $v = 2(\log L(\boldsymbol{\beta}_\Omega) - \log L(\boldsymbol{\beta}_\omega))$, is a positive constant, which will be close to zero when our model of interest fits the data as well as the saturated model. Hence the deviance has the approximate distribution (Dobson and Barnett, 2008, p. 80)

$$D \sim \chi^2(m - p, v), \tag{2.37}$$

where v is the non-centrality parameter. A non-central chi-squared distribution, denoted $\chi^2(n, \lambda)$, is defined as the distribution of

$$\sum_{i=1}^{n}(Z_i + c_i)^2,$$

where the Z_i's are i.i.d. standard normal and the c_i's are arbitrary constants. Then, the non-centrality parameter is defined as $\lambda = \sum_i^n c_i^2$ (Dobson and Barnett, 2008, p. 8).

In Section 2.4.2 the deviance is used to test nested models.

2.3.3 Goodness of Fit

The deviance can also be used as a summary statistic to test the goodness of fit of a model. As seen in (2.33), the deviance compares the likelihoods of the model of interest ω and the saturated model Ω. As stated above, the likelihood function under the saturated model is greater than the likelihood under any other model. Hence, any model that has a small value of the deviance is considered to fit the data well. This argument gives an informal approach to comparing two models, say ω_1 and ω_2, that need not be nested. E.g. say $D(\omega_1) < D(\omega_2)$, then model ω_1 is considered a better fit to the data than model ω_2. Note, this is not a statistical test, but only an informal comparison. The goodness of fit of a model can be tested using the likelihood ratio (LR) test, as described in Section 2.4.2.

Another goodness of fit statistic worth mentioning is the Akaike's information criterion (AIC). The AIC is defined as (Dobson and Barnett, 2008, p. 137)

$$AIC = -2 \log L(\hat{\boldsymbol{\beta}}; \mathbf{y}) + 2p, \tag{2.38}$$

where p is the number of parameters in the model of interest that can be estimated from the data. As in the case for the deviance, we can use the AIC to informally compare the fit of two models ω_1 and ω_2. Whichever model has the lowest AIC is usually preferred.

Note that both the deviance and the AIC are based on the likelihood function. Hence, if we are to compare two competing models using the deviance or the AIC values, both models need to assume that the data are from the same exponential family and use the same link function.

Kullback-Leibler (K-L) Information However, it turns out that the AIC is an approximation to the Kullback-Leibler (K-L) information, which is defined as (Burnham and Anderson, 2002, p. 51)

$$I(f, g) = \int f(x) \log\left(\frac{f(x)}{g(x|\theta)}\right) dx, \tag{2.39}$$

where f is considered the true distribution of the data x, and g is our postulated model parametrised by θ. The K-L information is a directed distance from g to f, and can be interpreted as the information lost when g is used to approximate

f. Hence, if we have a set of candidate models $\mathcal{G} = \{g_1, ..., g_n\}$, the best model is the model $g_i \in \mathcal{G}$ that gives the smallest K-L information. Note that (2.39) can be written as (Burnham and Anderson, 2002, p. 58)

$$\begin{aligned} I(f, g) &= \int f(x) \log(f(x)) dx - \int f(x) \log(g(x|\theta)) dx \\ &= \mathrm{E}_x[\log(f(x))] - \mathrm{E}_x[\log(g(x|\theta))] \\ &= C - \mathrm{E}_x[\log(g(x|\theta))], \end{aligned} \tag{2.40}$$

where E_x denotes the expected value with respect to the distribution f. The last step in (2.40) emphasizes that $\mathrm{E}_x[\log(f(x))]$ can be treated as a constant when evaluating the K-L information for the candidate models in $\mathcal{G}$. This is convenient because the true distribution of the data f is usually not known. Hence, the problem of finding the best model in $\mathcal{G}$ reduces to evaluating the relative K-L information

$$I(f, g) - C = -\mathrm{E}_x[\log(g(x|\theta))]. \tag{2.41}$$

However, since we need to estimate the model parameters θ for each candidate model g_i, we can only obtain an estimate of the true relative K-L information. Akaike showed that the *expected* value of the relative K-L information in (2.41) can be estimated, that is, it is possible to estimate (Burnham and Anderson, 2002, p. 60)

$$\mathrm{E}_y \mathrm{E}_x \Big[\log\Big(g(x|\hat{\theta}(y)) \Big) \Big], \tag{2.42}$$

where x and y are independent random samples from the same distribution and both expectations are taken with respect to the distribution f. In large samples, an approximate unbiased estimator of (2.42) can be obtained from the log-likelihood function as (Burnham and Anderson, 2002, p. 61)

$$\log L(\hat{\theta}; y) - p, \tag{2.43}$$

where p is the number of estimable parameters in our candidate model. Multiplying this expression with -2 gives the AIC in (2.38).

This discussion shows that the AIC can be interpreted as a relative distance from a candidate model to the true distribution that generated the data, or as the information that is lost when using the candidate model. Yet, the absolute value of the AIC is not so much of interest, as is the difference in the AIC values between candidate models. Furthermore, unlike deviance values, AIC values can be compared between models that have different probability distributions. This is because the log-likelihood function in (2.43) appears as an estimator of the expected relative K-L information, whereas the deviance in (2.34) is actually defined using the log-likelihood function.

2.3.4 The Deviance for a Logistic Model

Consider again n independent observations $y_1, ..., y_n$, where y_i is a realisation of a random variable $Y_i \sim \text{binomial}(n_i, p_i)$. From the probability distribution function

of the binomial distribution in (2.10), the log-likelihood is

$$\log L(\boldsymbol{\beta}; \mathbf{y}) = \sum_{i=1}^{n} \Big(y_i \log\Big(\frac{p_i}{1-p_i} \Big) + n_i \log(1-p_i) + \log \binom{n_i}{y_i} \Big). \tag{2.44}$$

The saturated model Ω has one parameter for each observation, such that $\boldsymbol{\beta}_\Omega = (p_1, ..., p_n)^T$. Furthermore, the corresponding MLE $\hat{\boldsymbol{\beta}}_\Omega$ is such that $\hat{p}_i = y_i/n_i$ for all i's (Dobson and Barnett, 2008, p. 81). This gives the log-likelihood under the saturated model evaluated at the MLE

$$\log L(\hat{\boldsymbol{\beta}}_\Omega; \mathbf{y}) = \sum_{i=1}^{n} \Big(y_i \log\Big(\frac{y_i/n_i}{1-y_i/n_i} \Big) + n_i \log(1-y_i/n_i) + \log \binom{n_i}{y_i} \Big).$$

Assume that the model of interest ω has $p < n$ parameters. Using (2.33) and the definition of the deviance (2.34), we get the deviance for a binomial model

$$D = 2 \sum_{i=1}^{n} \Big(y_i \log\Big(\frac{y_i}{\hat{y}_i} \Big) - (n_i - y_i) \log\Big(\frac{n_i - y_i}{n_i - \hat{y}_i} \Big) \Big), \tag{2.45}$$

where the $\hat{y}_i$'s are the fitted values obtained from the model ω by using the MLE $\hat{\boldsymbol{\beta}}_\omega$.

2.3.5 Deviance Residuals

To test if our fitted model has captured all the systematic effects in the observed data, we assess the residuals. In essence, residuals are estimates of errors, defined as the difference between the observed values and the fitted values. Conceptually this can be written as (McCullagh and Nelder, 1989, p. 37)

$$\text{datum} = \text{fitted values} + \text{residuals}.$$

If our fitted model has indeed captured all the systematic effects in the data, the residuals should only contain randomness. Usually this test is done graphically by plotting the residuals and looking for any pattern. The absence of patterns in the residual plot entails randomness.

The deviance D, as defined in (2.34), is decomposed into units of d_i, one for each pair of observed and fitted value such that $D = \sum d_i$. Hence we can define the deviance residual as (McCullagh and Nelder, 1989, p. 39)

$$r_i = \text{sign}(y_i - \hat{y}_i) \sqrt{d_i},$$

where $\text{sign}(x)$ is a function that extracts the sign of a real number x, and y_i and $\hat{y}_i$ are the observed values and the fitted values, respectively.

Using (2.45) we get the deviance residuals for a binomial model

$$r_i = \text{sign}(y_i - \hat{y}_i) \big(2(y_i \log(y_i/\hat{y}_i) - (n_i - y_i) \log((n_i - y_i)/(n_i - \hat{y}_i))) \big)^{1/2}.$$

2.4 Hypothesis Testing

In this section we describe the Wald test and the likelihood ratio (LR) test, which are used in our analyses to test the significance of the regression parameters.

2.4.1 The Wald Test

Consider hypotheses of the form

$$H_0 : \boldsymbol{\beta} = \boldsymbol{\beta}_0 \quad \text{vs.} \quad H_1 : \boldsymbol{\beta} \neq \boldsymbol{\beta}_0, \tag{2.46}$$

where $\boldsymbol{\beta}$ is the $p \times 1$ vector of parameters and $\boldsymbol{\beta}_0$ a $p \times 1$ vector of fixed values. Such hypotheses can be tested using the Wald test, where the test statistic is (Rodríguez, 2007, Appendix A, p. 6)

$$W = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)^T \mathrm{Cov}(\hat{\boldsymbol{\beta}})^{-1} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0),$$

where $\mathrm{Cov}(\hat{\boldsymbol{\beta}})$ denotes the $p \times p$ variance-covariance matrix of the estimated parameters. The Wald test statistic has approximately in large samples a chi-squared distribution (Dobson and Barnett, 2008, p.85)

$$W \sim \chi^2(p) \qquad \text{(approx.)}, \tag{2.47}$$

where p is the degrees of freedom. This follows from the fact that in large samples (that is, for large values of n, the total number of observations), the MLE follows approximately a multivariate normal distribution (Rodríguez, 2007, Appendix A, p. 6)

$$\hat{\boldsymbol{\beta}} \sim N_p(\boldsymbol{\beta}, \mathrm{Cov}(\hat{\boldsymbol{\beta}})), \tag{2.48}$$

where $\mathrm{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$ denotes the true parameter values. The variance-covariance matrix of the MLE can be replaced by any consistent estimator, without altering the asymptotic distribution of W in (2.47). Particularly, the variance-covariance matrix can be estimated by

$$\widehat{\mathrm{Cov}}(\hat{\boldsymbol{\beta}}) = \mathbf{I}^{-1}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1}, \tag{2.49}$$

where we've used the expression for the information matrix in GLMs in (2.26). The elements of $\mathbf{W}$ are the weights obtained in the last iteration of the IWLS-algorithm in (2.31). Hence the Wald statistic for an MLE $\hat{\boldsymbol{\beta}}$ in a GLM-framework can be written

$$W = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)^T \mathbf{X}^T \mathbf{W} \mathbf{X} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0).$$

In our analysis, such Wald tests are used to test the significance of each estimated parameter $\hat{\beta}_j$, where the null and alternative hypotheses are

$$H_0 : \beta_j = 0 \quad \text{vs.} \quad H_1 : \beta_j \neq 0, \tag{2.50}$$

In such cases where the MLE $\hat{\beta}_j$ is a scalar, it is common to take the square root of the Wald statistic (Dobson and Barnett, 2008, p. 78)

$$z = \frac{\hat{\beta}_j}{\sqrt{\text{Var}(\hat{\beta}_j)}}, \tag{2.51}$$

where the z-statistic has a standard normal distribution in large samples. In our analyses, we test several such hypotheses simultaneously, that is, one set of hypotheses (2.50) for each regression parameter $\hat{\beta}_j$ in the model. Hence, a multiple testing correction is needed, as we discuss in Chapter 3.

2.4.2 Likelihood Ratio (LR) Test

Instead of testing for a fixed value for each estimated parameter, as in (2.46), we can alternatively compare the goodness of fit of two models. Consider hypotheses of the form

$$H_0 : \boldsymbol{\beta} = \boldsymbol{\beta}_0 = (\beta_1, ..., \beta_q)^T \tag{2.52}$$

$$H_1 : \boldsymbol{\beta} = \boldsymbol{\beta}_1 = (\beta_1, ..., \beta_p)^T, \tag{2.53}$$

where the model under the null hypothesis, say ω_0, has parameters $\boldsymbol{\beta}_0$, and the model under the alternative hypothesis, say ω_1, has parameters $\boldsymbol{\beta}_1$. Furthermore, these two models need to be nested such that $\omega_0 \subset \omega_1$, and $q < p < n$ where n is the total number of observations. These hypotheses are tested by comparing their maximum likelihoods, based on the likelihood ratio

$$\lambda = \frac{L(\hat{\boldsymbol{\beta}}_{\omega_1})}{L(\hat{\boldsymbol{\beta}}_{\omega_0})},$$

where we've replaced the parameters $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$ by their MLEs under their respective models. Multiplying this likelihood ratio by two and taking the logarithm on both sides, we get that

$$\begin{aligned} 2\log\lambda &= 2(\log L(\hat{\boldsymbol{\beta}}_{\omega_1})) - \log L(\hat{\boldsymbol{\beta}}_{\omega_0})) \\ &= 2(\log L(\hat{\boldsymbol{\beta}}_{\Omega}) - \log L(\hat{\boldsymbol{\beta}}_{\omega_0})) - 2(\log L(\hat{\boldsymbol{\beta}}_{\Omega}) - \log L(\hat{\boldsymbol{\beta}}_{\omega_1})) \\ &= D_0 - D_1 = \Delta D, \end{aligned} \tag{2.54}$$

where we've used (2.34) in the last step of this expression. If both models ω_0 and ω_1 fit the data well, then $D_0 \sim \chi^2(n-q)$ and $D_1 \sim \chi^2(n-p)$ according to (2.37) where the non-centrality parameter is set to zero for both models. Hence we get that $\Delta D \sim \chi^2(p-q)$ (Dobson and Barnett, 2008, p. 11).

This result can be used to construct analysis-of-deviance tables (McCullagh and Nelder, 1989, p. 35). The nested models in this case correspond to the sequence in which we introduce the covariates in the linear predictor in (2.4).

Example 1 (Analysis-of-deviance table)**.** Consider a model with two covariates x_1 and x_2 in addition to an intercept, such that the regression model is

$$g(\mu) = \beta_0 + x_1\beta_1 + x_2\beta_2, \tag{2.55}$$

where $\mu = \mathrm{E}(Y|x_1, x_2)$ and the response Y has a distribution from the exponential family. The corresponding analysis-of-deviance table is given in Table 2.1. In each row of this table the test statistic ΔD is used to evaluate the significance of adding the corresponding covariate, given the preceding covariate(s). This is an example of the so-called forward-selection algorithm, which continuous to add parameters to the model as long their inclusion is significant.

Model	H_0	H_1	ΔD
$g(\mu) = \beta_0 + x_1\beta_1$	$\boldsymbol{\beta} = \beta_0$	$\boldsymbol{\beta} = (\beta_0, \beta_1)$	$D_0 - D_1$
$g(\mu) = \beta_0 + x_1\beta_1 + x_2\beta_2$	$\boldsymbol{\beta} = (\beta_0, \beta_1)$	$\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$	$D_1 - D_2$

Table 2.1: Analysis-of-deviance table for model (2.55).

Now, let the alternative model in (2.53) be the saturated model Ω, which usually has one parameter for each observation. That is, we compare our model of interest ω with q parameters to the saturated model Ω with n parameters, such that (2.52) and (2.53) become

$$H_0 : \boldsymbol{\beta} = \boldsymbol{\beta}_\omega = (\beta_1, ..., \beta_q)^T \tag{2.56}$$

$$H_1 : \boldsymbol{\beta} = \boldsymbol{\beta}_\Omega = (\beta_1, ..., \beta_n)^T. \tag{2.57}$$

The saturated model has deviance equal to zero, since Ω is a perfect fit to the data. Then, (2.54) becomes

$$2\log\lambda = D_\omega,$$

where $D_\omega \sim \chi^2(n - q)$. Hence, the deviance can be used as a statistic to test the goodness of fit of a model, compared to the saturated model.

Chapter 3

Multiple hypothesis testing

In this chapter we discuss challenges in multiple hypothesis testing, and associated correction methods such as the Bonferroni and the Bonferroni-Holm methods.

When conducting a hypothesis test, there is a possibility of arriving at the wrong conclusion. Consider the single two-sided hypothesis test

$$H_0 : \beta_j = 0 \quad \text{vs.} \quad H_1 : \beta_j \neq 0, \tag{3.1}$$

where β_j could be the jth parameter in the regression model (2.4). In this case there are two possible ways of making a mistake, called type I error and type II error, as presented in Table 3.1. A common way of reporting the result of a hypothesis test is to report the p-value.

Definition 1. *A p-value $p(\mathbf{X})$ is a test statistic satisfying $0 \leq p(\mathbf{x}) \leq 1$ for every sample point $\mathbf{x}$. Small values give evidence that H_1 is true. A p-value is valid if*

$$P(p(\mathbf{X}) \leq \alpha) \leq \alpha, \tag{3.2}$$

for all α, $0 \leq \alpha \leq 1$, whenever H_0 is true. (Casella and Berger, 2002, p. 397).

A p-value is called an exact p-value if $P(p(\mathbf{X}) \leq \alpha) = \alpha$. For valid p-values, a test that rejects H_0 if $p(\mathbf{X}) \leq \alpha$, has significance level α. That is, the probability of a type I error is controlled at level α. However, when conducting multiple hypotheses tests, the probability of at least one type I error becomes inflated.

	Not reject H_0	Reject H_0
H_0 true	Correct	Type I error
H_0 false	Type II error	Correct

Table 3.1: Two types of errors in single hypothesis testing.

Consider testing m null hypotheses $\mathcal{H} = \{H_1, ..., H_m\}$. In our case, m could be the number of parameters in the regression model (2.4), and each H_j corresponds

to the hypothesis of no effect of parameter β_j (comparable to H_0 in (3.1)). An unknown number m_0 of these null hypotheses are true, and the set of these true hypotheses is termed $\mathcal{T} \subseteq \mathcal{H}$. Similarly, there is an unknown number of false hypotheses $m - m_0$ in the set $\mathcal{F} = \mathcal{H} \setminus \mathcal{T}$. We aim to infer a collection of hypotheses $\mathcal{R} \subseteq \mathcal{H}$ that is as close to $\mathcal{F}$ as possible (ideally, $\mathcal{R} = \mathcal{F}$). Assuming each individual hypothesis H_j has a corresponding p-value p_j, we let $\mathcal{R} = \{H_j; p_j \leq \alpha_{\text{loc}}\}$, where α_{loc} is a chosen threshold called the local significance level. That is, $\mathcal{R}$ is the set of rejected null hypotheses, each rejected at significance level α_{loc}. In this situation, a type I error is in the set $\mathcal{R} \cap \mathcal{T}$, and a type II error is in the set $\mathcal{F} \setminus \mathcal{R}$. Table 3.2 shows a summary of the numbers of type I and type II errors committed (Goeman and Solari, 2014), where only m and $R = |\mathcal{R}|$ are observable. This table shows that there was a total number of $V = |\mathcal{R} \cap \mathcal{T}|$ hypotheses that were falsely rejected, called false positive findings. When performing a large number of hypothesis tests, the risk of false positive findings increases. To highlight this problem, we consider the expected number of false positives, $\mathrm{E}(V)$.

Consider the special case $\mathcal{H} = \mathcal{T}$. Then, for exact p-values, the probability of a type I error is α_{loc} for each individual hypothesis. Hence, $\mathrm{E}(V) = m \cdot \alpha_{\text{loc}}$. That is, as the number of hypotheses m increases, we should expect an increasing number of false positive findings. E.g. consider we have $m = 20$ regression parameters, and would like to test if they have a significant effect on the response by testing hypotheses such as (3.1). By choosing the conventional significance level $\alpha_{\text{loc}} = 0.05$, we would expect to find $\mathrm{E}(V) = 1$ false positive. Furthermore, consider the probability of at least one false positive result, $P(V > 0)$. Assuming that the p-values of all m hypotheses tests are independent (and $\mathcal{H} = \mathcal{T}$ still), we get

$$\begin{aligned} P(V > 0) &= 1 - P(V = 0) \\ &= 1 - P\Big(\bigcap_{j=1}^{m} p_j > \alpha_{\text{loc}}\Big) \\ &= 1 - \prod_{j=1}^{m} P(p_j > \alpha_{\text{loc}}) \\ &= 1 - (1 - \alpha_{\text{loc}})^m. \end{aligned}$$

Since $0 < \alpha_{\text{loc}} < 1$, we see that $P(V > 0) \to 1$ as m increases. In the regression example, with $m = 20$ and $\alpha_{\text{loc}} = 0.05$, we get that $P(V > 0) = 0.64$. That is, when testing the significance of 20 regression parameters, each controlled at significance level 0.05, the probability of at least one false positive is 64%. For these reasons, it is necessary to take into account the number of hypotheses m being evaluated when controlling the type I error.

	Not reject H_0	Reject H_0	Total
H_0 true	$m_0 - V$	V	m_0
H_0 false	$m_1 - (R - V)$	$R - V$	m_1
Total	$m - R$	R	m

Table 3.2: Contingency table for multiple hypothesis testing. Only the total number of hypotheses m and the number of rejected hypotheses R, are observed.

3.1 Family-wise error rate (FWER)

The concept of a type I error can be generalized in different ways in the context of multiple hypothesis testing. The family-wise error rate is one such generalization, defined as the probability of at least one type I error (Goeman and Solari, 2014)

$$\text{FWER} = P(V > 0),$$

where $V = |\mathcal{R} \cap \mathcal{T}|$ is the number of type I errors as shown in Table 3.2. We will consider two methods of controlling the FWER, the Bonferroni method and its improvement, the Bonferroni-Holm method.

Consider first a collection of test statistics $S_1, ..., S_m$, one for each hypothesis tested. The corresponding p-values $p_1, ..., p_m$ are called *raw* p-values. (In the regression example, S_j could be the z-statistic (2.51), and the p_j is the p-value derived from the asymptotic normality of the jth regression parameter).

3.2 Bonferroni

The method of Bonferroni controls the FWER at level α by rejecting a hypothesis if its corresponding raw p-value is smaller than $\alpha_{\text{loc}} = \alpha/m$. To see that this controls the FWER, denote the p-values of the m_0 true hypotheses $q_1, ..., q_{m_0}$, such that the event of a type I error is $q_i \leq \alpha/m$. Then, we get that

$$\begin{aligned} \text{FWER} &= P\Big(\bigcup_{i=1}^{m_0} q_i \leq \alpha/m\Big) \\ &\leq \sum_{i=1}^{m_0} P(q_i \leq \alpha/m) \\ &\leq m_0 \frac{\alpha}{m} \leq \alpha. \end{aligned} \tag{3.3}$$

The first inequality follows from Boole's inequality.[1] The second inequality follows from the definition of a p-value of a true null hypothesis (3.2). Note that the Bonferroni method does not make any assumption regarding the dependency structure of the p-values.

[1] For any finite or countable set of events $A_1, A_2, ...$, we have $P(\cup_i A_i) \leq \sum_i P(A_i)$.

Furthermore, the inequalities in (3.3) illustrate when the Bonferroni method is conservative (meaning that the rejection criterion is overly strict, that is, $\alpha_{\text{loc}} = \alpha/m$ is smaller than it needs to be). The first inequality shows that the Bonferroni method is conservative in cases where the hypotheses are dependent, because the equality is true only when the rejection regions $q_i \leq \alpha/m$ are disjoint. The last inequality shows that the Bonferroni method controls the FWER at level $m_0\alpha/m$ rather than the intended level α. Hence, in situations were there are many false hypotheses (such that the ratio m_0/m is small), the Bonferroni method will be conservative.

It is possible to construct so called *adjusted* p-values following the Bonferroni method, given by

$$\tilde{p}_j = \min(p_j \cdot m, 1). \tag{3.4}$$

3.3 Bonferroni-Holm

The conservative nature of Bonferroni's method in cases with many false hypotheses is partly remedied by the Bonferroni-Holm method, which is an iterative variant of the Bonferroni method. In each iteration, this method expands the rejection region, in order to reject as many hypotheses as possible while controlling the FWER. The steps of the Bonferroni-Holm method are as follows.

1. Reject all hypotheses with p-values at most α/m. This step is equivalent to the Bonferroni method.

2. Let h be the number of hypotheses left unrejected, and reject all all hypotheses with p-values at most α/h.

3. Repeat step 2 until either all hypotheses are rejected, or no additional rejections occur.

The proof that this iterative procedure controls the FWER at level α is based on Boole's inequality, and no other assumptions. Hence, the Bonferroni-Holm method is as general as the original Bonferroni method, and should be preferred, specially in cases where the proportion of false hypotheses is large. In any other situation, the actual gain in power is small compared to Bonferroni's method.

Chapter 4

Regularization

In this chapter we extend the GLM-framework to incorporate a penalty on the size of the regression parameters. Specifically, we introduce the lasso estimator, and we briefly mention ridge regression. Next, we discuss some aspects of the resulting optimization problem, from which we can derive parameter estimation algorithms, such as pathwise coordinate descent. Finally, we note the difficulty of obtaining a distribution for the estimated parameters, hence, we describe inference through hypothesis testing using the pragmatic multi sample-splitting method.

4.1 The Optimization Problem

From the definition of the MLEs in (2.17), we can restate the optimization problem of fitting a generalized linear model as

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}} \log L(\boldsymbol{\beta}; \mathbf{y}), \tag{4.1}$$

where $\mathbf{y} = (y_1, ..., y_n)^T$ and $\boldsymbol{\beta} = (\beta_0, ..., \beta_p)^T$. This can be extended by imposing a penalty term on the size of the coefficients (Hastie et al., 2015, p. 30)

$$\hat{\boldsymbol{\beta}}(\nu) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{-\frac{1}{n} \log L(\boldsymbol{\beta}; \mathbf{y}) + \nu \|\boldsymbol{\beta}\|_q\}, \tag{4.2}$$

where $\nu > 0$ is the regularization parameter, and $\|\cdot\|_q$ denotes the ℓ_q-norm and is given by

$$\|\boldsymbol{\beta}\|_q = \sum_{j=0}^{p} |\beta_j|^q,$$

for any fixed real number $q \geq 0$. In (4.2) we've scaled the log-likelihood with the sample size n, which makes values of ν comparable for different sample sizes (Hastie et al., 2015, p. 9). Usually, the intercept parameter is not penalized (Buhlmann and van de Geer, 2011, p. 47). Hence, (4.2) can be written

$$\hat{\boldsymbol{\beta}}(\nu) = \underset{\beta_0, \boldsymbol{\beta}}{\operatorname{argmin}}\{-\frac{1}{n} \log L(\beta_0, \boldsymbol{\beta}; \mathbf{y}) + \nu \|\boldsymbol{\beta}\|_q\}, \tag{4.3}$$

where β_0 is the intercept parameter, and $\boldsymbol{\beta}$ is now the column vector of the remaining p model parameters, that is $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)^T$.

In short, regularization is useful when fitting a model to high-dimensional data. That is, when the number of unknown parameters p in the model is of much larger order than the sample size n (Buhlmann and van de Geer, 2011, p. 1). In order to adequately estimate the unknown parameters, $\boldsymbol{\beta}$ needs to be sparse in some sense. Sparsity can be measured in different ways depending on the ℓ_q-norm, and thus the optimization problem and the resulting parameters will have different properties depending on $q \geq 0$.

Often, regularization is used to increase prediction accuracy. The unrestricted problem (4.1) results in unbiased parameter estimates (asymptotically), as mentioned briefly in Section 2.4.1 (where $\mathrm{E}(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$ in (2.48)). However, in a prediction setting, the variance of these unrestricted parameter estimates can be large (Hastie et al., 2015, p. 7). The sparse parameter estimates following a regularized problem are more biased, but have lower variance in a prediction setting.

Another reason for considering regularization is interpretability of the resulting model. E.g. for $q = 1$ in (4.3), some of the resulting parameter estimates can be set to zero (as discussed below in Section 4.2.1), leading to a smaller subset of parameters in the estimated model.

In our analyses, we've used regularization, specifically the lasso regularization (using the ℓ_1-norm in (4.3)), for two reasons. First, the observed data in our analyses $\mathbf{y}$ has two classes (described in more detail in Section 6.1). The size of the least frequent class is generally not greater than 10 times the size of the parameter vector $\boldsymbol{\beta}$, which violates a rule-of-thumb known as the "rule of 10" (Veierød et al., 2012, Chapter 3). Essentially, this means that we generally don't have enough information in the observed data to adequately estimate all model parameters. For this reason, the unrestricted problem (4.1) fails to converge. Second, we don't expect all model parameters to be non-zero (as will be clarified in Section 6.2.3). Hence, the lasso with its variable selection property (Section 4.2.1) is useful.

4.2 The Lasso

An important case of the optimization problem in (4.3) is the lasso, where the model parameters are constrained using the ℓ_1-norm, that is, when $q = 1$ in (4.3). Furthermore, we let the likelihood in (4.3) be from the normal distribution. This is because the statistical properties of the lasso are more straightforward to develop in the context of the usual linear regression model (2.8). Later in Section 4.4 we introduce the binomial likelihood into (4.3), obtaining the lasso regularized logistic regression model central in our analyses.

For now, we consider n independent observations $y_1, ..., y_n$, where y_i is a realization of a random variable $Y_i \sim N(\mu_i, \sigma^2)$. From the probability distribution function of the normal distribution in (2.7), the log-likelihood is given by

$$\log L(\beta_0, \boldsymbol{\beta}, \sigma^2; \mathbf{y}) = -\frac{n}{2} \log\big(2\pi\sigma^2\big) - \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_0 - \mathbf{x}_i\boldsymbol{\beta})^2/\sigma^2.$$

Substituting this expression into (4.3), and considering σ^2 to be fixed, we find that the lasso estimator is (Hastie et al., 2015, p. 8)

$$\hat{\boldsymbol{\beta}}(\nu) = \underset{\beta_0,\,\boldsymbol{\beta}}{\operatorname{argmin}}\{\frac{1}{2n}\|\mathbf{y} - \beta_0\mathbf{1} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \nu\|\boldsymbol{\beta}\|_1\},$$

where $\mathbf{1}$ denotes a column vector of n ones, and $\|\cdot\|_2$ the usual Euclidean norm. For convenience, we assume that the predictors in $\mathbf{X}$ are standardized ($\sum_{i=1}^n x_{ij} = 0$ and $\frac{1}{n}\sum_{i=1}^n x_{ij}^2 = 1$) and that the response $\mathbf{y}$ is centered ($\sum_{i=1}^n y_i = 0$), such that the intercept parameter β_0 can be dropped from the model.[1] Hence, the lasso estimator can we written

$$\hat{\boldsymbol{\beta}}(\nu) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \nu\|\boldsymbol{\beta}\|_1\}, \tag{4.4}$$

This optimization problem is written in the so-called Lagrangian form (Hastie et al., 2015, p. 9), which is a convenient way of stating the problem. However, to help intuition, (4.4) can be re-expressed as a bounded optimization problem (Hastie et al., 2015, p. 8)

$$\begin{aligned}\hat{\boldsymbol{\beta}}(t) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\{&\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2\} \\ &\text{subject to } \|\boldsymbol{\beta}\|_1 \leq t,\end{aligned} \tag{4.5}$$

where t limits how well the model is fit to the data. Smaller values of t gives smaller values of $\boldsymbol{\beta}$, and for large enough t we obtain the ordinary least squares (OLS) estimates. There is a one-to-one correspondence between t in the bounded problem (4.5) and ν in the Lagrangian form (4.4) (Hastie et al., 2015, p. 9). This equivalence holds because both the objective function in the minimization and the ℓ_1 constraint region are convex in $\boldsymbol{\beta}$ (Buhlmann and van de Geer, 2011, p. 9) (convexity is defined in Definitions 2 and 3 below.)

4.2.1 Variable Selection Property

One of the desirable properties of the lasso estimator is its variable selection property. For small enough values of t in (4.5), only some of the parameters in $\hat{\boldsymbol{\beta}}(t)$ will be non-zero (Hastie et al., 2015, p. 2). Since the regularization constrains the size of the parameters, these non-zero parameters can be interpreted as the parameters that influence the response variable the most. That is, the lasso can also be understood as a model selection procedure, which is more automated than other model selection procedures such as forward-selection (mentioned briefly in

[1] In linear regression, the intercept parameter can be recovered from the model parameters estimated on the centered data, as

$$\hat{\beta}_0 = \bar{y} - \sum_{j=1}^{p} \bar{x}_j\hat{\beta}_j,$$

where $\bar{y}$ and $\bar{x}_j$ are the empirical means.

Section 2.4.2). The following is an intuitive explanation of why the lasso is able to set some of the estimated parameters exactly to zero, based on the geometry of the ℓ_1-norm.

Consider only two parameters in the linear model, that is, $\boldsymbol{\beta} = (\beta_1, \beta_2)$. The constraint region of the lasso estimator is then $|\beta_1| + |\beta_2| \leq t$, shown as the solid diamond in Figure 4.1. The solution to the bounded problem in (4.5) is the first point where the elliptical contours hit the constraint region (Hastie et al., 2015, p. 12). The situation depicted in Figure 4.1 shows that this point is located on one of the corners of the constraint region, which results in one of the parameters set to zero ($\hat{\beta}_1 = 0$).

For comparison, consider using the ℓ_2-norm as the constraint region. That is, setting $q = 2$ in the regularization (4.3). The resulting optimization problem is called ridge regression (Hastie et al., 2015, p. 10). For normally distributed data, the bounded problem is

$$\begin{aligned} \hat{\boldsymbol{\beta}}(t) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \{ \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \} \\ \text{subject to } \|\boldsymbol{\beta}\|_2 \leq t^2, \end{aligned} \tag{4.6}$$

which is similar to the lasso bounded problem (4.5) except for the constraint. For the two-parameter case $\boldsymbol{\beta} = (\beta_1, \beta_2)$, the constraint region is the solid disk shown in Figure 4.1. Unlike the lasso, the geometry of the bounded problem (4.6) does not allow any of the estimated parameters to be exactly zero.

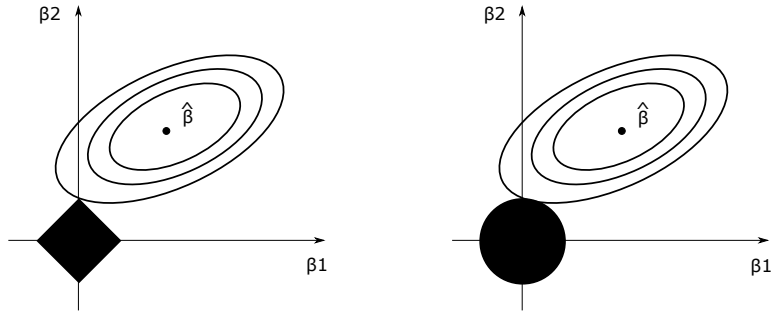

Figure 4.1: Lasso estimation (left) and ridge regression (right). The solid diamond is the constraint region in the lasso, $|\beta_1| + |\beta_2| \leq t$. The solid disk is the constraint region in ridge regression, $\beta_1^2 + \beta_2^2 \leq t^2$. The ellipses are the contour lines of the residual sum of squares function, where $\hat{\boldsymbol{\beta}}$ is the OLS solution. This figure is taken from Buhlmann and van de Geer (2011), with permission from Springer.

4.3 Parameter Estimation

In this section we consider a parameter estimation algorithm for the lasso, called pathwise coordinate descent. We rely on the lasso for the linear model, for which the

core step of this algorithm is available in closed form, called the soft-thresholding operator. This operator can be derived from the Karush-Kuhn-Tucker (KKT) optimality conditions (4.12). Hence, we begin this section by inspecting the lasso as an optimization problem. At the end of this section, an example is given where we apply the lasso to a test data set, which illustrates the parameter path and the associated cross-validation with respect to the regularization parameter ν.

4.3.1 Optimality Conditions

The following definitions from Hastie et al. (2015, p. 95) are useful.

Definition 2 (Convex set)**.** *A set $C \subseteq \mathbb{R}^p$ is convex if for all $x, y \in \mathbb{C}$, and for all scalars $s \in [0, 1]$, $z = sx + (1 - s)y \in C$.*

Definition 3 (Convex function)**.** *A function $f : \mathbb{R}^p \mapsto \mathbb{R}$ is convex if for any $x, y \in \mathbb{R}^p$, and any scalar $s \in (0, 1)$*

$$f(z) = f(sx + (1 - s)y) \leq sf(x) + (1 - s)f(y).$$

In general terms, bounded problems such as (4.5) can be stated as (Hastie et al., 2015, p. 95)

$$\begin{aligned} \boldsymbol{\beta}^* &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, f(\boldsymbol{\beta}) \\ &\text{subject to } \boldsymbol{\beta} \in C, \end{aligned} \tag{4.7}$$

where we assume that the objective function $f : \mathbb{R}^p \mapsto \mathbb{R}$ is a convex function, and that the constraint $C \subset \mathbb{R}^p$ is a convex set. The constraint C can be described using sublevel sets of several convex functions $g_l : \mathbb{R}^p \mapsto \mathbb{R}$, for $l = 1, ..., m$. Specifically, the sublevel sets have the form $\{\boldsymbol{\beta} \in \mathbb{R}^p | g_l(\boldsymbol{\beta}) \leq 0\}$. Since the g_l's are convex functions, so are their sublevel sets (Hastie et al., 2015, p. 96). Hence, (4.7) can be re-written as

$$\begin{aligned} \boldsymbol{\beta}^* &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, f(\boldsymbol{\beta}) \\ &\text{subject to } g_l(\boldsymbol{\beta}) \leq 0 \text{ for } l = 1, ...m. \end{aligned} \tag{4.8}$$

This constrained optimization problem can be reduced to an equivalent unconstrained optimization problem using the Lagrangian, $L : \mathbb{R}^p \mapsto \mathbb{R}^m_+$, defined by (Hastie et al., 2015, p. 97)

$$L(\boldsymbol{\beta}; \boldsymbol{\nu}) = f(\boldsymbol{\beta}) + \sum_{l=1}^{m} \nu_l g_l(\boldsymbol{\beta}),$$

where $\boldsymbol{\nu} = (\nu_1, ..., \nu_m)$, and the weights $\nu_l \geq 0$ are called Lagrange multipliers. Furthermore, we have that

$$\sup_{\boldsymbol{\nu}} L(\boldsymbol{\beta}; \boldsymbol{\nu}) = \begin{cases} f(\boldsymbol{\beta}) & \text{if } g_l(\boldsymbol{\beta}) \leq 0 \text{ for } l = 1, ...m, \\ +\infty & \text{otherwise.} \end{cases} \tag{4.9}$$

There exists optimal multipliers ν_l^* that impose a penalty whenever the corresponding constraint $g_l(\boldsymbol{\beta}) \leq 0$ is violated. This ensures that we're in the topmost case in equation (4.9). Hence, the solution to the bounded problem (4.8) can be written

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, L(\boldsymbol{\beta}; \boldsymbol{\nu}^*).$$

Furthermore, this solution needs to satisfy the following condition (Hastie et al., 2015, p. 97)

$$\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^*; \boldsymbol{\nu}^*) = \nabla f(\boldsymbol{\beta}^*) + \sum_{l=1}^{m} \boldsymbol{\nu}_l^* \nabla g_l(\boldsymbol{\beta}^*) = 0, \tag{4.10}$$

For $l = 1$ this condition reduces to $\nabla f(\boldsymbol{\beta}^*) = -\nu^* \nabla g(\boldsymbol{\beta}^*)$. Geometrically, this means that at the solution $\boldsymbol{\beta}^*$, the normal vector to the contour line of f and the normal vector to the constraint curve $g(\boldsymbol{\beta}) = 0$ point in opposite directions. (In the discussion above, regarding Figure 4.1, this point was referred to as the first point where the elliptical contours of the residual sum of squares function hit the constraint region.)

An example of the convex function g is the ℓ_1-norm, that is, $g(\boldsymbol{\beta}) = \sum_{j=1}^{p} |\beta_j|$. However, this function is nondifferentiable (Hastie et al., 2015, p. 98), hence the Lagrangian condition (4.10) is not directly applicable as it requires evaluation of the gradient of g. Nonetheless, for convex functions, the notion of a gradient can be generalized by subgradients (Hastie et al., 2015, p. 99).

Definition 4 (Subgradient and subdifferential)**.** *Let $f : \mathbb{R}^p \mapsto \mathbb{R}$ be a convex function. $z_i \in \mathbb{R}^p$ is a subgradient of f at $\beta \in \mathbb{R}^p$ if* [2]

$$f(\beta') \geq f(\beta) + \langle z_i, \beta' - \beta \rangle$$

for all $\beta' \in \mathbb{R}^p$. The set of all subgradients at β is called the subdifferential, denoted $\partial f(\beta) = \{z_i\}_i$.

Whenever the function f in Definition 4 is differentiable at β, the subdifferential contains a single vector $\partial f(\beta) = \{\nabla f(\beta)\}$ (Hastie et al., 2015, p. 99). Hence, a more general form of the Lagrangian condition is[3]

$$\partial_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^*; \boldsymbol{\nu}^*) = \partial f(\boldsymbol{\beta}^*) + \sum_{l=1}^{m} \boldsymbol{\nu}_l^* \partial g_l(\boldsymbol{\beta}^*) \ni 0. \tag{4.11}$$

Example 2 (Subgradients of the absolute value function)**.** Let $\beta \in \mathbb{R}$ and $g(\beta) = |\beta|$. Two subgradients of the absolute value function g at $\beta = 0$ are shown in Figure 4.2. From this illustration, we can deduce that this function has infinitly many subgradients at $\beta = 0$. That is, $\partial g(\beta)$ is a set that contains all planes that lower bound the function g at β, and can be written

$$\partial g(\beta) = \partial |\beta| = \begin{cases} \{+1\} & \text{if } \beta > 0 \\ \{-1\} & \text{if } \beta < 0 \\ [-1, +1] & \text{if } \beta = 0. \end{cases}$$

[2] Where $\langle a, b \rangle = a^T b$.

[3] Because the subdifferentiables are sets, the vector of zeros is written as an element of the set.

It is convenient to write the subgradient $z \in \partial|\beta|$ as $z \in \text{sign}(\beta)$.

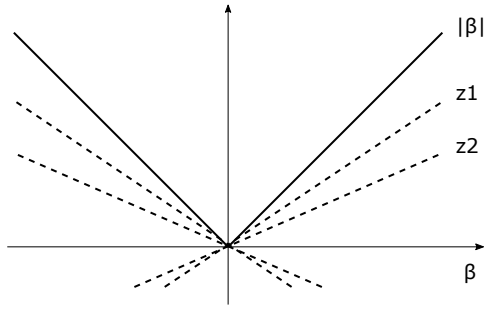

Figure 4.2: The absolute value function $g(\beta) = |\beta|$ and two of its subgradients z_1 and z_2 at $\beta = 0$.

Consider again the lasso problem for the linear regression model (4.4). To express the ℓ_1 constraint region in terms of sublevel sets $\{\boldsymbol{\beta} \in \mathbb{R}^p | g(\boldsymbol{\beta}) \leq 0\}$, we let $g(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1 - t$, where t is the radius of the ℓ_1-norm as in (4.5). Then, the Lagrangian condition (4.11) is

$$\partial_{\boldsymbol{\beta}} L(\boldsymbol{\beta}^*; \boldsymbol{\nu}^*) = \frac{1}{2n} \nabla \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*\|_2^2 + \nu^* \partial \|\boldsymbol{\beta}^*\|_1,$$

which evaluated gives the Karush-Kuhn-Tucker (KKT) optimality conditions (Hastie et al., 2015, p. 9)

$$-\frac{1}{n}\langle \mathbf{y} - \mathbf{X}\boldsymbol{\beta}^*, \mathbf{x}_j \rangle + \nu^* z_j^* = 0 \quad \text{for} \quad j = 1, ..., p, \tag{4.12}$$

where $z_j^* \in \text{sign}(\beta_j^*)$ (as in Example 2). That is, the solution to this system of equations, $\boldsymbol{\beta}^*$, is the lasso estimator in (4.4), $\hat{\boldsymbol{\beta}}$.

4.3.2 Soft thresholding

For a single covariate in the linear regression model, the lasso estimator is available in closed-form by evaluating the KKT conditions (4.12)

$$-\frac{1}{n}\sum_{i=1}^{n} y_i x_i + \hat{\beta}\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \nu^* \hat{z} = 0, \tag{4.13}$$

and assuming that the data are standardized (such that $\frac{1}{n}\sum_{i=1}^{n} x_i^2 = 1$), the solution can be written (Hastie et al., 2015, p. 15)

$$\hat{\beta} = \begin{cases} \frac{1}{n}\langle \mathbf{y}, \mathbf{x} \rangle - \nu^* & \text{if} \quad \frac{1}{n}\langle \mathbf{y}, \mathbf{x} \rangle > \nu^* \\ \frac{1}{n}\langle \mathbf{y}, \mathbf{x} \rangle + \nu^* & \text{if} \quad \frac{1}{n}\langle \mathbf{y}, \mathbf{x} \rangle < \nu^* \\ 0 & \text{if} \quad \frac{1}{n}|\langle \mathbf{y}, \mathbf{x} \rangle| \leq \nu^*, \end{cases} \tag{4.14}$$

which is compactly written as

$$\hat{\beta} = \mathcal{S}_{\nu^*}\Big(\frac{1}{n}\langle \mathbf{y}, \mathbf{x}\rangle\Big), \tag{4.15}$$

where

$$\mathcal{S}_\nu(x) = \text{sign}(x)(|x| - \nu)_+,$$

is called the soft-thresholding operator. This operator translates its argument towards zero by the amount ν, and sets it to exactly zero whenever the absolute value of the argument is smaller than ν. The argument to the soft-thresholding operator in (4.15) is the OLS solution (Hastie et al., 2015, p. 15). That is, the lasso estimator for the single covariate in a linear model is a shrunken version of the corresponding OLS estimate.

4.3.3 Cyclical coordinate descent

Consider solving the lasso problem for more than one covariate, that is, we consider optimization problems of the form (4.4). The idea of coordinate descent is to update a single covariate at a time, keeping all other covariates fixed. That is, updating the jth covariate amounts to performing a univariate optimization (Hastie et al., 2015, p. 110)

$$\beta_j^{t+1} = \underset{\beta_j}{\text{argmin}}\, h(\beta_1^t, ..., \beta_{j-1}^t, \beta_j, \beta_{j+1}^t, ..., \beta_p^t),$$

and $\beta_k^{t+1} = \beta_k^t$ for $k \neq j$, where h represents a general multivariate objective function. This algorithm is guaranteed to converge to the global minimum for problems that have the following separable form (Hastie et al., 2015, p. 110)

$$h(\boldsymbol{\beta}) = f(\boldsymbol{\beta}) + \sum_{j=1}^{p} g_j(\beta_j),$$

where $f : \mathbb{R}^p \mapsto \mathbb{R}$ is a convex and differentiable function and $g_j : \mathbb{R} \mapsto \mathbb{R}$ is a convex but not necessarily differentiable function. This separability applies to the lasso problem (4.4), where $f(\boldsymbol{\beta}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$, and $g_j(\beta_j) = \nu|\beta_j|$. In general, when the nondifferentiable part of h is a sum of univariate functions over the covariates β_j, coordinate descent converges to the global solution.

Consider again the KKT conditions for a linear regression model (4.12), which evaluated at $\hat{\boldsymbol{\beta}}$ can be written

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i - \sum_{k\neq j}^{p} x_{ik}\hat{\beta}_k - x_{ij}\hat{\beta}_j)x_{ij} + \nu^*\hat{z}_j = 0 \quad \text{for} \quad j = 1, ..., p,$$

and defining the partial residuals $r_i^{(j)} = y_i - \sum_{k\neq j}^{p} x_{ik}\hat{\beta}_k$ we can re-write this as

$$-\frac{1}{n}\sum_{i=1}^{n} r_i^{(j)} x_{ij} + \hat{\beta}_j \frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 + \nu^*\hat{z}_j = 0 \quad \text{for} \quad j = 1, ..., p.$$

Comparing this equation to (4.13), and assuming that the covariates are standardized (such that $\frac{1}{n}\sum_{i=1}^{n} x_{ij}^2 = 1$), we see that the jth parameter update is (Hastie et al., 2015, p. 112)

$$\hat{\beta}_j = \mathcal{S}_{\nu^*}\Big(\frac{1}{n}\langle \mathbf{r}^{(j)}, \mathbf{x}_j \rangle\Big).$$

Note that $\mathbf{r}^{(j)} = \mathbf{r} + \mathbf{x}_j\hat{\beta}_j$, where $\mathbf{r} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$ are the full residuals. Hence, the update can be written (Hastie et al., 2015, p. 16)

$$\hat{\beta}_j^{t+1} \leftarrow \mathcal{S}_{\nu^*}\Big(\hat{\beta}_j^{t} + \frac{1}{n}\langle \mathbf{r}^{t}, \mathbf{x}_j \rangle\Big), \tag{4.16}$$

where superscript t denotes the iteration. This updating scheme is referred to as *naive updates* (Friedman et al., 2010, p. 5). There are other updating schemes (not developed here), such as covariance updates, sparse updates and weighted updates, each designed to increase computational efficiency.

4.3.4 Pathwise Coordinate Descent

Generally we are not interested in solving the lasso problem (4.4) for a single value of the regularization parameter ν^*, but rather, for a sequence of (decreasing) values $\{\nu_l^*\}_{l=0}^{L}$ (Hastie et al., 2015, p. 17, 114). The strategy is to start with a value $\nu_0^* = \nu_{\max}^*$ just large enough so that all coordinates are zero, that is, $\hat{\boldsymbol{\beta}}(\nu_{\max}^*) = 0$. Then, a smaller ν_1^* is chosen and coordinate descent is run to estimate $\hat{\boldsymbol{\beta}}(\nu_1^*)$. For the next value in the sequence, ν_2^*, coordinate descent is initialized by the estimate from the previous step, $\hat{\boldsymbol{\beta}}(\nu_1^*)$. Such initializations are called *warm starts*. In Friedman et al. (2010), the authors report that using warm starts results in a stable and fast algorithm.

The largest value of the regularization parameter needed, $\nu_{\max}^*$, can be found by considering the coordinatewise update (4.16) (Friedman et al., 2010, p. 7). For $\hat{\boldsymbol{\beta}}^t = 0$ to stay zero, that is, $\hat{\boldsymbol{\beta}}^{t+1} = 0$, we see from the structure of the soft-thresholding operator (4.14) that

$$\frac{1}{n}|\langle \mathbf{r}^t, \mathbf{x}_j \rangle| \leq \nu^* \quad \text{for} \quad j = 1, ..., p,$$

and since $\hat{\boldsymbol{\beta}}^t = 0$, the residual is $\mathbf{r}^t = \mathbf{y}$. Hence, we get

$$\nu_{\max}^* = \max_k \frac{1}{n}|\langle \mathbf{y}, \mathbf{x}_k \rangle|.$$

The minimum value in the decreasing sequence $\{\nu_l^*\}_{l=0}^{L}$ is chosen to be $\nu_{\min}^* = \epsilon\nu_{\max}^*$. The remaining K values in this sequence are equally spaced on the log-scale between $\nu_{\max}^*$ and $\nu_{\min}^*$. Typical values in the `glmnet`-package are $\epsilon = 0.001$ and $K = 100$.

The main steps of pathwise coordinate descent are summarized in Algorithm 1.

Algorithm 1: Pathwise coordinate descent.

Result: Estimates the path of lasso solutions $\{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_l$ for a linear regression model.

initialize $\boldsymbol{\beta}^0 = 0$;

foreach $\nu^* \in \{\nu_l^*\}_{l=1}^L$ **do**

  while *not converged* **do**

    foreach $j \in \{1, ..., p\}$ **do**

      $\beta_j^{t+1} \leftarrow \mathcal{S}_{\nu^*}\Big(\beta_j^t + \frac{1}{n}\langle \mathbf{r}^t, \mathbf{x}_j \rangle\Big)$;

    end

  end

  return $\hat{\boldsymbol{\beta}}(\nu^*) = (\beta_1^{t+1}, ..., \beta_p^{t+1})$;

end

return $\hat{\boldsymbol{\beta}} = \{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_{l=1}^L$;

4.4 Lasso Regularized Logistic Regression

Our analysis is based on the lasso for a logistic regression model. That is, we consider n independent observations $y_1, ..., y_n$, where $y_i \in \{0, 1\}$ is a realization of a random variable $Y_i \sim \text{Bernoulli}(p_i)$. Setting $n_i = 1$ in the binomial log-likelihood in (2.44), we get the log-likehood for the logistic model

$$\log L(\beta_0, \boldsymbol{\beta}; \mathbf{y}) = \sum_{i=1}^{n} \Big(y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \Big). \tag{4.17}$$

Substituting this into (4.3) (and using $q = 1$), we get the corresponding lasso estimator

$$\hat{\boldsymbol{\beta}}(\nu) = \underset{\beta_0, \boldsymbol{\beta}}{\text{argmin}} \{ -\frac{1}{n} \sum_{i=1}^{n} \Big(y_i(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}) - \log(1 + \exp(\beta_0 + \mathbf{x}_i\boldsymbol{\beta})) \Big) + \nu \|\boldsymbol{\beta}\|_1 \}, \tag{4.18}$$

where we've used $p_i = \exp(\beta_0 + \mathbf{x}_i\boldsymbol{\beta})/(1 + \exp(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}))$ from (2.11). In theory, we could solve (4.18) directly by applying coordinate descent (for a single value of ν). However, the updates along each coordinate are not explicitly available, as is the case with the lasso for a linear regression model that uses the soft-thresholding operator (4.16). The `glmnet` package uses an approach that reduces the optimization problem (4.18) to a penalized weighted least-squared problem, that is, a weighted form of (4.4). Then, the resulting optimization problem can be solved by the regular cyclical coordinate descent algorithm, with coordinate updates of the form (4.16).

In more detail, let $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ be our current parameter estimates. The log-likehood part of (4.18) can be approximated by a second order Taylor expansion around

$(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ (as in (2.35)), which gives (Hastie et al., 2015, p. 116)

$$\log L_Q(\beta_0, \boldsymbol{\beta}) = -\frac{1}{2}\sum_{i=1}^{n} w_i(z_i - \beta_0 - \mathbf{x}_i\boldsymbol{\beta})^2 + C(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}), \tag{4.19}$$

where C is a constant (independent of $(\beta_0, \boldsymbol{\beta})$), and the weights w_i and so-called working response z_i are

$$\begin{aligned} w_i &= \tilde{p}_i(1 - \tilde{p}_i), \\ z_i &= \tilde{\beta}_0 + \mathbf{x}_i\tilde{\boldsymbol{\beta}} + \frac{y_i - \tilde{p}_i}{\tilde{p}_i(1 - \tilde{p}_i)}, \end{aligned}$$

obtained from (2.27) and (2.30), respectively, where $\tilde{p}_i$ is evaluated using the current parameter estimates $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$. Subscript Q in (4.19) emphasizes that this is a quadratic function in the parameters $(\beta_0, \boldsymbol{\beta})$, contrary to (4.17). Then, the optimization problem (4.18) can be written

$$\hat{\boldsymbol{\beta}}(\nu) = \underset{\beta_0, \boldsymbol{\beta}}{\operatorname{argmin}}\{-\frac{1}{n}\log L_Q(\beta_0, \boldsymbol{\beta}) + \nu\|\boldsymbol{\beta}\|_1\}. \tag{4.20}$$

The full nested algorithm for estimating the parameter path of a lasso regularized logistic regression can thus be summarized (Hastie et al., 2015, p. 116)

Outer loop Decrement ν.

Middle loop Update the quadratic approximation $\log L_Q$ using the current parameters $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$.

Inner loop Run the coordinate descent algorithm on the penalized weighted-least-squares problem (4.20).

4.5 Cross-Validation

As we've seen, both in the algorithm above (outer loop) and in pathwise coordinate descent (Algorithm 1), we calculate parameter estimates $\hat{\boldsymbol{\beta}}$ for different values of the regularization parameter. A particular solution from the ensemble $\{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_{l=1}^{L}$ can be chosen by considering the prediction error for each value of the regularization parameter ν_l^*, using cross-validation (Hastie et al., 2015, p. 13).

In essence, K-fold cross-validation partitions the data into K roughly equal-sized parts (Hastie et al., 2009, p. 241). And for each $k = 1, ..., K$, the model is fitted on the remaining data by leaving out the kth partition. Then, this fitted model is used to calculate the prediction error on the kth partition. For the lasso use case, the main steps of the cross-validation scheme are sketched out in Algorithm 2, where usually $K = 10$. The function d used to calculate the error terms in Algorithm 2, is the mean-squared prediction error for linear regression (Hastie et al., 2015, p. 13) and the binomial deviance for logistic regression (Friedman et al., 2010, p. 18).

Algorithm 2: K-fold cross-validation for the lasso path.

Result: Estimate the expected prediction errors of the lasso solutions $\{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_l$.

initialize data into K partitions;

foreach $k \in \{1, ..., K\}$ **do**

  $test_k \leftarrow$ the kth partition;

  $training_k \leftarrow$ the remaining $k-1$ partitions;

  compute the lasso path $\{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_{l=1}^L$ using data in $training_k$;

  foreach $\nu^* \in \{\nu_l^*\}_{l=1}^L$ **do**

    $err_{k,l} \leftarrow d(\text{test}_k, \hat{\boldsymbol{\beta}}(\nu^*))$;

  end

end

return $\{\frac{1}{K}\sum_k err_{k,l}\}_{l=1}^L$;

Example 3 (Lasso estimator for linear regression)**.** We apply the lasso to the `abalone` data set (Nash et al., 1994), provided by the UCI Machine Learning Repository, which consists of $n = 4177$ instances. There are 8 predictors in this data set: sex, length, diameter, height, whole weight, shucked weight, viscera weight and shell weight, which are labelled $x_1, ..., x_8$. The response variable y is the number of rings of the abalone, directly related to its age. That is, we consider the observed variables to follow a normal distribution, and impose an ℓ_1-penalty on the parameters. Then, the optimization problem is as stated in equation (4.4).

Figure 4.3 (left panel) shows the corresponding coefficient path $\{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_{l=1}^L$. This plot shows that all coefficients are zero at $\nu^* \approx 1$, and gradually increase in size as ν^* decreases. When $\nu^* = 0$, the coefficients are the OLS estimates. This exemplifies the variable selection property of the lasso, as discussed above in association with Figure 4.1. Figure 4.3 (right panel) also shows the mean-squared prediction error curve following a 10-fold cross-validation. This plot can be used to select one specific set of coefficients on the path $\{\hat{\boldsymbol{\beta}}(\nu_l^*)\}_{l=1}^L$, that is, for a specific value of ν_l^*. Usually, this value is either chosen to be $\nu_{\min}^*$ or $\nu_{1\text{se}}^*$. The former is where the mean-squared prediction error has its minimum, and the latter is the largest value of ν_l^* that gives an error no larger than one standard deviation above the minimum (Hastie et al., 2015, p. 13). Since $\nu_{\min}^* < \nu_{1\text{se}}^*$, the coefficients $\hat{\boldsymbol{\beta}}(\nu_{1\text{se}}^*)$ are more sparse than $\hat{\boldsymbol{\beta}}(\nu_{\min}^*)$. This is shown in Table 4.1. This table shows that the linear model that best predicts the age of an abalone is achieved (at $\nu_{\min}^*$) by retaining all but one of the predictors in the data set. Yet, a more parsimonious model can be achieved (at $\nu_{1\text{se}}^*$) by excluding an additional predictor from this best predicting model, whilst keeping the prediction error within one standard deviation of the smallest error.

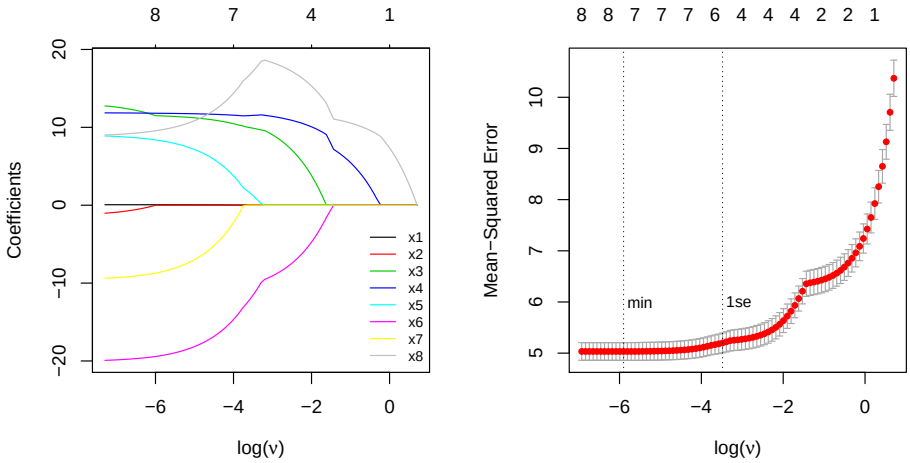

Figure 4.3: The lasso coefficient path (left) and the 10-fold cross-validated mean-squared prediction error (right) for the `abalone` data set (Nash et al., 1994), as functions of the regularization parameter ν, plotted on the log-scale. The topmost horizontal axes show the number of non-zero coefficients in the model.

j	$\hat{\beta}_j(\nu^*_{\text{min}})$	$\hat{\beta}_j(\nu^*_{\text{1se}})$
1	0.06	0.01
2	-	-
3	11.48	9.87
4	11.80	11.54
5	8.31	1.15
6	−19.35	−11.42
7	−8.62	-
8	9.64	17.30

Table 4.1: The estimated lasso coefficients for the `abalone` data set (Nash et al., 1994), at $\nu^*_{\text{min}} = 0.27 \times 10^{-2}$ and $\nu^*_{\text{1se}} = 3.07 \times 10^{-2}$. A "–" indicates that the coefficient was set to zero by the lasso.

4.6 Inference

Due to the adaptive nature of the lasso estimator, it is difficult to find the distribution of the estimated parameters $\hat{\boldsymbol{\beta}}$. This implies that it is not possible to construct test statistics and obtain p-values in the traditional sense, as done in Section 2.4 for regular GLMs, in order to test hypotheses about $\boldsymbol{\beta}$. In this section we describe the multi sample-splitting procedure, which is a generic way of constructing p-values. Multi sample-splitting is especially useful in a high-dimensional setting ($p \gg n$), where the fitting method has a variable selection property (Dezeure et al., 2015, p. 3).

4.6.1 Multi Sample-Splitting

Consider the lasso for a GLM (4.2) in a possibly high-dimensional setting, with $\mathbf{X}$ an $n \times p$ model matrix, $\mathbf{Y}$ an $n \times 1$ response vector and $\boldsymbol{\beta}$ a $p \times 1$ parameter vector. Let $I = \{1, ..., n\}$ denote the set of sample indexes. The idea is to split this set into two equally sized halves, I_1 and I_2, and do model selection on one of these halves and construct p-values using the other half.

More precisely, let $I_r \subset I$ with $r = \{1, 2\}$, such that $I_1 \cup I_2 = I$, $I_1 \cap I_2 = \varnothing$, $|I_1| = \lfloor n/2 \rfloor$ and $|I_2| = n - \lfloor n/2 \rfloor$. Denote by $\hat{S}(I_1)$ the set of indexes of non-zero parameters following a lasso fit using samples in I_1. That is,

$$\hat{S}(I_1) \subseteq \{1, ..., p\}.$$

If $|\hat{S}(I_1)| \leq n/2 \leq |I_2|$ (which is generally true for the lasso), then the model matrix $\mathbf{X}_{I_2}^{\hat{S}(I_1)}$ has full rank $|\hat{S}(I_1)|$, where the subscript denotes the sample set and the superscript denotes the parameter set in the model matrix. In that case, we fit the usual GLM (2.4) (with no penalization term) with response vector $\mathbf{Y}_{I_2}$ and model matrix $\mathbf{X}_{I_2}^{\hat{S}(I_1)}$, and use the asymptotic normality of the MLEs to obtain p-values (based on z-statistics (2.51)) for the parameters $\boldsymbol{\beta}_{\hat{S}(I_1)} = \{\beta_j; j \in \hat{S}(I_1)\}$. The raw p-values of $\boldsymbol{\beta}$ are then given by

$$p_{\text{raw},j} = \begin{cases} p_{\text{z-test},j} \text{ based on GLM fit with } \mathbf{Y}_{I_2}, \mathbf{X}_{I_2}^{\hat{S}(I_1)}, & \text{if} \quad j \in \hat{S}(I_1), \\ 1, & \text{otherwise.} \end{cases} \tag{4.21}$$

Consider adjusting these raw p-values for multiple testing. According to Dezeure et al. (2015), it is not necessary to control the family-wise error rate (FWER) over the complete set of considered null hypotheses $\mathcal{H} = \{H_1, ..., H_p\}$ (H_j corresponds to H_0 in (2.50)). The authors state that it is enough to control the FWER over the hypotheses set $\mathcal{H}_{\hat{S}(I_1)} = \{H_j; j \in \hat{S}(I_1)\}$. Furthermore, rather than adjusting the p-values using the Bonferroni-Holm method, the authors use the Bonferroni method for simplicity. Hence the adjusted p-values are given by (3.4)

$$p_{\text{corr},j} = \min(p_{\text{raw},j} \cdot |\hat{S}(I_1)|, 1). \tag{4.22}$$

The resulting p-values are highly dependent on the sample splits I_1 and I_2. That is, the p-values in (4.21), and hence in (4.22), are not likely to be reproducible.

To overcome this, the sample-splitting is performed B times, with typical choices $B = 50$ or $B = 100$. This gives a total of B p-values for each null hypothesis H_j, that is, we have

$$p_{\text{corr},j}^{(1)}, ..., p_{\text{corr},j}^{(B)} \quad \text{for} \quad j = 1, ..., p. \tag{4.23}$$

These p-values are dependent (for fixed j), since the sample splits in each iteration of B are over the same data. Hence, to aggregate these to a single p-value, we need an appropriate method. A simple solution is to aggregate by using a γ-quantile (Dezeure et al., 2015, p. 4)

$$Q_j(\gamma) = \min(\gamma\text{-quantile}\{p_{\text{corr},j}^{(b)}/\gamma; b = 1, ..., B\}, 1),$$

where $0 < \gamma < 1$. For $\gamma = 1/2$, the quantile evaluates to the sample median of $\{p_{\text{corr},j}^{(1)}, ..., p_{\text{corr},j}^{(B)}\}$ multiplied by a factor 2. Usually, a search is performed to find the most appropriate value of $\gamma \in (\gamma_{\min}, 1)$, where e.g. $\gamma_{\min} = 0.05$. Then, the final p-values are given by

$$p_j = \min((1 - \log(\gamma_{\min})) \inf_{\gamma \in (\gamma_{\min}, 1)} Q_j(\gamma), 1) \quad \text{for} \quad j = 1, ..., p, \tag{4.24}$$

where the term $(1 - \log(\gamma_{\min}))$ can be considered a penalty for searching for an appropriate $\gamma \in (\gamma_{\min}, 1)$.

The multi sample-splitting method just described is said to produce p-values (4.24) that are approximately reproducible. In addition, this method controls the FWER at level $\alpha + B\delta$ for some $0 < \delta < 1$, given that the following assumption is fulfilled.

4.6.2 Screening property

As discussed in Section 4.2, the lasso is able to do variable selection by setting some of the model parameters so zero. For the multi sample-splitting to control the FWER, however, a more precise assumption needs to be satisfied. Consider the (unknown) set of indexes of parameters that are truly non-zero

$$S_0 = \{j; \beta_j^0 \neq 0, j = 1, ..., p\}. \tag{4.25}$$

Then the screening property is (Dezeure et al., 2015, p. 2)

$$\hat{S} = \{j; \hat{\beta}_j \neq 0\} \supseteq S_0,$$

and the assumption that needs to be fulfilled to control the FWER is

$$P(\hat{S}(I_1) \supseteq S_0) \geq 1 - \delta, \tag{4.26}$$

for some $0 < \delta < 1$.

It is proven that the lasso for a linear regression model satisfies (4.26) with $\delta \to 0$, given some constraints on the design matrix $\mathbf{X}$ and the (unknown) size of the parameters $\boldsymbol{\beta}$ (not recited here). This latter constraint is susceptible to

criticism, since an assumption on the sizes of the parameters will have an effect on the resulting hypothesis test.

Based on the assumption (4.26), it can be proven that (Meinshausen et al., 2009)

$$\text{FWER} = P\Big(\bigcup_{j \in S_0^c} p_j \leq \alpha\Big) \leq \alpha + B\delta, \tag{4.27}$$

where S_0^c is the complement set of S_0 in (4.25), $0 < \alpha < 1$ is a chosen significance level, B the number of times the samples are split, and δ as in (4.26).

Chapter 5

Additive Models

Thus far, we've assumed that the effects of the predictors are linear, as modelled in (2.4). This might not be an adequate description. E.g. consider the model with two predictors Example 1, where only linear effects are included

$$g(\mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2,$$

where $\mu = \mathrm{E}(Y|x_1, x_2)$ as before. A more realistic model may be achieved by including higher order polynomial terms as

$$g(\mu) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2. \tag{5.1}$$

In general, we can write this as a so-called *additive model* (Hastie et al., 2009, p. 296)

$$g(\mu) = \beta_0 + f_1(x_1) + f_2(x_2),$$

where the f_j's can be any unspecified smooth functions.[1] There exists methods that estimate functions f_j in a flexible manner, using a so called scatterplot smoother (e.g. a cubic smoothing spline) (Hastie et al., 2009, p. 297). This approach lets the data dictate the shape of the f_j's. However, we limit ourselves to consider predefined functions f_j, each expressed in terms of an expansion of basis functions.

5.1 Basis Function Expansion

Let $X_1, ..., X_p$ denote the predictors, such that the additive model is

$$g(\mu) = \sum_{j=1}^{p} f_j(X_j), \tag{5.2}$$

where we represent each f_j by M_j basis functions (Hastie et al., 2009, p. 140)

$$f_j(X_j) = \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j),$$

[1] In (5.1), $f_1(x_1) = \beta_1 x_1 + \beta_3 x_1^2$, and $f_2(x_2) = \beta_2 x_2 + \beta_4 x_2^2$.

where $h_{jm} : \mathbb{R} \mapsto \mathbb{R}$. For $M_j = 1$ and $h_{jm}(X_j) = X_j$, we recover the original model that is linear in the predictors. Other notable examples of the basis function h_{jm} are

- $h_{jm}(X_j) = X_j^d$, polynomial terms. This leads to models of the form (5.1).
- $h_{jm}(X_j) = \log(X_j)$ or $h_{jm}(X_j) = \sqrt{X_j}$, some non-linear transformation.
- $h_{jm}(X_j) = I(L_m < X_j < U_m)$, an indicator for a region. Using non-overlapping regions, these models lead to piecewise contributions of the predictors.

In any case, if we predefine the number M_j of basis functions for each j, and also their parametric form h_{jm}, the f_j's are completely determined. That is, we can proceed to fit the resulting model

$$g(\mu) = \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j),$$

in the usual way (by running the IWLS algorithm for this GLM, as shown in Section 2.2) to obtain the estimates $\hat{\beta}_{jm}$. We will see an example of this model in Section 6.2.2.

Part II

Data Specific Regression

Chapter 6

Regression Model

In this chapter we define the lasso penalized logistic regression model used in our analyses, which is given at the end in Section 6.2.3. But first, we define the response variables and the model covariates in Sections 6.1 and 6.2, respectively.

6.1 Spike Trains as Response Variables

The activity of a neuron is recorded as a spike train, which is a sequence of event times at which a neuron fires, as explained in Section 1.2.2. To reiterate, a neurons can be in either of two states, a resting state or an active state. The point in time when a neuron transitions from a resting state to an active state is called a firing or a spike. When analyzing neural data, the rate at which a neuron fires, and the proportion of time it fires, is of interest. A fundamental concept in neurophysiology is that neurons respond to a stimulus or contribute to an action by increasing their firing rate (Kass et al., 2014, p. 563). Formally, the firing rate at time t is defined as

$$FR(t|x_t) = \lim_{\Delta t \to 0} \frac{\text{E(number of spikes in } (t, t+\Delta t)|x_t)}{\Delta t}, \tag{6.1}$$

where x_t can incorporate any experimental conditions, any effects of previous spikes (called history effects) or even activity of neighbouring neurons. The numerator in (6.1) is the (conditional) expected number of spikes in a time interval of length Δt.

Consider an observed spike train $s_1, s_2, ..., s_m$ over a time interval $(0, T]$, where the s_i's are the times at which a spike occurs. Such a spike train is modelled as a point processes $S_1, S_2, ...$ on $(0, \infty)$ (Kass et al., 2014, p. 564). To analyze a point process within the framework of (restricted or unresticted) GLMs, we need to discretize the spike times (Kass et al., 2014, p. 568). A way towards discretization is the counting process representation of the point process, denoted $N(t)$. The function $N(t)$ counts the total number of spikes that have occurred up until and including time t. Next, we divide the finite observed time interval $(0, T]$ into n bins of equal length $\Delta t = T/n$. The number of spikes in bin i can now be counted as $\Delta N_i = N(t_i) - N(t_{i-1})$, where $t_i = i \cdot \Delta t$. The set $\{\Delta N_i; i = 1, ..., n\}$ is

called the discrete increments, and it is displayed in Figure 6.1 along with the spike times and the counting process. Note that both the point process $S_1, S_2, ...$ and the counting process $N(t)$ are stochastic processes in continuous time, while the discrete increments $\{\Delta N_i; i = 1, ..., n\}$ are a sequence of random variables in discrete time.

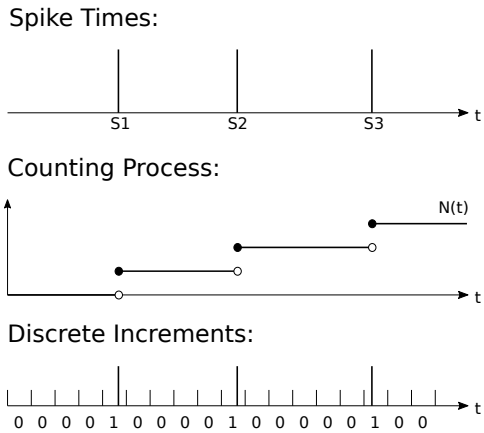

Figure 6.1: The spike times $S_1, S_2, ...$ are random variables, each representing the point in time when a spike occurs. $N(t)$ is the cumulative count of spikes that have occurred up until and including time t. The discrete increments ΔN_i count the number of spikes in bin i. This figure is copied from Kass et al. (2014, p. 567), with permission from Springer.

Let $Y_i = \Delta N_i$. If we choose a small enough bin size Δt, it is unlikely that there will be more than one spike occurrence in a single bin (Kass et al., 2014, p. 568) (this is the case illustrated in Figure 6.1). That is, $P(Y_i > 1) \approx 0$. Hence, we have that

$$Y_i \sim \text{Bernoulli}(p_i), \tag{6.2}$$

where $p_i = P(Y_i = 1)$, as described in (2.9). Since $\text{E}(Y_i) = p_i$, we can rewrite the firing rate defined in (6.1) as

$$FR(t|x_t) = \lim_{\Delta t \to 0} \frac{P(\text{spike in } (t, t + \Delta t)|x_t)}{\Delta t}. \tag{6.3}$$

That is, we consider the discretized (binary) sequence $\mathbf{y} = (y_1, ..., y_n)^T$ as the response variable in the logistic regression model (2.14) (as described in Section 2.1), and get that the linear predictor

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} \tag{6.4}$$

is such that $\boldsymbol{\eta} = \text{logit}(\mathbf{p})$, and $\mathbf{p} = (p_1, ..., p_n)^T$, where $p_i = P(Y_i = 1)$. Note that p_i can also be interpreted as the expected number of spikes in bin i (since $n_i = 1$ in (2.12)). In the next section we elaborate on the contents of the $n \times p$ model matrix $\mathbf{X}$.

6.2 Model Covariates

As expressed in (6.1), the firing rate of a neuron can depend on a stimulus, in addition to its previous spiking activity and the activity of its neighbouring neurons. Hence, inspired by Kass et al. (2014, p. 566), the linear predictor in our regression model can symbolically be written as

$$\boldsymbol{\eta} = \text{stimulus effects} + \text{history effects} + \text{connectivity effects}.$$

In the next two sections, we elaborate on how these three effects are modelled.

6.2.1 History and Connectivity Effects

In this section we outline an approach to model history effects and connectivity effects based on basis function expansions, to incorporate non-linear effects of these predictors, as described in Chapter 5.

Since there are connectivity effects involved, we need to consider observations from multiple neurons at once. To reflect this, we introduce the following notation: let $\mathbf{y}_j = (y_{1j}, ..., y_{nj})^T$ denote the n observations of the jth neuron. Furthermore, to reflect that we are modelling history effects, that is, that we are including observations from previous bins, we let $y_{ij} = y_j(t_i)$, where $t_i = i \cdot \Delta t$ is the midpoint in time of the ith bin.

Consider modelling the observations of neuron j by taking into accounts its own previous spiking history, in addition to the spiking history of neighbouring neurons. Thus, the regression model can be written

$$\eta_j(t_i) = \alpha_{0j} + \sum_{k=1}^{N} \sum_{m=1}^{M} \alpha_{jkm} y_k(t_i - t_m), \tag{6.5}$$

where $\eta_j(t_i) = \text{logit}\{p_j(t_i)\}$, N denotes the total number of neurons, and $M \geq 1$ denotes the number of bins included. That is, M represents the number of steps we go backwards in time, since $y_k(t_i - t_m)$ is the observation of neuron k from the previous mth bin (relative to bin i). The intercept α_{0j} represents the background firing probability (on the logit-scale) of neuron j. The remaining parameters (the α_{jkm}'s) represent the effect of the directed connection from neurons $k \neq j$ to neuron j. These parameters can be gathered in M-dimensinal vectors $\boldsymbol{\alpha}_{jk} = (\alpha_{jk1}, ..., \alpha_{jkM})$, which are then interpreted as the overall effect of the spiking activity of neuron k on the spiking activity of neuron j. Whenever $k = j$, $\boldsymbol{\alpha}_{jj}$ represents the history effects of neuron j. There are a total of N^2 such vectors $\boldsymbol{\alpha}_{jk}$ representing history and connectivity effects between all N observed neurons, but the model in (6.5) estimates N of these parameters at a time. That is, for a fixed neuron j, the model (6.5) estimates $\boldsymbol{\alpha}_{jk}$, for $k = 1, ..., N$.

Example 4. Consider estimating $\boldsymbol{\alpha}_{jk}$, using a time window of $\tau_M = 160\,\text{ms}$ ms to include the past activity of neuron k. That is, the observations of neuron j in each time bin are modelled using the observations of neuron k in the previous $M = \tau_M / \Delta t$ bins. Using $\Delta t = 10\,\text{ms}$, we get that $\boldsymbol{\alpha}_{jk}$ is a 16-dimensional vector.

As shown by Example 4, the connectivity effects $\boldsymbol{\alpha}_{jk}$ depend on the number of previous bins included M and the bin size Δt. That is, whether we wish to increase the time window (increase M), or model a finer time resolution (decrease Δt), the number of parameters in (6.5) increases. To separate the connectivity effects $\boldsymbol{\alpha}_{jk}$ from the time window and resolution, we can use a set of basis functions. Consider then the regression model

$$\eta_j(t_i) = \alpha_{0j} + \sum_{k=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{L} \alpha_{jkl} b_l(t_i) y_k(t_i - t_m),$$

where $b_l(t_i)$ denotes the lth basis function evaluated at time t_i. The set of basis functions are predefined, and equal for all neurons. This model can also be written as

$$\eta_j(t_i) = \alpha_{0j} + \sum_{k=1}^{N}\sum_{l=1}^{L} \alpha_{jkl} \mathbf{b}_{l,M}^T \mathbf{y}_{k,M}(t_i), \tag{6.6}$$

where $\mathbf{b}_{l,M} = (b_l(t_1), ..., b_l(t_M))^T$ and $\mathbf{y}_{k,M}(t_i) = (y_k(t_i - t_1), ..., y_k(t_i - t_M))^T$. For each neuron pair, the directed connection $\boldsymbol{\alpha}_{jk} = (\alpha_{jk1}, ..., \alpha_{jkL})$ is now an L-dimensional vector. That is, the size of $\boldsymbol{\alpha}_{jk}$ is only dependent on the number of basis functions.

In summary, the history and connectivity parameters in (6.6) can be written as an $N \times N \times L$ matrix (where N is the number of observed neurons)

$$\begin{pmatrix} \boldsymbol{\alpha}_{11} & \boldsymbol{\alpha}_{12} & \cdots & \boldsymbol{\alpha}_{1N} \\ \boldsymbol{\alpha}_{21} & \boldsymbol{\alpha}_{22} & \cdots & \boldsymbol{\alpha}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}_{N1} & \boldsymbol{\alpha}_{N2} & \cdots & \boldsymbol{\alpha}_{NN} \end{pmatrix}, \tag{6.7}$$

where

$$\boldsymbol{\alpha}_{jk} = (\alpha_{jk1}, ..., \alpha_{jkL}).$$

Note that the parameters $\boldsymbol{\alpha}_{jk}$ in this matrix are estimated one row at a time. That is, using observations of the jth neuron $\mathbf{y}_j$ as the response variable, and the remaining neurons as covariates, the regression model (6.6) estimates $\boldsymbol{\alpha}_{jk}$ for $k = 1, ..., N$.

Cosine Bases The basis functions used in (6.6) are so-called raised cosine "bumps", given as (Pillow et al., 2008, Methods)

$$b_l(t) = \begin{cases} \frac{1}{2}\cos(a\log(t+c) - \phi_l) + \frac{1}{2}, & \text{if} \quad a\log(t+c) \in [\phi_l - \pi, \phi_l + \pi], \\ 0, & \text{otherwise}, \end{cases} \tag{6.8}$$

where t represents the time after a spike event (so-called *lag*), the ϕ_l's are comparable to the placement of the peaks (or "bumps") of the cosines which are separated by $\pi/2$, and a and c are constants. According to Pillow et al. (2008), these constants (a and c) need to be chosen by evaluating the auto- and cross-correlation

functions of the activity of the neurons $\mathbf{y}_k$. This recommendation is not followed in our analysis. (See Appendix C.1 for our choice of a and b). Furthermore, following Pillow et al. (2008), we've used $L_{\text{hist}} = 10$ cosine basis functions in (6.6) to represent history effects, and $L_{\text{connect}} = 4$ to represent connectivity effects. The resulting basis functions (6.8) are shown in Figure 6.2 (top row), along with their orthogonal equivalent (bottom row). In our analyses, we use these orthogonal cosine bases. This figure shows that the cosine bases allow for a finer temporal representation at short lags, and a coarser representation at long lags. This means that we can model with more detail the effects near the time of a spike, which is a desirable feature, since more emphasis is placed on this time interval in the neuroscience literature, which can display effects such as refractory periods (Kass et al., 2014, p. 568) (as mentioned in Section 1.2.3).

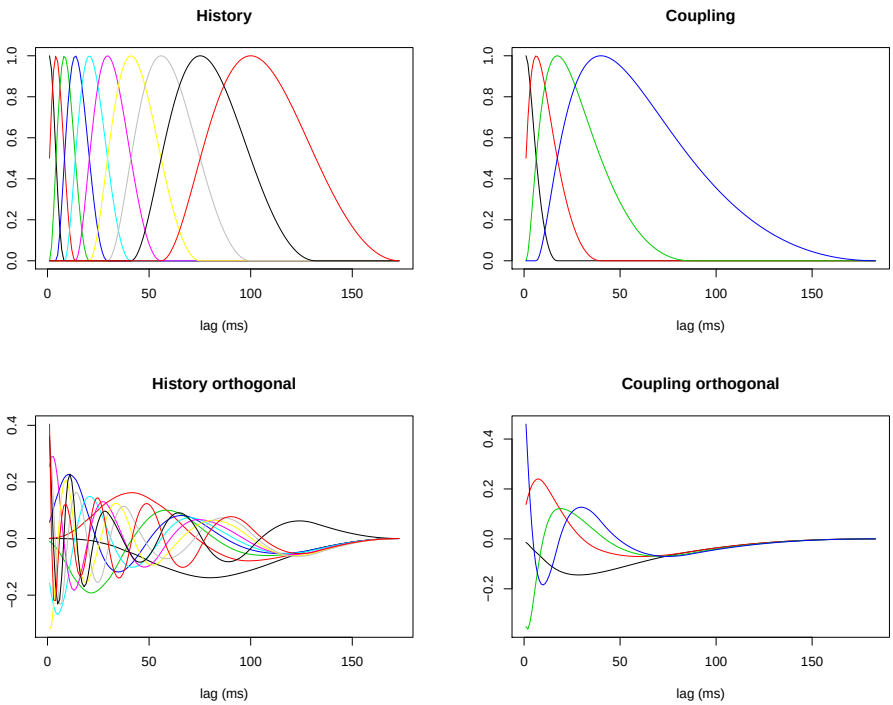

Figure 6.2: Cosine bases (6.8) from Pillow et al. (2008, Methods). There are $L_{\text{hist}} = 10$ bases used to represent history effects (left column), and $L_{\text{connect}} = 4$ to represent connectivity effects (right column). These bases span a time window of approximately $\tau_M = 160$ ms. The top row shows the bases as given in (6.8), while the bottom row are their orthogonal equivalent. That is, these latter bases are such that b_{l_1} and b_{l_2} are orthogonal for all l_1 and l_2. We've used these orthogonal bases in our analyses.

6.2.2 Stimulus Effects

As described in Section 1.3.3, a stimulus is provided in each trial. This stimulus varies as the trial progresses through its three stages: sample epoch, delay epoch and response epoch (as shown in Figure 1.7). A simple approach to incorporate the effects of these periods is to include the *trial time* as a covariate. We've observed that the trial time may have non-linear effects on the firing rate (6.3) of a neuron. To illustrates this, consider an empirical approximation to the firing rate of neuron j, which can be defined as

$$\widehat{FR}_j = \frac{\sum_{i=1}^{n} y_j(t_i)/n}{\Delta t}. \tag{6.9}$$

Figure 6.3 shows $\widehat{FR}_j$ calculated in the three bins corresponding to the sample, delay and response epochs (averaged over the number of trials), for neurons $j = 2, 6, 9$ from session `20130702` of mouse `ANM210861`. This plot shows that the trial time can have both a linear effect (neuron 2) and a non-linear effect (neurons 6 and 9) on the firing rate. Thus, we choose to represent the stimulus effects as a polynomial in trial time of degree D. That is,

$$\eta_j(t_i) = \alpha_{0j} + \sum_{d=1}^{D} \gamma_{dj} t_i^d, \tag{6.10}$$

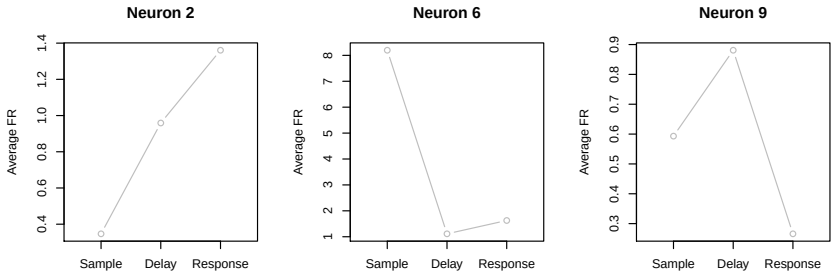

Figure 6.3: Empirically approximated firing rate $\widehat{FR}_j$, for neurons 2, 6 and 9 from session `20130702` of mouse `ANM210861`. $\widehat{FR}_j$ was calculated in three bins corresponding to each trial epoch. Each point is then the number of spikes in that trial epoch divided by the length of that trial epoch Δt, and averaged over the number of trials in `GoodTrials`.

However, in our analyses we represent this polynomial with a set of orthogonal bases $\{P_1(t_i), ..., P_D(t_i)\}$ called Legendre polynomials[1] (Süli and Mayers, 2003,

[1] Legendre polynomials can be obtained in `R` by the built-in function `poly`.

p. 263). Each $P_k(t_i)$ is a polynomial of degree k, and can be expressed using Rodriguez' formula (Lamb, 1995, p. 451)

$$P_k(t) = \frac{1}{2^k k!} \frac{d^k}{dt^k}(t^2 - 1)^k. \tag{6.11}$$

Inserting these polynomial bases into (6.10), the stimulus effect can be modelled as

$$\eta_j(t_i) = \alpha_{0j} + \sum_{d=1}^{D} \gamma_{dj} P_d(t_i). \tag{6.12}$$

Figure 6.4 shows an example (using $D = 5$) contrasting the usual polynomial bases in (6.10) and the Legendre polynomials in (6.12).

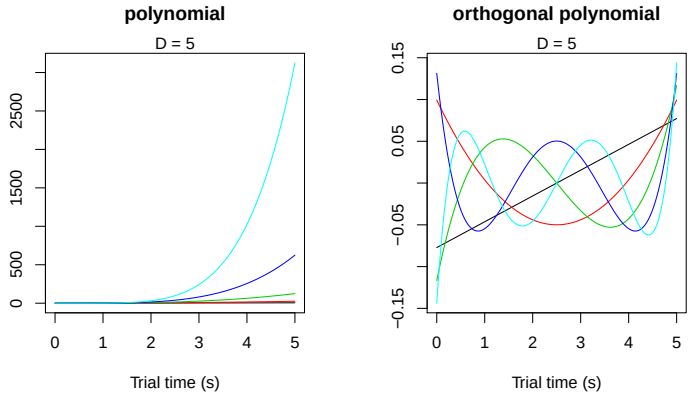

Figure 6.4: The left plot shows the usual polynomial bases $\{t, ..., t^5\}$ used in (6.10), and the right plot shows the orthogonal Legendre bases $\{P_1(t), ..., P_5(t)\}$ used in (6.12), where each $P_k(t)$ is given by (6.11). However, we've used the built-in function `poly` in `R` to obtain the set of Legendre polynomials.

6.2.3 Lasso Penalized Logistic Regression

Combining the history and connectivity effects (6.6), expressed in terms of cosine basis functions, and the stimulus effects (6.12), expressed as a polynomial, we get that

$$\begin{aligned} \eta_j(t_i) = \alpha_{0j} &+ \sum_{l=1}^{L_{\text{hist}}} \alpha_{jjl} \mathbf{b}_{l,M}^T \mathbf{y}_{j,M}(t_i) \\ &+ \sum_{k=1, k\neq j}^{N} \sum_{l=1}^{L_{\text{connect}}} \alpha_{jkl} \mathbf{b}_{l,M}^T \mathbf{y}_{k,M}(t_i) + \sum_{d=1}^{D} \gamma_{dj} P_d(t_i), \end{aligned} \tag{6.13}$$

which is an example of the additive model in (5.2). Recall that $t_i = i \cdot \Delta t$ is the midpoint of the ith bin, where Δt is the bin length used to discretize the time

interval $(0, T]$ in which we've observed the data, as mentioned in Section 6.1. In our analyses we've chosen $\Delta t = 1\,\text{ms}$, which ensures that the number of spikes in the ith bin does not exceed one, at least for the data considered in the analyses of Chapter 7. (Until Section 7.4, all regression models are fitted by discretizing $(0, T]$ using $\Delta t = 1\,\text{ms}$). That is, since $y_j(t_i) \leq 1$, we can consider $y_j(t_i)$ to be a realization of a random variable Y_i following a Bernoulli-distribution, as in (6.2). Furthermore, we have from (6.6) that $\mathbf{y}_{k,M}(t_i) = (y_k(t_i - t_1), ..., y_k(t_i - t_M))^T$ (and hence $\mathbf{b}_{l,M} = (b_l(t_1), ..., b_l(t_M))^T$). That is, the linear predictor (6.13) includes observations from the past M bins (relative to bin i), which is a fixed number. In our analyses we've used $M = 160$, which means that the linear predictor at time t_i includes data from the previous $\tau_M = M \cdot \Delta t = 160\,\text{ms}$.

In this model (6.13) we've introduced $1 + L_{\text{hist}} + (N-1) \cdot L_{\text{connect}} + D$ parameters, which might be an over-parameterization. That is, we don't expect neuron j to have a connection to each and every one of the remaining $k \neq j$ neurons in the model. As discussed in Section 1.2.3, we don't expect that many connections in the underlying network (5% estimate). For these reasons, we impose a restriction on the size of the parameters in the form of a lasso penalty, as introduced in Chapter 4. Let then the linear predictor in (6.13) be

$$\eta_j(t_i) = \alpha_{0j} + \mathbf{x}_{t_i}\boldsymbol{\beta}_j,$$

where $\boldsymbol{\beta}_j$ is a combination of the (rearranged) jth row of the history and connection matrix in (6.7), and the parameters γ_d of the stimulus polynomial. $\mathbf{x}_{t_i}$ incorporates observations of all neurons $\mathbf{y}_{k,M}$ in the past M bins (relative to the ith bin), weighted by the (orthogonal) cosine bases (6.8), in addition to D polynomial terms of the trial time. Then, the optimization problem, as characterized in (4.18), can be written

$$\begin{aligned}(\hat{\alpha}_{0j}, \hat{\boldsymbol{\beta}}_j)(\nu) = \underset{\alpha_{0j},\, \boldsymbol{\beta}_j}{\text{argmin}}\{-\frac{1}{n}\sum_{i=1}^{n}\Big[y_j(t_i)(\alpha_{0j} + \mathbf{x}_{t_i}\boldsymbol{\beta}_j) \\ - \log\big(1 + \exp\big(\alpha_{0j} + \mathbf{x}_{t_i}\boldsymbol{\beta}_j\big)\big)\Big] \\ + \nu\|\boldsymbol{\beta}_j\|_1\},\end{aligned} \tag{6.14}$$

where $n = T/\Delta t$.

Chapter 7

Data Analysis

In this chapter we present the main results of our analyses. As mentioned introductory in Section 1.1, our main goal is to estimate the underlying network of neurons. The analyses leading to these networks is done in Section 7.4. In Sections 7.2 and 7.3 we analyse the stimulus effects and the history and connectivity effects, respectively. However, we begin with Section 7.1 with some exploratory data analyses.

7.1 Exploratory Data Analysis

7.1.1 Session-Wise Activity

We've arbitrarily chosen to analyze data from mouse `ANM210861`, which has three sessions tagged `20130701`, `20130702` and `20130703`, which from here on are termed sessions 1, 2 and 3. These sessions have recorded activity of 30, 16, and 12 neurons, respectively. Figure 7.1 shows raster plots[1] of the activity of these neurons for all three sessions. That is, each row in these plots is a spike train $s_1, s_2, ..., s_m$, as defined in Section 6.1.

The raster plots in Figure 7.1 show that some neurons are active throughout the whole session, seen as rows that are mostly black. Examples are neuron 12 in session 1, neuron 5 in session 3 and the majority of neurons in session 2. As described in Section 1.3.3, the stimulus is provided periodically (during the sample epoch in each trial, where the trials occur repeatedly throughout the session). That is, it might be possible that neurons that display an almost continuous activity throughout the session are not directly related to the stimulus. However, it is difficult to assess the relation between neuronal activity and stimulus by inspecting these raster plots on a session-wise time scale. In fact, we should expect the majority of the neurons to be related to the stimulus, as most neurons are classified as pyramidal neurons,

[1] A raster plot shows a tick at the time a spike was present in the recorded voltage trace. Raster plots are commonly used as exploratory tools in the analysis of neural data. See Kass et al. (2014, p. 4) for examples.

which are neurons that project activity from the ALM towards the motor-related areas in the brainstem, as mentioned in Section 1.3.1. The classification of each neuron is summarized in Table 7.1, which is done as part of the pre-processing of the raw data (Li et al., 2015, Methods).

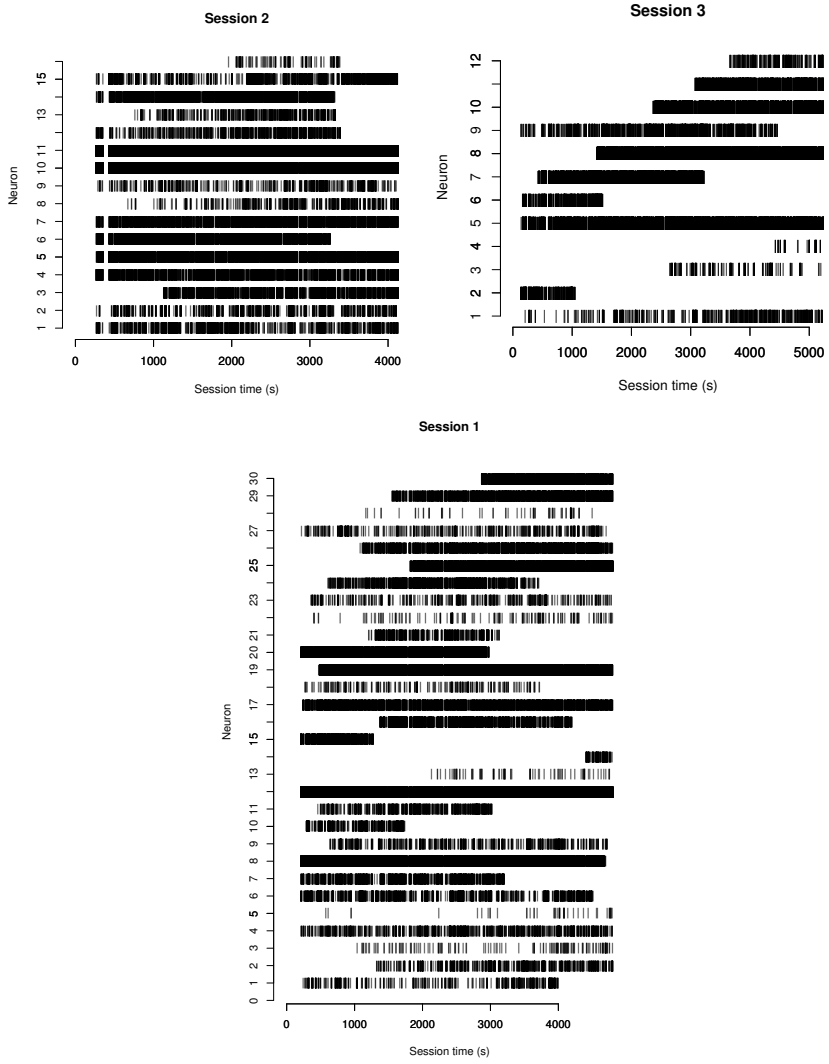

Figure 7.1: Raster plots of the activity of neurons, based on data from `GoodTrials`. For each neuron, a vertical bar indicates the point in (session) time when a spike occurred.

Neuron Type	Session 1	Session 2	Session 3
Pyramidal	2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 23, 25, 26, 28, 30	1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 14, 16, 15	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12
Fast Spiking (FS)	1, 16, 24, 27	11	5
Not Classified	8, 15, 29	7	-

Table 7.1: Classification of the neurons. Pyramidal neurons project activity out from the ALM-circuit towards motor-related areas in the brainstem. Fast spiking (FS) neurons project activity to other neurons inside the ALM-circuit (also called intratelencephalic neurons in Section 1.3.1). These classifications are done by inspecting so-called waveforms, as parts of the pre-prosessing of the data, explained in Li et al. (2015, Methods).

Additionally, the activity of the neurons in sessions 1, 2, and 3 can be summarized by calculating their empirical firing rates $\widehat{FR}_j$, as in (6.9) (using bin size $\Delta t = 1\,\text{ms}$), averaged over all trials. The distribution of the firing rates in each session is shown as boxplots in Figure 7.2. The outliers in these boxplots are exactly those neurons that show an almost continuous activity in the raster plots of Figure 7.1. Moreover, Figure 7.2 shows that the neurons were most active during session 2. The median firing rate over all neurons in session 1, 2 and 3 is 0.63 Hz, 1.92 Hz and 0.61 Hz, respectively.

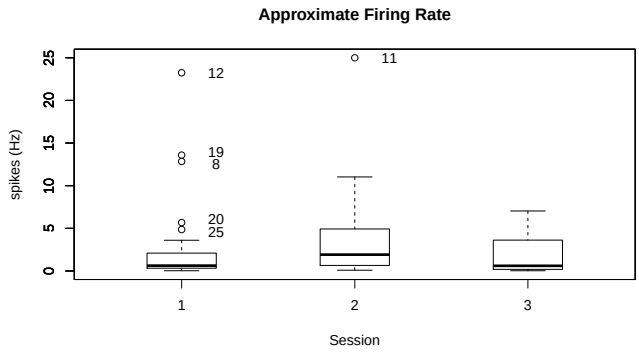

Figure 7.2: Boxplots of the empirically approximated firing rate $\widehat{FR}_j$ (6.9), using bin size $\Delta t = 1\,\text{ms}$, averaged over all trials in `GoodTrials`.

Note that some neurons in Figure 7.1 are only active during some parts of the session. This is clearly visible for most of the neurons in session 3, where for example neuron 2 is only seen to be active during the early part of the session, while neuron 11 is not active until late in the session. Furthermore, these particular two neurons in session 3 are recorded on the same electrode/channel on the silicon probe, as

seen in Figure 1.8. Therefore, it might be possible that these neurons in reality represent the same physical neuron. Another such example is neurons 14 and 16 in session 1. Still, there are plenty other neurons that are active only in some parts of the session, which might be a correct representation of their true activity. However, it might also be the case that the electrodes/channels that recorded these neurons were not functioning properly during the whole session. Yet, in our analyses, we have not adjusted for these effects. That is, we've consider the spike trains in Figure 7.1 to be the actual activity of the identified neurons.

We note that there are times during the session where the majority of the neurons, if not all, are inactive. This is most clearly visible just before the 500 s mark in session 2 (thick white vertical line cutting through all rows in Figure 7.1). These periods of inactivity might be related to so-called *photostimulation* trials, which are trials where the experimenters themselves inactive the ALM. Using photostimulation, the experimenters are also able to target and inactive specific types of neurons, which can explain why some neurons are seen to be active only in parts of the session. For details, see Li et al. (2015, Methods). In our analyses, we have not taken such effects of photostimulation into account.

For completeness, similar raster plots as in Figure 7.1 are plotted using only either correct left lick trials and correct right lick trials, which are shown in Appendix B.1. These plots don't give any additional insight into the activity of the neurons.

7.1.2 Trial-Wise Activity

Next, we explore the firing rate in each trial epoch (similarly as for neurons 2, 6 and 9 from session 2 in Figure 6.3). That is, we calculate the empirical firing rate $\widehat{FR}_j$, as in (6.9), once for each trial epoch and average it over all trials. This is done for the 16 neurons in session 2, as shown in Figure 7.3. These plots show that some neurons are most active during the sample epoch, such as neurons 5 and 6. These neurons might be directly related to the sensing of the metal pole, which is provided as the stimulus in the sample epoch (as explained in Section 1.3.3). There are also neurons that are most active during the response epoch (when the licks occur), such as neurons 4 and 8. This kind of activity, that peaks during the response epoch, is called *peri-movement* activity (Li et al., 2015, p. 52).

In addition to showing how the firing rate changes in different trial epochs, the plots in Figure 7.3 also show that there is a difference in activity in correct lick left (red plots) and correct lick right (blue plots) trials. For example, Figure 7.3 shows that neurons 1 and 2 are generally most active during lick left trials (the red points are considerably higher than the blue points), while neuron 13 is generally most active during lick right trials. Such neurons are said to display a so-called *selective* activity. Generally, selective activity begins during the sample epoch, and reaches a maximum in the delay epoch. Additionally, these particular three neurons (1, 2 and 13) have different firing rates in correct left/right trials during the delay epoch (that is, right before a lick occurs). This type of activity is called *selective preparatory* activity. (Recall from Section 1.3.1 that we expect neurons in the ALM to display preparatory activity). Furthermore, Figure 7.3 shows that

there are neurons that aren't selective, that is, neurons that display rather similar firing rates regardless of trial type. Neurons 7 and 11 are such examples. Similar plots as in Figure 7.3 for neurons in session 1 and session 3 are shown in Appendix B.2. These plots also show examples of selectivity. Thus, from Figures 7.3, B.4 and B.5, we note that the provided stimulus does affect the firing rate, and it does so differently in right/left trials. In addition, it seems that individual neurons are *tuned*, so to speak, to respond to the stimulus in different trial epochs.

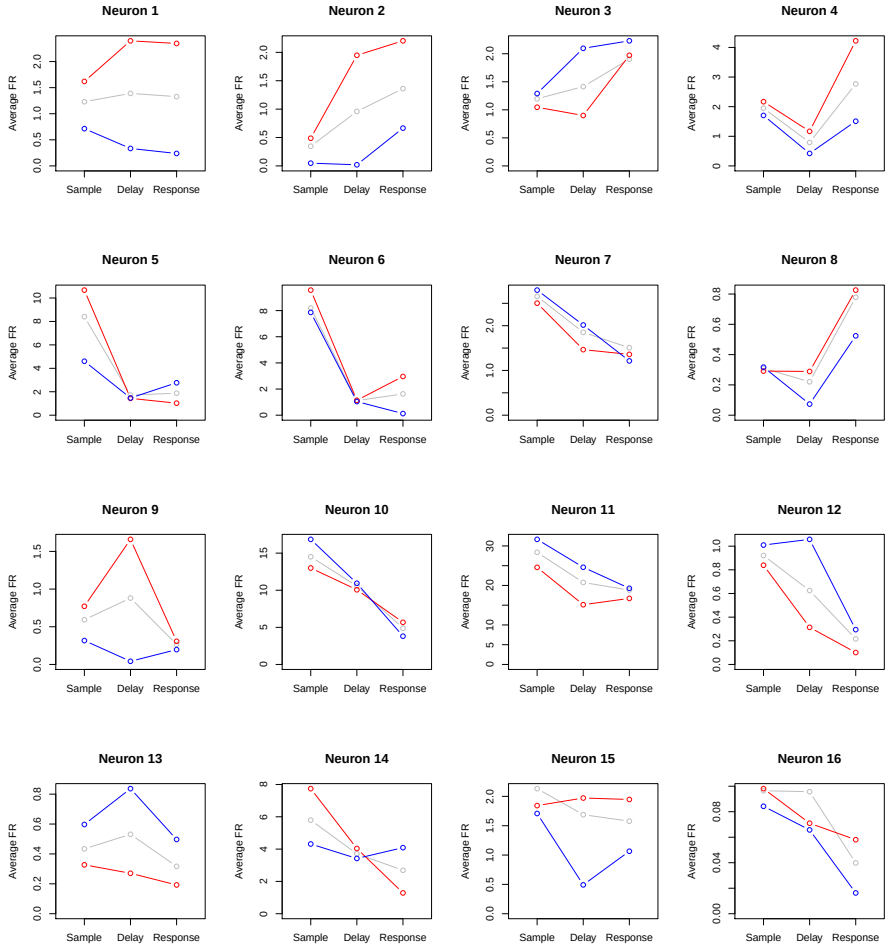

Figure 7.3: The empirically approximated firing rate $\widehat{FR}_j$ (6.9) calculated separately for each trial epoch with Δt equal to the length of the relevant trial epoch, averaged over all trials. The gray, red and blue plots are based on data from `GoodTrials`, correct left lick trials, and correct lick right trials, respectively. These 16 neurons are from session 2.

7.2 Tuning Curves

Influenced by the work of Stevenson et al. (2012), we first fit a regression model with only stimulus effects in Section 7.2.1, and assess the effect of the estimated parameters on the firing probability. Then, in Section 7.2.2, we fit the so-called full regression model, by also including history and connectivity effects, and compare it to the model with only stimulus effects.

7.2.1 Regression Model with only Stimulus Effects

In this section we fit the logistic penalized regression model from Section 6.2.3 for each neuron $j = 1, ..., N$, but with only stimulus effects as covariates. Let $\mathbf{x}_{t_i} = (P_1(t_i), ..., P_D(t_i))$, where P_d is given in (6.11), and the set $\{P_1(t), ..., P_D(t)\}$ is shown in Figure 6.4 (for $D = 5$). Inserting this $\mathbf{x}_{t_i}$, with $\boldsymbol{\beta}_j = (\gamma_{1j}, ..., \gamma_{Dj})^T$, into (6.14), we get that

$$(\hat{\alpha}_{0j}, \hat{\gamma}_{1j}, ..., \hat{\gamma}_{Dj})(\nu) = \underset{\alpha_{0j},\, \gamma_{1j}, ..., \gamma_{Dj}}{\operatorname{argmin}} \{-\frac{1}{n} \sum_{i=1}^{n} \Big[y_j(t_i)(\alpha_{0j} + \sum_{d=1}^{D} \gamma_{dj} P_d(t_i)) - \log\left(1 + \exp\left(\alpha_{0j} + \sum_{d=1}^{D} \gamma_{dj} P_d(t_i)\right)\right)\Big] + \nu \|(\gamma_{1j}, ..., \gamma_{Dj})\|_1\}. \tag{7.1}$$

As an example, we fit this model using observations from neuron $j = 1$, $\mathbf{y}_1$, from session 2, as the response variable, and setting $D = 5$. Figure 7.4 shows the resulting coefficient path (left panel), along with the 10-fold cross-validated binomial deviance curve (right panel), as functions of the regularization coefficient ν. (An interpretation of these plots is given in Example 3 in Section 4.5). As noted in Section 2.3.3, the model with the lowest deviance can be considered to fit the data best (when comparing models using the same likelihood). As shown by the cross-validation curve in Figure 7.4, this model is located at $\nu_{\min} = 1.97 \times 10^{-5}$ (shown by the leftmost vertical dashed line), and retains all five parameters $(\gamma_{1,1}, ..., \gamma_{5,1})$ in the model. The parameter estimates of this model are shown in Table 7.2 (first column under "Logistic Lasso"). Note that the regularization parameter of this best fit model (where the binomial deviance has its minimum) is close to zero ($\nu_{\min} \approx 0$). That is, this penalized model is close to the usual (unrestricted) logistic regression model. Hence, in this particular example, we fit a logistic GLM to neuron $j = 1$ (from session 2) with stimulus effects as the only covariates (which is equivalent to setting $\nu = 0$ in (7.1), as discussed in Section 4.1 regarding the optimization problem for a GLM given in (4.1)). Table 7.2 also shows the parameter estimates of this unrestricted regression model (first column under "Logistic GLM"), which are seen to be similar to the restricted model. This table shows that when there are only a few number of parameters in the regression model (here we have five), then the parameter estimates from the lasso (at $\nu_{\min}$) are close to the parameter estimates from the equivalent unrestricted regression model. The $\hat{\gamma}_{d,1}$'s

in Table7.2 can be interpreted as weights used to evaluate a 5th degree polynomial that represents the effect of trial time on the firing rate of neuron 1.

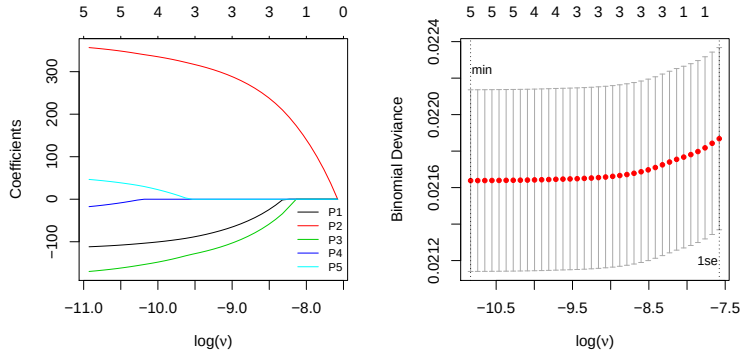

Figure 7.4: The lasso coefficient path (left) and the 10-fold cross-validated binomial deviance curve (right) for the stimulus model in (7.1) fitted using observations of neuron 1 $\mathbf{y}_1$ (from session 2) from `GoodTrials`, as functions of the regularization parameter ν, plotted on the log-scale. The topmost horizontal axes shows the number of non-zero coefficients in the model.

	Logistic Lasso at $\nu_{\min}$		Logistic GLM	
	Estimate	FWER adjusted p-value	Estimate	FWER adjusted p-value
$\hat{\alpha}_{0,1}$	-6.60	NA	-6.61	0.00
$\hat{\gamma}_{1,1}$	-111.13	2.14×10^{-3}	-119.06	5.47×10^{-6}
$\hat{\gamma}_{2,1}$	355.15	1.38×10^{-26}	372.13	1.10×10^{-44}
$\hat{\gamma}_{3,1}$	-168.43	2.58×10^{-6}	-185.28	5.45×10^{-12}
$\hat{\gamma}_{4,1}$	-15.82	1.00	-34.49	8.66×10^{-1}
$\hat{\gamma}_{5,1}$	45.11	1.00	61.56	7.16×10^{-2}

Table 7.2: The first two columns are the estimated parameters and adjusted p-values of the lasso penalized logistic model (7.1) at $\nu_{\min} = 1.97 \times 10^{-5}$. The adjusted p-value for the intercept $\alpha_{0,1}$ is not available, since the `multi.split`-function in `R` calculates adjusted p-values for parameters that are constraint by the ℓ_1-norm. The last two columns are the estimates and adjusted p-values from an equivalent logistic GLM. These models where fit using observations of neuron 1 $\mathbf{y}_1$ (from session 2) from `GoodTrials`. The gray boxes highlight adjusted p-values that are below the cut-off level 0.05.

As explained in Section 4.6.1, adjusted p-values (adjusted for multiple testing, as given in (4.24)) can be obtained for the lasso by multi sample-splitting. This is done for the model in (7.1) using the `R`-function `multi.split` from the `hdi`-

package with $B = 50$ sample-splits (see (4.23)). These adjusted p-values are given in Table 7.2 (second column under "Logistic Lasso"). The p-values for the GLM are obtained in the usual manner, however, those given in Table 7.2 (second column under "Logistic GLM") are adjusted using Bonferroni's method (3.4) (with $m = 5$). Say we choose a significance level 5% for testing whether each regression parameter $\gamma_{d,1}$ is different from zero. (Since the p-values are adjusted, this controls the FWER at 5%, as stated in (4.27) in Section 4.6.2). Then, we see from the adjusted p-values in Table 7.2 that both the lasso regularized logistic regression model and the usual logistic regression model conclude that $\gamma_{1,1}$, $\gamma_{2,1}$ and $\gamma_{3,1}$ are significantly non-zero (shown by the gray boxes, which highlight adjusted p-values below the cut-off level 0.05).

Next, we plot the estimated stimulus effect on the firing rate of neuron 1, using the parameters from the lasso (7.1) (first column under "Logistic Lasso" in Table 7.2). That is, we plot the linear predictor

$$\hat{\eta}_{\text{stim},1}(t_i) = \hat{\alpha}_{0,1} + \sum_{d=1}^{5} \hat{\gamma}_{d,1} P_d(t_i),$$

on the probability scale

$$\hat{p}_1(t_i) = \text{logit}^{-1}\{\hat{\eta}_{\text{stim},1}(t_i)\}, \tag{7.2}$$

where the inverse of the logit-function is given by (2.11), and $\hat{p}_1(t_i)$ can be interpreted as the estimated probability of neuron 1 to fire in the ith bin. Since we've chosen $\Delta t = 1\,\text{ms}$ small enough to ensure that $y_j(t_i) \leq 1$ (as mentioned in Sections 6.1 and 6.2.3), $\hat{p}_1(t_i)$ is proportional to the firing rate of neuron 1, as given by the relation in (6.3). Hence, in the following, we interpret the effect any covariate might have on the activity of a neuron either on the probability scale, or equivalently, in terms of the firing rate.

Figure 7.5 shows the estimated stimulus effect on the firing rate of neuron $j = 1$ (7.2). (Note that we use all five estimated parameters given in Table 7.2 in the first column under "Logistic Lasso", including those that aren't significantly different than zero according to the 5% significance test above). Such plots, where the firing rate is plotted on the stimulus domain, are called *tuning curves* in the neuroscience literature (Stevenson et al., 2012), a terminology that implies that a neuron is *tuned* to a certain type of stimulus. Figure 7.5 shows that the firing rate is relatively high in the beginning of the trial, but drops considerably during the sample epoch, then rises slightly during the delay and the response epochs. These highs and lows are relative to a baseline firing rate, plotted as $\text{logit}^{-1}\{\hat{\alpha}_{0,1}\}$ in Figure 7.5 (horizontal gray line). This tuning curve should be compared to the empirically approximated firing rate $\widehat{FR}_j$ of neuron $j = 1$ in Figure 7.3 (gray plot, which uses data from `GoodTrials`). At first glance, the tuning curve does not seem to match the trend in the approximated firing rate. However, note that the tuning curve is estimated using the whole 5 s of the trial (recall the trial structure from Figure 1.7), while the firing rate is approximated in the sample epoch and onwards. That is, the activity of the neuron in the interval from the start of the trial to the start of the sample

epoch (roughly 0 to 0.6 s in trial time) is ignored in the plots of Figure 7.3, but this activity is taken into account when fitting the tuning curve. Considering this, note that the tuning curve estimates that the activity in the sample epoch is lower than the activity in the delay and the response epochs, which is somewhat reflected in the corresponding plot of the approximated firing rate $\widehat{FR}_1$ in these three epochs.

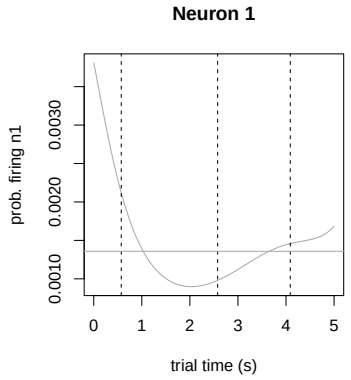

Figure 7.5: Tuning curve of neuron 1 from session 2, estimated using data $\mathbf{y}_1$ from `GoodTrials`. The curve is the evaluated polynomial (7.2), using the fitted values from the lasso (7.1) shown in Table 7.2 (first column under "Logistic Lasso"). The horizontal line is the inverse logit-function of the intercept $\hat{\alpha}_{0,1}$. The vertical dashed lines are the start of the sample, delay and response epoch, respectively.

As noted, the main point of a tuning curve is to display how the activity of a neuron relates to a stimulus. For this reason, we've estimated two additional tuning curves for each neuron j by fitting the lasso (7.1) using observations $\mathbf{y}_j$ from correct lick left trials and from correct lick right trials, respectively. These tuning curves (along with the tuning curve using data from `GoodTrials`) are shown in Figure 7.6 for the 16 neurons from session 2.[2] In cases where a neuron is relatively less active in the beginning of a trial (before the start of the sample epoch), the tuning curves of Figure 7.6 correspond to the approximated firing rates in Figure 7.3 (neuron 2 is an example). Most importantly, these tuning curves show that there are neurons that respond differently to left/right stimulus (called selective activity), while other neurons have the same activity pattern regardless of trial type (e.g. neurons 4 and 7). An example of a selective neuron is neuron 9, which has an increased activity in correct left trials, compared to correct right trials. Additionally, this increased activity is manifested in the delay epoch, that is, neuron 9 displays a selective preparatory activity. Other patterns (such as peri-movement activity) can be deduced, as done in Section 7.1.2.

[2]For simplicity, the baseline firing rates, in terms of the estimated intercept parameters $\hat{\alpha}_{0j}$ are not shown in Figure 7.6, which would have corresponded to three additional horizontal lines (one for each of the trial types: `GoodTrials`, correct lick left and correct lick right trials).

Similarly, tuning curves for the 30 neurons in session 1, and the 12 neurons in session 3, are shown in Appendix B.3.

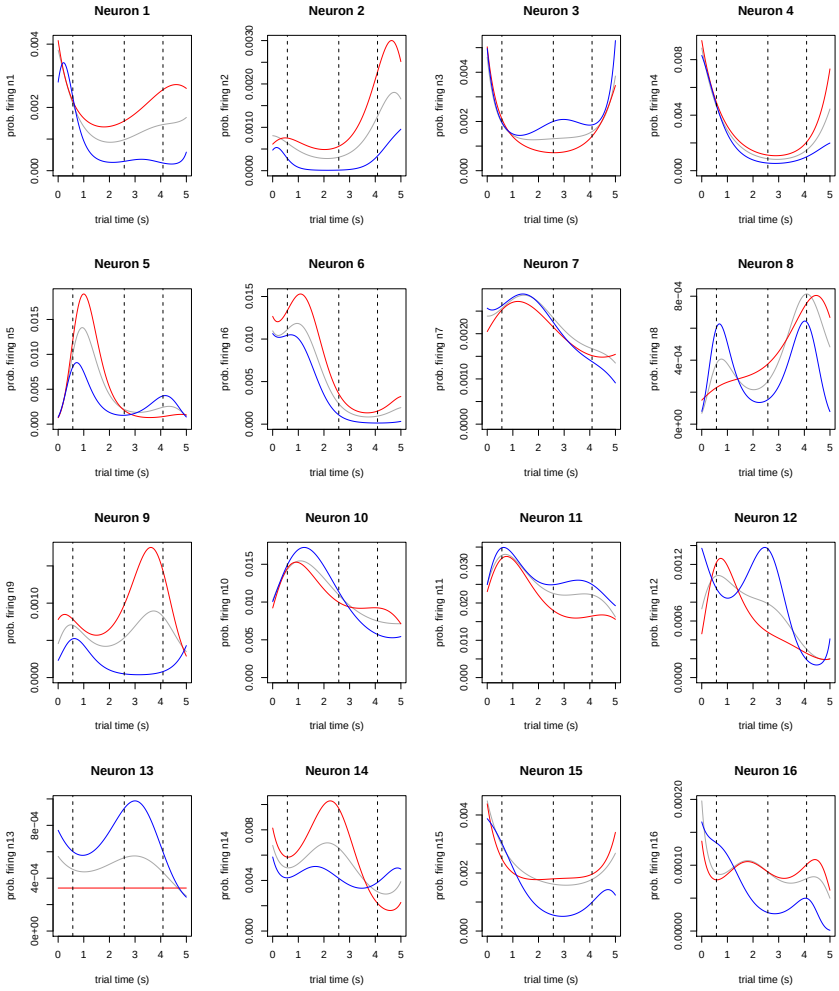

Figure 7.6: Tuning curves for the 16 neurons from session 2. The parameters are estimated using the lasso with only stimulus effects (7.1), and choosing the values at $\nu_{\min}$. The gray curves are estimated using data from `GoodTrials`, and the red and blue curves from correct lick left and right trials, respectively. The dashed vertical lines are the start of the sample, delay and response epochs, respectively.

7.2.2 Full Regression Model

We now fit the so-called full regression model (6.14) from Section 6.2.3. That is, in addition to stimulus effects, as represented by a degree $D = 5$ (orthogonal) polyno-

mial, we include history and connectivity effects, using $L_{\text{hist}} = 10$ and $L_{\text{connect}} = 4$ (orthogonal) cosine bases, respectively.

As an example, we fit this model to observations from neuron $j = 1$ from session 2 using data from `GoodTrials`, that is, we set $\mathbf{y}_1$ as the response in (6.14), and use the observations from the remaining $k \neq j$ 15 neurons (there are 16 neurons in session 2) to evaluate the cosine bases as covariates. Then, the total number of parameters in this model is $1 + 10 + 15 \cdot 4 + 5 = 76$. Figure 7.7 shows the coefficient path (left panel) and the 10-fold cross-validation curve (right panel) for this model. The model with the lowest binomial deviance (leftmost vertical dashed line in the cross-validation curve), at $\nu_{\min} = 2.13 \times 10^{-5}$, results in 55 non-zero parameter estimates. The estimated coefficients representing the stimulus effects are given in Table 7.3 (second column under `GoodTrials`), which are the focus of this section. (Tables 7.4 and 7.5 show the estimated coefficients representing the history and connectivity effects, respectively, which will be discussed in the next section). Note that the coefficients representing the stimulus effects have the largest absolute value among all estimated coefficients in the full model, as shown by the coefficient path in Figure 7.7. However, comparing the sizes of the estimated stimulus effects and estimated history and connectivity effects may be meaningless, since the bases used to represent these effects are not on the same scale. (This is seen by comparing the y-axis of the orthogonal cosine bases in Figure 6.2, which ranges from -0.4 to 0.4, to the y-axis of the orthogonal Legendre polynomial bases in Figure 6.4, which ranges from -0.15 to 0.15).

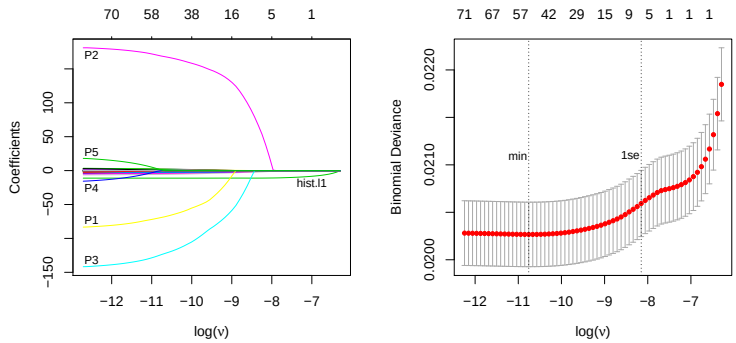

Figure 7.7: The lasso coefficient path (left) and the 10-fold cross-validated binomial deviance (right) for the regression model in (6.14) fitted to observations from neuron $j = 1$, $\mathbf{y}_1$, from session 2, using data from `GoodTrials`. The topmost horizontal axes shows the number of non-zero coefficients in the model. The labels in the coefficient path represent the estimated coefficients $\hat{\gamma}_{1,1}, ..., \hat{\gamma}_{5,1}$ of the five (orthogonal) polynomial bases $P_1, ..., P_5$ (shown in Figure 6.4) and the estimated coefficient $\hat{\alpha}_{1,1,1}$ for the first (orthogonal) history basis b_1 (shown in Figure 6.2).

Table 7.3 shows a comparison of the stimulus coefficients between the full re-

gression model (6.14) and the regression model with only stimulus effects (7.1), known as net and gross effects, respectively. The first column of this table (under `GoodTrials`) is identical to the first column of Table 7.2 (under "Logistic Lasso"), which is added for completeness. Table 7.3 shows that the regularization parameter $\nu_{\min}$ (for the minimum binomial deviance model) is larger for the full model compared to the stimulus only model, which leads to a *shrinkage* in the parameter estimates of the full model. That is, the estimated (stimulus) coefficients from the full model are generally smaller (in absolute value) than the coefficients from the stimulus only model. Furthermore, this table shows that the deviance (2.34) is smaller for the full regression model compared to the stimulus only model, which indicates that the full regression is a better fit to the data (regardless of which trial type the data is from), as discussed in Section 2.3.3.

	`GoodTrials`		Left Tr.		Right Tr.	
	Stim. only	Full	Stim. only	Full	Stim. only	Full
$\nu_{\min}(10^{-5})$	1.97	2.13	3.23	4.18	1.34	3.50
Dev. (10^3)	30.82	28.77	19.76	18.51	6.67	6.39
$\hat{\alpha}_{0,1}$	−6.60	−6.86	−6.22	−6.48	−7.66	−7.61
$\hat{\gamma}_{1,1}$	−111.13	−69.09	61.20	1.59	−501.22	−406.02
$\hat{\gamma}_{2,1}$	355.15	168.92	186.36	72.55	342.22	235.79
$\hat{\gamma}_{3,1}$	−168.43	−123.85	−111.42	−49.92	−126.33	−73.68
$\hat{\gamma}_{4,1}$	−15.82	-	−2.85	−10.73	-	-
$\hat{\gamma}_{5,1}$	45.11	0.70	-	-	184.35	61.58

Table 7.3: Estimated coefficients of the stimulus effect from the full regression model (6.14) (net effect) and the regression model with only stimulus effects (7.1) (gross effect). These two models are fitted for neuron 1 in session 2, once for data from each of the three trial types: `GoodTrials`, correct left trials, and correct right trials. The deviance is as defined in (2.34).

Next, we use the estimated coefficients in Table 7.3 to plot tuning curves for the three types of trials (`GoodTrials`, correct left lick trials, and correct right lick trials). That is, we plot (7.2) as in the previous section. These tuning curves are shown in Figure 7.8. (Note that the left plot in this figure is the same as the leftmost plot in the first row in Figure 7.6). Figure 7.8 shows that the tuning curves from the full model (6.14) retain the same shape as the tuning curves from the stimulus only model (7.1). This indicates that the design matrix in the full regression model may be nearly block-wise orthogonal with respect to stimulus effects and history and connectivity effects. Furthermore, as indicated by the shrinkage in Table 7.3, the tuning curves from the full model have smaller numerical values, which can be seen by comparing the range of the y-axes in these two plots. This means that once we've accounted for history and connectivity effects (in the full model), the stimulus affects the firing rate of neuron 1 to a smaller degree. (That is, the net effect of the stimulus is smaller than the gross effect of the stimulus). The tuning

curves from the full regression model for all 16 neurons from session 2 are shown in Figure 7.9. Comparing the plots in this figure, to the plots in Figure 7.6, we see the same trend; that the shape of the tuning curves are retained, while the net effect of the stimulus is smaller than its gross effect.

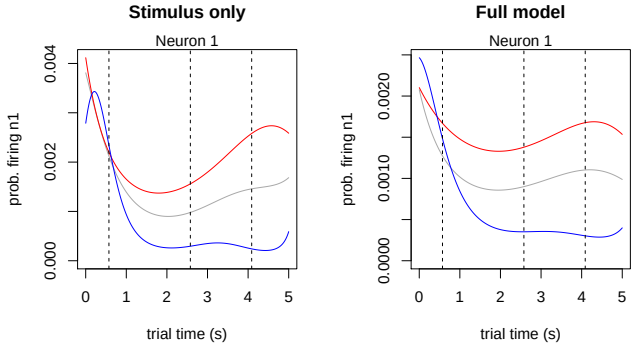

Figure 7.8: Tuning curves from the regression model with only stimulus effects (7.1) (left), and from the full regression model (6.14) (right), fitted for neuron 1. The gray, red and blue curves are tuning curves estimated by using data from `GoodTrials`, correct left lick trials, and correct right lick trials, respectively. The parameter estimates for these curves are shown in Table 7.3. The dashed vertical lines represent the start of the sample, delay and response epochs, respectively.

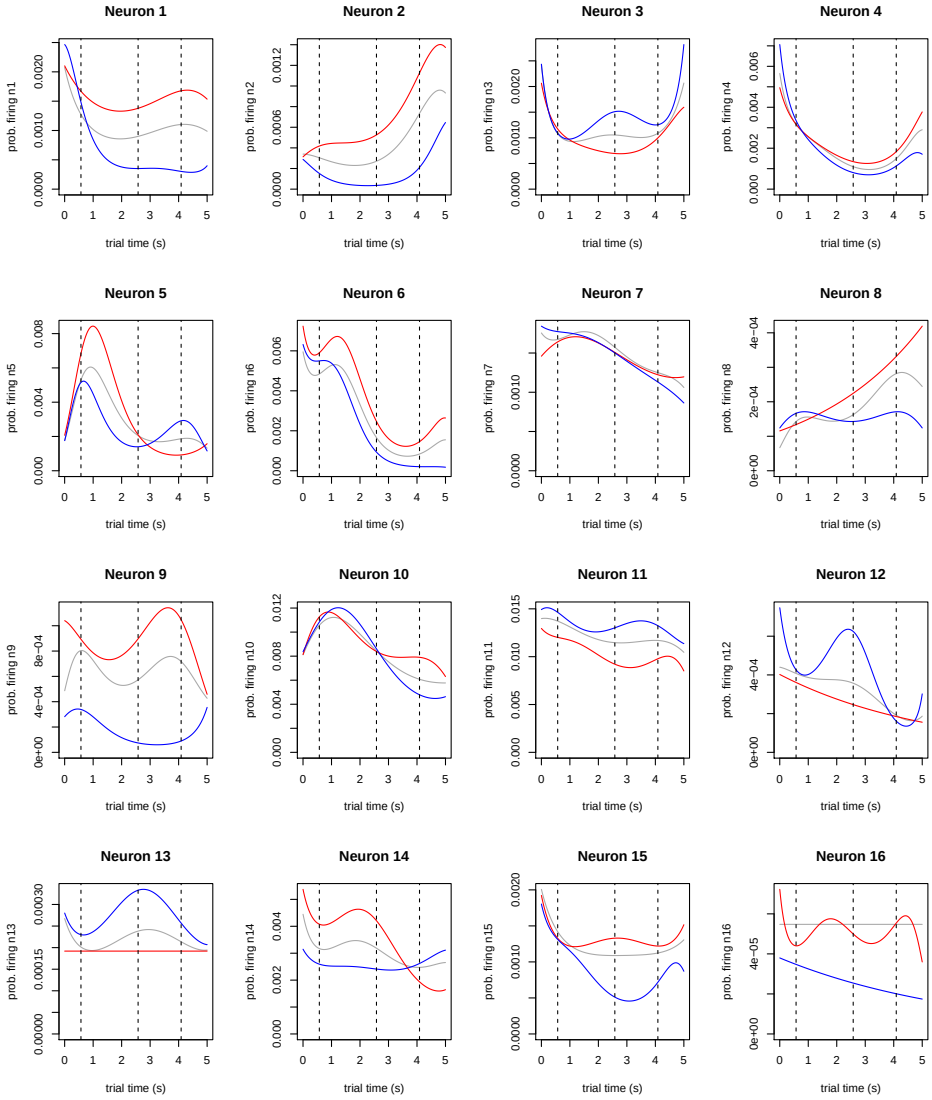

Figure 7.9: Tuning curves for the 16 neurons from session 2. The parameters are estimated using the so-called full regression model with history, connectivity and stimulus effects (6.14), and choosing the values at $\nu_{\min}$. The gray curves are estimated using data from `GoodTrials`, and the red and blue curves from correct lick left and right trials, respectively. The dashed vertical lines are the start of the sample, delay and response epochs, respectively.

7.3 History and Connectivity Effects

In this section we plot history and connectivity effects by using their estimated coefficients from the full regression model (6.14) to evaluate the bases functions discussed in Section 6.2.1. That is, considering the linear predictor in (6.13), we can plot the estimated history effects

$$\hat{\eta}_{\text{hist},j}(t_i) = \hat{\alpha}_{0j} + \sum_{l=1}^{L_{\text{hist}}} \hat{\alpha}_{jjl} \mathbf{b}_{l,M}^T \mathbf{y}_{j,M}(t_i), \tag{7.3}$$

and the estimated connectivity effects

$$\hat{\eta}_{\text{connect},jk}(t_i) = \hat{\alpha}_{0j} + \sum_{l=1}^{L_{\text{connect}}} \hat{\alpha}_{jkl} \mathbf{b}_{l,M}^T \mathbf{y}_{k,M}(t_i), \tag{7.4}$$

on the probability scale (by using the inverse logit-function). That is, we plot history effects as $\text{logit}\{\hat{\eta}_{\text{hist},j}(t_i)\}$ and connectivity effects as $\text{logit}\{\hat{\eta}_{\text{connect},j}(t_i)\}$.

Continuing the example of fitting the full regression model (6.14) using observations from neuron $j = 1$ from session 2, we get the estimated history and connectivity effects shown in Tables 7.4 and 7.5, respectively. Note that the estimated history effects in Table 7.4 are relatively large (in absolute value) compared to the connectivity effects in Table 7.5. (The estimated history effects and the estimated connectivity effects are comparable since the (orthogonal) cosine bases used to represent these effects are on the same scale, seen by comparing the y-axes in Figure 6.2). This was indicated by the plot of the coefficient path in Figure 7.7, where the effect of the first history basis is seen to be prominent (which has an estimated value $\hat{\alpha}_{1,1,1} = -11.13$ at $\nu_{\min} = 2.13 \times 10^{-5}$). Furthermore, both in Tables 7.4 and 7.5 we see that the $\hat{\alpha}_{jkl}$'s have a tendency to be closer to zero as the index l increases. That is, the estimated effects on the firing rate of neuron $j = 1$ become smaller as the lag increases, as discussed in Section 6.2.1 with regards to the placement of the cosine bases in time after a spike event (Figure 6.2).

l	$\hat{\alpha}_{1,1,l}$
1	-11.13
2	-3.50
3	2.09
4	-5.51
5	1.87
6	-4.00
7	-
8	-2.80
9	-
10	-1.23

Table 7.4: Estimated history effects for the penalized logistic model in (6.14) at $\nu_{\min} = 2.13 \times 10^{-5}$, for neuron $j = 1$ from session 2 using data from `GoodTrials`.

Connect. Neuron (k)	$\hat{\alpha}_{1,k,1}$	$\hat{\alpha}_{1,k,2}$	$\hat{\alpha}_{1,k,3}$	$\hat{\alpha}_{1,k,4}$
2	-1.55	0.90	-1.14	0.12
3	-2.07	0.70	-0.54	-
4	-1.18	0.30	-0.16	-0.42
5	-0.08	-0.52	-0.46	-0.47
6	0.11	-	-	-0.04
7	-	-0.08	-	0.48
8	-0.25	-	0.82	-0.21
9	-0.12	0.21	-	-
10	-	-0.56	-	0.16
11	0.80	-0.23	0.10	0.03
12	2.14	-	-	-
13	1.26	0.89	1.21	-
14	0.26	-	0.03	0.16
15	-0.49	0.94	-	-0.26
16	1.62	-	0.23	-

Table 7.5: Estimated connectivity effects for the penalized logistic model in (6.14) at $\nu_{\min} = 2.13 \times 10^{-5}$, for neuron $j = 1$ from session 2 using data from `GoodTrials`.

Setting the estimated coefficients in Tables 7.4 and 7.5 into (7.3) and (7.4), we can plot history and connectivity effects for neuron $j = 1$. These plots are shown in the top row of Figure 7.10, where we've included plots of two connectivity effects, $\hat{\eta}_{\text{connect},1,2}$ and $\hat{\eta}_{\text{connect},1,3}$. (The parameter estimates of these two connectivity effects are shown in the first two rows of Table 7.5). To interpret these connectivity plots, recall the conceptualized connectivity effect from Figure 1.4 in Section 1.2.3. In general, the (j,k)th connectivity effect $\hat{\eta}_{\text{connect},jk}$ can be interpreted as the effect of neuron k's activity on the firing rate of neuron j, as discussed in Section 6.2.1 in the context of the $N \times N \times L$ matrix in (6.7). The plots in the second row of Figure 7.10 are obtained by fitting the full regression model (6.14) using the observations from neuron $j = 2$ ($\mathbf{y}_2$) as the response variable (the off-diagonal plots in this row are the connectivity effects $\hat{\eta}_{\text{connect},2,1}$ and $\hat{\eta}_{\text{connect},2,3}$). Similarly, the plots in the third row of Figure 7.10 are from the full regression model fitted using observations from neuron $j = 3$ ($\mathbf{y}_3$) as the response variable (the off-diagonal plots in this row are the connectivity effects $\hat{\eta}_{\text{connect},3,1}$ and $\hat{\eta}_{\text{connect},3,2}$).

The plots of the history effects in Figures 7.10a, 7.10e and 7.10i show that right after a neuron has fired, the probability of it firing again decreases. This can be seen for all three neurons, where the firing rate is below the background firing rate (horizontal line) until lag 50 ms. As noted in Section 1.2.3, this is known as the refractory period. Furthermore, these history plots show that after the refractory period, the firing rate increases until it reaches a peak at lag 100 ms. Meaning, if neuron j (for $j = \{1, 2, 3\}$) fired anywhere from 50 to 100 ms ago, it is more likely to fire again. This points towards some oscillating activity pattern[3], that is,

[3]This is a know phenomena in neuroscience, called *neural oscillation*, where a neuron fires in

activity that occurs in regular intervals.

The connectivity plots of Figure 7.10 show that these three neurons seem to have excitatory connections among themselves. For example, the connectivity plot in Figure 7.10b shows that right after neuron 2 has fired, neuron 1 is also likely to fire, since the connectivity curve is above the background firing rate of neuron 1. The same pattern can be observed for the directed connection from neuron 3 to neuron 1, shown in Figure 7.10c. Furthermore, these two plots also show that the connectivity effect goes towards zero as the lag increases, as noted in the estimated coefficients in Table 7.5. Note that the excitatory effects among these three neurons appear at different lags. For the directed connections in Figures 7.10b and 7.10c just discussed, the excitatory effect appears rather immediately (at short lags). This is also the case for the directed connection from neuron 3 to neuron 2 shown in Figure 7.10f. However, the excitatory effect appears at later lags for the directed connection from neuron 1 to neuron 2 in Figure 7.10d, and from neuron 1 to neuron 3 in Figure 7.10g. For example, for the directed connection in Figure 7.10d, this excitatory effect appears in the interval 50 to 100 ms. Additionally, a connectivity effect that is excitatory at first can change to an inhibitory effect. This is seen for the directed connection from neuron 2 to neuron 3 in Figure 7.10h, which is excitatory in the first 50 ms and inhibitory in the interval 50 to 100 ms.

A connectivity effect that appears at specific lags is related to how the information flows in the underlying physical network of neurons, as discussed in Section 1.2.3 regarding Figure 1.4. As shown in Figure 1.4, the lag can be divided into three intervals: from 0 to 3 ms, from 3 to 15 ms and from 15 to 100 ms. The connectivity effect that manifests in these intervals indicates the type of underlying connection: common, direct and indirect, respectively. From the connectivity plots of Figure 7.10, the type of underlying connection is not immediately clear, because the lag axis is scaled from 0 to 160 ms, which is a consequence of placing the (orthogonal) cosine bases such that we include observations from the previous $\tau_M = 160$ ms, as stated in Section 6.2.3. This scaling does not reveal the details in the intervals from 0 to 3 ms and from 3 to 15 ms.

a rhythmical fashion.

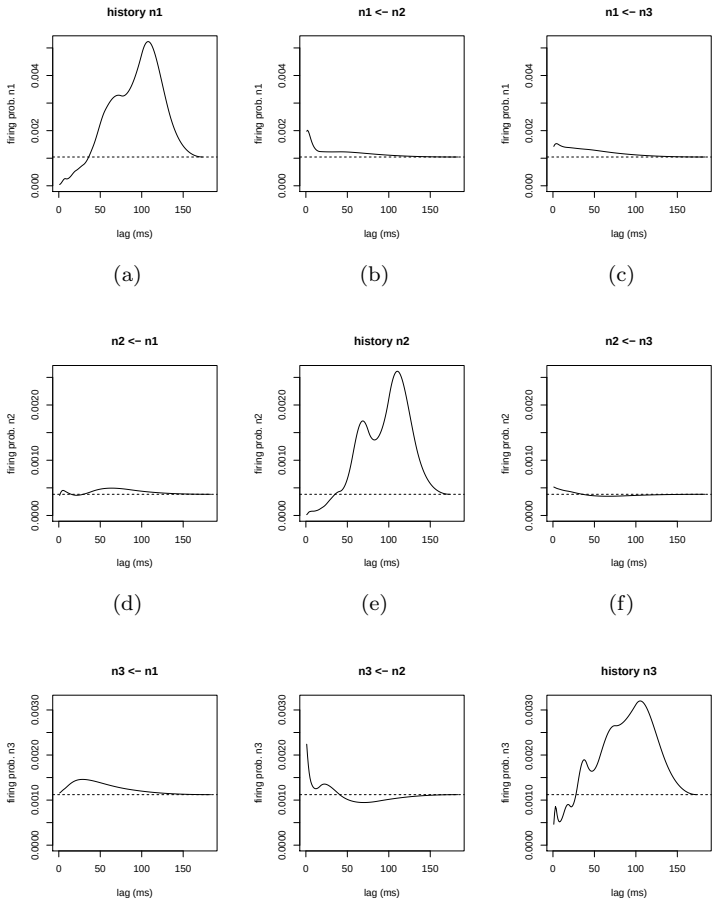

Figure 7.10: History and connectivity effects of neurons 1, 2 and 3 from session 2, estimated using data from `GoodTrials`. The plots on the jth row are obtained by fitting the lasso (6.14) using observations from the jth neuron, $\mathbf{y}_j$, as the response variable. The plots on the diagonal show the estimated history effects $\text{logit}^{-1}\{\hat{\eta}_{\text{hist},j}\}$. The remaining plots in the jth row and the kth column are the estimated connectivity effects $\text{logit}^{-1}\{\hat{\eta}_{\text{connect},jk}\}$. The (j,k)th connectivity represents the effect of neuron k on the activity of neuron j. The dashed horizontal line is the estimated intercept $\text{logit}^{-1}\{\hat{\alpha}_{0j}\}$, which represents the baseline firing rate of neuron j.

7.4 Network of Neurons

In this section we attempt to estimate the underlying network of neurons, by defining significant connectivity effects. Recall from Section 6.2.1 that a directed connection from neuron k to neuron j is represented by $\boldsymbol{\alpha}_{jk} = (\alpha_{jk1}, \alpha_{jk2}, ..., \alpha_{jkL})$ (where $L = 10$ for history effects and $L = 4$ for connectivity effects). In the following, we have chosen to conclude that the connection $\boldsymbol{\alpha}_{jk}$ is significant if at least one of its element is significant. To test the significance of each α_{jkl}, we calculate its adjusted p-value, and use a cut-off level 0.05. (This controls the FWER at 5%, as stated in (4.27) in Section 4.6.2). Similarly, we can also test if neuron j is significantly tuned to a certain stimulus, by testing the significance of each γ_{dj}. That is, if at least one γ_{dj} is significant, for $d = 1, ..., D$, then we conclude that neuron j has a significant tuning curve. Hence, we obtain adjusted p-values for the α_{jkl}'s and the γ_{dj}'s by the multi sample-splitting procedure described in Section 4.6.1. This is done using the function `multi.split` from the `hdi`-package in `R`, where we use $B = 50$ sample-splits (see (4.23)).

7.4.1 Significant Effects

Consider the full regression model (6.14) fitted for neuron $j = 1$ from session 2, using data from `GoodTrials`, for which the estimated coefficients at $\nu_{\min}$ are summarized in Tables 7.3 (stimulus), 7.4 (history) and 7.5 (connectivity). These estimates, along with their adjusted p-values, are summarized in Tables 7.6 (history and connectivity) and 7.7 (stimulus), in the columns corresponding to $\Delta t = 1\,\text{ms}$. This table also shows parameter estimates, and their adjusted p-values, for the same regression model (6.14), except that the interval in which the data is observed $(0, T]$ is discretized using $\Delta t = 10\,\text{ms}$. The reason for considering this coarser discretization is to reduce the computation time needed to calculate the adjusted p-values. For example, it took 6.73 h to calculate the adjusted p-values in Table 7.6 in the $\Delta t = 1\,\text{ms}$ discretization, and only 19 min to calculate the adjusted p-values in the $\Delta t = 10\,\text{ms}$ discretization. Details will be discussed in Section 8.2. Hence, for all remaining neurons in the `alm-1` data set, adjusted p-values will be calculated using the $\Delta t = 10\,\text{ms}$ discretization.

		$\hat{\alpha}_{1,k,l}$		FWER adjusted p-value	
Neuron (k)	Basis (l)	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$
1	1	−11.13	−12.39	6.80×10^{-204}	1.92×10^{-203}
	2	−3.50	−2.61	1.86×10^{-10}	2.29×10^{-6}
	3	2.09	1.41	4.60×10^{-3}	2.48×10^{-1}
	4	−5.51	−4.32	1.22×10^{-10}	1.06×10^{-8}
	5	1.87	−0.82	1.00	1.00
	6	−4.00	−1.72	4.15×10^{-7}	9.81×10^{-1}
	7	-	−1.65	1.00	1.00
	8	−2.80	-	1.82×10^{-4}	1.00
	9	-	2.60	1.00	1.22×10^{-2}

		$\hat{\alpha}_{1,k,l}$		FWER adjusted p-value	
Neuron (k)	Basis (l)	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$
	10	-1.23	-	3.74×10^{-1}	1.00
2	1	-1.55	-1.66	1.00	1.00
	2	0.90	0.29	1.00	1.00
	3	-1.14	-0.75	1.00	1.00
	4	0.12	-	1.00	1.00
3	1	-2.07	-1.85	7.08×10^{-5}	1.32×10^{-4}
	2	0.70	0.52	1.00	1.00
	3	-0.54	-0.28	1.00	1.00
	4	-	-	1.00	1.00
4	1	-1.18	-1.12	3.03×10^{-2}	2.46×10^{-1}
	2	0.30	0.36	1.00	1.00
	3	-0.16	-0.37	1.00	1.00
	4	-0.42	-0.60	1.00	1.00
5	1	-0.08	-0.10	1.00	1.00
	2	-0.52	-0.76	1.00	1.00
	3	-0.46	-0.34	1.00	1.00
	4	-0.47	-0.49	1.00	1.00
6	1	0.11	0.15	1.00	1.00
	2	-	-	1.00	1.00
	3	-	-	1.00	1.00
	4	-0.04	-	1.00	1.00
7	1	-	-	1.00	1.00
	2	-0.08	-0.07	1.00	1.00
	3	-	0.18	1.00	1.00
	4	0.48	0.35	1.00	1.00
8	1	-0.25	-	1.00	1.00
	2	-	-	1.00	1.00
	3	0.82	-	1.00	1.00
	4	-0.21	-	1.00	1.00
9	1	-0.12	-	1.00	1.00
	2	0.21	0.26	1.00	1.00
	3	-	-	1.00	1.00
	4	-	-	1.00	1.00
10	1	-	-	1.00	1.00
	2	-0.56	-0.40	1.00	1.00
	3	-	-	1.00	1.00
	4	0.16	0.75	1.00	1.00
11	1	0.80	0.75	1.05×10^{-5}	5.43×10^{-5}
	2	-0.23	-0.15	1.00	1.00
	3	0.10	-	1.00	1.00
	4	0.03	-	1.00	1.00
12	1	2.14	2.20	1.00	1.00

		$\hat{\alpha}_{1,k,l}$		FWER adjusted p-value	
Neuron (k)	Basis (l)	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$
	2	-	-	1.00	1.00
	3	-	0.10	1.00	1.00
	4	-	-	1.00	1.00
13	1	1.26	1.50	1.00	1.00
	2	0.89	0.84	1.00	1.00
	3	1.21	0.69	1.00	1.00
	4	-	-	1.00	1.00
14	1	0.26	0.30	1.00	1.00
	2	-	-	1.00	1.00
	3	0.03	-	1.00	1.00
	4	0.16	0.06	1.00	1.00
15	1	−0.49	−0.33	1.00	1.00
	2	0.94	0.79	1.00	1.00
	3	-	-	1.00	1.00
	4	−0.26	−0.32	1.00	1.00
16	1	1.62	1.62	1.00	1.00
	2	-	-	1.00	1.00
	3	0.23	-	1.00	1.00
	4	-	3.07	1.00	1.00

Table 7.6: Parameter estimates of history and connectivity effects of neuron $j = 1$ (from session 2) $\hat{\alpha}_{1,j,l}$, and their adjusted p-values (with respect to the FWER, as shown in (4.27)). The parameter estimates are obtained from the penalized logistic model in (6.14) using data from `GoodTrials`, fitted once for each discretization $\Delta t = 1\,\text{ms}$ and $\Delta t = 10\,\text{ms}$. The adjusted p-values are obtained using the function `muli.split` from the `hdi`-package, with $B = 50$ iterations. The gray boxes highlight p-values that are below the cut-off level 0.05.

	$\hat{\gamma}_{d,1}$		FWER adjusted p-value	
Degree (d)	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$	$\Delta t = 1\,\text{ms}$	$\Delta t = 10\,\text{ms}$
1	−69.09	−24.71	1.00	1.00
2	168.92	54.89	2.74×10^{-4}	2.90×10^{-4}
3	−123.85	−41.65	3.31×10^{-2}	3.60×10^{-3}
4	-	-	1.00	1.00
5	0.70	-	1.00	1.00

Table 7.7: Parameter estimates of stimulus effects of neuron $j = 1$ (from session 2) $\hat{\gamma}_{d,1}$, and their adjusted p-values (with respect to the FWER, as shown in (4.27)). The parameter estimates are obtained from the penalized logistic model in (6.14) using data from `GoodTrials`, fitted once for each discretization $\Delta t = 1\,\text{ms}$ and $\Delta t = 10\,\text{ms}$. The adjusted p-values are obtained using the function `muli.split` from the `hdi`-package, with $B = 50$ iterations. The gray boxes highlight p-values that are below the cut-off level 0.05.

Note that the parameter estimates from the coarser discretized model with $\Delta t = 10\,\text{ms}$ are similar to the parameter estimates from the finer discretized model with $\Delta t = 1\,\text{ms}$ (in Tables 7.6 and 7.7). This is to be expected, since the parameters in these models represent the same effects, regardless of what bin length is used to discretize the observed interval $(0, T]$. To see this, consider the regression parameters α_{jkl} and γ_{dj} as weights that are used to evaluate the basis functions they represent (orthogonal cosines b_l (6.8) and orthogonal polynomials P_d (6.11), respectively). To illustrate, Figure 7.11 shows evaluated cosine bases, using the two discretizations $\Delta t = 1\,\text{ms}$ and $\Delta t = 10\,\text{ms}$. These plots show that the bases retain the same shape, and hence, capture the same effects, regardless of the bin length. (This is also true for the orthogonal versions of these bases, which are actually used to fit the regression models). However, it is preferable to discretize using $\Delta t = 1\,\text{ms}$, which leads to bases functions that capture the effects in more detail. This can be seen by comparing the history and connectivity effects in Figure 7.12 (which are fitted using the coarser discretized model with $\Delta t = 10\,\text{ms}$), to the corresponding effects in Figure 7.10 (fitted using the finer discretized model with $\Delta t = 1\,\text{ms}$). This comparison shows that the plots in 7.12 resemble the corresponding plots in Figure 7.10, however, the former results in less detailed and more jagged plots.

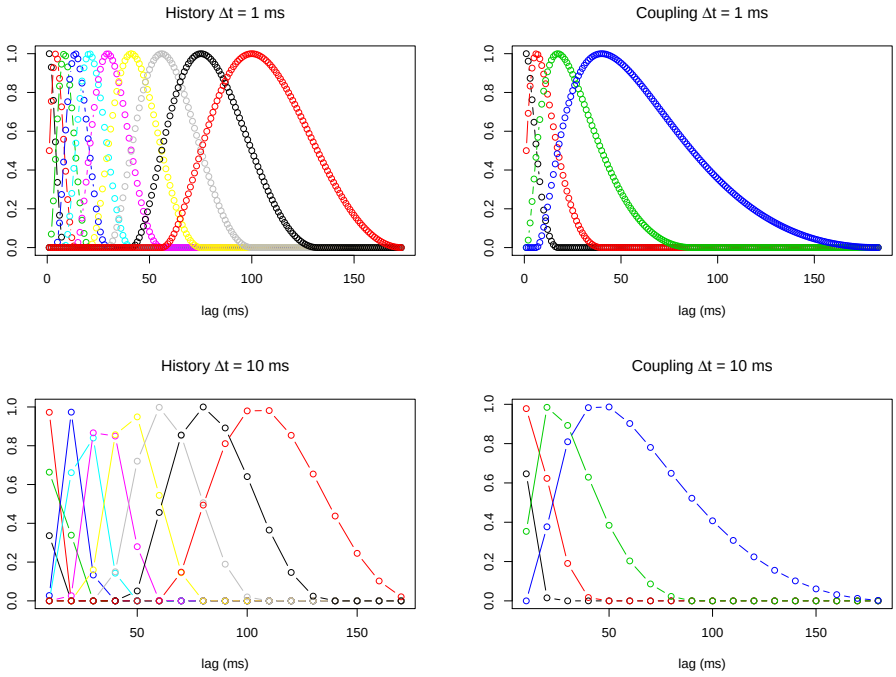

Figure 7.11: Evaluated cosine bases b_l from (6.8), for $l = 1, ..., L$. The left column shows the $L = 10$ history bases, and the right column shows the $L = 4$ connectivity bases. Each plotted point is $b_l(t_i)$, where $t_i = i \cdot \Delta t$ is the midpoint of the ith bin for $i = 1, ..., M$, where M is the number of bins we go back (relative to bin i). In the top row $\Delta t = 1\,\text{ms}$, and $M = 160$. And in the bottom row $\Delta t = 10\,\text{ms}$, and $M = 16$.

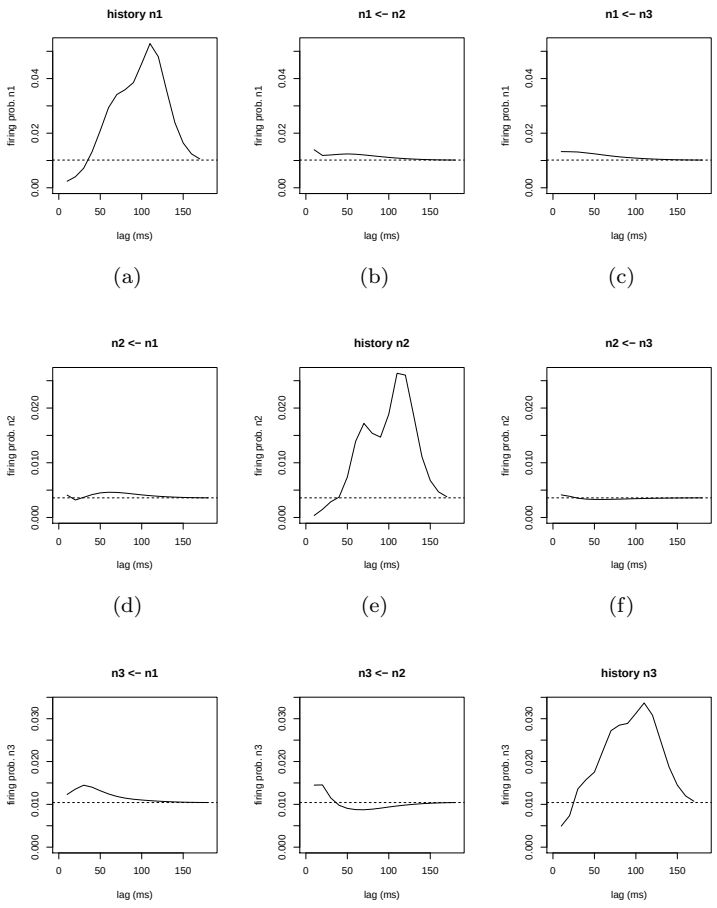

Figure 7.12: History and connectivity effects of neurons 1, 2 and 3 in session 2, using data from `GoodTrials`, discretized using bin length $\Delta t = 10\,\text{ms}$. The plots on the jth row are obtained by fitting the lasso (6.14) using observations from the jth neuron, $\mathbf{y}_j$, as the response variable. The plots on the diagonal show the estimated history effects $\text{logit}^{-1}\{\hat{\eta}_{\text{hist},j}\}$. The remaining plots in the jth row and the kth column are the estimated connectivity effects $\text{logit}^{-1}\{\hat{\eta}_{\text{connect},jk}\}$. The dashed horizontal line is the estimated intercept $\text{logit}^{-1}\{\hat{\alpha}_{0j}\}$.

Furthermore, note the difference in the y-axes between the effects in Figure 7.12 and their corresponding effects in Figure 7.10. For example, the history effect of neuron 1 in Figure 7.12a shows that neuron 1 has a baseline firing probability of 0.01 (intercept), and a maximum firing probability of 0.05 (reached at lag 100 ms). The same history effect in Figure 7.10a, however, shows a baseline firing probability of 0.001 and a maximum firing probability of 0.005. The fact that the probability scale

decreases by a factor 10, when the interval $(0, T]$ is discretized using $\Delta t = 1\,\text{ms}$ rather then $\Delta t = 10\,\text{ms}$, is related to the number of bins in which there are no observed firings. Recall the discretization process from Section 6.1, illustrated in Figure 6.1. As the number of bins $n = T/\Delta t$ increases by a factor 10 when using $\Delta t = 1\,\text{ms}$ rather than $\Delta t = 10\,\text{ms}$, the number of bins where $y_j(t_i) = 0$ increases by the same factor. Hence, the firing probabilities in our analyses should not be considered as absolute quantities. However, these probabilities can be used to assess the change in activity of a neuron compared to a baseline activity. For example, the history effect of neuron 1 shows that the maximum firing probability is five times higher than its baseline firing probability in both discretizations.

More importantly, Tables 7.6 and 7.7 show that the calculated adjusted p-values are similar (in most cases) for the estimated effects from the $\Delta t = 1\,\text{ms}$ discretized model and the $\Delta t = 10\,\text{ms}$ discretized model. For example, using significance level 5%, there are 6 (out of $L_{\text{hist}} = 10$) history effects that are significant in the $\Delta t = 1\,\text{ms}$ discretized model (shown by the gray boxes in Table 7.6). Similarly, there are 4 history effects that are significant in the $\Delta t = 10\,\text{ms}$ discretized model. Hence, using the rule that $\boldsymbol{\alpha}_{jk}$ is significant if at least one of its element is significant, we get that the history effect for neuron 1 ($\boldsymbol{\alpha}_{1,1}$) is significant, in both diszcretizations. Furthermore, in both diszcretizations, we get that the directed connections from neuron 3 to neuron 1 ($\boldsymbol{\alpha}_{1,3}$) and from neuron 11 to neuron 1 ($\boldsymbol{\alpha}_{1,11}$) are also significant. And additionally, in both diszcretizations, we also get that neuron 1 has a significant tuning curve (as shown by the gray boxes in Table 7.7). However, according to this pragmatic significance rule, the directed connection from neuron 4 to neuron 1 ($\boldsymbol{\alpha}_{1,4}$) is significant under the $\Delta t = 1\,\text{ms}$ discretized model, as the adjusted p-value of $\alpha_{1,4,1}$ is under the cut-off level 0.05. But, $\boldsymbol{\alpha}_{1,4}$ is not significant under the $\Delta t = 10\,\text{ms}$ discretized model. That is, we trade-off accuracy in detecting significant effects in favour of decreasing computation time, when we calculate the adjusted p-values from the $\Delta t = 10\,\text{ms}$ discretized model rather than from the $\Delta t = 1\,\text{ms}$ discretized model.

In summary, we've observed that using cut-off level 0.05 we get almost the same conclusions regarding significance from both discretizations for neuron $j = 1$ from session 2. Hence, for the remaining neurons in session 2 (and generally for the remaining neurons from any session and any mouse in the `alm-1` data set), we test for significant effects using adjusted p-values calculated from the $\Delta t = 10\,\text{ms}$ discretized model.

7.4.2 Excitatory and Inhibitory Connections

In estimating the network of neurons, we focus on direct connections, which are described in the discussion regarding Figure 1.4 in Section 1.2.3. Using this figure, we can define excitatory and inhibitory connections by evaluating the area between the connectivity effect curve and the baseline. That is, consider the dark gray area (from 3 to 15 ms) in Figure 1.4, which illustrates how the direct connection from neuron a affects the activity of neuron b. As seen, the effect is excitatory at first, but then changes to a slight inhibitory effect. However, to classify this connectivity as either excitatory or inhibitory, we calculate the area under the effect curve that

is above the baseline (intercept), and the area that is under the baseline. We approximate these areas by a simple trapezoidal rule. Then, if the area above the baseline is greater than the area under the baseline, we classify the connectivity as excitatory. (This is the case illustrated in Figure 1.4). Similarly, if the area under the baseline is greater, then the connection is classified as inhibitory. Furthermore, by evaluating the difference in the area above and below the connectivity curve and the baseline we can estimate to some extent the *strength* of the connection. For example, if the area above the baseline is much larger than the area under the baseline, we consider this connection as a strong excitatory connection. To accurately evaluate these areas, we use the estimated connectivity effects from models where the interval $(0, T]$ is discretized using $\Delta t = 1\,\text{ms}$. (Recall that the connectivity plots form the $\Delta t = 10\,\text{ms}$ discretized model where less detailed and more jagged, as shown in Figure 7.12).

7.4.3 Network of Connections Between the 16 Neurons from Session 2

Figures 7.13, 7.14 and 7.15 show the estimated network of neurons from session 2, using data from `GoodTrials`, correct left lick trials, and correct right lick trials, respectively. Each directed connection from neuron k to neuron j in these networks is a significant connectivity effect $\boldsymbol{\alpha}_{jk}$, evaluated using cut-off level 0.05 on the adjusted p-values calculated from the full regression model (6.14), where the interval $(0, T]$ is discretized using $\Delta t = 10\,\text{ms}$. For example, as shown in Table 7.6, neuron 1 has significant connections from neurons 3 ($\boldsymbol{\alpha}_{1,3}$) and 11 ($\boldsymbol{\alpha}_{1,11}$). This is shown as arrows pointing from neurons 3 and 11 to neuron 1 in Figure 7.13. These two connections are evaluated as excitatory and inhibitory, respectively. Figure 7.10c shows why the former connection was evaluated as excitatory, as the plotted connectivity effect lies above the baseline, seemingly for all lags, especially for lags 3 to 15 ms. Additionally, Table 7.6 also shows that neuron 1 has significant history effects ($\boldsymbol{\alpha}_{1,1}$). This is displayed in Figure 7.13 as the curved dashed arrow from neuron 1 to itself. Furthermore, Table 7.7 shows that neuron 1 has a significant tuning curve. This is shown in Figure 7.13 by coloring neuron 1 as a light gray node (otherwise neuron 1 would've been colored as a dark gray node).

Note, for each estimated network, the full regression model (6.14) is fitted N times, independently, where N is the number of observed neurons in the session. That is, the regression model is fitted using observations from the jth neuron, $\mathbf{y}_j$, as the response variable, and the remaining $k \neq j$ neurons as covariates, for $j = 1, ..., N$.

As mentioned in Section 1.2.3, we can expect there to be $(N^2 - N) \cdot 5\%$ directed connections in a network of N neurons. In session 2, there are $N = 16$ neurons, and hence we can expect there to be 12 directed connections. In the estimated networks in Figures 7.13, 7.14 and 7.15 there are 35, 21 and 12 directed connections. First off, we note that N is not necessary the total number of recorded neurons, but rather, the number of so-called principal neurons (Section 1.2.3). Recall that principle neurons are neurons that transmit an excitatory signal to other neurons.

Whats more, it is unclear whether neurons that provide both an excitatory and an inhibitory connectivity should be considered principle neurons (neuron 14 is such an example, which seems to excite neurons 6 and 11, but inhibit neuron 15, see Figure 7.13). Secondly, we have classified a connection as excitatory/inhibitory by only considering the connectivity effects in the 3 to 15 ms range, as outlined in the previous section, which also adds to the uncertainty in the number of expected connections. Furthermore, as noted above, we've made a trade-off between accuracy and computation time by evaluating significance (and hence the existence of a connection in these networks) using $\Delta t = 10\,\text{ms}$ to discretize the interval $(0, T]$. Additionally, we note that the number of connections decreases when estimating the network using data from `GoodTrials`, correct left and right trials. Recall from Section 1.3.3 that data from correct left trials and correct right trials are subsets of `GoodTrials`. There are 285 `GoodTrials`, 134 correct lick left trials and 116 correct lick right trials in session 2. That is, the number of estimated connections seems to decrease as the available data decrease in size. However, the difference in the estimated networks of Figures 7.14 and 7.15 might be because of an actual underlying difference in how the ALM responds to left/right licks.

Note that neuron 11 seems to stand out in the estimated networks in all three trial types, with regards to the number of connections both from and to this neuron. Recall from Table 7.1 that this neuron was the only neuron in session 2 that was classified as an FS neuron (in the pre-processing of the raw data, done by Li et al. (2015, Methods)), which are neurons that transmit signals to other neurons in the ALM. Secondly, note that there is a strong excitatory connection from neuron 15 to neuron 2, in all of the three estimated networks. In Figure 1.8, which shows the relative location of the recorded neurons, we can see that neurons 2 and 15 were recorded on the same electrode/channel in session 2. Finally, note that neuron 16 is not displayed in the estimated networks in Figures 7.14 and 7.15. This is because neuron 16 has too few observations, which can be seen in the raster plot for session 2 in Figure 7.1 where neuron 16 is active during approximately one third of the session time. Hence, when fitting the regression model (6.14) using $\mathbf{y}_{16}$ as the response variable, the (nested) coordinate descent algorithm from Section 4.4 does not converge. This seems to apply generally for neurons that have a relatively small number of spikes, which will be discussed in Section 8.2.

Estimated networks for session 1 and session 3 are shown in Appendix B.4, where there are several neurons with too few spikes to fit the regression model. (These are colored black in Figures B.8 and B.9).

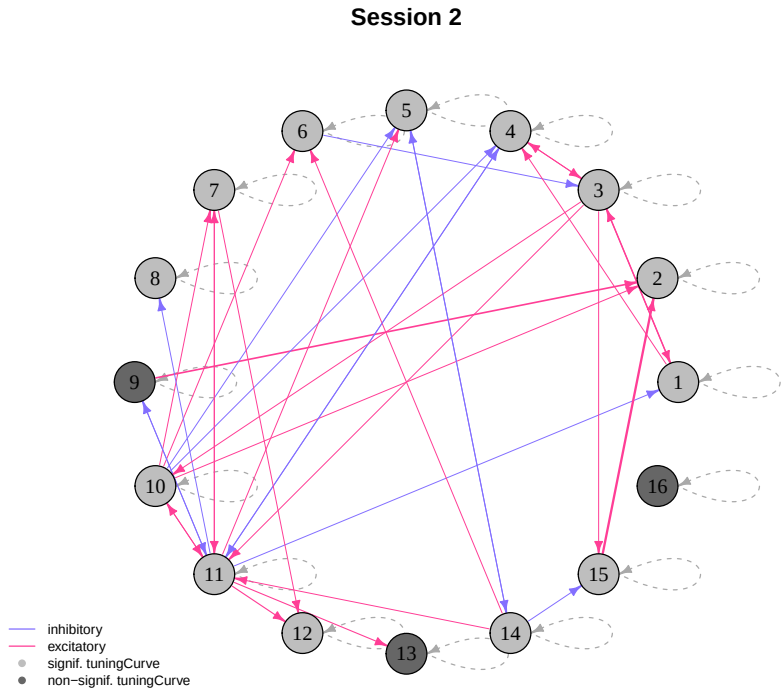

Figure 7.13: Estimated network of the 16 neurons from session 2, using data from `GoodTrials`. A directed arrow from neuron k to neuron j indicates a significant connection $\boldsymbol{\alpha}_{jk}$, tested at 5% significance level. Light gray nodes represent neurons that have a significant tuning curve, also tested at 5% significance level. Excitatory and inhibitory connections are assigned by evaluating the difference in the area between the connectivity effect and the baseline firing probability. The width of an arrow shows the strength of the connectivity, as the (normalized) absolute value of the difference in the area above and below the connectivity curve and the baseline.

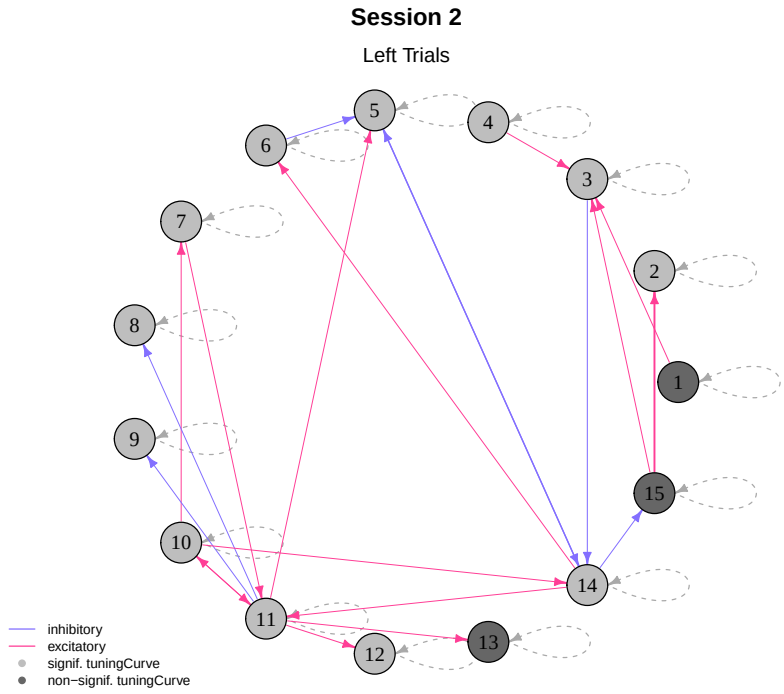

Figure 7.14: Estimated network of the 16 neurons from session 2, using data from correct lick left trials. A directed arrow from neuron k to neuron j indicates a significant connectivity $\boldsymbol{\alpha}_{jk}$, tested at 5% significance level. Light gray nodes represent neurons that have a significant tuning curve, also tested at 5% significance level. Excitatory and inhibitory connections are assigned by evaluating the difference in the area between the connectivity effect and the baseline firing probability. The width of an arrow shows the strength of the connectivity, as the (normalized) absolute value of the difference in the area above and below the connectivity curve and the baseline.

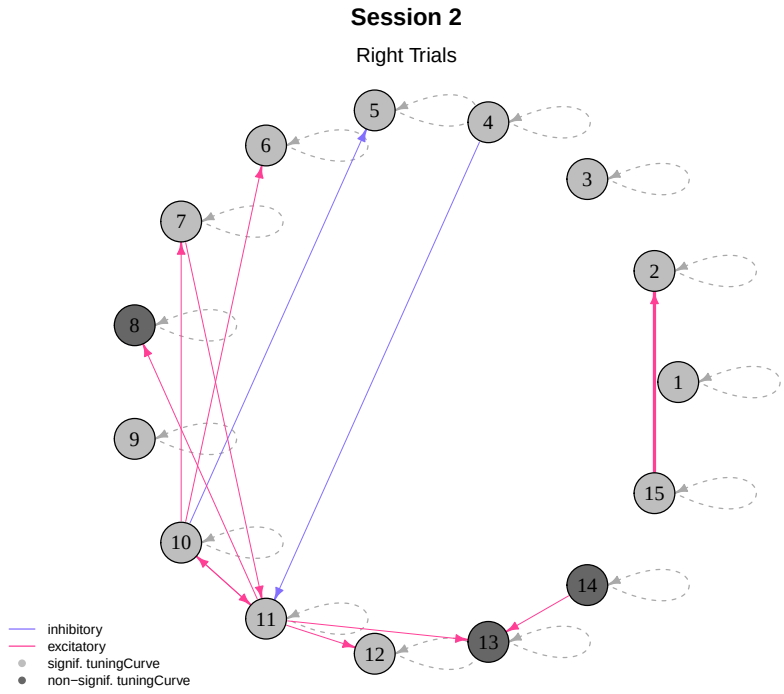

Figure 7.15: Estimated network of the 16 neurons from session 2, using data from correct lick right trials. A directed arrow from neuron k to neuron j indicates a significant connectivity $\boldsymbol{\alpha}_{jk}$, tested at 5% significance level. Light gray nodes represent neurons that have a significant tuning curve, also tested at 5% significance level. Excitatory and inhibitory connections are assigned by evaluating the difference in the area between the connectivity effect and the baseline firing probability. The width of an arrow shows the strength of the connectivity, as the (normalized) absolute value of the difference in the area above and below the connectivity curve and the baseline.

Chapter 8

Discussion and Conclusion

8.1 Summary of Results

The analyses in Chapter 7 are based on data from mouse `ANM210861`, which has three sessions called session 1, 2 and 3. In these three sessions, there are extracellular recordings of 30, 16 and 12 neurons, respectively. As part of the exploratory data analysis in Section 7.1, we observed that some neurons were active throughout the whole session, while some were active only in parts of the session. This is shown by the raster plots in Figure 7.1. The boxplots of (empirical) firing rates in Figure 7.2 showed that neurons were most active during session 2. The median firing rates over all neurons in sessions 1, 2 and 3 were 0.63 Hz, 1.92 Hz and 0.61 Hz, respectively. Table 7.1 showed that the majority of the neurons are pyramidial neurons, which are neurons that connect activity from the ALM into motor-related areas in the brainstem. This motivated the analyses of stimulus effects.

Hence, in Section 7.1.2, we calculated empirical firing rates in the three trial epochs (delay, sample and response). The plots in Figure 7.3 indicated that a neurons activity is related to the stimulus. From these plots we could observe different types of stimulus related activity, such as selectivity, which is a neurons predisposition to increase its activity during either lick left or lick right trials.

In Section 7.2.1, we fitted a logistic penalized regression model that included stimulus effects as covariates. These stimulus effects where represented by a 5th degree polynomial, using a set of orthogonal Legendre bases $\{P_1, ..., P_5\}$ (which are shown in Figure 6.4). As an example, we fitted this model using observations from neuron $j = 1$, $\mathbf{y}_1$, from session 2. The resulting minimum binomial deviance model retained all five parameters (none were set to zero), since the regularization parameter was close to zero ($\nu_{\min} = 1.97 \times 10^{-5}$). Since there were only five parameters in the model, we also fitted a logistic GLM. The estimated coefficients and adjusted p-values were similar between the lasso restricted logistic model and the unrestricted logistic model, as shown by Table 7.2. Most importantly, we used the estimated parameters from the lasso to plot a tuning curve, as shown in Figure 7.5. This tuning curve captured to some extent the trend of the activity of

neuron 1 shown in Figure 7.2. The tuning curves in Figure 7.6 showed that the fitted regression models captured the selectivity, that is, the difference in neuronal activity in correct lick left and correct lick right trials.

In Section 7.2.1, we fitted a logistic penalized regression model that, in addition to including stimulus effects, also included history and connectivity effects. These additional effects were represented by $L_{\text{hist}} = 10$ and $L_{\text{connect}} = 4$ (orthogonal) cosine bases (which are shown in Figure 6.2). As an example, we fitted this so-called full regression model to observations from neuron $j = 1$, $\mathbf{y}_1$, from session 2, and observed that the minimum binomial deviance model (at $\nu_{\min} = 2.13 \times 10^{-5}$) retained 55 (out of 76) of the parameters in the model. Table 7.3 compared the stimulus effects from the full regression model (net effect) to the stimulus effects from the stimulus only model (gross effect). The regularization parameter $\nu_{\min}$ was generally larger for the full regression model, and hence, lead to the net effects being smaller (in absolute value) than the gross effects. However, note that the value of $\nu_{\min}$ in the full regression model is still in the order of 10^{-5}, which is close to zero. That is, we could have fitted the usual (unrestricted) GLM using $\mathbf{y}_1$, but we prefer to fit the lasso restricted model to allow the estimated parameters to be exactly zero. (This is because we would like to achieve sparsity in the estimated parameters, in order to estimate network of neurons that are as simple as possible, as will be discussed below). Figure 7.8 showed that the tuning curves for neuron 1 from session 2 have similar shapes whether they're estimated using the full regression model, or the stimulus only model. The same trend can be seen for all 16 neurons from session 2 by comparing the plots in Figure 7.9 to the plots in Figure 7.6.

The estimated history and connectiviy effects from the full regression model were the focus of Section 7.3. Figure 7.10 plotted these effects for neurons 1, 2 and 3 from session 2. These plots showed that the history of neuron j had the largest effect (in absolute value) on its firing rate, relative to a connectivity effect from neuron k (for $j, k \in \{1, 2, 4\}$ and $j \neq k$). Furthermore, the history effect seemed to capture the refractory periods (see Section 1.2.3) of each neuron, in addition to some oscillatory pattern. The connectivity effects between neurons 1, 2 and 3 seemed excitatory.

In Section 7.4 we attempted to estimated the underlying network of neurons. This was based on two concepts: significance of (history, connectivity and stimulus) effects, and classification of a connection between two neurons (excitatory or inhibitory). In Section 7.4.1 we incorporated a pragmatic approach to finding significant effects: the directed connectivity effect from neuron k to neuron j, $\boldsymbol{\alpha}_{jk} = (\alpha_{jk1}, ..., \alpha_{jkL})$, is significant if at least one of its element is significant. And similarly, we could also conclude that neuron j was significantly tuned to a stimulus if at least one of the parameters representing stimulus effects $\gamma_{j1}, ..., \gamma_{jD}$ was significant. For neuron $j = 1$ from session 2, we found that calculating adjusted p-values using the `R`-function `multi.split` took 6.73 h, when the observations in the time interval $(0, T]$ used to fit the regression model were discretized using $\Delta t = 1$ ms. For this reason, we increased the bin length to $\Delta t = 10$ ms, which resulted in a considerable reduction in computation time. `multi.split` took 19 min to calculate the adjusted p-values for neuron $j = 1$ from session 2, when the time interval $(0, T]$

was discretized using $\Delta t = 10\,\text{ms}$. The adjusted p-values from the $\Delta t = 10\,\text{ms}$ discretized model were similar (in all but one case) to the p-values from the $\Delta t = 1\,\text{ms}$ discretized model. The former adjusted p-values resulted in the significance of $\boldsymbol{\alpha}_{1,1}$, $\boldsymbol{\alpha}_{1,3}$, $\boldsymbol{\alpha}_{1,11}$ and the tuning curve. The latter adjusted p-values, however, resulted in the significance of the same effects as the former, but in addition, also concluded that $\boldsymbol{\alpha}_{1,4}$ was a significant effect. Hence, it seems that we trade-off accuracy in detecting significant effects in favour of decreasing computation time, when calculating adjusted p-values using the $\Delta t = 10\,\text{ms}$ discretization rather than the $\Delta t = 1\,\text{ms}$ discretization.

In addition to evaluating the significance of a connection, we also classified its effect as either excitatory or inhibitory. This was done by evaluating the plot of the connectivity curve, and its deviation from the baseline firing rate, by calculating the area between these quantities, as described in Section 7.4.2. Note, these areas were only calculated for lags in the interval from 3 to 15 ms, which are direct connections as illustrated by Figure 1.4.

Finally, in Section 7.4.3, we presented the resulting network of the 16 neurons from session 2. These networks were estimated by fitting the lasso penalized regression model with stimulus, history and connectivity effects, once for each neuron $j = 1, ..., N$. Figures 7.13, 7.14 and 7.15 show the estimated networks based on data from `GoodTrials`, correct lick left trials, and correct lick right trials. These three networks are seemingly different, however, that might be attributed to the size of the data used to fit the regression model, since we observed that as the number of trials reduces, so does the number of significant connections. Furthermore, we observed in all Figures 7.13, 7.14 and 7.15 that there were most connections from and to neuron 11, which was the only neuron in session 2 that was classified as an FS neuron in the pre-processing of the raw data (Li et al., 2015, Methods). Lastly, we noted that neurons 2 and 15 had a strong connection between themselves, and they were recorded on the same electrode/channel, as shown in Figure 1.8. Furthermore, the majority of the neurons in session 2 seem to have significant tuning curves, as shown by the blue nodes in these estimated networks.

8.2 Challenges in the Data Analysis

Representing Stimulus Effects As stated in Section 6.2.2, we represented the stimulus effects by choosing a set of (orthogonal) Legendre bases $\{P_1, ..., P_D\}$, with $D = 5$ fixed. These bases are shown in Figure 6.4. That is, we completely determine the number of bases and their parametric shape, such that the resulting additive model can be fitted by the usual IWLS algorithm, as stated in Section 5.1. However, by letting the bases function be so-called *thin plate regression splines* (Wood, 2006, p. 153), we can let the data estimate the degrees of freedom, which is a comparable quantity to the degree of a polynomial D. This was done by using the `R`-function `gam` from the package `mgcv`, by fitting the stimulus only model using observations of neuron 1, $\mathbf{y}_1$, from session 2 using data from `GoodTrials`. The estimated tuning curve is shown in Figure 8.1, which has 4.15 estimated degrees of freedom. Compare this tuning curve to the tuning curve estimated using the orthogonal Legendre

bases with $D = 5$ in Figure 7.5 from Section 7.2.1, where we fitted a comparable regression model with only stimulus effects to observations from neuron 1 from session 2. These two tuning curves, fitted using two different methods, are rather similar. That is, by letting the fitting procedure underlying the `R`-function `gam` estimate the degrees of freedom, and hence the shape of the tuning curve, we obtain a similar result compared to when predefining the degree D of a Legendre polynomial. This gives us confidence that the tuning curves of Section 7.2 rely more so on the data used to fit the regression model, rather than the imposed degree D of the polynomial that represents the stimulus effect.

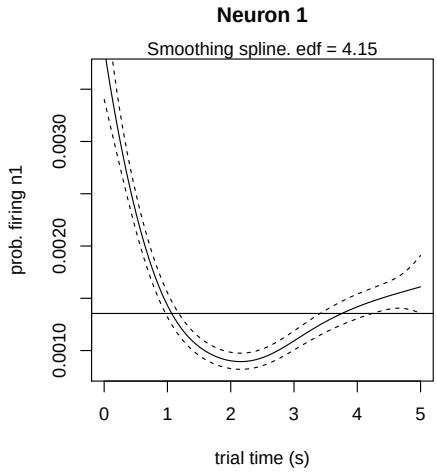

Figure 8.1: Estimated tuning curve for a regression model with only stimulus effects, using observations from neuron 1 from session 2. This curve is bases on so-called thin plate regression splines, which is estimated using the `R`-function `gam` from the package `mgcv`. The horizontal line is the inverse logit-function of the estimated intercept, which represents a baseline probability of firing. The dashed lines are the 2-standard error estimates of the fitted curve. The estimated degrees of freedom for the fitted curve is 4.15.

Computation Time As noted in Section 7.4.1, calculating the adjusted p-values for regression models where the observed time interval $(0, T]$ is discretized by $\Delta t = 1\,\text{ms}$, takes a considerable amount of time. For example, for neuron 1 from session 2, the calculation (of the adjusted p-values in Tables 7.6 and 7.7, based on data from `GoodTrials`) took 6.73 h, when taking advantage of parallelization on a server with multiple cores.[1] We've observed that the computation time is related to the dimensions of the model matrix used to fit the regression model. For example, in session 2 there 285 `GoodTrials`. Since each trial lasts 5 s, the observed time

[1] The server has 24 (logical) cores, as reported by the `R`-function `detectCores` from the package `parallel`.

interval is $T = 285 \cdot 5 = 1450\,\text{s}$ long. Hence, discretizing using $\Delta t = 1\,\text{ms}$, we get that the length of the observation vector of a neuron is $|\mathbf{y}_j| = 1.425 \times 10^6$. That is, the model matrix $\mathbf{X}$ has 1.425×10^6 rows. When we increased the bin size by a factor 10, that is, $\Delta t = 10\,\text{ms}$, the resulting model matrix $\mathbf{X}$ had 10 times fewer rows (1.425×10^5), which decreased the computation time of the adjusted p-values for neuron 1 from session 2 to 19 min.

Calculation of the adjusted p-values using the multi-sample approach hinges on estimating the parameters of a lasso regularized regression model at $\nu_{\min}$, B times (where B is the number of sample splits), as explain in Section 4.6.1, where usually $B = 50$ or $B = 100$ (we used $B = 50$ sample splits when calculating the adjusted p-values in Section 7.4.1). Recall that the parameter estimates of the lasso at $\nu_{\min}$ are obtained by fitting several lasso models for different values of the regularization parameter ν (called the lasso path), and cross-validating with respect to some loss function (binomial deviance in the case of logistic lasso model), as outlined in Section 4.5. Consider fitting the lasso path once, which is done by the cyclical coordinate descent algorithm, which is described in Section 4.3.3. At the heart of this algorithm lies the soft-thresholding operator (4.15), which is used to update each parameter in the model, as shown in (4.16). For conveniance, this parameter-wise updating scheme is re-stated here

$$\hat{\beta}_j^{t+1} \leftarrow \mathcal{S}_\nu\Big(\hat{\beta}_j^t + \frac{1}{n}\langle \mathbf{r}^t, \mathbf{x}_j \rangle\Big),$$

where $\mathcal{S}_\nu$ is the soft-thresholding parameter. Note that in the argument to $\mathcal{S}_\nu$, an inner-product is calculated between the jth column of the model matrix ($\mathbf{x}_j$) and the residuals from the previous step t ($\mathbf{r}^t$). This is an inner product between two vectors of length $T/\Delta t$. Figure 8.2 shows that the computation time of an inner product calculation seems to increase exponentially, as the length of the vectors increase. Granted that if the vectors in the inner product have lengths in the order of 10^5 and 10^6 (which is the case for the $\Delta t = 10\,\text{ms}$ and the $\Delta t = 1\,\text{ms}$ discretized models in our analyses, as noted above), the calculation is in the order of $10^{-2}\,\text{s}$ (to be exact: $4.9 \times 10^{-2}\,\text{s}$ and $9.9 \times 10^{-2}\,\text{s}$, respectively). These computation times seem small, but relatively, the computation time for the inner product is twice as long for the $\Delta t = 1\,\text{ms}$ discretized model, compared to the $\Delta t = 10\,\text{ms}$ discretized model. This difference adds up for all model parameters β_j (in the full regression model fitted to neurons from session 2 in Section 7.2.2, there were 76 model parameters) and the number of iterations t that are needed (for the cyclical coordinate descent algorithm to converge). Furthermore, this difference is multiplied when estimating the lasso path for several values of the regularization parameter ν, performing the 10-fold cross-validation, and obtaining the adjusted p-values for B sample splits.

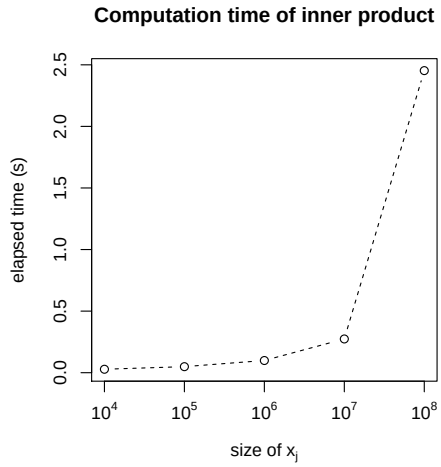

Figure 8.2: Computation time of $\langle \mathbf{x}_j, \mathbf{x}_k \rangle = \sum_i x_{ij} x_{ik}$, where $\mathbf{x}_j$ and $\mathbf{x}_k$ are two vectors of integers, where the integers were picked at random (with replacement) from the interval $[0, 999]$. These computation times were obtained on the same multi core server used to calculate the adjusted p-values in Section 7.4.1. (Parallelization was not used to calculate these computation times.)

There might also be other reasons that affect the computation time for fitting a lasso penalized regression model, and hence, also for calculating adjusted p-values. We've observed that (nested) coordinated descent algorithm (outlined in Section 4.4) takes longer to converge for neurons that have few spikes relative to the number of bins $n = T/\Delta t$, that is, neurons for which there are few observations where $y_i \neq 0$ (i is the bin number). It is not obvious how the lack of observations affect the computation time. In the extreme case, the (nested) coordinated descent algorithm does not converge at all, and we are not able to obtain any parameter estimates. This happened for neuron 16 from session 2, when estimating the network of neurons based data from lick left trials and lick right trials (neuron 16 is hence not shown in Figures 7.14 and 7.15). In the estimated networks of session 1 and 3, such neurons, that lead to non-convergence of the (nested) coordinated descent algorithm, are shown as black nodes in Figures B.8 and B.9. The non-convergence of this algorithm might be related to the so-called "rule of 10" mentioned in Section 4.1. For example, neuron 16 from session 2 has 169 observations that are non-zero. Recall that the network of neurons in Section 7.4.3 were fitted using the full regression model, which had a total of 76 parameters. That is, the number of observations (which is the least frequent class) is not greater than 10 times the number of parameters in the model, which violates the "rule of 10". Intuitively, this means that there isn't enough information in the observations of neuron 16 to fit the full regression model. Nevertheless, we would expect the lasso to converge in this situation. As mentioned in Section 4.1, our motivation for choosing to fit

a lasso restricted logistic regression model, rather than the (unrestricted) logistic regression model, was exactly because we believed that the lasso would be able to estimate the model parameters under such conditions where there are few observations. This reasoning was based on the ability of the lasso (or rather, regularized models in general) to adequately estimate the the unknown model parameters, even in a high dimensional setting $p \gg n$ (where p is the number of unknown parameters, and n is the sample size, as in Section 4.1).

We believe that the non-convergence of the lasso might be related to the effects of photostimulation (the experimenters ability to inactive targeted neurons, see (Li et al., 2015, Methods)). That is, there is a systematic effect that causes neurons to be inactive in certain parts of the session. This can clearly be seen in the raster plot of the 12 neurons from session 3, shown in Figure 7.1. The lasso did not converge for most of these neurons, as seen in the estimated network in Figure B.9 (note the number of black nodes). In Section 8.4 we suggest that photostimulation effects should be taken into account.

8.3 Neuroscientific Findings

Tuning Curves The tuning curves of Section 7.2 showed some characteristic neuronal activity, such as selectivity. That is, for the 16 neurons from session 2, we could for example observe a relative difference between lick left and lick right trials, as shown by the red and blue plots of Figure 7.6 (from stimulus only model) and of Figure 7.9 (from full regression model). However, it is interesting to note that all 16 neurons from session 2 have seemingly different tuning curves. That is, none of these neurons appear to be tuned to the same stimulus. This is also the case for the tuning curves for the 30 neurons from session 1 and the 12 neurons from session 3, shown in Figures B.6 and B.7 (both of which are estimated by the stimulus only regression model). This might be an actual phenomena, that the neurons in the ALM are all tuned to different stimuli. However, it is possible that the underlying data used to estimate the tuning curves might include noise. As mentioned in Section 1.2.4, the recording is done by measuring the change in activity in the extracellular (meaning "outside the cell") environment. That is, there is a chance that the observations of neuron j, $\mathbf{y}_j$, might actually include some activity of neuron k. We imagine that this chance increases if neurons j and k are located nearby, such that their activity is recorded on the same electrode/channel (for example neurons 2 and 15 from session 2, see Figure 1.8). If the underlying data used to estimate the tuning curves truly is noisy, the stimulus effects should be represented by a lower degree polynomial than degree $D = 5$, to restrict the shape of the resulting tuning curves.

In any case, whether the variety of tuning curves is because the underlying neurons actually are tuned to different stimuli, or whether the variety is due to noisy data, this can explain why the tuning curves retained their shape when they were estimated from the full regression model (Section 7.2.2) rather than the stimulus only regression model (Section 7.2.1). (Figure 7.8 shows a clear example, using data from neuron 1 from session 2, where the tuning curves retain their

shape). That is, if neurons j and k are tuned to completely different stimuli, their tuning curves will retain their shapes whether we include the connectivity between these neurons or not. On the other hand, if neurons j and k were tuned to the same stimuli, say for example that they were both active during the sample epoch only, then we can imagine that their tuning curves will change if we include their connectivity. This might explain why Stevenson et al. (2012) reported that their estimated tuning curves (based on a different data set than the `alm-1` data set considered in this thesis) changed as the authors included connectivity effects.

Sink/Source of FS Neurons in the Estimated Networks As noted in Section 7.1, there is a single neuron in session 2 that was classified as an FS neuron, as part of the pre-processing of the raw data (see Table 7.1). In the estimated network of neurons in Figures 7.13, 7.14 and 7.15 we observed that neuron 11 has the highest amount of connections. That is, it seems that neuron 11 is a central part of the estimated network of neurons, which is in accordance with the neurological characteristics of an FS neuron. As explained in Section 1.3.1, FS neuorons (also called intratelencephalic neurons) send signals to other neurons in the ALM. That is, FS neurons are a driving force, so to speak, behind the activity in the ALM. In the estimated network of session 3, shown in Figure B.9, we observe the same trend. In this figure, neuron 5 seems to be a central part of the network, which was classified as an FS neuron (Table 7.1). However, in the estimated network of session 1, it is not clear if the four FS neurons (neurons 1, 16, 24 and 27) have a central role. This is because there are too many connections in the estimated network of session 1, which makes it difficult to visually assess its structure.

There's a rule of thumb in neuroscience known as *Dale's principle*, which states that FS neurons act inhibitory (and pyramidial neurons act excitatory). That is, according to this Dale's principle, the connection from an FS neuron should be inhibitory. However, in the estimated networks of session 2 and session 3, we've generally observed the opposite case. For example, neuron 5 from session 3 (which is an FS neuron, see Table 7.1) has three outgoing connections (to neurons 1, 7 and 9), as seen in the estimated network shown in Figure B.9. All these three connections are excitatory, which contradicts Dale's principle. For neuron 11 from session 2, all four connections leaving neuron 11 (towards neurons 8, 10, 12 and 13) are also excitatory, seen in the estimated network based on data from lick right trials, shown in Figure 7.15. For the other two estimated networks based on data from `GoodTrials` and lick left trials (Figures 7.13 and 7.14, respectively) neuron 11 has outgoing connections that are both excitatory and inhibitory. Although, most of them are excitatory.

However, note that we classified a connection as either excitatory/inhibitory using only the shape of the connectivity curve from 3 to 15 ms, as explained in Section 7.4.2. If we considered the whole lag (in the time window from 0 to 160 ms), the resulting networks might have been different regarding the classification of connections.

8.4 Future Work

Resolution of the Lag Time Window Consider the resolution of the time window of the lag, which goes from 0 to 160 ms, which we chose by setting the number of bins we go back in time to $M = 160$ (in the $\Delta t = 1$ ms discretized model, as stated in Section 6.2.3). This time window did reveal some interesting oscillating behaviour for the history effects, as seen in Figure 7.10. However, we believe that this time window is too wide to capture any interesting connectivity effects. As indicated by the discussion in Section 1.2.3, regarding the conceptualized connectivity effect of Figure 1.4, the most interesting connectivity effects might be in the intervals 0 to 3 ms, 3 to 15 ms and 15 to 100 ms. It would be interesting to place the cosine bases (that represent the connectivity effects, as explained in Section 6.2.2) such that they specifically covered these intervals. This can be achieved by simply letting $M = 100$ (when $\Delta t = 1$ ms), which would lead to a time window that goes from 0 to 100 ms. Going further, to tailor the cosine bases to capture the effects in the three distinct intervals of Figure 1.4, the constants (such as a and c in (6.8)) and the placement of the cosine bases (in terms of ϕ_l in (6.8)) should experimented with, and changed. The values used in our analyses were directly copied from Pillow et al. (2008). Tailoring the cosine bases to capture the effects in the intervals 0 to 3 ms, 3 to 15 ms and 15 to 100 ms might be a worthwhile effort, which could lead to estimated connectivity effects that represent common, direct and indirect connections more precisely. From Pillow et al. (2008), we also copied the number of history bases $L_{\text{hist}} = 10$ and the number of connectivity bases $L_{\text{connect}} = 4$. These numbers should also be experimented with, to see how the resulting estimated history and connectivity effects change, and hence, also how the estimated networks change.

Reducing the Number of Connections in the Estimated Networks The estimated network of the 30 neurons from session 2, shown in Figure B.8, shows no clear underlying structure, since there are many connections (too many to visually discover any structure, at least). To some extent, this is also the case for the estimated network of the 16 neurons from session 2, shown in Figure 7.13 (based on data from `GoodTrials`). A possible solution to this is to enforce fewer connections in the estimated networks. This can be achieved by considering a stricter cut-off level when evaluation the adjusted p-values to test the significance of a connection. Recall that we used a cut-off level of 0.05 to test the significance of a connection in Section 7.4.1, hence, a stricter cut-off level could be 0.001. However, it is unusual to test for significance using a cut-off level below 0.05 when evaluating FWER adjusted p-values.

A second approach to reducing the number of connections in the estimated networks is to select the estimated coefficients from the lasso path at ν_{1se}, rather than at ν_{min}. This is because, generally, there are fewer non-zero coefficients at ν_{1se}, compared to ν_{min}, which will lead to fewer connections in the estimated networks. For example, recall that the full regression model fitted to observations from neuron 1 from session 2 had 55 non-zero coefficients at ν_{min} (see the lasso path shown in Figure 7.7). However, if we had chosen the estimated coefficients at ν_{1se} in this

example, we would've retained only 8 coefficients. It would be interesting to see the network of neurons that result by choosing a more sparse solution to the lasso regression models.

Expanding the Regression Models As noted in Section 7.1.1, we ignored the effects of photostimulation (the experimenters ability to inactive targeted neurons, see (Li et al., 2015, Methods)) and the relative location of the neurons as given by the implanted MEA used to record the activity of the neurons (see Sections 1.2.4 and 1.3.3). Trials where the experimenters used photostimulation are tagged in the `alm-1` data set, and can be accounted for in the regression models by, for example, using an index, or simply by removing the data from these trials. To account for the locations of the recordings, however, some type of distance metric between the neurons need to be developed. A simple distance metric would be the usual Euclidean distance in two dimensions, calculated between the electrodes/channels of Figure 1.8. That is, in addition to the connectivity effects as represented by the cosine bases, it would be interesting to include location effects between neurons, by adding additional covariates to the regression model. Ideally, this would lead to estimated networks where the connections are not affected by the location of the recordings. For example, we can imagine that taking location effects into account would not lead to a strong connection between neurons 2 and 15 from session 2 (which are recorded on the same electrode/channel, see Figure 1.8), as seen in the estimated networks of the 16 neurons from session 2 in Figures 7.13, 7.14 and 7.15.

Regarding the fitted regression models, there is a need to evaluate their goodness of fit, in a more extensive manner than done in this thesis. In Section 7.2.2, we mentioned briefly that the deviance seems to decrease, when fitting the full regression model compared to fitting the stimulus only model (see Table 7.3). If possible, perhaps we could have developed so-called analysis-of-deviance tables (as in Example 1 in Section 2.4.2), to compare the full and stimulus only regression models. This would've yielded a test on the significance of the stimulus effect, or the history and connectivity effect (depending on the order in which these effects are introduced into the regression models). It would be beneficial to investigate what types of goodness of fit measures there exist for the lasso penalized regression model. (As outlined in Section 4.6, regarding inference based on regularized regression models, we've only focused on calculating the adjusted p-values in order to assess the significance of the estimated parameters.)

Finally, our regression models used a set of parameters to represent an effect (stimulus, history and connectivity). For example, we represented stimulus effects by a set of five parameters $\{\gamma_{j1}, ..., \gamma_{j5}\}$ (where j denotes the response neuron), one for each of the polynomial bases. As explained in Section 4.1, the lasso (or rather, regularized models in general) sets a constraint on the size of each of the parameters in the model. However, to reflect that all five of the γ_{jd}'s in actuality represent a single effect (stimulus), we could instead constrained them as a group. This would lead to parameter estimates where all of the γ_{jd}'s are either set to zero or not. By reflecting this grouped structure in the constraint, we fit a so-called grouped lasso regression model. It would be interesting to fit this type of regression model, and

perhaps construct the estimated networks by including connectivity effects that were non-zero.

Analyzing the other 18 Mice in the `alm-1` Data Set The `R`-code developed in this thesis, given in Appendix C, is general enough to analyze data from the remaining 18 mice in the `alm-1` data set, with regards to estimating the tuning curves and the networks of neurons. However, keep in mind that there will be neurons in each session with few observations, such that the lasso does not converge, as noted above in Section 8.2. It would be interesting to see if the trends observed in our analyses based on data from mouse `ANM210861` (such as different tuning curves for each neuron, and FS neurons that are seemingly a central part of the estimated network), can also be found for the other mice in the `alm-1` data set.

8.5 Conclusion

In this thesis we considered neural data recorded from the ALM of an adult mouse brain, under experimental conditions where the mouse is given a certain stimulus. We observed that the neural activity is related to the stimulus. Thus, we fitted a logistic penalized regression model, where the activity of a neuron is set as the response variable, and the stimulus effect as a covariate. This resulted in tuning curves (which plot neural activity against stimulus) that were seemingly different for each neuron, indicating that each neuron was related to the stimulus in its own unique way. When we included history and connectivity effects into the regression model, these tuning curves retained their shape. By plotting the estimated history effects, we could observe characteristics such as refractory periods and neural oscillation. And by plotting the estimated connectivity effects, we could see that neurons affected each other in an excitatory and/or inhibitory manner. By evaluating the significance of the connections, we estimated networks of neurons. These networks show how the recorded neurons relate to each other, and hence, how the activity/information flows between them. We observed that so-called FS neurons had the highest amount of connections (compared to so-called pyramidial neurons), and hence, seem to be a central part of the underlying network. This is of relevance from a neuroscientific point of view, where central research questions concern the transformation of activity in the brain that eventually translates to, for example, body movements.

Bibliography

Buhlmann, P. and S. van de Geer (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications* (1st ed.). Springer Publishing Company, Incorporated.

Burnham, K. and D. Anderson (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer New York.

Casella, G. and R. Berger (2002). *Statistical Inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning.

Dezeure, R., P. Bühlmann, L. Meier, and N. Meinshausen (2015, 11). High-dimensional inference: Confidence intervals, p-values and r-software hdi. *Statist. Sci. 30*(4), 533–558.

Dobson, A. J. and A. G. Barnett (2008). *An Introduction to Generalized Linear Models, Third Edition*. Texts in Statistical Science. Boca Raton, FL: Chapman & Hall/CRC Press.

Friedman, J., T. Hastie, and R. Tibshirani (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software 33*(1), 1–22.

Goeman, J. J. and A. Solari (2014). Multiple hypothesis testing in genomics. *Statistics in Medicine 33*(11), 1946–1978.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The elements of statistical learning: data mining, inference and prediction* (2nd ed.). Springer.

Hastie, T., R. Tibshirani, and M. Wainwright (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis.

Kass, R. E., U. T. Eden, and E. N. Brown (2014). *Point Processes*, pp. 563–603. New York, NY: Springer New York.

Lamb, G. L. (1995). *Appendix E: Legendre Polynomials*, pp. 450–461. John Wiley & Sons, Inc.

Li, N., T.-W. Chen, Z. V. Guo, C. R. Gerfen, and K. Svoboda (2015, 03). A motor cortex circuit for motor planning and movement. *Nature 519*(7541), 51–56.

McCullagh, P. and J. Nelder (1989). *Generalized Linear Models, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis.

Meinshausen, N., L. Meier, and P. Bühlmann (2009). P-values for high-dimensional regression. *Journal of the American Statistical Association 104*(488), 1671–1681.

Nash, W. J., T. L. Sellers, S. R. Talbot, A. J. Cawthorn, and W. B. Ford (1994). UCI machine learning repository: Abalone data set. `http://archive.ics.uci.edu/ml/datasets/Abalone`.

Pillow, J. W., J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. J. Chichilnisky, and E. P. Simoncelli (2008, Aug). Spatio-temporal correlations and visual signaling in a complete neuronal population. *Nature 454*(7206), 995–999.

Rodríguez, G. (2007). Lecture notes on generalized linear models. `http://data.princeton.edu/wws509/notes/`.

Schroter, M., O. Paulsen, and E. T. Bullmore (2017, 03). Micro-connectomics: probing the organization of neuronal networks at the cellular scale. *Nat Rev Neurosci 18*(3), 131–146.

Stevenson, I. H., B. M. London, E. R. Oby, N. Sachs, J. Reimer, B. Englitz, S. V. David, S. Shamma, T. J. Blanche, K. Mizuseki, A. Zandvakili, N. G. Hatsopoulos, L. E. Miller, and K. P. Kording (2012, November). Functional Connectivity and Tuning Curves in Populations of Simultaneously Recorded Neurons. *PLoS Computational Biology 8*(11), e1002775.

Süli, E. and D. Mayers (2003). *An Introduction to Numerical Analysis*. Cambridge University Press.

Veierød, M., P. Laake, and S. Lydersen (2012). *Medical Statistics: In Clinical and Epidemiological Research*. Gyldendal akademisk.

Wood, S. N. (2006). *Generalized additive models: an introduction with R*. Texts in Statistical Science. Chapman Hall/CRC.

Appendix A

A.1 Summary of the `alm-1` Data Set

Mouse tag	Session tag	Number of trials	Number of neurons
ANM210861	20130701	423	30
	20130702	384	16
	20130703	483	12
ANM210862	20130626	292	25
	20130627	299	21
	20130628	380	7
ANM210863	20130626	350	27
	20130627	319	29
	20130628	362	34
ANM214427	20130805	320	12
	20130806	243	3
	20130807	550	5
	20130808	372	6
ANM214429	20130805	467	9
	20130806	575	10
	20130807	524	14
	20130808	485	15
ANM214430	20130820	333	10
	20130821	375	18
	20130822	343	7
	20130823	194	7
ANM218453	20131014	327	16
	20131015	306	10
	20131016	307	13
	20131017	289	8
	20131018	337	7
ANM218457	20131003	300	21
	20131004	289	23
	20131005	344	16

Mouse tag	Session tag	Number of trials	Number of neurons
	20131006	403	15
	20131007	297	23
	20131008	273	20
ANM218693	20131203	366	14
	20131204	325	20
	20131205	331	25
ANM219030	20130829	308	19
	20130830	344	6
	20130831	391	10
	20130901	270	13
	20130903	363	9
ANM219031	20131021	396	22
	20131022	316	21
	20131023	318	15
	20131024	145	11
	20131025	312	24
ANM219033	20131116	216	19
	20131117	214	6
	20131118	309	11
	20131119	360	6
	20131120	314	2
	20131121	403	3
	20131122	253	10
ANM219036	20131116	275	16
	20131117	303	18
	20131118	208	10
	20131119	233	13
	20131120	217	23
	20131121	319	12
	20131122	285	5
ANM219037	20131116	292	11
	20131117	357	17
	20131118	315	21
	20131119	363	17
	20131120	319	11
	20131121	312	11
	20131122	251	12
ANM219038	20131021	328	25
	20131022	318	22
	20131023	313	22
	20131024	321	9
	20131025	246	11
	20131026	275	11
	20131027	294	9

Mouse tag	Session tag	Number of trials	Number of neurons
	20131028	325	10
ANM219047	20130919	343	17
	20130920	311	12
	20130921	366	13
	20130922	376	13
	20130923	294	5
ANM219048	20130919	414	26
	20130920	406	30
	20130921	365	22
	20130922	395	20
	20130923	398	29
	20130924	350	14
	20130925	412	7
ANM219253	20140117	337	10
	20140118a	150	9
	20140118b	149	16
	20140119a	153	5
	20140119b	144	5
	20140120a	128	9
	20140120b	191	12
	20140121a	115	5
	20140121b	149	7
	20140122	295	15
ANM221977	20140115	337	8
	20140116	415	13
	20140118	178	15

Table A.1: Summary of the `alm-1` data set.

Appendix B

Figures

B.1 Raster Plots

The following raster plots correspond to the raster plots in Figure 7.1. The difference is that these plots also show activity from only correct lick left and correct lick right trials.

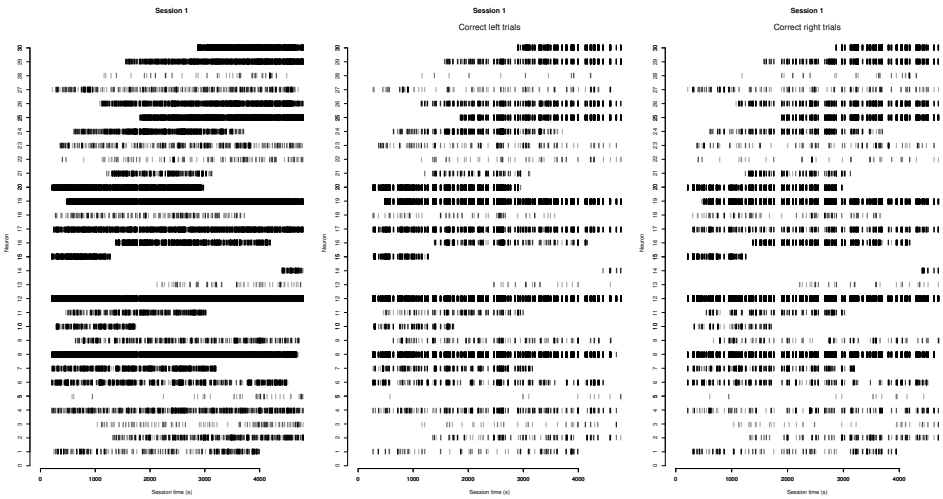

Figure B.1: Session `20130701`, mouse `ANM210861`. Raster plots of the activity of neurons, based on `GoodTrials`, correct lick left trials, and correct lick right trials, respectively.

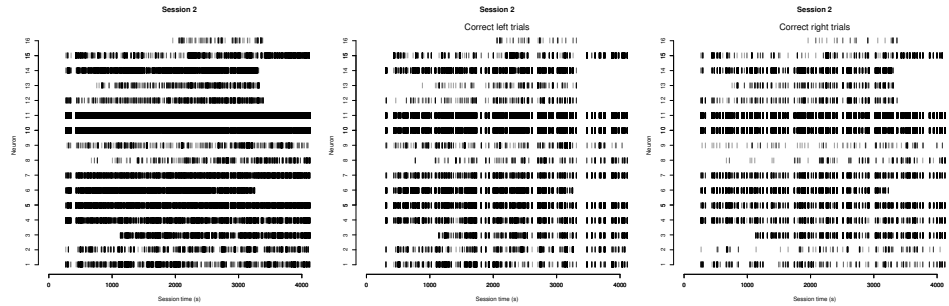

Figure B.2: Session `20130702`, mouse `ANM210861`. Raster plots of the activity of neurons, based on `GoodTrials`, correct lick left trials, and correct lick right trials, respectively.

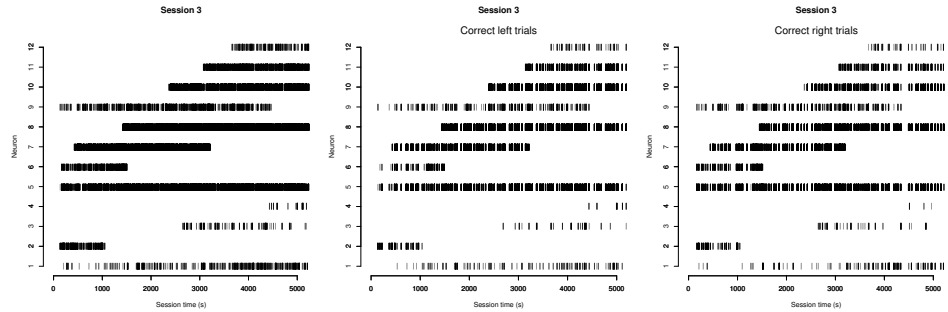

Figure B.3: Session `20130703`, mouse `ANM210861`. Raster plots of the activity of neurons, based on `GoodTrials`, correct lick left trials, and correct lick right trials, respectively.

B.2 Approximated Firing Rate

The following figures are bases on data from session `20130701` and `20130702` (called session 1 and 3 in Section 7.1) from mouse `ANM210861`.

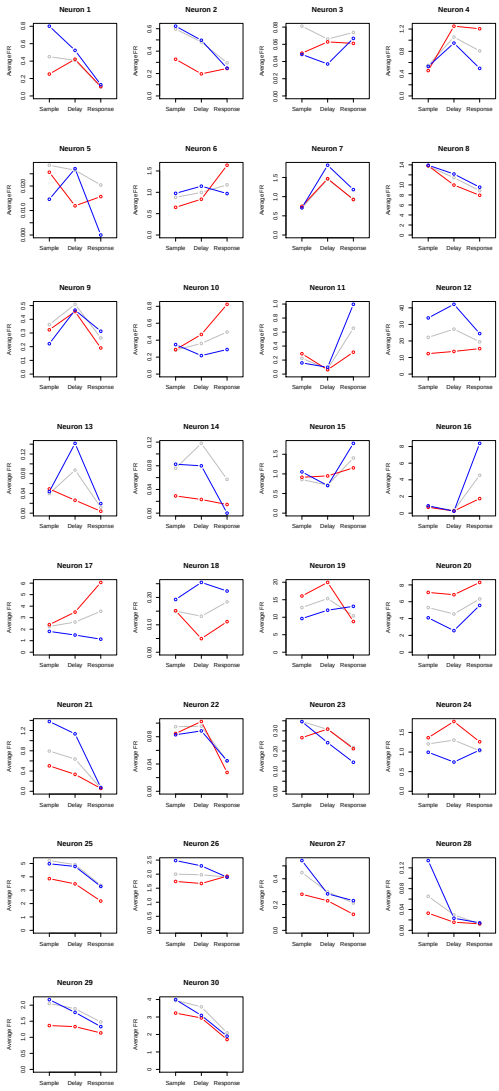

Figure B.4: The empirically approximated firing rate $\widehat{FR}_j$ (6.9), for the 30 neurons are from session 1.

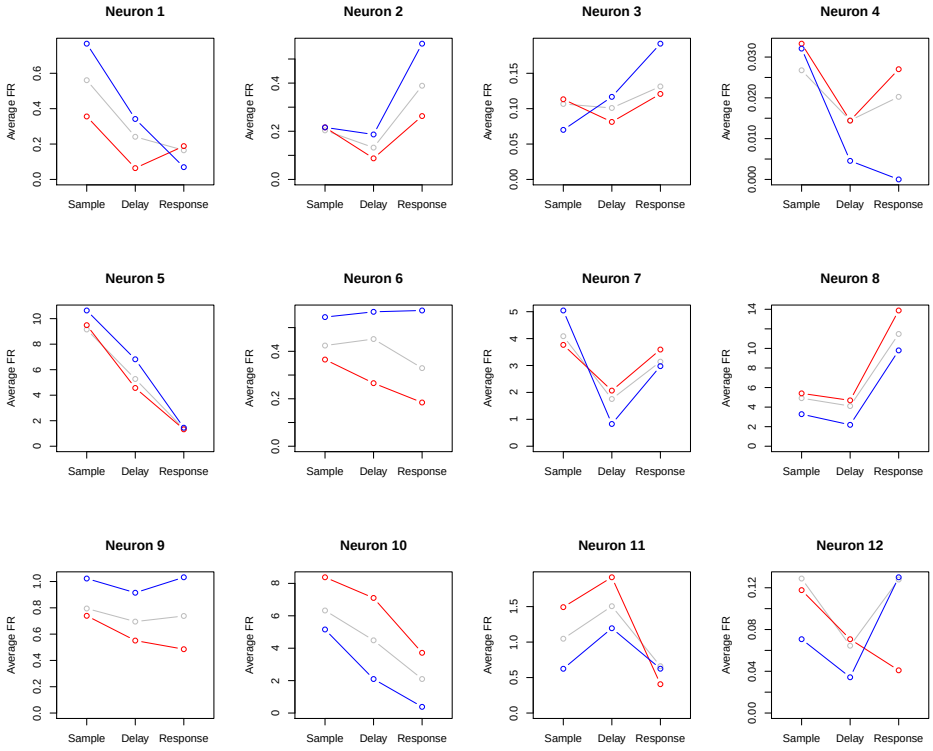

Figure B.5: The empirically approximated firing rate $\widehat{FR}_j$ (6.9), for the 12 neurons are from session 3.

B.3 Tuning Curves

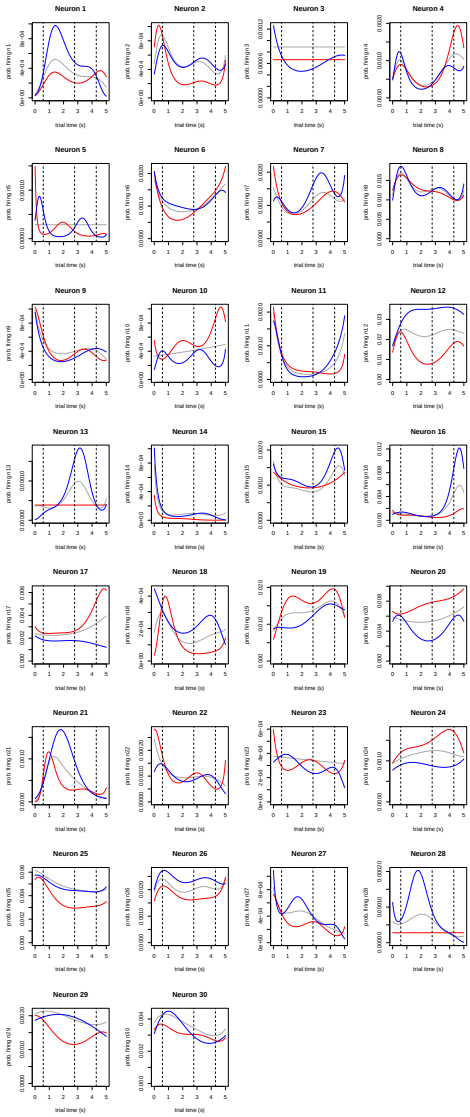

Figure B.6: Tuning curves for the 30 neurons from session 1. The parameters are estimated using the lasso with only stimulus effects (7.1), and choosing the values at $\nu_{\min}$. The gray curves are estimated using data from `GoodTrials`, and the red and blue curves from correct lick left and right trials, respectively. The dashed vertical lines are the start of the sample, delay and response epochs, respectively.

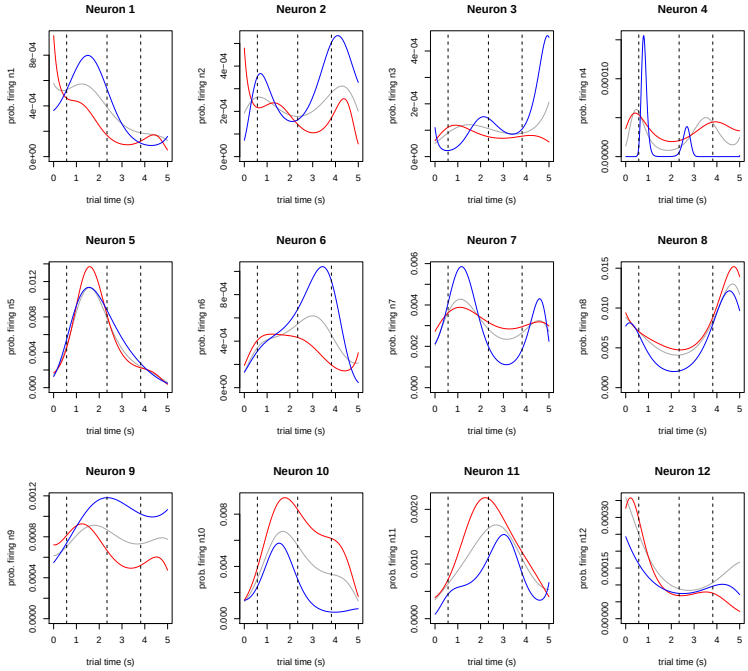

Figure B.7: Tuning curves for the 12 neurons from session 3. The parameters are estimated using the lasso with only stimulus effects (7.1), and choosing the values at $\nu_{\min}$. The gray curves are estimated using data from `GoodTrials`, and the red and blue curves from correct lick left and right trials, respectively. The dashed vertical lines are the start of the sample, delay and response epochs, respectively.

B.4 Network of Neurons

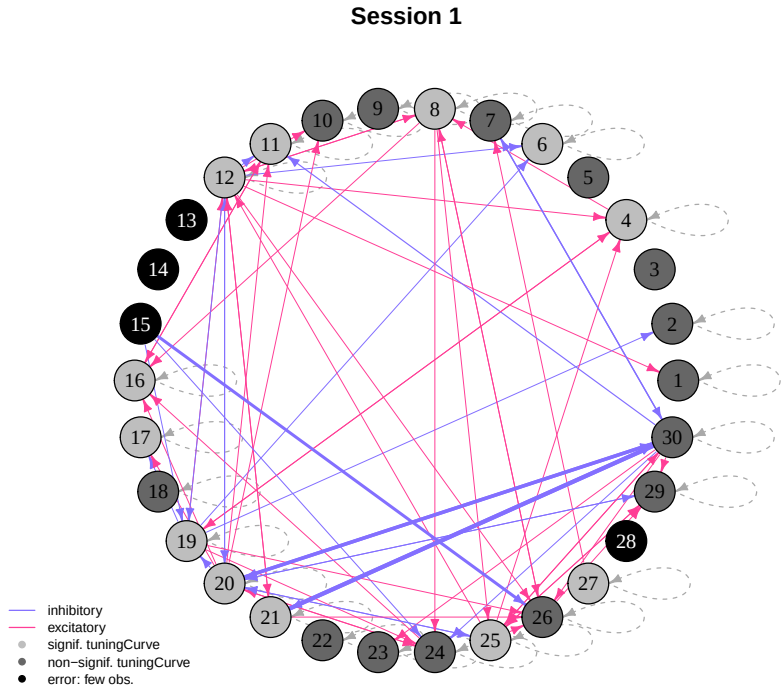

Figure B.8: Estimated network of the 30 neurons in session 1, using data from `GoodTrials`.

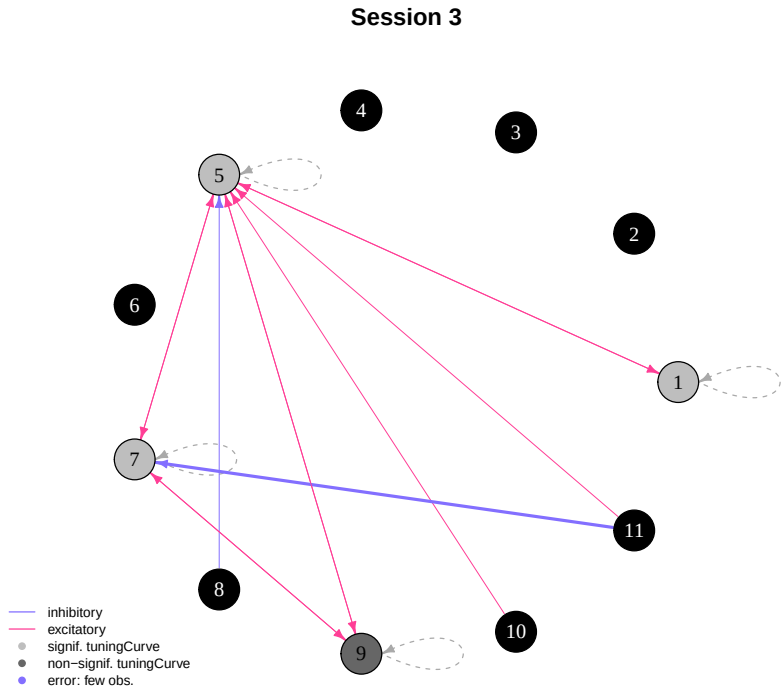

Figure B.9: Estimated network of the 12 neurons in session 3, using data from `GoodTrials`.

Appendix C

R-code

C.1 Cosine Bases

The following R-code is replicated from the Matlab-code that accompanied Pillow et al. (2008), which can be downloaded at `http://pillowlab.princeton.edu/code_GLM.html`.

```
getBasis = function(nBases,binSize){
  b = binSize*nBases # IN THESIS TEXT: c <- b
  peaks = c(binSize,binSize*10*nBases)

  # nonlinearity for stretching x axis (and its inverse)
  nlin = function(x){log(x+1e-20)}
  invnl = function(x){exp(x)-1e-20}

  # Generate basis of raised cosines
  yrange = nlin(peaks+b)
  db = diff(yrange)/(nBases-1)
  centers = seq(yrange[1],yrange[2],db) # IN THESIS TEXT: phi_j <- pi*
      c[j] / (2*db)
  maxt = invnl(yrange[2]+2*db)-b
  iht = seq(binSize,maxt,binSize)
  nt = length(iht)

  raisedCosineBasis = function(x,c,dc){
    (cos(max(-pi,min(pi,(x-c)*pi/dc/2)))+1)/2 # IN THESIS TEXT: a <-
        pi / (2*db)
  }

  ihbasis = matrix(NA,nrow = nt,ncol = nBases)
  for(i in seq(1,nt)){
    for(j in seq(1,length(centers))){
      ihbasis[i,j] = raisedCosineBasis(nlin(iht+b)[i],centers[j],db)
    }
  }


  # orthogonal bases
  library(pracma)
```

```
  ihbas = orth(ihbasis)

  return(list(bas=ihbasis,bas_orth=ihbas,tau_N=maxt))
}
```

C.2 Model Matrix

The following code is used to construct the model matrices used to fit the full regression model (6.14).

```
library(R.matlab)
library(foreach)
library(doParallel)
library(parallel)
library(Matrix)

####################
# INPUT PARAMETERS
####################
mouseTag = "ANM210862"
sessionTag = "20130626"
trialType = c("goodTrials","leftTrials","rightTrials")[1]
binSize = c(0.01,0.001)[2] # should only choose 0.001 or 0.01 (see the
    function weightedSpikeData)
deg = 5 # degree of lickOnset polynomial
nBases_history = 10
nBases_connectivity = 4
####################

# read data
data = readMat(paste("/global/work/harisf/mette/data/data_structure_",
    mouseTag,"/data_structure_",mouseTag,"_",sessionTag,".mat",sep="")
    )
totalNumberOfNeurons = length(data$obj[[12]][[1]])

# get trials that should be analyzed
getTrials = function(trialType){
  if(trialType == "goodTrials"){
    trials.good = which(data$obj[[9]][[3]][[4]][[1]] == 1) # trials
        where mice are performing (should be tested)
    trials.photostimConfig = which(is.nan(data$obj
        [[9]][[3]][[5]][[1]])) # trials where photostimulation
        configuration is tested (should NOT be tested)
    trials.good = trials.good[!is.element(trials.good,trials.
        photostimConfig)] # trials where mice are performing, AND we'
        ve taken out trials where photostimulation configuration is
        tested
    trials = trials.good
  }
  if(trialType == "rightTrials"){
    trials_correctR = which(data$obj[[8]][1,] == 1)
    trials_correctR = trials_correctR[is.element(trials_correctR,
        trials.good)]
    trials = trials_correctR
  }
```

```
36    if(trialType == "leftTrials"){
37      trials_correctL = which(data$obj[[8]][2,] == 1)
38      trials_correctL = trials_correctL[is.element(trials_correctL,
            trials.good)]
39      trials = trials_correctL
40    }
41    return(trials)
42  }
43  trials = getTrials(trialType)
44
45  discretizeSpikeData = function(neuron,TRIALS,binSize){
46    eventTrials = data$obj[[12]][[3]][[neuron]][[1]][[3]]
47    eventTimes = data$obj[[12]][[3]][[neuron]][[1]][[2]]
48    lickOnset = mean(c(data$obj[[9]][[3]][[3]][[1]][TRIALS]),na.rm=TRUE)
49
50    timeInterval = seq(0,5,binSize) # each trial lasts 5 seconds (this
          is true in all 3 sessions of  mouse ANM210861, but is it true
          for all other mice?)
51
52    mat_j = matrix(NA,ncol=3,nrow=length(timeInterval)-1)
53
54    registerDoParallel(cores = detectCores()-1)
55    mat = foreach(trial_j = TRIALS,.combine = rbind) %dopar% {
56      trialStartTime_j = data$obj[[7]][1,trial_j]
57      eventTimes_j = eventTimes[which(eventTrials == trial_j)] -
            trialStartTime_j
58
59      mat_j[,1] = rep(trial_j,length(timeInterval)-1)
60      mat_j[,2] = timeInterval[2:length(timeInterval)]-lickOnset
61      mat_j[,3] = as.vector(table(cut(eventTimes_j,breaks=timeInterval))
            )
62
63      mat_j
64    }
65    stopImplicitCluster()
66
67    colnames(mat) = c("trialId","lickOnset",paste("spikeCountj",neuron,
          sep=""))
68    mat = as.data.frame(mat)
69    return(mat)
70  }
71
72  discretizeAndAlignSpikeData = function(mainNeuron,TRIALS,binSize){
73    spikeData_mainNeuron = discretizeSpikeData(mainNeuron,TRIALS,binSize
          )
74
75    otherNeurons = setdiff(seq(1,totalNumberOfNeurons),mainNeuron)
76
77    timeInterval = seq(0,5,binSize)
78
79    registerDoParallel(cores = detectCores()-1)
80    spikeData_otherNeurons = foreach(i = seq(1,totalNumberOfNeurons-1),.
          combine = cbind) %dopar% {
81      neuron = otherNeurons[i]
82      eventTimes = data$obj[[12]][[3]][[neuron]][[1]][[2]]
83      eventTrials = data$obj[[12]][[3]][[neuron]][[1]][[3]]
84
```

```
85      registerDoParallel(cores = detectCores()-1)
86      spikesInTrial = foreach(trial_j = unique(spikeData_mainNeuron$
            trialId), .combine = list) %dopar% {
87        trialStartTime_j = data$obj[[7]][1, trial_j]
88        eventTimes_j = eventTimes[which(eventTrials == trial_j)] -
              trialStartTime_j
89        spikevector = as.vector(table(cut(eventTimes_j, breaks=
              timeInterval)))
90      }
91      stopImplicitCluster()
92
93      spikeData_neuron = unlist(spikesInTrial)
94    }
95    stopImplicitCluster()
96
97    colnamesTxt = NULL
98    for(i in seq(1,totalNumberOfNeurons-1))
99      colnamesTxt = c(colnamesTxt, paste("spikeCountj",otherNeurons[i],
            sep = ""))
100   colnames(spikeData_otherNeurons) = colnamesTxt
101
102   return(as.data.frame(cbind(spikeData_mainNeuron, spikeData_
          otherNeurons)))
103
104 }
105
106 getBasis = function(nBases, binSize){
107   b = binSize*nBases
108   peaks = c(binSize, binSize*10*nBases)
109
110   # nonlinearity for stretching x axis (and its inverse)
111   nlin = function(x){log(x+1e-20)}
112   invnl = function(x){exp(x)-1e-20}
113
114   # Generate basis of raised cosines
115   yrange = nlin(peaks+b)
116   db = diff(yrange)/(nBases-1)
117   centers = seq(yrange[1], yrange[2], db)
118   maxt = invnl(yrange[2]+2*db)-b
119   iht = seq(binSize, maxt, binSize)
120   nt = length(iht)
121
122   raisedCosineBasis = function(x,c,dc){
123     (cos(max(-pi, min(pi, (x-c)*pi/dc/2)))+1)/2
124   }
125
126   ihbasis = matrix(NA, nrow = nt, ncol = nBases)
127   for(i in seq(1,nt)){
128     for(j in seq(1,length(centers))){
129       ihbasis[i,j] = raisedCosineBasis(nlin(iht+b)[i], centers[j], db)
130     }
131   }
132
133   #matplot(ihbasis, type="b", pch=seq(1,5))
134   # for plotting model coefficients
135   lags = invnl(centers)-b
136
```

```
137   library(pracma)
138   ihbas = orth(ihbasis) # orthogonal bases
139
140   return(list(bas=ihbasis,bas_orth=ihbas,lags=lags,tau_N=maxt))
141 }
142
143 getMedianBasis = function(nBases,binSize_original,binSize_median){
144   bas_original = getBasis(nBases,binSize = binSize_original)
145
146   pointsToCombine = binSize_median / binSize_original
147   totalCombinedPoints = floor(dim(bas_original$bas)[1] /
          pointsToCombine)
148
149   indexes = list()
150   for(i in seq(0,totalCombinedPoints - 1))
151     indexes = c(indexes,list(seq(1,pointsToCombine) + pointsToCombine
          * i))
152
153   binSize_median = list(bas = matrix(NA,ncol = dim(bas_original$bas)
          [2],nrow = length(indexes)),
154                         bas_orth = matrix(NA,ncol = dim(bas_original$
                              bas)[2],nrow = length(indexes)),
155                         tau_N = bas_original$tau_N)
156
157   for(i in seq(1,length(indexes))){
158     binSize_median$bas[i,] = bas_original$bas[floor(median(indexes[[i
          ]])),]
159     binSize_median$bas_orth[i,] = bas_original$bas_orth[floor(median(
          indexes[[i]])),]
160   }
161
162   binSize_median
163 }
164
165 weightedSpikeData = function(spikeData,nBases_history,nBases_
      connectivity,binSize){
166   if(binSize == 0.001){
167     bas_hist = getBasis(nBases_history,binSize)$bas_orth
168     bas_connect = getBasis(nBases_connectivity,binSize)$bas_orth
169   }
170   if(binSize == 0.01){
171     bas_hist = getMedianBasis(nBases_history,binSize_original = 0.001,
          binSize_median = binSize)$bas_orth
172     bas_connect = getMedianBasis(nBases_connectivity,binSize_original
          = 0.001,binSize_median = binSize)$bas_orth
173   }
174
175   #history
176   registerDoParallel(cores = nBases_history)
177   basisWeights = foreach(k = seq(1,nBases_history),.combine = cbind) %
          dopar%{
178     if(binSize == 0.01)
179       bsWght = convolve(c(0,spikeData[,3]),rev(bas_hist[,k]),type="
              open")[2:dim(spikeData)[1]]
180     if(binSize == 0.001)
181       bsWght = convolve(c(0,spikeData[,3]),rev(bas_hist[1:160,k]),type
              ="open")[2:dim(spikeData)[1]] # ideally, should convolve
```

```
            using bas_hist[,k], but to shorten computation time we use
            the first 160 rows
182       bsWght
183     }
184     stopImplicitCluster()
185
186     txt=NULL
187     for(k in seq(1,nBases_history)){
188       txt = c(txt,paste(sub("spikeCount","",colnames(spikeData)[3]),".k"
              ,k,sep=""))
189     }
190     colnames(basisWeights) = txt
191
192     spikeData_basis = cbind(spikeData[-1,seq(1,3)],basisWeights) #when
          using convolution
193     colnames(spikeData_basis)[1:3] = colnames(spikeData)[1:3]
194     spikeData_basis = as.data.frame(spikeData_basis)
195
196     #connectivity
197     registerDoParallel(cores = detectCores() - 1)
198     spikeData_basis_connectivity = foreach(j = seq(4,dim(spikeData)[2])
          ,.combine = cbind) %dopar% {
199       registerDoParallel(cores = nBases_connectivity)
200       basisWeights = foreach(k = seq(1,nBases_connectivity),.combine =
            cbind) %dopar% {
201         if(binSize == 0.01)
202           bsWght = convolve(c(0,spikeData[,j]),rev(bas_connect[,k]),type
                ="open")[2:dim(spikeData)[1]]
203         if(binSize == 0.001)
204           bsWght = convolve(c(0,spikeData[,j]),rev(bas_connect[1:160,k])
                ,type="open")[2:dim(spikeData)[1]] ## ideally, should
                convolve using bas_connect[,k], but to shorten computation
                 time we use the first 160 rows
205         bsWght
206       }
207       stopImplicitCluster()
208
209       txt=NULL
210       for(k in seq(1,nBases_connectivity)){
211         txt = c(txt,paste(sub("spikeCount","",colnames(spikeData)[j]),".
              k",k,sep=""))
212       }
213       colnames(basisWeights) = txt
214
215       basisWeights
216     }
217     stopImplicitCluster()
218
219     spikeData_basis = cbind(spikeData_basis,spikeData_basis_connectivity
          )
220
221     return(spikeData_basis)
222
223   }
224
225   saveModelmatrixIn = paste("/global/work/harisf/mette/modelmatrix/",
        mouseTag,"/",sessionTag,"/",trialType,sep="")
```

```
226 if(!dir.exists(saveModelmatrixIn))
227   dir.create(saveModelmatrixIn,recursive = TRUE)
228
229 for(responseNeuron in seq(1,totalNumberOfNeurons)){
230   start = Sys.time()
231   spikeData <- discretizeAndAlignSpikeData(responseNeuron,trials,
          binSize)
232
233   spikeData_basis <- weightedSpikeData(spikeData,nBases_history,nBases
          _connectivity,binSize)
234
235   predMat = cbind(spikeData_basis[,-c(1,2,3)],poly(spikeData_basis
          [,2],degree = deg))
236   txtPolyname = NULL
237   for(i in seq(1,deg))
238     txtPolyname = c(txtPolyname,paste("poly(lickOnset)",i,sep = ""))
239   colnames(predMat)[seq(dim(predMat)[2]-deg+1,dim(predMat)[2])] =
          txtPolyname
240
241   # save data
242   evaluatedData = cbind(spikeData_basis[,3],predMat)
243   colnames(evaluatedData)[1] = paste("spike.j",responseNeuron,sep="")
244   tol = 1e-10
245   for(i in seq(2,dim(evaluatedData)[2])){
246     evaluatedData[which(abs(evaluatedData[,i]) < tol),i] = 0
247   }
248   evaluatedData = Matrix(as.matrix(evaluatedData),sparse = TRUE)
249
250   saveRDS(evaluatedData,paste(saveModelmatrixIn,"/n",responseNeuron,"_
          b",binSize*1000,"ms.rds",sep=""))
251   end = Sys.time() - start
252   cat("Model matrix saved for neuron ",responseNeuron,". (Time used: "
          ,as.numeric(end)," ",attr(end,"units"),").\n",sep="")
253 }
```

C.3 Lasso Penalized Regression Model

The following code uses the model matrices from the code in Section C.2, to actually fit the full regression model (6.14), using the `R`-function `cv.glmnet` from the `glmnet`-package.

```
1  library(glmnet)
2  library(doMC)
3  library(Matrix)
4
5  ####################
6  # INPUT PARAMETERS
7  ####################
8  mouseTag = "ANM210862"
9  sessionTag = "20130626"
10 trialType = c("goodTrials","leftTrials","rightTrials")[1]
11 binSize = c(0.01,0.001)[2] # normally, we fit lasso models only for
       binSize = 0.001
12 ####################
13
```

```
modelmatrixDirectory = paste("/global/work/harisf/mette/modelmatrix/",
    mouseTag,"/",sessionTag,"/",trialType,sep="")
modelmatrixFiles = list.files(modelmatrixDirectory,pattern=paste("_b",
    binSize*1000,"ms.rds",sep=""))

saveLassofitIn = paste("/global/work/harisf/mette/lassofit/",mouseTag,
    "/",sessionTag,"/",trialType,sep="")
if(!dir.exists(saveLassofitIn))
  dir.create(saveLassofitIn,recursive = TRUE)

for(fileName in modelmatrixFiles){
  modelMatrix = readRDS(file.path(modelmatrixDirectory, fileName))

  x = modelMatrix[,-1]
  y = modelMatrix[,1]

  y[which(y > 1)] = 1 # since family = "binomial". Certainly for
      binSize = 0.01 we get that some y > 1

  startTime = Sys.time()
  # fit cv.glmnet model
  registerDoMC(cores = 10)
  model_lasso_cv = cv.glmnet(x,y,
                             family = "binomial",alpha = 1, nfolds =
                                 10,
                             parallel = TRUE)
  endTime = Sys.time() - startTime
  saveRDS(model_lasso_cv,file.path(saveLassofitIn, fileName))
  cat("Lasso model fitted for neuron ",substr(fileName,start = 2,stop
      = regexpr("_",fileName)[1]-1),". (Time used: ",as.numeric(
      endTime),"",attr(endTime,"units"),").\n",sep="")
}
```

C.4 Multi-Sample Split

The following code uses the model matrices from the code in Section C.2 to calculate the FWER adjusted p-values for the parameters in the full regression model (6.14), using the `R`-function `multi.split` from the `hdi`-package.

```
library(Matrix)
library(parallel)
library(hdi)
library(glmnet)

####################
# INPUT PARAMETERS
####################
mouseTag = "ANM210862"
sessionTag = "20130626"
trialType = c("goodTrials","leftTrials","rightTrials")[1]
binSize = c(0.01,0.001)[1] # normally, we run multisplit only for
    binSize = 0.01
####################

```

```
15 modelmatrixDirectory = paste("/global/work/harisf/mette/modelmatrix/",
       mouseTag,"/",sessionTag,"/",trialType,sep="")
16 modelmatrixFiles = list.files(modelmatrixDirectory,pattern=paste("_b",
       binSize*1000,"ms.rds",sep=""))
17
18 saveMultisplitIn = paste("/global/work/harisf/mette/multisplit/",
       mouseTag,"/",sessionTag,"/",trialType,sep="")
19 if(!dir.exists(saveMultisplitIn))
20   dir.create(saveMultisplitIn,recursive = TRUE)
21
22 lasso.cv.lambda.min = function (x, y, nfolds = 10, grouped = nrow(x) >
       3 * nfolds,...){
23   suppressMessages(library(doMC))
24   registerDoMC(cores=10)
25   fit.cv <- cv.glmnet(x, y, nfolds = nfolds, grouped = grouped,
         parallel = TRUE,
26                       ...)
27   sel <- predict(fit.cv, type = "nonzero", s = "lambda.min")
28   sel[[1]]
29 }
30
31 glm.pval.x.as.matrix = function (x, y, family = "binomial", verbose =
       FALSE, ...){
32   fit.glm <- glm(y ~ as.matrix(x), family = family, ...)
33   fit.summary <- summary(fit.glm)
34   if (!fit.glm$converged & verbose) {
35     #print(fit.summary)
36     cat("␣glm.fit:␣algorithm␣did␣not␣converge.\n")
37   }
38   pval.sel <- coef(fit.summary)[-1, 4]
39   names(pval.sel) <- colnames(x)
40   pval.sel
41 }
42
43 for(fileName in modelmatrixFiles){
44   modelMatrix = readRDS(file.path(modelmatrixDirectory, fileName))
45
46   x = modelMatrix[,-1]
47   y = modelMatrix[,1]
48   y[which(y > 1)] = 1 # since family = "binomial". Certainly for
         binSize = 0.01 we get that some y > 1
49
50
51   startTime = Sys.time()
52   fit <- multi.split(x,y, ci = FALSE, B = 50,
53                      classical.fit = glm.pval.x.as.matrix, #args.
                           classical.fit = list(verbose = TRUE),
54                      model.selector = lasso.cv.lambda.min, args.model.
                           selector = list(family = "binomial"),
55                      parallel = TRUE, ncores = 10,
56                      return.selmodels = FALSE, verbose = FALSE)
57   endTime = Sys.time() - startTime
58   saveRDS(fit,file.path(saveMultisplitIn, fileName))
59   cat("Multisplit␣done␣for␣neuron␣",substr(fileName,start = 2,stop =
         regexpr("_",fileName)[1]-1),".␣(Time␣used:␣",as.numeric(endTime)
         ,"␣",attr(endTime,"units"),").␣\n",sep="")
60 }
```