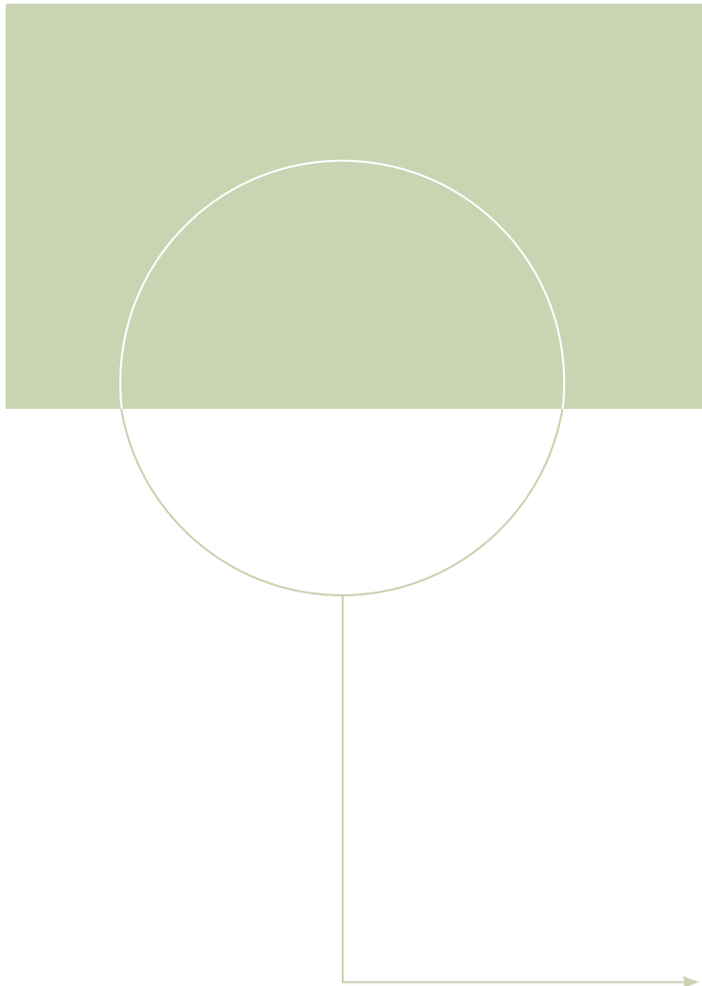

Master's thesis

2019

Dag Johnsrud Kristiansen

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

Master's thesis

Dag Johnsrud Kristiansen

Detecting Neuronal Activity with Lasso Penalized Logistic Regression

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Detecting Neuronal Activity with Lasso Penalized Logistic Regression

Dag Johnsrud Kristiansen

Master of Science in Physics and Mathematics
Submission date: June 2019
Supervisor: Mette Langaas, IMF
Co-supervisor: Benjamin Adric Dunn, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Summary

The brain is the most complex organ in animals, constantly transferring signals between its cellular components, also known as neurons. We will in this thesis investigate how such information flows, and try to establish certain connections. To investigate the matter, we fit a lasso penalized logistic regression model to relate the activity of one single neuron to all other neurons (that are included in the data set). It is known that a neurons activity is both dependent on its own historic activity – as well as the activity of other surrounding neurons. As such, our regression model will relate the activity of one neuron to its previous activity, as well as the previous activity of all nearby neurons using a special set of cosine bases. To estimate the underlying networks, we use both regression parameters regularized using different values for the hyperparameter, as well as family-wise error-rate corrected p-values.

The analyses are based on data from an experiment with cellular recordings of 12 neurons, where the lasso penalized logistic regression was fit using each neuron as response. We include theory on the formulation of a generalized linear model, specifically focusing on binomial variables, and how to extend it to an additive model. Regularization is also thoroughly explained, and how parameter estimates in the lasso is performed, before presenting theory regarding multiple hypothesis testing, focusing on the family-wise error rate and how this may be controlled at a pre-specified level through multi-sample splitting.

Preface

This thesis constitutes the course TMA4900: 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 I specialized in Industrial Mathematics. The work in this thesis was done through the spring semester of 2019. In advance, a project thesis was also completed during the fall semester of 2018, with focus of identifying neuronal connections. This gave a warm-up for the work further carried out in this thesis, as I had already encountered some neuroscientific concepts. The topic of this thesis is to detect neuronal activity with a lasso penalized logistic regression.

I would like to direct a huge thanks to my supervisor Mette Langaas at the Department of Mathematical Sciences for her guidance, motivation and great help during the process of writing this thesis – I am sure it would not be what it is without you. I would also like to thank my co-supervisor Benjamin Adric Dunn to happily answer all of the questions I had related to neuroscience. Lastly, I would like to thank my parents for all the motivation and guidance they have given me throughout my academic career. You always knew what to say to keep me going when facing what seemed like an unattainable problem, and for that I am forever grateful.

Dag Johnsrud Kristiansen
Trondheim, Norway
June, 2019

Table of Contents

Part I

Background and theory

Chapter 1

Introduction

We will begin this chapter with a problem description, explaining why the work in this thesis is interesting, both from a statistical and neuroscientific point of view.

1.1 Problem description

In this project we focus on analyzing functional connectivity for neurons in the brains of mice. Previous studies have shown that whenever one neuron is active, it may alter the way surrounding neurons behave, either in an inhibitory (less active) or excitatory (more active) manner. As such, we know that neurons communicate, and we will through this thesis try to find which neurons communicate for a given experiment. The main goal of our analysis is therefore to estimate the underlying network of neurons for said experiment. This can be done by modelling the activity of one neuron j using the remaining $k \neq j$ neurons as explanatory variables in the experiment, and systematically alternate the modelled focus neuron.

The modelling tool is statistical regression, more specifically a generalized linear model (GLM), including a lasso penalty to regularize coefficients. Following the work of Pillow et al. (2008), this regression model will relate the activity of neuron j to all neurons $k \neq j$ using a special set of cosine bases. These were developed as it is known that the activity of neuron j at time t_b is dependent on neuron j's own past activity at time t_a, with $t_a < t_b$. The regression models considered in this thesis will therefore relate activity of neuron j both with it's own historic activity, as well as the so called coupling effects of neurons $k \neq j$, as the nearby neurons $k \neq j$ at time t_a may also affect the behaviour of neuron j at

time t_b, also for $t_a < t_b$. With this is mind, we aim to fit lasso regularized logstic regression models.

To assess our underlying network of neurons, we will search for significant connections between neurons by introducing p-values. Statistically, this is interesting as it is rather difficult to construct p-values from a lasso model, as the distribution of estimated parameters remain unknown. To counter this, we use the concept of multi-sample splitting, which is proven by Dezeure et al. (2015) to control the family-wise error-rate (FWER) at a pre-specified level.

1.2 Outline

The continuation of this thesis will include the following. In Chapter 2 we will present key neuroscientific concepts, particularly how every neuron fires action potentials which may be used to construct spike trains. Chapter 3 and 5 will introduce theory regarding concepts that will be used in our analysis, including the generalized linear model, additive models and multiple hypothesis testing. In Chapter 4 we derive the lasso regularized logistic regression, before presenting our analysis in Chapter 6. Lastly, in Chapter 7 we present some discussion of the results from our data analysis, and a sketch for how one could continue this work, before concluding the thesis as a whole.

Chapter 2

Neuroscience

2.1 Introduction

This chapter will give a brief introduction to key concepts in neuroscience, specifically regarding the nervous system and action potentials. In Section 2.2 we focus on the nervous system as a whole, before continuing to gain insight in the general structure of a brain and its cellular components in Section 2.3, specifically the neuron. We end the chapter by explaining functions of the neuron, what it means for the neuron to be active, and how it may communicate with nearby neurons.

2.2 The nervous system

Most animals have a nervous system that controls the body through a continuous transmission of chemical and electrical signals between different parts of the body. In vertebrates, this nervous system consists of two main components, namely the Peripheral Nervous System (PNS) and the Central Nervous System (CNS). These two may be seen as subdivisions of the nervous system, where the central nervous system includes the brain and spinal cord, and the peripheral system contains everything else, that is, nerves and ganglia.

Even though they both are part of the nervous system, their tasks differ greatly. However, these two systems complement each other in a very organised and rigid manner. The peripheral nervous system connects the central nervous system to the organs and limbs of the body, while the central nervous system is the one to process the given information, and coordinates how the animal should react upon

changes.

While both of these systems are of great relevance, there is no secret that there is one part of the central nervous system that stands out – the brain. In neuroscience, the brain is often the matter of discussion, and the data set on which we will do our statistical analysis is the neuronal activity in the brain of mice. We will now give a brief summary of how the brain is structured and what it consists of.

2.3 Brain structure

In most animals, the brain is the most complex organ that may be found in the body. Furthermore, there are often similarities between brain structure among different animals, for example between humans and rodents. As rodents are the model organism for the experiment on which we are to do our statistical analysis, an overview of the brain structure and its main components will be given.

For humans, it is common to divide the brain dependent on location and in which order the part was developed. As such, one may speak of the forebrain (telencephalon and diencephalon), the midbrain (mesencephalon) and the hindbrain (metencephalon and myencephalon). A graphic view is seen in Figure 2.1. The order from which these divisions arise is that the hindbrain is the first to de-

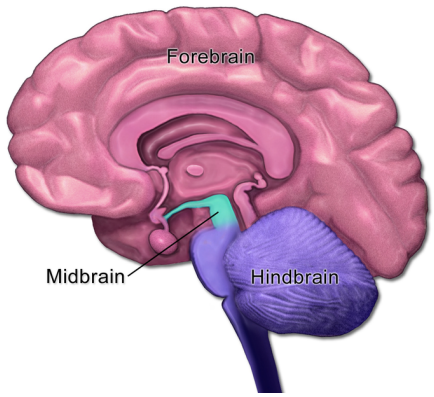

Figure (2.1) A visual representation of the three mentioned parts of the brain, where the pink area represents the forebrain, the cyan area represents the midbrain and the purple area represents the hindbrain.

Source: **https://commons.wikimedia.org/wiki/File:Brain_Anatomy_-_Mid-Fore-HindBrain.png** , licensed under CC BY-SA 4.0

velop, followed by the midbrain and lastly the forebrain. To delve even further, it should be mentioned that these are all successors from more primitive brain regions, where the hindbrain forms from the rhombencephalon, and the forebrain forms from the prosencephalon. The rhombencephalon (hindbrain) is what gives rise to the adult cerebellum, pons and adult medulla, whereas the prosencephalon (forebrain) is what gives rise to the rudiments of the cerebral cortex, hippocampus, thalamus and the basal forebrain, among others. For even more information about what these are and their function see Purves et al. (2018), Chapter 22.

There are often similarities when it comes to brains in different species, some are more similar than others. When it comes to our model organism, it is known that rodents in many ways are genetically similar to humans, though they also differ – both humans and rodents are mammals. What is very different is how our model organism has whiskers, which are as important for them as eyes are for humans. This is one of the many things that make their brain different than the human brain. Other than that, the shape is also different, mainly due to different shapes of their head compared to human head – and the human brain is significantly larger. Another curious difference is how the human brain is folded, whereas our model organism's brain is not. What the folding does for the human brain is to make a greater surface than an unfolded one. A human brain will thus function at a higher cognitive level than our model organism, both due to its size as well as the folding. A last interesting matter is how the brain develops in accordance with what is distinctive for each species. For humans, as an example, one may observe the language-part of the brain to be larger than most other animals, as we are dependant on a well developed language system.

Rodents are often used as a model organism as it found that they are advantageous over other model organisms. A few of the reasons why includes the fact that their genome is closely related to the human one (approximately 99%), there has been developed a decent molecular toolbox, and the fact that the animals require little space makes the use of rodents very cost efficient.

2.3.1 The cellular components of the nervous system

As neurons and glial cells have been mentioned, a brief explanation of what these are would be in order. Through the histological studies of Cajal, Golgi, and a host of successors, a consensus that the cells of the nervous system can be divided into two broad categories was made (Purves et al., 2018). These two types are commonly known as nerve cells (neurons) and supporting glial cells (neuroglia/glia). These are the main components in which the different parts of the body communicate with each other.

Nerve cells specialises in long-range communication using electrical signalling. This is in contrast to the glial cells, as they support electrical signalling, rather than generating them. Although this is an important function of the glial cells, perhaps even more important is how the glial cells may contribute to repair nerves that have been damaged. Known examples includes how glia can repair brain regions by acting as stem cells.

2.4 Action potential and neuronal connections

The *membrane potential* is the electric potential which arises when ions of opposite sign are separated across the plasma membrane, that is, the difference in electrical charge between the inside and outside of the cell. This means that the membrane potential equals the potential inside the cell minus the potential outside the cell. Any neuron has what is called the *resting potential* or *resting state*, which is the typical difference between the inner and outer of the cell. It has been found that most neurons have a resting potential between -60mV and -70mV.

However, a neuron is not always at rest. Neurons may be stimulated, which will give a change in the membrane potential. The stimuli provided are often found to be signals from other neurons, and such signals may be either *inhibitory* or *excitatory*. An inhibitory signal will decrease the membrane potential, and thus, in some sense, prevent a neuron from firing as the threshold potential is never reached. An excitatory signal does the exact opposite, and increases the membrane potential, so that the neuron which receives the signal is more likely to reach the threshold potential and fire. An action potential is therefore initiated when the change in membrane potential becomes significant, and reaches what is called the threshold potential. However, it is important to note that the action potential will only initiate if the change in membrane potential is large enough, otherwise the neuron will simply fall back to its resting potential.

The action potential is a primary electrical signal generated by excitable cells, and is important for neurons in the sense that they need fast and reliable signalling. It should also be mentioned that action potentials are important for muscle fibres as they initiate muscle fibre contraction. Briefly summarized, the action potential may thus be described as a transient change in the membrane potential from negative to positive. The nomenclature of an action potential may be seen in Figure 2.2.

It is action potentials like these that may give rise to what is known as *spike trains*. A spike train is a sequence of recorded times for when a neuron fired using an action potential. For this thesis, we will use a discretization of time, and spike trains will therefore be represented using 0/1-vectors. Such a vector will therefore,

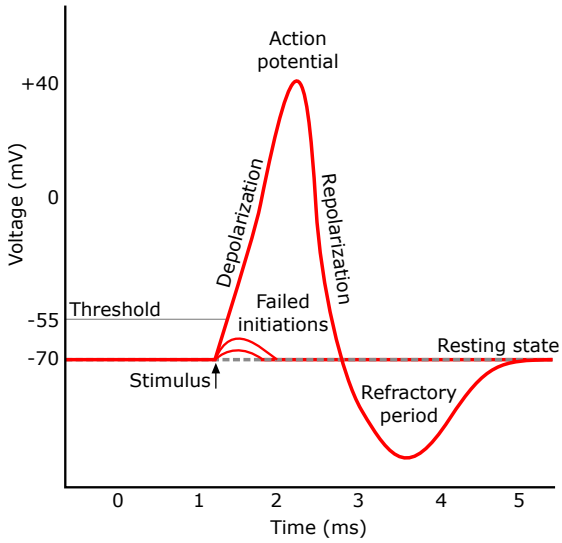

Figure (2.2) An illustrative figure of the different stages of a membrane potential, which (if it reaches the threshold) ultimately leads to an action potential (peak). The vertical axis shows the value of the membrane potential.

Source: **`https://commons.wikimedia.org/wiki/File:Action_potential.svg`** , licensed under CC BY-SA 3.0

as an example, be of the form [0 0 0 1 0 1 0 0 0 0 0 1]. Each number represents a time-stamp in milliseconds, and as this vector contains 12 numbers, this vector represents the neuronal activity for a neuron during 12 consecutive milliseconds. Examples of different spike trains and their respective 0/1-vectors may be seen in Figure 2.3.

As neurons exchange signals, there exists connections between neurons, and that these connections may reveal how information flows. As an example, consider two neurons, say n_1 and n_2. These may signal to one another using different forms of connectivity. If n_1 is to send a signal to n_2, it is not always the case that n_1's signal travels directly from its axon to neuron n_2's dendrite. It is often the case that the signal n_1 emits travels through a handful of other neurons before arriving at neuron n_2. The different forms of connectivity are therefore often represented using the terms *common connection, direct connection* and *indirect connection*, and the use of these thus say something about the time it takes from when a signal is sent to when it is received. This time is often called the time lag. A conceptual plot of the connectivity effect may be seen in Figure 2.4.

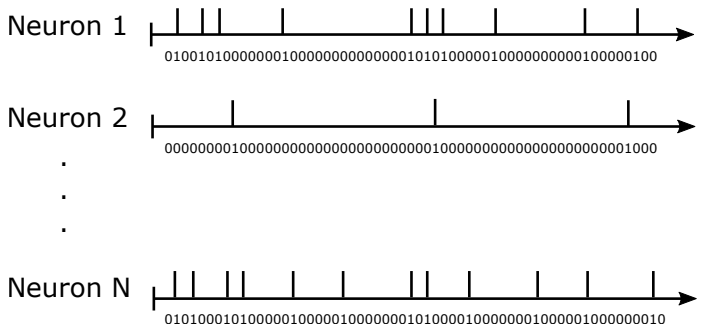

Figure (2.3) Three realisations of what a spike train could look like. The horizontal axis represents time, and the vertical lines represents at which times a neuron was active, that is, when a neuron spiked. For each neuron we have also included how the 0/1 vector would be for each neuron.

common connection We refer to a common connection if the time lag is in the interval from 0 to 3 ms. This means that the almost simultaneous firing of two neurons, say n_1 and n_2, may be explained by a third neuron, n_3, which made them both fire at almost the same time.

direct connection We refer to a direct connection if the time lag is in the time interval from 3 ms to 15 ms. This may be understood as direct path between two neurons, say n_1 and n_2. The signal that originated in neuron n_1 thus directly travelled to neuron n_2.

indirect connection We refer to a indirect connection if the time lag is in the interval from 15ms to 100 ms. This is commonly understood as a signal that may originate in neuron n_1, and travels through one ore more neurons, before arriving in neuron n_2. Notation was based on Fawad (2017), page 4 and 5.

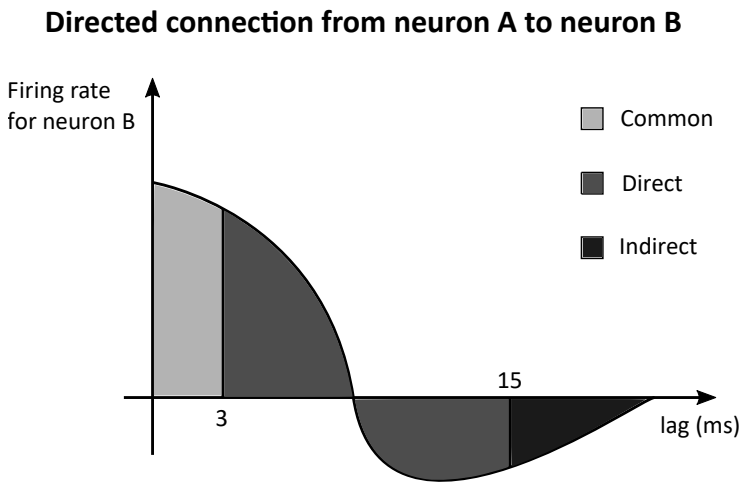

Figure (2.4) A conceptual plot of a so-called connectivity effect. The effect in the intervals 0 ms to 3 ms, 3 to 15 ms and 15 to 100ms is attributed to a common, direct and indirect connection from neuron A to neuron B, respectively. The y-axis is the so called firing rate of neuron B, which represents how often neuron B fires. The baseline represents neuron B's average background firing rate. The x-axis is the time since neuron A fired. The image is based on the one found on page 5 in Fawad (2017).

Chapter 3

Regression

In this chapter we introduce the generalized linear model (GLM) and it's framework, and how it is shaped in the case of a binary variable. This forms the logistic regression, and we continue to explain the concept of odds, and how one could interpret changes of the parameters in such a model. We also introduce additive models and basis functions, which is used by Pillow et al. (2008) to create the cosine bases. We end the chapter with theory of general hypothesis testing.

3.1 The GLM-framework

A generalized linear model (GLM) is an expansion of the ordinary linear regression, allowing the response variables to follow distributions other than the normal distribution. In general, the GLM generalizes the linear regression using three elements,

1. a random component, meaning a response variable with a probability distribution originating from the exponential family,
2. a systematic component, meaning a linear predictor η, and
3. a link function g which allows the mean of the probability distribution to be connected to the linear predictor. An alternative approach includes using a response function h, where $g = h^{-1}$.

By using these elements, we can build a unified framework where the maximum likelihood estimation can be written on a generalized form.

In linear regression, we observe n outcomes of a random response variable Y_i, denoted $y = (y_1, y_2, \ldots, y_n)$. By assuming Y_i has a probability function following the form of a univariate exponential family,

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

the probability distribution function can be expressed as (McCullagh and Nelder, 1989, p. 28)

$$f_{Y_i}(y_i; \theta_i, \phi) = \exp\left(\frac{y_i\theta_i - b(\theta)}{a(\phi)} + c(y_i, \phi)\right), \tag{3.1}$$

where $a(\phi), b(\theta_i)$ and $c(y_i, \phi)$ are known functions. In most cases, θ_i and ϕ can be considered as the location and scale parameters of the family. Specifically, θ_i is called the canonical parameter, and is what is of interest to us, whereas ϕ is called the nuisance parameter. Furthermore, it can be shown that there exists a link between the mean and variance of the distribution and the known functions a and b,

$$\begin{aligned} \mu_i &= \mathrm{E}(Y_i) = b'(\theta_i) \\ \mathrm{Var}(Y_i) &= b''(\theta_i)a(\phi). \end{aligned} \tag{3.2}$$

The linear predictor can be expressed as

$$\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}, \tag{3.3}$$

where $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_n)^T$, $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown parameters, and $\boldsymbol{X}$ is an $n \times p$ design matrix with rows x_i^T, and $p = k + 1$. p is thus comprised of k predictors and one intercept term. In multiple linear regression there exists two ways for estimating the unknown parameters $\boldsymbol{\beta}$; maximum likelihood and least squares. In a normal linear regression model, these will give the same estimate for $\boldsymbol{\beta}$, but for non-normal responses, the maximum likelihood estimation (MLE) is the favorable option.

At last, the link function g links the random and systematic component,

$$\eta_i = g(\mu_i). \tag{3.4}$$

When using a distribution function having canonical parameter θ_i, the canonical link function is the function expressing θ_i in terms of μ_i, that is,

$$\theta_i = g(\mu_i). \tag{3.5}$$

While other choices for the link function exists, the canonical is preferred because then some of the results for the GLM are simplified, and the log-likelihood is concave.

3.1.1 Example: Poisson distribution

Consider the case where the random variable follows a Poisson process, with the following probability mass function

$$f(y_i; \mu_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!},$$

for $\mu_i > 0$ and $y_i = 0, 1, 2, \ldots$. When rewritten to the form

$$f(y_i; \mu_i) = \exp\left\{y_i \log \mu_i - \mu_i - \log(y_i!)\right\},$$

one may observe that $\theta_i = \log \mu_i$, $b(\theta_i) = \exp(\theta_i)$ and $\phi = 1$, as well as the normalizing function $c(y_i, \phi) = 1/(\log(y!))$. As such, we may then conclude that the Poisson distribution is an exponential family.

3.1.2 Example: Exponential distribution

Assume $Y_1, Y_2, \ldots, Y_n$ are independent, random variables following an exponential distribution, with probability density function

$$f(y_i; \alpha_i) = \alpha_i \exp(-\alpha_i y_i),$$

for $y_i > 0$, $\alpha_i > 0$ and $i = 1, 2, \ldots, n$. When rewritten, it takes the form

$$f(y_i; \alpha_i) = \exp\left(\theta_i y_i - \ln(-1/\theta_i)\right),$$

and we observe that $\phi = 1$, $c(y_i, \phi) = 0$, $\theta_i = -\alpha_i$ and $b(\theta_i) = -\ln(\alpha_i) = -\ln(-\theta_i)$. As such, we conclude that the exponential distribution is an exponential family.

3.2 Binary variables – logistic regression

Consider a binary random variable (Dobson and Barnett, 2008, p. 123),

$$Z = \begin{cases} 1, & \text{if the outcome is a success,} \\ 0, & \text{if the outcome is a failure,} \end{cases}$$

with probabilities $P(Z = 1) = \pi$ and $P(Z = 0) = 1 - \pi$, which is the same as the Bernoulli distribution, $\mathcal{B}(\pi)$. In the case where we have m such independent random variables, $Z_1, Z_2, \ldots, Z_m$, and we assume all π_l equal, we may define

$$Y = \sum_{l=1}^{m} Z_l,$$

where Y represents the number of successes in m trials. The random variable Y will thus have a binomial distribution, $Y \sim \text{Bin}(m, \pi)$, with probability mass function

$$P(Y = y) = \binom{m}{y} \pi^y (1 - \pi)^{m-y}, \qquad y = 0, 1, \ldots, m.$$

Let us now consider the case of n such independent random variables, $Y_1, Y_2, \ldots, Y_n$. These are now distributed according to the binomial distribution, with parameters n_i and π_i, $Y_i \sim \text{Bin}(n_i, \pi_i)$. The distribution may then be rewritten on exponential form,

$$\begin{aligned} f_{Y_i}(y_i : \theta_i, \phi) &= \binom{n_i}{y_i} {\pi_i}^{y_i} (1 - \pi_i)^{n_i - y_i} \\ &= \exp\left[y_i \log\left(\frac{\pi_i}{1 - \pi_i} \right) + n_i \log(1 - \pi_i) + \log\binom{n_i}{y_i} \right]. \end{aligned} \tag{3.6}$$

By carefully comparing expression (3.1) and (3.6), one may observe that

$$\begin{aligned} \theta_i &= \log\left(\frac{\pi_i}{1 - \pi_i} \right) \\ a(\phi) &= \phi = 1, \text{ and} \\ b(\theta_i) &= n_i(\theta_i + \log(1 + \exp(-\theta_i))), \end{aligned}$$

where we have used that

$$\pi_i = \frac{1}{1 + \exp(-\theta_i)}. \tag{3.7}$$

Using the equations in (3.2), we may thus find the corresponding mean and variance,

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

As may be observed from the expected value, we have a term n_i included, alongside the probability π_i. To make use of the result from Equation (3.5) to find the canonical link function, we thus have to get rid of the extra term. According to Fahrmeir et al. (2013, p. 277), the original distribution may be divided by n_i, which results in a scaled binomial distribution,

$$\bar{Y}_i = \frac{Y_i}{n_i} \sim \frac{\text{Bin}(n_i, \pi_i)}{n_i}.$$

While this distribution no longer follows the form of an exponential family, the desired expected value is obtained,

$$\bar{\mu_i} = \mathrm{E}(\bar{Y_i}) = \pi_i.$$

Using the new, scaled binomial distribution, we may use the results of Equation (3.5), and the canonical link function becomes

$$g(\mu_i) = \log\left(\frac{\pi_i}{1-\pi_i}\right),$$

which is referred to as the logit-function. By combining Equation (3.3) and (3.4), the regression model for binary data becomes

$$\mathrm{logit}(\pi_i) = \boldsymbol{\eta_i} = \boldsymbol{x_i}^T\boldsymbol{\beta}.$$

3.2.1 The concept of odds

It is common to use the odds ratio for interpretation of the logit model. This comes from a non-linear change in the probability π_i when the value for a covariate changes from x_{i1} to $x_{i1}+1$.

As such, we introduce the concept of odds. The odds ratio is defined

$$\frac{P(Y_i=1)}{P(Y_i=0)} = \frac{\pi_i}{1-\pi_i},$$

and represents a multiplicative model following

$$\begin{aligned}\frac{\pi_i}{1-\pi_i} = \frac{P(Y_i=1)}{P(Y_i=0)} &= \frac{\frac{\exp(\eta_i)}{1+\exp(\eta_i)}}{1-\frac{\exp(\eta_i)}{1+\exp(\eta_i)}} \\ &= \exp(\eta_i) = \exp(\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik}) \\ &= \exp(\beta_0)\cdot\exp(\beta_1 x_{i1})\ldots\exp(\beta_k x_{ik}),\end{aligned}$$

when considering a logit-model of the form

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{ik}.$$

Consider the case where the covariate x_{i1} is increased by one, and all other covariates are kept fixed. In such a case, the odds is multiplied by $\exp(\beta_1)$,

$$\begin{aligned}\frac{P(Y_i=1|x_{i1}+1)}{P(Y_i=0|x_{i1}+1)} &= \exp(\beta_0)\cdot\exp(\beta_1(x_{i1}+1))\cdots\exp(\beta_k x_{ik}) \\ &= \exp(\beta_0)\cdot\exp(\beta_1 x_{i1})\exp(\beta_1)\cdots\exp(\beta_k x_{ik}) \\ &= \frac{P(Y_i=1|x_{i1})}{P(Y_i=0|x_{i1})}\cdot\exp(\beta_1).\end{aligned}$$

As such, whenever a covariate, say x_{i1}, increases by one, we have three cases for a change in the odds dependent on the value for β_1. These are

- A decrease in the odds when $\beta_1 < 0$, because $\exp(\beta_1) < 1$,
- No change in the odds when $\beta_1 = 0$, because $\exp(\beta_1) = 1$ and
- An increase in the odds when $\beta_1 > 0$, because $\exp(\beta_1) > 1$.

While the odds is the general ratio used to interpret logit models, it is possible to make a connection to the probability π, as it will behave similarly. In short, when the odds increase, the probability π will also increase, and if the odds decrease, the probability will decrease. However, the amount of change in the probability remains unknown. This may be visualised.

Consider a linear predictor $\eta_i = \beta_0 + \beta_1 x_{i1}$. If the covariate x_{i1} is increased by one, as explained above, the odds will be multiplied by $\exp(\beta_1)$, and thus increase if $\beta_1 > 0$, or decrease if $\beta_1 < 0$. The linear predictor will change to

$$\eta_i = \beta_0 + \beta_1(x_{i1} + 1) = \beta_0 + \beta_1 x_{i1} + \beta_1.$$

This means that the term β_1 will be added to the linear predictor, which will either increase or decrease dependent on the value of β_1. If one continues to plot the link function against the probability, an interesting result is found. The function turns out to be a strictly increasing function, meaning that when a value is added to η_i, the value for π_i must increase, and if the value for η_i decreases, so will the value for π_i. This is represented in Figure 3.1. This means that when the odds increases the probability will follow and also increase, whereas when the odds decreases, the probability will decrease. The change in probability is however dependent on where one would be on the appurtenant curve.

3.3 Parameter estimation

As mentioned, the linear predictor from Equation (3.3) contains unknown parameters $\boldsymbol{\beta}$. These may be estimated by maximizing the distributions corresponding likelihood function, where the obtained estimates are called maximum likelihood estimates (in short: MLEs), and are generally denoted $\hat{\boldsymbol{\beta}}$.

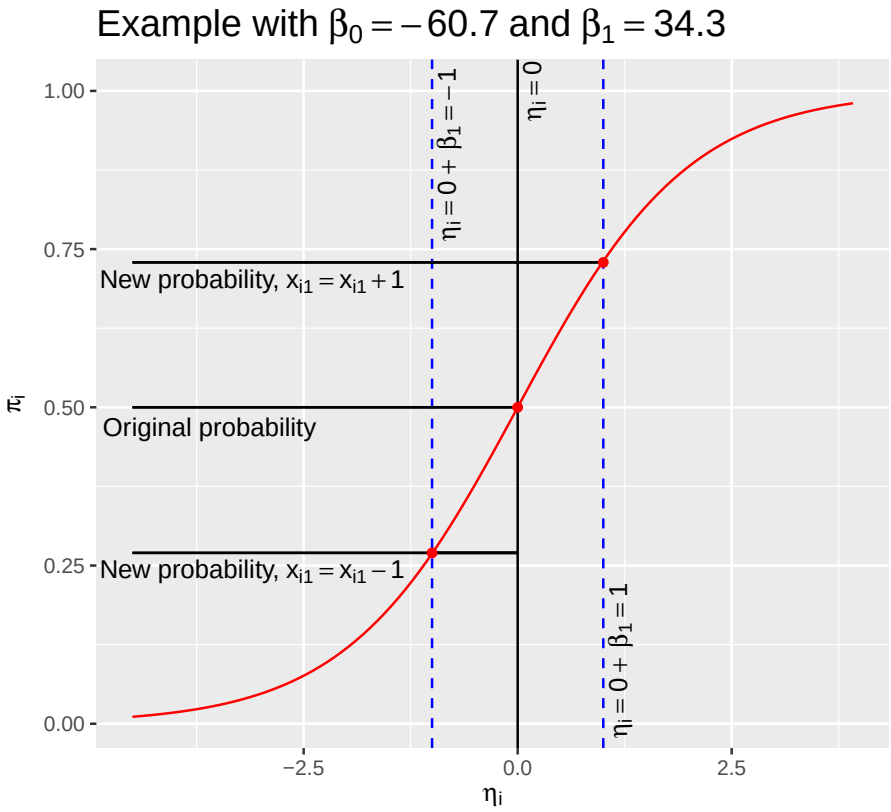

Figure (3.1) The link function plotted against the probability using an example with parameters $\beta_0 = -60.7$ and $\beta_1 = 34.3$. The original values for η and π are included ($\eta_i = 0$, $\pi_i = 0.5$), as well as the cases in which the odds increase ($x_{i1} \to x_{i1} + 1$, $\beta_1 > 0$, $\eta_i = 1$, $\pi_i = 0.729$), and decrease ($x_{i1} \to x_{i1} - 1$, $\beta_1 < 0$, $\eta_i = -1$, $\pi_i = 0.27$).

3.3.1 Introducing the log-likelihood function

For an exponential family of the form as expressed in Equation (3.1), the log-likelihood function for a single observation y_i is defined

$$\begin{aligned} \log L_i(\theta_i, \phi; y_i) &= \log f_{Y_i}(y_i; \theta_i, \phi) \\ &= \frac{\theta_i y_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi). \end{aligned} \tag{3.8}$$

When considering a set of independent observations $\boldsymbol{y} = (y_1, \ldots, y_n)^T$, the log-likelihood takes the form

$$\log L(\boldsymbol{\theta}, \phi; \boldsymbol{y}) = \sum_{i=1}^{n} \log L_i(y_i; \theta_i, \phi). \tag{3.9}$$

The fact that Equations (3.8) and (3.9) are functions of $\boldsymbol{\beta}$ is however not apparent. This result comes from the fact that there exists a connection between θ_i and μ_i, as expressed in (3.2). Furthermore, (3.3) links the linear predictor η_i and the unknown parameters $\boldsymbol{\beta}$.

Finally, a formal definition of the MLE $\hat{\boldsymbol{\beta}}$ may be expressed using the notation from Dobson and Barnett (2008, p. 12). By letting Ω denote the set of all possible values of the parameter vector $\boldsymbol{\beta}$; Ω is called the parameter space. The maximum likelihood estimator of β is the value $\boldsymbol{\beta}$ which maximizes the likelihood function, that is,

$$L(\hat{\boldsymbol{\beta}}; \boldsymbol{y}) \geq L(\boldsymbol{\beta}; \boldsymbol{y}) \quad \text{for all } \boldsymbol{\beta} \text{ in } \Omega.$$

3.4 Additive models

As an extension to the linear predictor, we will now present additive models, as it is not always the case that the effects of predictors are linear. As such, these models are needed when a straight line is not an accurate description for the regression.

As mentioned, the linear predictor is linear in the coefficients $\boldsymbol{\beta}$, which will be the case for everything we are to consider,

$$\boldsymbol{\eta_i} = \boldsymbol{x_i}^T \boldsymbol{\beta} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots.$$

This model has assumed that the expected response is a linear combination of some explanatory variables. However, the model may be extended to include higher order polynomial terms, for example the square of an explanatory variable, as linear effects are not always representable for realistic data. An additive model may therefore be introduced according to the notation of Hastie et al. (2009, p. 295),

$$\eta_i = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_k(x_k),$$

where $x_1, \ldots, x_k$ is the predictors and the functions $f_1(), \ldots, f_k()$ are unspecified smooth functions. While it is possible to let the data model the shape of the functions in a flexible manner, we will in this thesis limit ourselves to consider predefined functions.

3.4.1 Basis functions

Once again, by letting $x_1, \ldots, x_k$ be our predictors, the additive model takes the form

$$\eta_i = \beta_0 + \sum_{j=1}^{k} f_i(x_i).$$

Each of the functions f_j can furthermore be represented by M_j basis functions (Hastie et al., 2009, p. 139),

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

where now $h_{jm} : \mathbb{R} \to \mathbb{R}$ represents the m'th transformation of x_j. This is a rather elegant solution, as there are already existing examples of the basis functions h_{jm}. These includes

- $h_{jm}(x_j) = x_j$, $M_j = 1$. This will simply recover our original linear model.
- $h_{jm}(x_j) = \log(x_j)$ or $h_{jm}(x_j) = \sqrt{x_j}$. These are nonlinear transformations of a single input.
- $h_{jm}(x_j) = x_j^2$. Such polynomial terms may be used to achieve higher order Taylor expansions.
- $h_{jm}(x_j) = I(L_m < x_j < U_m)$. These are basis functions dividing the range of x_j into non-overlapping regions, so that the resulting model has a piecewise constant contribution from x_j.

3.4.2 Cosine bases

As we will perform our analysis on a dataset from neuroscience, we will continue to present the cosine basis functions. These basis functions are used to model history effects and coupling effects – as we need to both look at what have already happened as well as we consider how the spiking behaviour of other neurons affect the neuron being modelled.

Starting off we present the necessary notation. Consider we are observing neurons for a time interval of length $[0, T)$. The observed time interval could then be divided into n different bins, where each bin would have length $\Delta t = T/n$. Furthermore, we introduce a count process $N(t)$, which would count the amount of spikes that had occurred up to some time t. With this notation, we would the be able to count the number of spikes that had occurred in one specific bin, by looking at the difference of our counting process from the start till the end of the bin, $\Delta N_i = N(t_i) - N(t_{i-1})$, where $t_i = \Delta t \cdot i$. Furthermore, we let $\boldsymbol{y}_j = (y_{1j}, y_{2j}, \ldots, y_{nj})^T$ denote the observations in the n bins. Now, consider a linear predictor for the j'th neuron in bin t_i. We will consider two choices of L, for the history effects ($j = k$) and coupling effects ($j \neq k$) respectively. As such, the

linear predictor takes the form

$$\eta_j(t_i) = \alpha_{0j} + \sum_{k=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{10}\alpha_{jkl}b_l(t_i)y_k(t_i - t_m), \quad j = k, \tag{3.10}$$

$$\eta_j(t_i) = \alpha_{0j} + \sum_{k=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{4}\alpha_{jkl}b_l(t_i)y_k(t_i - t_m), \quad j \neq k \tag{3.11}$$

Here, N denotes the total number of neurons, whilst $M \geq 1$ represents the number of bins. Specifically, M represents how many steps we will go backwards in time, and we will in this thesis let $M = 161$, and thus only consider the former 161 ms. The α's represents the effect of any connection from neuron $k \neq j$ on neuron j, with α_{0j} as the background firing probability of neuron j (Fawad, 2017, p. 51). To separate the coupling effects $\boldsymbol{\alpha}_{jk}$ from time window and resolution, we introduced a set of basis functions $b_l(t_i)$, which denotes the l'th basis function which is evaluated at time t_i. These are our cosine basis functions.

Pillow bases

The basis functions $b_l(t_i)$ from equations (3.10) and (3.11) are basis functions developed by Pillow et al. (2008, Methods), and are defined

$$b_l(t) = \begin{cases} \frac{1}{2}\cos(a\log(t+c) - \phi_l) + \frac{1}{2}, & \text{if } a\log(t+c) \in [\phi_l - \pi, \phi_l + \pi] \\ 0, & \text{otherwise.} \end{cases} \tag{3.12}$$

In reality, these are raised cosine "bumps", where a and c are set constants, t represents the time after a spiking event, and the ϕ_l's can be seen as the placement of each bump. Furthermore, we will in this thesis work with $L_{\text{hist}} = 10$ ($j = k$ in Equation (3.10)) and $L_{\text{connect}} = 4$ ($j \neq k$ in Equation (3.11)), representing the cosine bases for history effects and coupling effects respectively. The bases are visually represented in the top of Figure 3.2. The bottom row shows an orthogonalized version of the bases, and we will use the orthogonal version.

3.5 Hypothesis testing

A hypothesis test is a statistical method for testing an assumption or declaration of properties based on one or more populations, where conclusions are generally drawn from a random sample from the population. To be able to perform such a hypothesis test, one has to state two hypotheses, the *null hypothesis*, often denoted

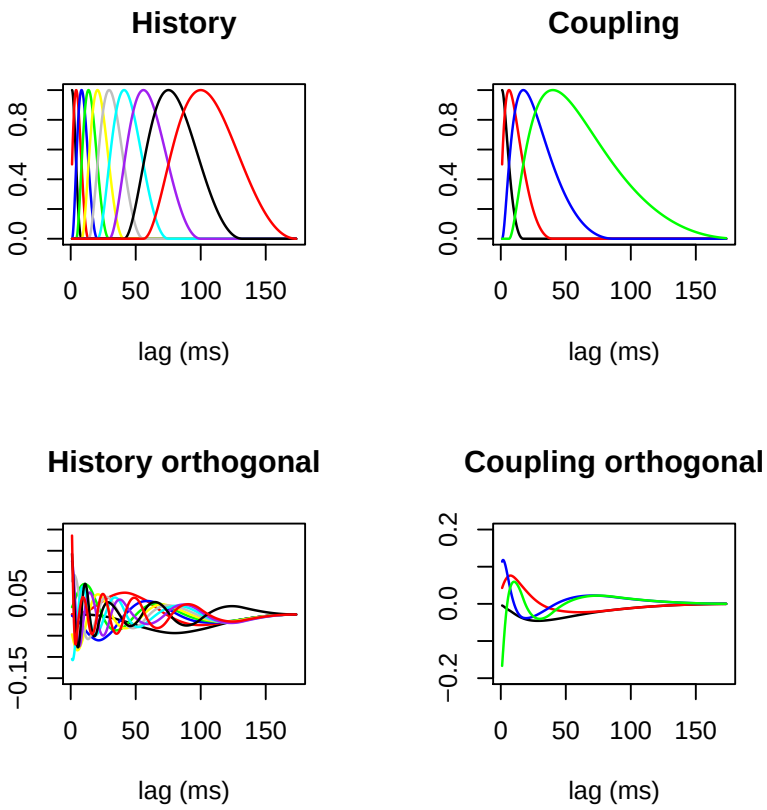

Figure (3.2) Pillow bases for history and coupling effects. Top row are the original bases, and the bottom row is the corresponding orthogonal bases.

H_0, and the *alternative hypothesis*, denoted H_1. With the hypotheses formulated, a statistician would then move forth and look at H_0, to examine whether we have the foundation to reject it or not. What is important to note is that the null- and alternative hypothesis has to be complementary.

When performing such a test, one needs a criterion to decide when to reject the null hypothesis or not. To do this we introduce a test statistic, which is a quantity derived from the sample. An important property of the test statistic is that it allows for the p-value to be calculated as it has a sampling distribution under the null hypothesis, which may be calculated either exactly, approximately or by simulation. The use of such a test statistic may thus guide us towards a decision to reject or not reject the null hypothesis and the test is now based on the chosen statistic. However, at times the result may be flawed, as such a hypothesis test

is rarely free of error. There exists two errors one could make when performing a hypothesis test, namely ***Type I error*** and ***Type II error***. Briefly summarised, a type I error occurs when one rejects H_0 even though it is the truth, and a type II error occurs when one does not reject H_0 even though it is wrong.

We will in this section introduce a frequently used test statistic for assessing the significance of regression parameters.

3.5.1 Wald test

Consider you are to test a hypothesis of the form

$$H_0 : \boldsymbol{\beta} = \beta_0 \quad \text{vs.} \quad H_1 : \boldsymbol{\beta} \neq \beta_0,$$

for a parameter vector $\boldsymbol{\beta}$ of dimension $p \times 1$, and some fixed $p \times 1$ values represented by β_0. Such hypotheses can be tested using the Wald test statistic, defined as

$$W = (\hat{\boldsymbol{\beta}} - \beta_0)^T \widehat{\text{Cov}}(\hat{\boldsymbol{\beta}})^{-1} (\hat{\boldsymbol{\beta}} - \beta_0) \tag{3.13}$$

based on notation from Dobson and Barnett (2008, p. 85). In (3.13), $\widehat{\text{Cov}}(\hat{\boldsymbol{\beta}})$ represents the estimated $p \times p$ variance-covariance matrix of estimated parameters. For large sample data, the asymptotic distribution of the Wald statistic is found to approximately follow a chi-squared distribution,

$$W \sim \chi^2(p),$$

where p represents the number of degrees of freedom. This comes from the fact that the maximum likelihood estimator $\hat{\boldsymbol{\beta}}$ approximately follows a normal distribution for large samples (Fahrmeir et al., 2013),

$$\hat{\boldsymbol{\beta}} \approx N_p \left(\boldsymbol{\beta}, I^{-1}(\boldsymbol{\beta}) \right),$$

where the mean equals the true parameter value, and the variance-covariance matrix is given by the inverse of the information matrix, defined by

$$\boldsymbol{I}(\boldsymbol{\beta}) = \boldsymbol{X}^T \mathrm{W}(\boldsymbol{\beta}) \boldsymbol{X}. \tag{3.14}$$

In (3.14), W is the $N \times N$ matrix with diagonal elements,

$$w_{ii} = \frac{1}{\text{Var}(Y_i)} \left(\frac{\partial \mu_i}{\partial \eta_i} \right)^2,$$

with $\partial\mu_i/\partial\eta_i$ evaluated at $\boldsymbol{\beta}$. Knowing this, the variance-covariance matrix may be approximately estimated using (3.14), as any consistent estimator may be used without changing the asymptotic distribution of the chi-square distribution,

$$\widehat{\text{Cov}}(\hat{\boldsymbol{\beta}}) = \boldsymbol{I}^{-1}(\hat{\boldsymbol{\beta}}) = (\boldsymbol{X}^T\widehat{\mathbf{W}}\boldsymbol{X})^{-1}.$$

The Wald test statistic now becomes, according to a GLM framework,

$$W = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0)^T\boldsymbol{X}^T\widehat{\mathbf{W}}\boldsymbol{X}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}_0). \tag{3.15}$$

We may thus use the Wald test statistic (3.15) to test the significance of different parameters, for example to test the significance of $\hat{\boldsymbol{\beta}}_j$ for the hypothesis $H_0 : \boldsymbol{\beta}_j = 0$ vs. $H_1 : \boldsymbol{\beta}_j \neq 0$. Whenever the parameter to be tested is a scalar, a common approach is to take the square root of the test statistic (Dobson and Barnett, 2008, p. 78),

$$z = \frac{\hat{\beta}_j}{\sqrt{\text{Var}(\hat{\beta}_j)}}.$$

The $z-$statistic is known to follow a standard normal distribution when the data sample is large.

Chapter 4

Regularization

In this chapter the GLM framework is extended to include a parameter for penalization, regularizing the logistic regression. Specifically, the lasso estimator is introduced, and we continue to discuss certain properties of the lasso, and the requirements for being able to solve such an optimization problem, including the Karush-Kuhn-Tucker conditions. We end by presenting solutions for the lasso problem.

4.1 Restating the optimization problem

One way of fitting a generalized linear model is based on utilizing the log-likelihood function, and maximize it. Mathematically, this is formulated

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{maximize}} \log L(\boldsymbol{\beta}; \boldsymbol{y}),$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T$ and $\boldsymbol{y} = (y_1, \ldots, y_N)^T$. Equivalently, based on the theory of Hastie et al. (2015, p. 30), one may minimize the negative log-likelihood. We will in addition add a penalty term to the negative log-likelihood,

$$\hat{\boldsymbol{\beta}}(\lambda) = \underset{\boldsymbol{\beta}}{\text{minimize}} \left\{ -\frac{1}{N} L(\boldsymbol{\beta}; \boldsymbol{y}) + \lambda \left\| \beta \right\|_v \right\}. \tag{4.1}$$

In (4.1), $\lambda \geq 0$ represents the regularization parameter, and $\left\| \cdot \right\|_v$ is the l_v-norm,

$$\left\| \beta \right\|_v = \sum_{j=1}^{p} |\beta_j|^v,$$

where $v \geq 0$ is any real number. The reasoning for the scaling parameter N equal to the sample size, is due the fact that it makes λ values comparable for different sample sizes (Hastie et al., 2015, p. 9). As we typically do not penalize the intercept β_0, expression (4.1) may be restated as

$$\hat{\boldsymbol{\beta}}(\lambda) = \underset{\boldsymbol{\beta}}{\text{minimize}} \left\{ -\frac{1}{N} \log L(\beta_0, \boldsymbol{\beta}; \boldsymbol{y}) + \lambda \left\| \boldsymbol{\beta} \right\|_v \right\}, \tag{4.2}$$

where now β_0 represents the intercept, while $\boldsymbol{\beta}$ includes the remaining p parameters, that is $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$.

4.2 The lasso estimator

In (3.3), we saw that it was common to state the regression problem as approximating the response variable using a linear combination of the predictors. When estimating the regression weights $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, a common approach is using the negative log likelihood.

However, there are alternative methods that could be considered as better, as of two reasons. First of all, one may discuss the prediction accuracy of the maximum likelihood estimation. Generally, this method is known to have low bias and large variance. As such, we could improve the prediction accuracy by either shrinking the regression coefficients, or even let some of them be set equal zero. This would mean that the bias would be increased, while the variance would decrease. Such a method may be better overall. Secondly, we may shed light on interpretation. If there are many predictors included in the model, there would often exist a subset of these predictors exhibiting the strongest effects, which we would like to identify. As such, we now introduce the lasso estimator.

In the lasso, model parameters are constrained using the ℓ_1-norm, that is, when $v = 1$ in (4.2). As the statistical properties of the lasso are more easily developed in the context of the usual linear regression model, we furthermore let the likelihood in (4.2) be from the normal distribution. As such, consider N independent observations $\boldsymbol{y} = (y_1, \ldots, y_N)$, with y_i being a realization of a random variable $Y_i \sim \mathcal{N}(\mu_i, \sigma^2)$. The log-likelihood follows to be

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

By substituting this expression into (4.2), and consider σ^2 to be fixed, the lasso estimator finds the solution $(\hat{\beta}_0, \hat{\boldsymbol{\beta}})$ to the optimization problem (Hastie et al., 2015,

p. 8)

$$\begin{aligned} \underset{\beta_0,\boldsymbol{\beta}}{\text{minimize}} \left\{ \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \right\} \\ \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq t, \end{aligned} \tag{4.3}$$

where the expression to be minimized is called the objective function, and the constraint is called the constraint function. The optimization problem may also be written using matrix notation,

$$\begin{aligned} \underset{\beta_0,\boldsymbol{\beta}}{\text{minimize}} \left\{ \frac{1}{2N} \left\| \boldsymbol{y} - \beta_0 \mathbf{1} - \boldsymbol{X}^T \boldsymbol{\beta} \right\|_2^2 \right\} \\ \text{subject to} \quad \|\boldsymbol{\beta}\|_1 \leq t. \end{aligned} \tag{4.4}$$

The matrix notation is introduced by letting $\boldsymbol{y} = (y_1, \ldots, y_N)$ be a N-vector of responses, $\mathbf{1}$ is simply a column vector of N ones, $\boldsymbol{X}$ is the $N \times p$ matrix where the i'th row contains $\boldsymbol{x}_i^T \in \mathbb{R}^p$, and $\|\cdot\|_2$ is the Euclidean norm on vectors.

By inspecting (4.3), it is easy to observe that a constraint term is introduced. This constraint is favorable as it shrinks some coefficients to zero. More specifically, the bound t acts as a kind of budget. It sets a limit for how much the absolute value of every parameter estimate may sum to, and thus shrinks some parameters to avoid the sum exceeding the limit. Whenever a parameter estimate is shrunk, this corresponds to a more constrained model, and we must therefore carefully pick a value for the budget t. Cross-validation is a frequently used method to find an optimal value for t.

Another key feature of the lasso is that it is common to make the predictors involved standardized,

$$\frac{1}{N} \sum_{i=1}^{N} x_{ij} = 0, \quad \text{and} \quad \frac{1}{N} \sum_{i=1}^{N} x_{ij}^2 = 1. \tag{4.5}$$

When standardized, one avoids any problems regarding the units (e.g., Celsius vs. Fahrenheit). However, if the features are already measured using the same units, standardization is not common practice. Furthermore, the outcome values are often also centered,

$$\frac{1}{N} \sum_{i=1}^{N} y_i = 0, \tag{4.6}$$

as this results in us being able to omit the intercept term β_0 in the regression.

The optimization problem (4.3) is known to be a convex problem. According to Boyd and Vandenberghe (2009, p. 7), a convex optimization problem is one in which the objective and constraint functions are convex, which means they satisfy the inequality

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y),$$

for all $x,\ y \in \mathbb{R}^n$ and all $\alpha,\ \beta \in \mathbb{R}$ with $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$. In the case of a convex objective function and a convex feasible region, there could only exist one optimal solution, that is, one globally optimal solution. Convexity thus simplifies calculation.

4.2.1 Lagrangian form – duality

We will now state the lasso problem on a different form, often referred to as the Lagrangian form. For this, we need to define the *Lagrangian*, L, associated with (4.4),

$$L(\boldsymbol{\beta}, \lambda) = \frac{1}{2N} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2 + \lambda \left\| \boldsymbol{\beta} \right\|_1 .$$

The parameter λ is often referred to as the *Lagrange multiplier* associated with the inequality constraint, and is our regularization parameter. Using the Lagrangian $L(\boldsymbol{\beta}, \lambda)$, we may now introduce the optimization problem on its Lagrangian dual form, which is based on the theory of taking the constraints into account by augmenting the objective function with a weighted sum of the constraint functions (Boyd and Vandenberghe, 2009, p. 215),

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\text{minimize}} \left\{ \frac{1}{2N} \left\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2 + \lambda \left\| \boldsymbol{\beta} \right\|_1 \right\}, \tag{4.7}$$

The reason why duality is interesting, is the fact that the solutions of the dual problem provides a lower bound on the optimal value of the solution of the primal problem. As such, there is a correspondence between the primal (original) problem, and its dual formulation. Furthermore, the solution of the primal and dual problem need not be equal. There might be a difference between the solutions, which is commonly referred to as the duality gap.

4.3 Variable selection property

An important aspect of the lasso estimator is the variable selection property. What is meant by this is that the lasso will shrink some parameters towards zero as the value for λ increases – the higher the λ the more parameters will be shrunk to zero.

As such, the lasso estimator performs model selection by default, and the resulting model will emphasize the parameters that matter the most for the response, leaving the unimportant parameters out of the regression. This is due to the geometry of the ℓ_1 constraint, and we will continue to give some insight in why this is the case.

To make the illustration easier, we start by considering the ridge regression, which takes the form,

$$\begin{aligned} \underset{\beta_0,\boldsymbol{\beta}}{\text{minimize}} & \left\{ \frac{1}{2N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \right\} \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_2 \leq t^2. \end{aligned} \tag{4.8}$$

As one may observe, (4.8) is almost identical to the problem of the lasso as stated in (4.4), with one slight twist as the constraints differ. Specifically, the ridge makes use of the ℓ_2 norm, whereas the lasso makes use of the ℓ_1 norm. As such, if $p = 2$, then the constraint regions will differ, as the ridge constraint makes a circle ($\beta_1^2 + \beta_2^2 \leq t^2$), while the lasso makes a diamond ($|\beta_1| + |\beta_2| \leq t$). The constraint regions are visually represented as the blue areas in Figure 4.1 for a linear model including two parameters ($\boldsymbol{\beta} = (\beta_1, \beta_2)$), and the red ellipses represent the contours of the scaled negative log-likelihood. Notice that the solution of either

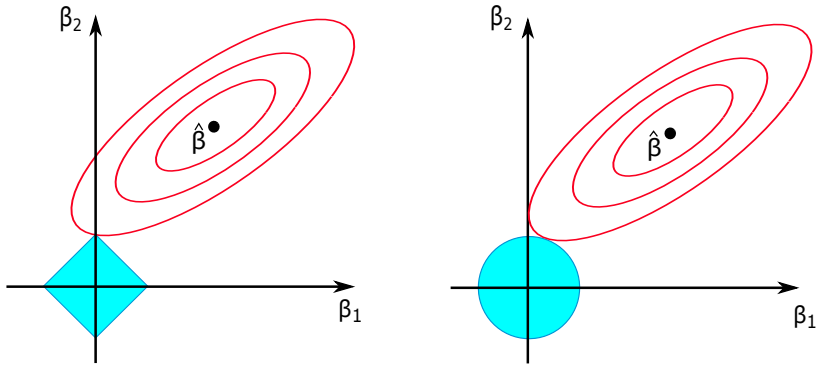

Figure (4.1) The figure is inspired from Figure 2.2 in Hastie et al. (2015, p. 11). The Lasso constraint is visualized to the left (diamond), whereas the Ridge constraint is visualized to the right (circle).

optimization problem (4.4) or (4.8) would be the first place where the contours of the scaled negative log-likelihood meet with the constraint area.

From Figure 4.1 we can observe that it is the geometry of the constraint that impacts the parameter estimate. As the lasso constraint has corners, it is much

more likely for the scaled negative log-likelihood estimates to be zero for the lasso than for ridge. This is due to the fact that a parameter β_j would be zero if the solution occurs at a corner. Furthermore, if one would choose to include more than parameters, the diamond would evolve into a rhomboid (Hastie et al., 2015, p. 12), which has more corners, edges and faces, and there are several more opportunities for the parameter estimates to become zero. The lasso is therefore known to give sparse solutions.

4.4 Karush-Kuhn-Tucker (KKT) conditions

For any optimization problem with differentiable objective and constraint functions, theory has shown that any pair of primal and dual optimal points must satisfy the Karush-Kuhn-Tucker conditions. Specifically, if looking at a optimization problem

$$\begin{aligned} &\text{minimize } f_0(x) \\ &\text{subject to } f_i(x) \leq 0,\ i = 1, \ldots, m \quad \text{(inequality constraints)} \\ &\qquad\qquad h_i(x) = 0,\ i = 1, \ldots, p, \quad \text{(equality constraints)} \end{aligned}$$

the Lagrangian L would be defined

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x).$$

By letting x^* and (λ^*, ν^*) be any primal and dual optimal points with zero duality gap, and due to the fact that x^* minimizes $L(x, \lambda^*, \nu^*)$ over x, it follows that the gradient must vanish at x^*,

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) = 0.$$

The resulting Karush-Kuhn-Tucker conditions may thus be defined (Boyd and Vandenberghe, 2009, p. 243),

$$\begin{aligned} f_i(x^*) &\leq 0, \quad i = 1, \ldots, m \\ h_i(x^*) &= 0, \quad i = 1, \ldots, p \\ \lambda_i^* &\geq 0, \quad i = 1, \ldots, m \\ \lambda_i^* f_i(x^*) &= 0, \quad i = 1, \ldots, m \\ \nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(x^*) &= 0. \end{aligned}$$

By applying this theory on the optimization problem (4.7), the resulting KKT conditions is found to be (Hastie et al., 2015, p. 9)

$$-\frac{1}{N}\langle x_j, \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\rangle + \lambda s_j = 0, \quad j = 1, \ldots, p. \tag{4.9}$$

In this expression, s_j represents an unknown quantity, which is equal to sign(β_j) whenever $\beta_j \neq 0$, or some value in $[-1, 1]$ otherwise.

However, as our primal problem was found to be convex, the KKT conditions changes from being a necessity for an optimal solution to be a sufficient set of conditions for an optimal solution. In other words, when solving the dual problem for a convex primal problem, the solution found will be equal, i.e. there will a zero duality gap between the two solutions.

We have thus introduced the lasso estimator, and presented related properties like its convexity and interpretation. Furthermore we introduced the dual problem, which may seem rather strange at first sight. However, it turns out that the dual formulation of the lasso estimator is perfect for the formulation of an algorithm able to solve the problem, namely coordinate descent. This algorithm is based on usage of the Lagrangian.

4.5 Coordinate descent

The lasso problem is often referred to as what we call a convex program, specifically a quadratic program (QP), meaning the optimization problem revolves around minimizing or maximizing a quadratic function of several variables. These are common problems, and as such, there exists many efficient algorithms to solve these problems. However, there is one particularly effective algorithm which gives insight in how the lasso works, and is based on the Lagrangian formulation of the dual problem in (4.7), and can be reformulated as

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}}}{\text{minimize}} \left\{ \frac{1}{2N}\sum_{i=1}^{N}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \right\}.$$

We continue to assume the both y_i's and x_{ij}'s to have been standardized, as explained in (4.5) and (4.6), meaning we once again can omit the intercept term β_0.

4.5.1 Soft thresholding

Starting off, we consider the case of a single covariate ($j = 1$) in the linear regression model, based on samples $\{(x_i, y_i)\}_{i=1}^{N}$. In such a case, the lasso estimator

becomes available in closed form by evaluating the KKT-conditions (4.9),

$$-\frac{1}{N}\langle \boldsymbol{y}, \boldsymbol{x}\rangle + \hat{\beta}\frac{1}{N}\sum_{i=1}^{N} x_i^2 + \lambda \widehat{s}_1 = 0,$$

where

$$\langle \boldsymbol{y}, \boldsymbol{x}\rangle = \sum_{i=1}^{N} y_i x_i.$$

In such a case, the solution may be written as (Hastie et al., 2015, p.15),

$$\hat{\beta} = \begin{cases} \frac{1}{N}\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \lambda & \text{if} \quad \frac{1}{N}\langle \boldsymbol{y}, \boldsymbol{x}\rangle > \lambda, \\ \frac{1}{N}\langle \boldsymbol{y}, \boldsymbol{x}\rangle + \lambda & \text{if} \quad \frac{1}{N}\langle \boldsymbol{y}, \boldsymbol{x}\rangle < -\lambda \\ 0 & \text{if} \quad \frac{1}{N}|\langle \boldsymbol{y}, \boldsymbol{x}\rangle| \leq \lambda, \end{cases}$$

or equivalently,

$$\hat{\beta} = \mathcal{S}_\lambda\left(\frac{1}{N}\langle \boldsymbol{x}, \boldsymbol{y}\rangle\right). \tag{4.10}$$

Here,

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

is called the soft-thresholding operator, which translates its argument towards zero by the amount λ, and sets it exactly equal to zero whenever the absolute value of the argument is smaller than λ, $|x| \leq \lambda$. Notice that the argument to the soft thresholding operator in (4.10) is the ordinary least squares (OLS) solution (Hastie et al., 2015, p. 15), meaning the lasso estimator for a single covariate in a linear model is the shrunken equivalent version of the OLS estimate.

4.5.2 Cyclical coordinate descent

Continuing the case of one covariate, we will now present the case of several covariates, that is, a problem of the form as in (4.7). The general idea is to update, or cycle, through the covariates, keeping all covariates fixed except the one that is to be updated. Specifically, if considering to update the j'th covariate, one would perform the univariate optimization (Hastie et al., 2015, p. 110),

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

where $\beta_k^{t+1} = \beta_k^t$ for $k \neq j$. The function h is a general multivariate objective function. This algorithm will converge as a result of a type of separability condition (Hastie et al., 2015, p. 110),

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

where $f : \mathbb{R}^p \to \mathbb{R}$ is a convex, differentiable function, whereas $g_j : \mathbb{R}^p \to \mathbb{R}$ is a convex, though not necessarily differentiable, function. This applies to the lasso in (4.7), as we identify the functions to be

$$f(\boldsymbol{\beta}) = \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \quad \text{and} \quad g_j(\beta_j) = \lambda|\beta_j|.$$

Once again considering the KKT conditions for the lasso as presented in (4.9), when evaluated at $\hat{\boldsymbol{\beta}}$, this becomes

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

From this we may define the partial residuals $r_i^{(j)} = y_i - \sum_{k\neq j}^{p} x_{ik}\hat{\beta}_k$, and rewrite the expression as

$$-\frac{1}{N}\sum_{i=1}^{N} r_i^{(j)} x_{ij} + \hat{\beta}_j \frac{1}{N}\sum_{i=1}^{N} x_{ij}^2 + \lambda s_j = 0 \quad \text{for} \quad j = 1, \ldots, p.$$

When comparing to the soft-threshold expression (4.10), and also assuming centering and standardization, the j'th parameter estimate could then be updated using (Hastie et al., 2015, p. 112)

$$\hat{\beta}_j = \mathcal{S}_\lambda\left(\frac{1}{N}\langle \boldsymbol{r}^{(j)}, \boldsymbol{x}_j\rangle\right). \tag{4.11}$$

Furthermore, by noticing that $\boldsymbol{r}^{(j)} = \boldsymbol{r} + \boldsymbol{x}_j\hat{\beta}_j$, where $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}$ are the full residuals, the expression may be more compactly written (Hastie et al., 2015, p. 16)

$$\hat{\beta}_j^{(t+1)} \leftarrow \mathcal{S}_\lambda\left(\hat{\beta}_j^t + \frac{1}{N}\langle \boldsymbol{r}^t, \boldsymbol{x}_j\rangle\right),$$

where, t denotes the iteration.

Orthogonal predictors

In the case of orthogonal predictors, the coordinate descent scheme becomes particularly easy as the inner product will always equal zero between different predictors, that is,

$$\frac{1}{N}\langle x_j, x_k\rangle = 0, \quad \text{for each} \quad j \neq k.$$

In such a case, the update (4.11) simplifies dramatically, as $\hat{\beta}_{\mathbf{j}}$ simply turns into the soft-thresholded version of the univariate least-squares estimate of y regressed against x_j (Hastie et al., 2015, p. 17), all because

$$\frac{1}{N}\langle x_j, r^{(j)}\rangle = \frac{1}{N}\langle x_j, y\rangle.$$

The reason of this being of interest to us is that when we have such an orthogonal design, we will have an explicit closed form solution for the lasso, and we may thus solve the optimization problem without iterations. In practice, this would mean that if we were to only consider the history effects, this result would apply and we would have a closed form solution.

4.6 Lasso regularized logistic regression

As in Section 3.2, we now consider n independent observations $Y_1, Y_2, \ldots, Y_n$, originating from the binomial distribution, $Y_i \sim \text{Bin}(n_i, \pi_i)$, where we will only consider the case when $n_i = 1$. In the case when $n_i = 1$, this reduces to the Bernoulli distribution, $Y_i \sim \mathcal{B}(\pi_i)$, and we find the log-likelihood for our logistic regression to be

$$\log L(\beta_0, \boldsymbol{\beta}; \boldsymbol{y}) = \sum_{i=1}^{N}(y_i \log(\pi_i) + (1 - y_i)\log(1 - \pi_i)).$$

Relating this to the lasso estimator, we substitute the expression into (4.1), let $v = 1$, and thus find

$$\hat{\boldsymbol{\beta}}(\lambda) = \text{minimize}_{\beta_0, \boldsymbol{\beta}}\left\{-\frac{1}{n}\sum_{i=1}^{n}\left(y_i(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}) - \log(1 + \exp(\beta_0 + \boldsymbol{x}_i^T\boldsymbol{\beta}))\right) + \lambda \left\|\boldsymbol{\beta}\right\|_1\right\}. \tag{4.12}$$

We have used the probability $\pi_i = \exp(\beta_0 + \boldsymbol{x}_i\boldsymbol{\beta})/(1 + \exp(\beta_0 + \boldsymbol{x}_i\boldsymbol{\beta})$ from (3.7). To solve (4.12), we introduce the `glmnet` package. This R-package is specifically made for fitting a GLM with lasso or elastic net regularization. What the package does it to alter the optimization problem (4.12), and turn it into an equivalent form based on penalized maximum likelihood, specifically, a weighted

form of (4.7). On this form, the optimization problem can be solved using regular cyclical coordinate descent.

To delve even further, we aim at understanding how this maximization process takes place. Consider $(\tilde{\beta}_0, \tilde{\boldsymbol{\beta}})$ to be our current parameter estimates. We may form a second order Taylor expansion approximating the log-likelihood part of (4.12) around the current parameter estimates, giving (Hastie et al., 2015, p. 116)

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

C represents a constant function which is independent of both $(\beta_0, \boldsymbol{\beta})$, while w_i represents the weights,

$$w_i = \tilde{\pi}_i (1 - \tilde{\pi}_i),$$

and z_i represents the working response,

$$z_i = \tilde{\beta}_0 + \boldsymbol{x}_i^T \tilde{\boldsymbol{\beta}} + \frac{y_i - \tilde{\pi}_i}{\tilde{\pi}_i (1 - \tilde{\pi}_i)}.$$

Even though it is possible to apply coordinate descent directly (4.12), it has been proven that a favourable option is to apply coordinate descent on the quadratic approximation, which results in a nested algorithm (Hastie et al., 2015, p. 116). Specifically, we now aim to solve the optimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\beta_0, \boldsymbol{\beta}}{\text{minimize}} \left\{ -\frac{1}{2n} \log L_Q(\beta_0, \boldsymbol{\beta}) + \lambda \left\| \boldsymbol{\beta} \right\|_1 \right\}. \tag{4.13}$$

The solution procedure is nested in the sense that it is comprised of three loops, that is, an *outer* loop, a *middle* loop and an *inner* loop.

- In the outer loop, λ is decremented.
- In the middle loop, the quadratic approximation L_Q is updated using the current parameter estimates $(\beta_0, \boldsymbol{\beta})$.
- In the inner loop, coordinate descent is used on the penalized weighted problem (4.13).

4.6.1 Cross-validation

We require an explicit value for our regularization parameter λ to solve the optimization problem (4.13). Specifically, as we calculate our parameter estimates $\hat{\boldsymbol{\beta}}$

based on different values of λ, we need a method to compare the estimates and select a value for the hyperparameter λ.

In order to find the optimal value for λ, we thus introduce the concept of K-fold cross-validation. This process revolves around partitioning the data into K groups, or folds, where $K > 1$. A typical choice would be to let $K = 10$. Starting off, one would choose one of the folds to equal what is called the test set, while the remaining $K - 1$ folds make the training set. Furthermore, one would fit the model on the training set, for a range of different λ values, and continue to use each fitted model to predict the responses in the test set, while the prediction error is also being recorded. This process is repeated K times, such that each of the folds will at one point have made the test set.

When cross-validation is finished, there are often two values for λ that are saved. These differs in the amount of penalization added to the model, and thus the amount of shrinking that is added to the parameter estimates. They are commonly referred to as λ_{min} and λ_{1se}. λ_{min} is the value for λ that gives minimum mean cross-validated error, whereas λ_{1se} gives the most regularized model such that error is within one standard unit of the minimum mean cross-validation error. What this means is that when fitting a lasso regression, more coefficients are likely to be shrunk to zero when using the λ_{1se} choice than when using λ_{min}. Furthermore, we will in this thesis use the binomial deviance as error measure.

Chapter 5

Multiple hypothesis testing

We will perform multiple hypotheses tests. As such, the question of choosing the cut-off value for the p-value arises, which is important, as the probability for type I error becomes enlarged in multiple hypothesis testing. We will in this section present an associated correction method used when performing multiple hypothesis testing.

We test m different null hypothesis $\mathcal{H} = \{H_1, \ldots, H_m\}$. Let m_0 denote the number of true null hypotheses, $\mathcal{T} \subseteq \mathcal{H}$, and $m - m_0$ the number of false hypotheses, $\mathcal{F} = \mathcal{H}/\mathcal{T}$. Our aim is to infer a subset of hypothesis, $\mathcal{R} \subseteq \mathcal{H}$, which is as close to $\mathcal{F}$ as possible. By assuming each and every one of the hypotheses H_j has a corresponding p-value p_j, we may define $\mathcal{R} = \{H_j : p_j \leq \alpha_{\text{loc}}\}$, where α_{loc} is a local threshold parameter, that we choose ourselves. This means that $\mathcal{R}$ is a set of our rejected null hypotheses, where each is rejected at significance level α_{loc}. The notation is summarised in Table 5.1. Notice that the only observable values from this table are m and $R = |\mathcal{R}|$. Now, we focus on $V = |\mathcal{R} \cap \mathcal{T}|$, the hypothesis that were falsely rejected, often referred to as false positive findings.

	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 (5.1) Table for multiple hypothesis testing. Only the number of rejected hypothesis R and the total number of hypothesis m are observed (Benjamini and Hochberg, 1995).

5.1 Family-wise error rate (FWER)

There exists multiple generalisations of the concept of type I error. One of these is the Family-wise error rate, which is defined as the probability of obtaining at least one type I error (Goemann and Solari, 2014),

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

where $V = |\mathcal{R} \cap \mathcal{T}|$ is the number of type I errors, as seen in Table 5.1.

When testing multiple hypotheses, one have to use a collection of test statistics, $T_1, \ldots, T_m$. These will produce each of their respective p-value, $p_1, \ldots, p_m$, which we refer to as *raw* p-values.

5.1.1 Bonferroni's method

The method of Bonferroni controls the family-wise error rate at a level α by rejecting a hypothesis if its respective raw p-value is smaller than $\alpha_{\text{loc}} = \alpha/m$. To prove that this actually controls the FWER, start by denoting the p-values of m_0 true hypotheses $w_1, \ldots, w_{m_0}$, so that the event of a type I error becomes $w_i \leq \alpha/m$. This yields

$$\begin{aligned} \text{FWER} &= P\left(\bigcup_{i=1}^{m_0} w_i \leq \alpha/m\right) \\ &\leq \sum_{i=1}^{m_0} P(w_i \leq \alpha/m) \\ &\leq m_0 \frac{\alpha}{m} \leq \alpha. \end{aligned} \tag{5.1}$$

What is nice about the Bonferroni method is that does not make any assumption when it comes to the dependency structures.

Furthermore, equation 5.1 illustrates how the Bonferroni method is said to be conservative, that is, the rejection criterion is strict, as $\alpha_{\text{loc}} = \alpha/m$ is smaller than what it actually needs to be. A last comment about the Bonferroni method is how it may be used to construct what is called *adjusted* p-values, using

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

5.2 Multi-sample splitting

Multi-sample splitting is a method for attaining p-values for data in a possible high-dimensional setting. Consider a the lasso as stated in (4.2), with a $n \times p$ design matrix $\boldsymbol{X}$, $n \times 1$ response vector $\boldsymbol{Y}$ and $p \times 1$ parameter vector $\boldsymbol{\beta}$. For multi-sample splitting, the general idea is to derive the p-values by splitting the sample set, denoted $I = \{1, \ldots, n\}$, into two equal sets, denoted I_1 and I_2. Specifically, $I_r \subset \{1, \ldots, n\} (r = 1, 2)$ with $|I_1| = \lfloor n/2 \rfloor$ and $|I_2| = n - \lfloor n/2 \rfloor$ where $I_1 \cap I_2 = \varnothing$ and $I_1 \cup I_2 = \{1, \ldots, n\}$ (Dezeure et al., 2015, p. 535). Note that $\lfloor$ $\rfloor$ represents the floor function of the given argument.

With such a split, it is then possible to use half of the sample set for variable selection, and the remaining set for statistical inference, only including the parameters that was deemed non-zero from the variable selection in the inference. The multi-sample splitting thus avoids using the problem of using data twice, both for variable selection and inference.

As an illustration, consider the set

$$\hat{\mathbf{S}}(I_1) \in \{1, \ldots, p\}$$

as the variables from I_1 whose parameter estimate is different to zero after performing variable selection, for example using the lasso. According to the multi-sample splitting methodology, one would continue to use the second half of the sample I_2 to construct p-values based on the selected variables from $\hat{\mathbf{S}}(I_1)$. In the case where $|\hat{\mathbf{S}}(I_1)| \leq n/2 \leq |I_2|$, which often occurs for the lasso, full rank ($|\hat{\mathbf{S}}(I_1)|$) is implicitly assumed for the matrix $\boldsymbol{X}_{I_2}^{(\hat{\mathbf{S}}(I_1))}$, where the subscript denotes the sample half and the superscript denotes the selected variables from the variable selection performed on I_1. One may then continue to perform ordinary GLM (3.3) on the set I_2, using $\boldsymbol{Y}_{I_2}$ as response vector and $\boldsymbol{X}_{I_2}^{(\hat{\mathbf{S}}(I_1))}$ as model matrix. Such a procedure thus yields p-values using Wald-statistics for the parameters $\hat{\boldsymbol{\beta}}_{\hat{\mathbf{S}}(I_1)} = \{\beta_j; j \in \hat{\mathbf{S}}(I_1)\}$, with the raw p-values defined as (Dezeure et al., 2015, p. 535)

$$P_{\text{raw},j} = \begin{cases} P_{t-\text{test},j} & \text{based on GLM fit with} \quad Y_{I_2}, \boldsymbol{X}_{I_2}^{(\hat{\mathbf{S}}(I_1))}, \quad \text{if} \quad j \in \hat{\mathbf{S}}(I_1), \\ 1, & \text{otherwise}, \end{cases}$$

which again means that we may continue to perform an ordinary GLM (3.3) on the set I_2. In the case of normality, the common approach is to use the t-statistic. For our matrix $\boldsymbol{X}_{I_2}^{(\hat{\mathbf{S}}(I_1))}$, the subscript denotes the sample half whereas the superscript denotes the selected variables from the variable selection performed on I_1. A key

feature of the multi-sample splitting is its extension to also correct for multiple testing.

One way performing such multiple testing is to control the family-wise error rate by naively applying a Bonferroni correction over the p tests. However, according Dezeure et al. (2015, p. 535), this is not necessary as we only need to control over the considered $|\hat{\mathbf{S}}(I_1)|$ tests in I_2. With this in mind, we can introduce the Bonferroni corrected p-value for $H_{0,j}$,

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

There is however a known problem occurring when performing the split of the entire sample, as p-values tend to not be reproducible. The reason why is simply that different sample splits lead to very different p-values. To address this problem, it is common to run the sample splitting method B times, for any large B. By doing this, a collection of p-values is obtained for every hypothesis $H_{0,j}$ $(j = 1, \ldots, p)$,

$$p^{(1)}_{\text{corr},j}, \ldots, p^{(B)}_{\text{corr},j} \quad (j = 1, \ldots, p).$$

With a collection of p-values for every hypothesis, the next goal would be to do an aggregation of these to obtain a single p-value for every hypothesis. By realizing that there is now a dependence among $\{P^{(b)}_{\text{corr},j}; b = 1, \ldots, B\}$ as all half samples origin from the same full sample, it is important to not naively aggregate the values. According to Dezeure et al. (2015, p. 536), an appropriate solution is to use an empirical γ-quantile,

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

where $0 < \gamma < 1$. A common approach is to do a search to find the best γ-quantile in range $(\gamma_{\text{min}}, 1)$ (for example $\gamma_{\text{min}} = 0.05$), which gives the aggregated p-value

$$P_j = \min\left((1 - \log(\gamma_{\text{min}})) \inf_{\gamma \in (\gamma_{\text{min}}, 1)} Q_j(\gamma), 1\right). \tag{5.2}$$

In this expression, the term $(1 - \log(\gamma_{\text{min}}))$ acts as a penalty for searching for the best $\gamma \in (\gamma_{\text{min}}, 1)$.

The approach in (5.2) is therefore favorable as it leaves us p-values that are approximately reproducible.

Part II

Analysis

Chapter 6

Data analysis

In this chapter we present the main results of our analyses, focusing on one experiment from English et al. (2017). We will do an in-depth analysis of one neuron in said experiment, presenting and explaining graphs and figures. Our main goal is to find the underlying network of neurons which is believed to exist in the brain of the mouse. We start by introducing the dataset, followed by an explanation of the regression model, before ending with presenting and assessing the network of neurons found.

6.1 The dataset

The complete dataset is presented in an article explaining the phenomenon of Pyramidal Cell-Interneuron Circuit Architecture and Dynamics in Hippocampal Networks (English et al., 2017). Specifically, we downloaded the data from McKenzie (2016), focusing on a single experiment containing 12 neurons (2 through 13), denoted by experiment ID 589 on the Buzakilab website. In addition, there were a lot of activity from unclassified fictive neurons included in the data, denoted as neuron 0.

The data was recorded by placing what is called a shank in the head of the mouse. This device would then be used to record all neuronal activity of nearby neurons. Furthermore, they inserted a glass pipette directly into one specific neuron, which became stimulated in intervals.

6.1.1 Experiment 589

As mentioned, we are focusing primarily on analyzing one experiment, specifically named experiment 589, consisting of the activity measurement of twelve neurons and one unclassified fictive neuron. As time is limited, we will only perform in-depth analysis of neuron 12 in the experiment, and briefly present the overall results of other neurons.

Experiment 589 was originally a stimulation experiment, and we have decided to exclude certain parts of the data on which the stimulation was performed, and only focus on the spontaneous parts of the data. In short, the data comprised of three parts, visually represented in Figure 6.1. The first part only contains the

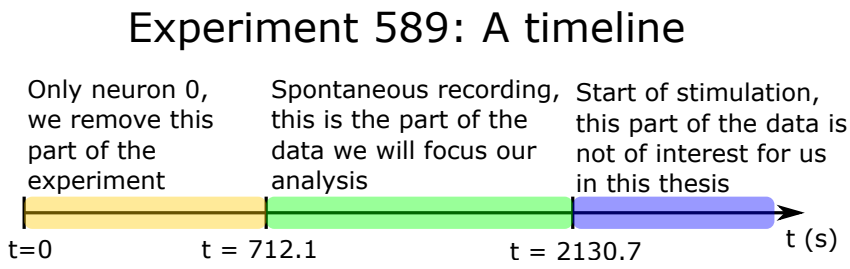

Figure (6.1) The green area between timestamps $t = 712.1$ and $t = 2130.7$ is what we define as the spontaneous part of the recording, on which we will perform our analysis.

firing of neuron 0, and is probably a result of tuning the shank to function properly before starting the actual recording, so that each firing neuron was simply labelled as zero. The second part is the spontaneous part of the experiment, and is the part of the data on which we will perform our analysis. The third part includes the stimulation, and will not be analyzed in this thesis. As can be seen from the figure, the length of the spontaneous interval is approximately 1400 seconds, or 1.4 million milliseconds. To shorten computational time, we have decided to only look at the first 1000 seconds of the spontaneous interval, or equivalently, the first 1 million milliseconds.

For most neural analyses regarding spike trains, raster plots are commonly introduced to show the activity of included neurons during some time period. In such a plot, the activity of each neuron is graphically represented as time passes by drawing a vertical line every time a neuron had an action potential, that is, spiked. As most neurons in Experiment 589 frequently fire, the raster plot has been divided into subplots 6, which together cover the entirety of the 1000 second interval. The raster plots in Figure 6.2 may thus gives us some information regarding the firing pattern of each neuron in the experiment. First of all one may notice that neuron

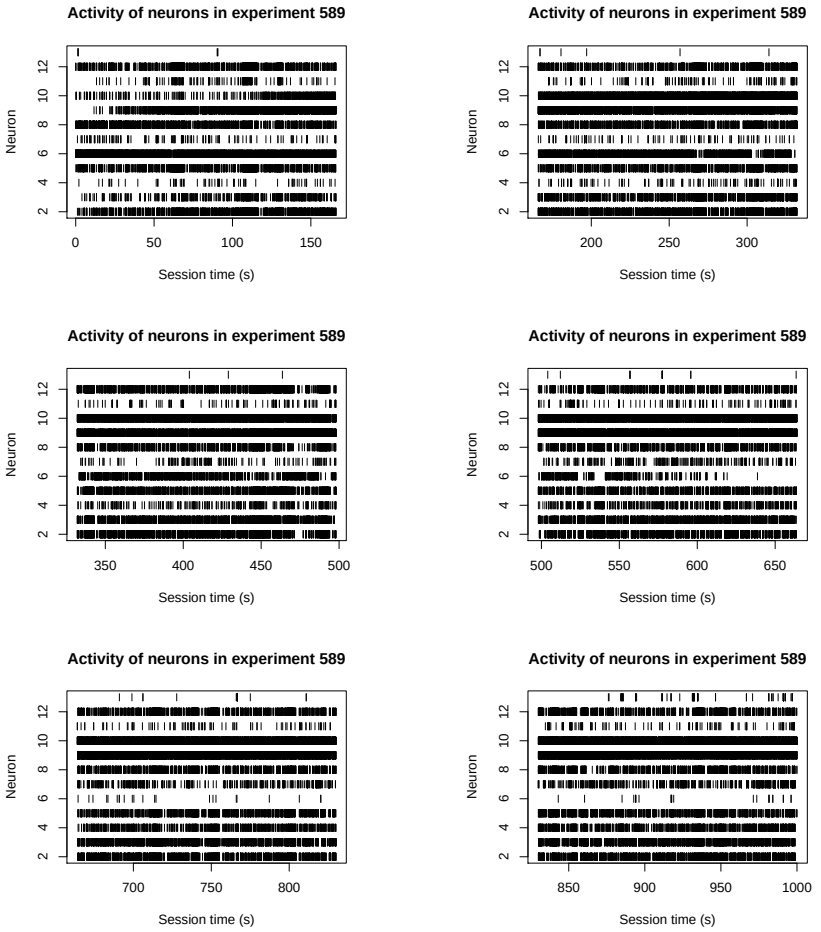

Figure (6.2) Raster plots of the activity of neurons for experiment with ID 589. For each neuron, a vertical line has been drawn at time t if the neuron spiked. The timeline has been divided into six parts to make it easier to observe firing patterns.

2, 3, 5, 8, 9 and 12 stay rather active throughout the whole interval. Furthermore, some neurons, like 4, 6, 7 have a change in spiking during the period, either from being very active to rather inactive (neuron 4), or from being almost inactive to very active (neuron 6). What should also be mentioned is that such recordings of neurons are very tricky to perform. As such, in most cases, signal contamination is entirely possible, or even likely. This could result in some neurons to be recorded twice, and to be labelled as different neurons. This could be the case if different neurons express a somewhat similar firing rate. In our experiment, an example may be neuron 2 and 9. However, this may only be speculation and is not something we will try to address or correct in this thesis.

6.2 Regression model

We now fit a lasso regularized regression model (4.12), with history and coupling effects in the linear predictors as presented in Equations (3.10) and (3.11). Specifically, we fit the model to observations from experiment 589, using neuron 12 as response, and the remaining neurons $2 - 11$ and 13 to evaluate coupling effects as covariates using the cosine bases. Multi unit activity was also included in the model, and is explained below.

The history effect was modelled at time t_1 by multiplying the 161 historic entries of the response vectors spike train (at times $t_{1-1}, t_{1-2}, \ldots, t_{1-161}$) with 161 predefined fixed points on each of the respective cosine bases ($L = 10$), before summing all 161 products together to find one value. This would be one of the 1 million entries in one column of the design matrix. The concept is illustrated in Figure 6.3.

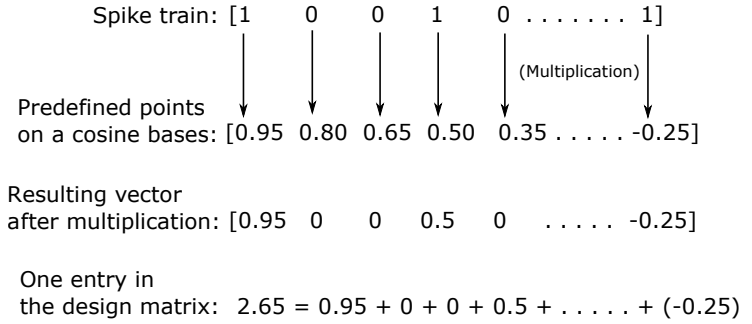

Figure (6.3) An example of how one entry of the design matrix is calculated for either history or coupling effects. The spike vector is multiplied by the predefined points on an appurtenant cosine basis, before the resulting vector is summed to end with one value in to the design matrix.

Coupling effects were modelled in the same way for the respective neighbouring neurons, with the use of coupling cosine bases ($L = 4$). The multi unit activity is modelling the fictive neuron 0, and is included by summing up the number of occurrences of neuron 0 in one bin of 1ms length, which is done for every millisecond of the 1 million milliseconds on which we perform our analysis. The sum of the occurrence of neuron 0 in one such bin of length 1ms would then be one entry of the MUA column of the design matrix. The frequency of neuron 0 in each bin varied, ranging from 1 to 33 occurrences. A barplot is included to show the distribution of occurrences per bin, presented in Figure 6.4. Observing that a $\log_{10}$

scale is used on the y-axis, we see that 1 occurrence of neuron 0 in per bin was by far the most frequent event.

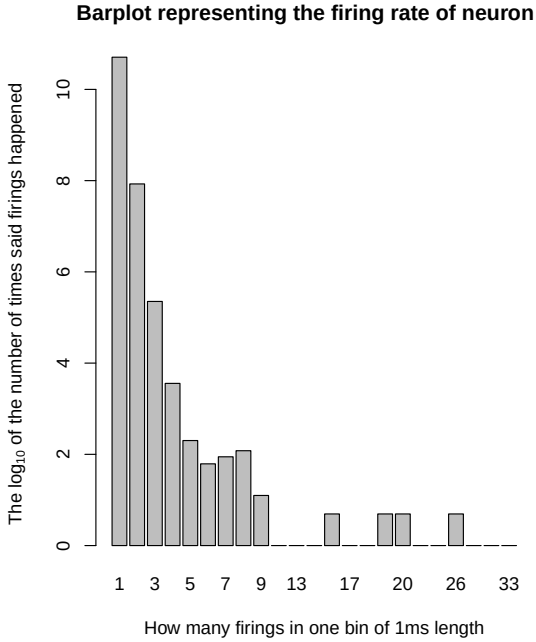

Figure (6.4) A barplot showing the frequency of how many times neuron 0 fired inside a bin of length 1 ms, using a $\log_{10}$ scale.

Knowing that our regression model includes history and coupling effects, as well as the multi unit activity, a conceptual plot of our lasso regularized logistic regression is presented in Figure 6.5.

We thus ended up with a model containing 56 parameters. This comes from the fact that the model includes an intercept, 10 history effects, $4 \times c$ coupling effects (where $c =$ the number of neighbouring neurons to the response neuron, excluding 0), and 1 for the MUA activity. This adds up to $1+10+11\times4+1 = 56$ parameters. Related to the fitting of the models, two plots may be presented. The first on is the coefficient path, as seen in Figure 6.6, and the second is the 10-fold cross-validation curve as seen in Figure 6.7. From the 10-fold cross-validation plot, it may be observed that the model with the lowest binomial deviance (line to the left in Figure 6.7), at $\lambda_{\text{min}} = 6.64 \times 10^{-5}$, resulting in 44 non-zero parameter estimates.

From observing both Figures 6.6 and 6.7 it is seen that there is a large dif-

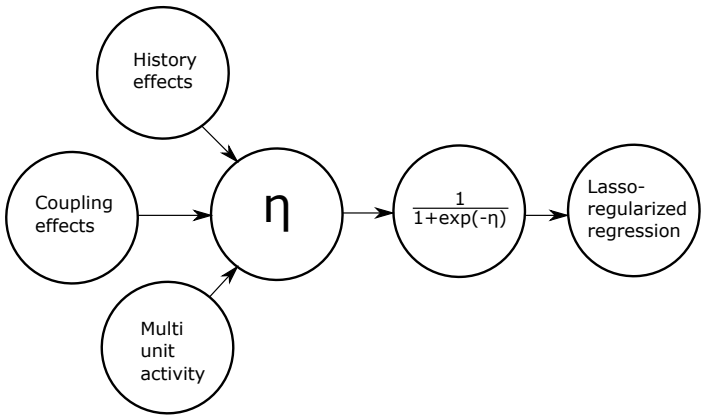

Figure (6.5) A conceptual plot showing how our model consists of history effects, coupling effects and multi unit activity.

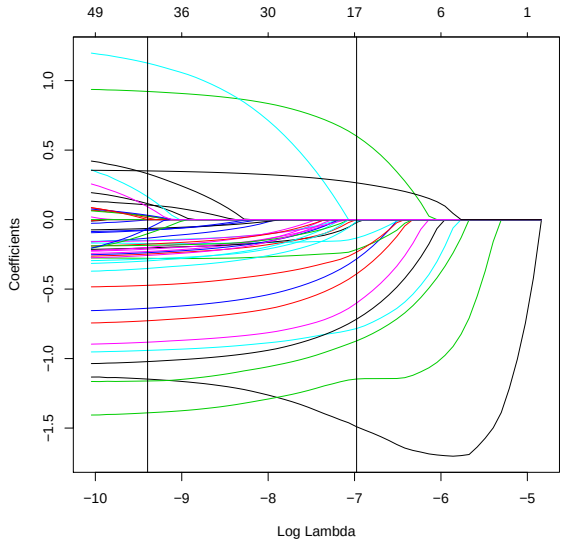

Figure (6.6) The lasso coefficient path for the full regression model fitted to observations for neuron 12 in experiment 589. The upper horizontal axis shows the number of nonzero coefficients included in the model. The two vertical lines represent the values for λ_{min} and λ_{1se}, from left to right, respectively.

ference when it comes to nonzero parameters dependent on whether we used the minimal lambda or the one at 1 standard deviation to regularize coefficients. As mentioned, choosing λ_{min} will result in 44 coefficients being nonzero, while the choice of λ_{1se} will result in only 16 coefficient being nonzero. We thus have two

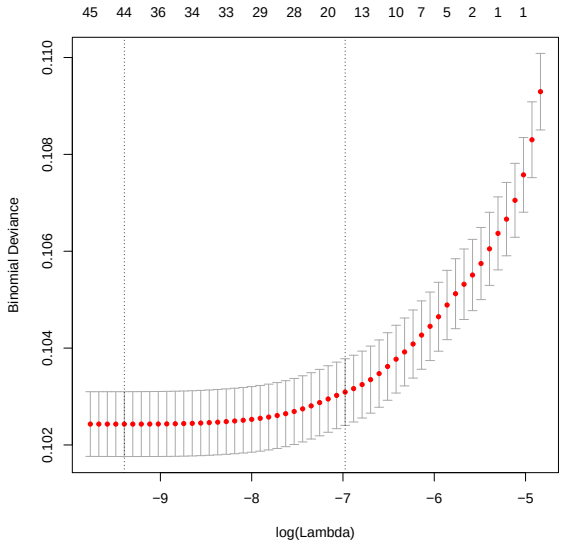

Figure (6.7) The 10-fold cross-validation binomial deviance plot for the full regression model fitted to observations for neuron 12 in experiment 589. Also here, the upper horizontal axis represents the number of nonzero coefficients included in the model. The two vertical lines represent the values for λ_{min} and λ_{1se}, from left to right, respectively.

possible choices to estimate our parameters based on either λ's, which again may be used to calculate two different FWER adjusted p-values.

6.3 A network of neurons

We will in this section try to estimate the true underlying network of neurons found in experiment 589, based on the significance of history and coupling effects. Recall from Section 3.4.2 if there is a connection from neuron k to j, it will be represented by $\hat{\boldsymbol{\alpha}}_{jkl} = (\hat{\alpha}_{j,k,1}, \hat{\alpha}_{j,k,2}, \ldots, \hat{\alpha}_{j,k,L})$, where $L = 4$ for coupling effects, and $L = 10$ for history effects. We have decided to consider that the connection $\boldsymbol{\alpha}_{j,k}$ is significant as long as one of its elements are significant. For each $\alpha_{j,k}$, a test of significance was performed by calculating the FWER adjusted p-values and using cut-off level 0.05. As such, the adjusted p-values were obtained based on the multi-sample splitting procedure as explained in Section 5.2, using the `multi.split` function in R from the `hdi`-package, where $B = 50$ splits was performed for each neuron included as response.

6.3.1 Significant effects

Once again consider the regression model fitted for neuron 12 from experiment 589. In Table 6.1 we have summarized the parameter estimates for history and connectivity effects, as well as the corresponding FWER adjusted p-values using both λ's to regularize coefficients. Calculation time to perform the multi-splitting took several hours for neuron 12 (approximately 6 hours), running on a computational server with 28 intel CPU's (2×14-core Xeon 2.6 GHz), with a memory of 768 GB. The computational time was approximately the same for every other neuron included in our analysis.

Neuron (k)	Basis (l)	$\hat{\alpha}_{j,k,l}$ based on		FWER adjusted p-value based on	
		λ_{min}	λ_{1se}	λ_{min}	λ_{1se}
12	1	-1.15	-1.49	7.91×10^{-117}	1.18×10^{-97}
	2	0.03	-	1	1
	3	-1.39	-1.15	3.55×10^{-40}	1.77×10^{-33}
	4	-0.13	-	1	1
	5	-0.94	-0.78	5.19×10^{-23}	1.73×10^{-23}
	6	-0.88	-0.60	2.45×10^{-23}	7.14×10^{-21}
	7	-1.02	-0.72	2.92×10^{-23}	6.06×10^{-23}
	8	-0.73	-0.39	1.35×10^{-13}	2.25×10^{-13}
	9	0.92	0.60	1.69×10^{-20}	1.55×10^{-18}
	10	-0.64	-0.28	2.46×10^{-9}	3.71×10^{-9}
0	MUA	0.35	0.26	4.93×10^{-35}	1.78×10^{-40}
2	1	-0.17	-0.13	3.39×10^{-7}	3.22×10^{-5}
	2	-0.23	-	1	1
	3	-0.20	-0.02	1	1
	4	-0.25	-	1	1
3	1	-0.28	-0.22	2.58×10^{-11}	9.11×10^{-9}
	2	-0.07	-	1	1
	3	-0.15	-	1	1
	4	-0.21	-	1	1
4	1	-	-	1	1
	2	-	-	1	1
	3	0.01	-	1	1
	4	-	-	1	1
5	1	-1.16	-0.87	7.26×10^{-35}	2.17×10^{-36}

	2	-0.002	-		1
	3	-0.35	-0.01	1	1
	4	-0.20	-	1	1
6	1	0.11	-	1	1
	2	-	-	1	1
	3	0.02	-	1	1
	4	0.03	-	1	1
7	1	1.12		1	1
	2	-	-	1	1
	3	-	-	1	1
	4	0.11	-	1	1
8	1	-0.47	-0.23	3.07×10^{-9}	6.23×10^{-9}
	2	-	-	1	1
	3	-0.24	-	1	1
	4	-0.30	-	1	1
9	1	-0.15	-	1	1
	2	-	-	1	1
	3	-0.06	-	1	1
	4	-0.18	-	1	1
10	1	-0.19	-	1	1
	2	-0.08	-	1	1
	3	-0.28	-	1	1
	4	-0.26	-	1	1
11	1	0.33	-	1	1
	2	-	-	1	1
	3	-	-	1	1
	4	-0.10	-	1	1
13	1	-0.06	-	1	1
	2	-	-	1	1
	3	0.16	-	1	1
	4	0.09	-	1	1

Table (6.1) Parameter estimates of neuron 12 in experiment 589, also including the adjusted p-values with respect to the FWER as explained in Section 5.1, using both λ_{min} and λ_{1se} to regularize coefficients. The parameter estimates were obtained from the penalized logistic regression model (lasso) as explained in (4.12) using linear predictors as in Equations (3.10) and (3.11). The adjusted p-values were constructed based on theory explained in Section 5.2, using this inbuilt function `multi.split` from the `hdi`-package, using $B = 50$ iterations. We have also highlighted the p-values that are below the cut-off level 0.05 using a gray background.

6.3.2 Network of connections between the 12 neurons and MUA from experiment 589

Based on our findings, we will now present three possible networks of neurons, each based on different estimates found during our analysis. The three networks are based on

1. FWER adjusted p-values using λ_{min} for regularization.
2. parameter estimates using λ_{min} for regularization,
3. parameter estimates using $\lambda_{1\text{se}}$ for regularization, and

While all neurons are included in the last two networks, we have excluded neuron 7 from the network based on FWER adjusted p-values due to convergence problems for the multi-sample splitting algorithm.

Figure 6.8 represents the estimated network of neurons based on the FWER adjusted p-values from the multi-sample splitting algorithm with data from experiment 589. In this plot, a line is drawn for every pair of neurons communicating, pointing towards the neuron being affected. As an example, as there is a line going out from neuron 6 and into neuron 9, this means that neuron 6 affects the firing rate of neuron 9 in some sense. One may also observe that every neuron has a significant history effect, represented as arrows originating from itself. This is probably due to the fact that every neuron has a refractory period, as presented in Figure 2.2. What this means is that a neuron is less likely to fire if it already fired not too long ago. Each of these lines from neuron k to neuron j is evaluated as significant coupling effects $\boldsymbol{\alpha}_{jkl}$ using the FWER adjusted p-values and a cut-off at 0.05.

It should also be mentioned that many of calculations are required for one such representation of a full network as seen in Figure 6.8. The entire regression model has to be fitted P times, independently, with P representing the number of neurons included in the model. As such, for the network in Figure 6.8, we ran the full regression model twelve times. This is a result of the experiment containing twelve neurons (2-13), where the regression model is fitted using observations from the j'th neuron, $\boldsymbol{y}_j$ as response variable, and the remaining $k \neq j$ neurons as covariates, for $j = 1, \ldots, P$. Furthermore, each regression was used to perform the multi-sample splitting algorithm using $B = 50$ to obtain the FWER corrected p-values.

Figure 6.9 represents the estimated network of neurons based on parameter estimates regularized using $\lambda_{1\text{se}}$, with data from experiment 589. As in Figure 6.8, also here a line is drawn for every pair of neurons communicating, pointing

towards the neuron being affected. A line is drawn in every case at least one of the history or coupling effects have a parameter estimate not shrunk to zero. To relate this to Table 6.1, every arrow going into neuron 12 in the network comes from other neurons which have at least one nonzero parameter estimate, as seen in the fourth column of the table. Observing the table, we see that neuron 12 has nonzero parameter estimates for history effects, as well as neuron $2, 3, 5, 8$, and the multi unit activity. This is reflected in Figure 6.9, as we see the neurons mentioned above have arrows going into neuron 12, as well as the history effect and the multi unit activity.

Finally, Figure 6.10 represents the estimated network of neurons based on parameter estimates regularized using λ_{min}. Lines are drawn between neurons as described in the paragraph above, and once again relating the figure to Table 6.1, we see in column 3 that every single neuron, including history effects and MUA have at least one nonzero parameter. This makes Figure 6.10 a full network, containing every possible connection that may occur.

Inhibitory and excitatory connections

Using Table 6.1, we may say something of the type of connections found specifically for neuron 12. Starting off, we observe that the parameter estimates are predominantly negative. Looking at the history effect parameters alone, we observe 8/10 and 6/7 negative estimates for λ_{min} and λ_{1se}, respectively. Based on this, we believe the history effect for neuron 12 to be inhibitory, meaning that historic firing of neuron 12 will decrease the chance of neuron 12 firing again. This agrees well with theory explained in Chapter 2, that every neuron has a refractory period after firing. Continuing in this manner, we believe that neuron $2, 3, 5, 8, 9$ and 10 also affects neuron 12 in an inhibitory manner, as the respective parameter estimates are also predominantly negative. Furthermore, we believe neurons $6, 7, 11$ and 13 affect neuron 12 in an excitatory manner, as their parameter estimates are predominantly positive. Relating this to theory from Chapter 2, this means that whenever one of these neurons fire, neuron 12 is more likely to follow and fire itself.

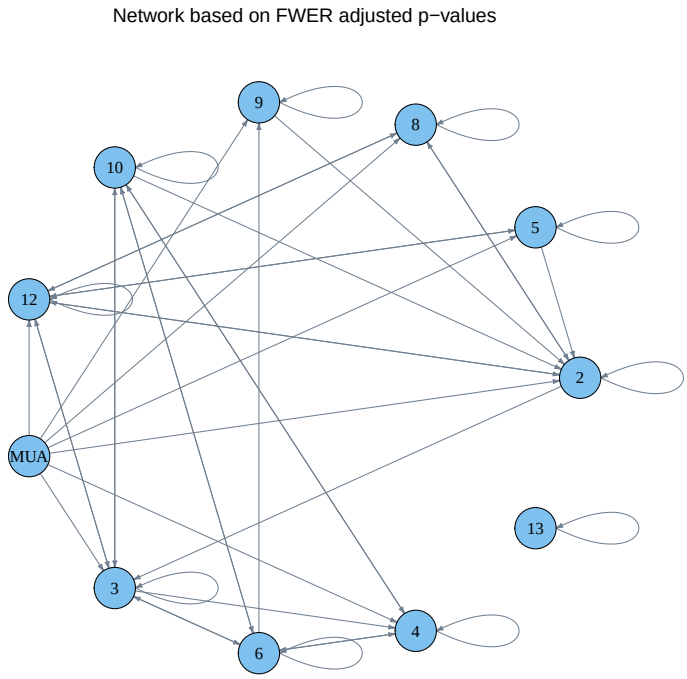

Figure (6.8) Estimated network of the twelve neurons (2-13), as well as the multi unit activity, for experiment 589. In this network, a directed arrow is drawn from one neuron m to another neuron n based on the significance of corresponding FWER adjusted p-value tested at 5% significance level, using λ_{min} to regularize coefficients.

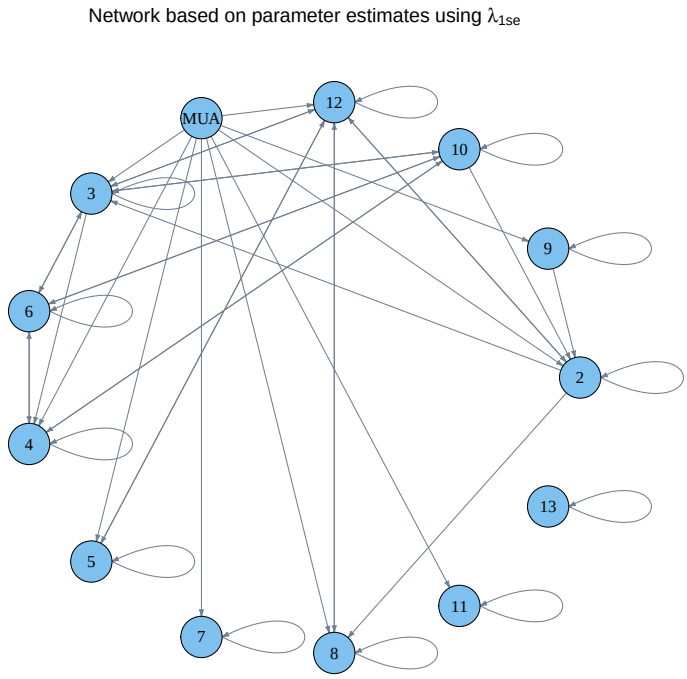

Figure (6.9) Estimated network of the twelve neurons (2-13), as well as the multi unit activity, for experiment 589. In this network, a directed arrow is drawn from one neuron m to another neuron n based on the nonzero parameter estimates using $\lambda_{1\text{se}}$ to regularize the parameters.

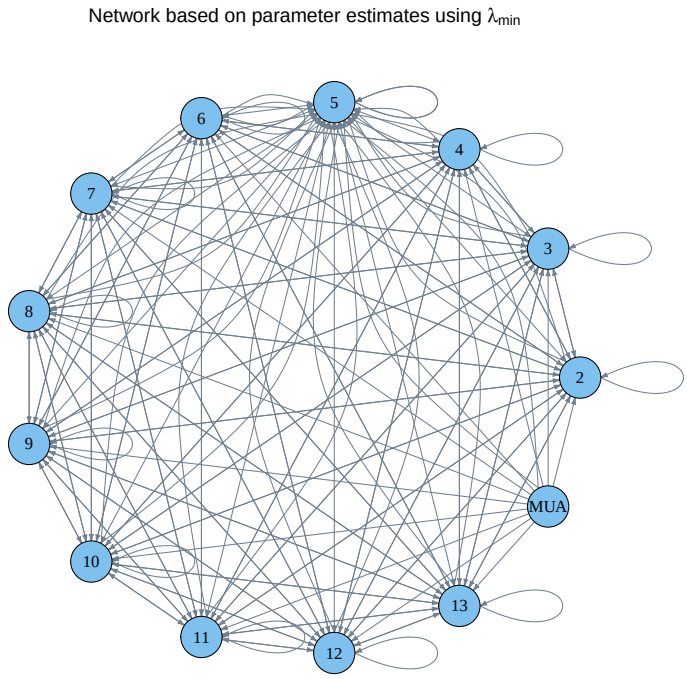

Figure (6.10) Estimated network of the twelve neurons (2-13), as well as the multi unit activity, for experiment 589. In this network, a directed arrow is drawn from one neuron m to another neuron n based on the nonzero parameter estimates using λ_{min} to regularize the parameters.

Chapter 7

Discussion and conclusion

In this chapter we start by discussing some aspects of what could be done differently in this thesis. We give a comparison of the original "ground truth" as developed by English et al. (2017), before discussing which of the λ's are most consistent and how an alteration of the multi-sample splitting algorithm could potentially benefit us. We proceed to elaborate on how one could continue the work started in this thesis, before presenting a conclusion.

7.1 Comparison with ground truth

What have yet to be mentioned is that English et al. (2017) also performed their own analysis of the dataset. We started this thesis with the aim of creating our own model to detect neuronal connections, and to continue by comparing our findings with their so called "ground truth". However, as we delved into the subject we quickly realized that we went in a whole other direction than how the data was originally analyzed, and thus excluded a comparison in our analysis. This is mainly due to two problems.

First of all, the dataset comprised of several different types of neurons, including what is known as pyramidal cells and interneurons. When originally analyzed, they classified each neuron in the experiment, and continued to only search for connections between interneurons and pyramidal cells, thus excluding some neurons in every experiment they analyzed. Identifying neurons to be of a specific type is both out of our scope, as well as it would give us no further statistical insight. As such, we chose to analyse the dataset as a whole, including every neuron

from the original dataset.

Furthermore, while both analyzing the spontaneous and stimulated part of the experiment, they delved into analyzing the stimulation part of experiment. During stimulation, a single neuron was stimulated, while English et al. (2017) looked at surrounding neurons to see how their behavioural firing pattern changed as the neuron was stimulated. What they found from this stimulation experiment was that neuron 13 was connected to neuron 12 for the experiment. As can be seen from our suggestions for networks in Figures 6.8, 6.9 and 6.10, we have no connection between neuron 13 and 12 for the two first figures, only including a connection for the full network as in the latter figure.

There might be several reasons to why we have not found the same connection between these two neurons. First of all, it might be a result of English et al. (2017) specifically searching for connections between some of the neurons in the dataset (specifically pyramidal-interneuron pairs), while we look at the entire experiment as a whole, treating every equally. This may affect some of the parameter estimates, and our model would likely change if we were to exclude certain neurons from the regression model. Furthermore, we have learned that when a neuron is stimulated, it may change the behavioural pattern for the stimulated neuron permanently. Specifically, as we chose to look at the spontaneous part of the dataset where no stimulation was applied to neuron 13, it may just be that this neuron had a completely different firing pattern in prior of the stimulation. Relating this to Figure 6.2, which is of the spontaneous period of the interval, we observe neuron 13 to be very inactive. Once the stimulation started, we believe that it caused a significant increase in the number of firings for said neuron, and that new connections could form between the neurons.

7.2 The effect of choosing λ

Another matter we would like to discuss is which of the λ's used to penalize coefficients that works best for our regression model. As explained in Section 4.6.1, λ_{min} gives the minimum mean cross-validated error, whereas λ_{1se} gives the most regularized model such that the error is within one standard unit of the minimum mean cross-validation error. As can be seen from Table 6.1, the resulting model contained 44 parameter estimates using λ_{min}, but only 16 parameters when using λ_{1se}. As such, the resulting networks varied greatly based on nonzero parameters as presented in Figures 6.9 and 6.10. One may thus discuss if choosing either of the λ's over the other is a better representation of the actual underlying network.

When addressing this matter, we chose to mainly base our discussion on the re-

sulting FWER adjusted p-values from the multi-sample splitting algorithm. What was observed was that the nonzero parameter estimates found regularizing using $\lambda_{1\text{se}}$ was very consistent with the resulting significant p-values from the multi-sample splitting algorithm, both when λ_{min} and $\lambda_{1\text{se}}$ was used. This may also be seen from the resulting networks of neurons as seen in Figure 6.8 and 6.9. Even though we should be careful to interpret anything as the "truth" itself in such neuroscientific matters, the consistency among the two models indicates that this is somewhat closer to the actual underlying network than what is presented from the full network in Figure 6.10. As such, we believe that Figures 6.8 and 6.9 are closest to the true underlying network, and therefore conclude that in our case, the usage of $\lambda_{1\text{se}}$ is superior over λ_{min} for estimating the network of neurons. However, for the network based on FWER adjusted p-values, there would be no difference on whether we were to use λ_{min} or $\lambda_{1\text{se}}$ to regularize coefficients.

7.3 Change the multi-sample splitting algorithm

We also thought of another aspect that could change the format of the underlying network based on the FWER corrected p-values. The multi-sample splitting algorithm divides the dataset into two bins each containing 50% of the data by default. It continues to perform the lasso penalized regression on one half of the dataset to obtain parameter estimates, before using the nonzero parameters on the other half to run a GLM and obtain p-values. This is repeated B times, before the resulting p-values from each parameters are aggregated to construct the FWER adjusted p-values. While it in general was no problem performing the multi-splitting, this was not the case when neuron 13 was used as response. The multi-sample splitting algorithm chose empty models for each split B, and the resulting corrected p-values were all equal to 1 for this neuron. What could help in this matter would be to alter the division of the dataset into bins of an unequal size, for example 70% of the data being used for the lasso regression, whereas only 30% could be used for construction of p-values. In addition to helping construct p-values for neuron 13, this could possibly also change the resulting corrected p-values for other neurons, resulting in a new, maybe better, representation of a network. As a final note, increasing B to a value larger than 50 could also improve the network, but was not done in this thesis as $B = 50$ already required days of computational time on a supercomputer.

7.4 Future work

Expanding the model

If one were to continue the work in this thesis, our first suggestion would be to expand the model. First of all, the one could try to include the neuroscientific concept of theta modulation to improve the model even further, specifically hippocampal theta modulation. Theta modulation, or theta waves, is a neural oscillatory pattern. It have been found that neurons in the hippocampus (part of the brain) react to such oscillations. Including such modulation in the regression model would probably result in a better model, although it would require very specific knowledge regarding the subjects of both neuroscience and statistics.

Furthermore, another aspect that could be introduced to the model is ripple modulation, as the original data also includes times of ripple events. Ripple events are also a neuroscientific concept in which neurons are excessively active, meaning that neurons fire more than usual. We believe that a model including both theta- and ripple modulation as covariates would better represent the actual underlying network. Both of these suggestions are motivated by English et al. (2017).

Only include PYR-INT pairs

As was originally done by English et al. (2017), the model could possibly be further improved by only including Pyramidal-Interneuron (PYR-INT) pairs in the regression model. When including every single neuron, our model may explain effects that are not wished to be included in the model. While we could say nothing of whether the resulting network from such a model would model the true underlying network any better than what our models do, it would be interesting to see a comparison between the two.

Lag and number of history and coupling bases

Lastly, we would like to address the matter of time lag and the different bases used. In this thesis, we have operated with a time lag window of 161ms, meaning that we at most consider what happened the previous 161ms milliseconds. However, if looking back at Figure 2.4, most forms of connections happen in the 0-100ms interval, which means that this may be a better interval to model the coupling effects. To do this one would place the cosine bases so they cover these intervals. The cosine bases could further be experimented with by changing the constants a and c in Equation (3.12), and also by changing the placement of the cosine bases,

as decided by the ϕ_l-parameter. This is because the values used in our thesis was directly matched from Pillow et al. (2008). Custom-fitting these parameters may therefore lead to a network representing the common, direct and indirect connections more accurately. We also matched the number of cosine bases directly from Pillow et al. (2008), with $L_{\text{hist}} = 10$ and $L_{\text{conn}} = 4$. The number of each could be experimented with, to also see how the resulting networks differs.

7.5 Conclusion

In this thesis, we considered the spontaneous part of a stimulation experiment where originally a neuron was stimulated. We implemented a lasso penalized regression model based on each neurons spike train, using the spike train itself as response variable, with history effects, coupling effects and multi unit activity as covariates. Based on on this model, we constructed three networks of neurons. Two of these were based on parameter estimates found using λ_{min}, $\lambda_{1\text{se}}$, whereas the third was based on the FWER corrected p-values when $\lambda_{1\text{se}}$ was used in the multi-sample splitting. These networks show how recorded neurons relate to each other, and thus how information flow between them. Our model works well for active neurons, though it ran into minor problems performing the multi splitting for somewhat inactive neurons.

Bibliography

Benjamini, Y., Hochberg, Y., 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society. Series B (Methodological) 57 (1), 289–300.

Boyd, S., Vandenberghe, L., 2009. Convex Optimization. Cambridge University Press.

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

Dobson, A. J., Barnett, A. G., 2008. An Introduction To Generalized Linear Models, 3rd Edition. Chapman & Hall / CRC Press.

English, D. F., McKenzie, S., Evans, T., Kim, K., Yoon, E., Buzsáki, G., 2017. Pyramidal cell-interneuron circuit architecture and dynamics in hippocampal networks. Neuron 96 (2), 505–520.

Fahrmeir, L., Kneib, T., Lang, S., Marx, B., 2013. Regression – Models, methods and applications. Springer.

Fawad, H., 2017. Modelling neuronal activity using lasso regularized logistic regression. Master's thesis, Norwegian University of Science and Technology. URL `https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2448349/17291_FULLTEXT.pdf?sequence=1&isAllowed=y`

Goemann, J. J., Solari, A., 2014. Multiple hypothesis testing in genomics. Statistics in Medicine 33 (11).

Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning; Data mining, inference, and prediction, 2nd Edition. Springer.

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

McCullagh, P., Nelder, J. A., 1989. Generalized Linear Models, 2nd Edition. Chapman & Hall.

McKenzie, S., 2016. Buzsaki lab webshare.
URL `https://buzsakilab.nyumc.org/datasets/McKenzieS/JS13/20161005_161005_110726/`

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E. J., Simoncelli, E. P., 2008. Spatio-temporal correlations and visual signalling in a complete neuronal population. Nature 545 (7206), 995–999.

Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., Lamantia, A.-S., White, L. E., 2018. Neuroscience; International Fifth Edition. Oxford University Press.

Appendix A

R code

A.1 Cosine bases

```
library(pracma)

getBasis = function(nBases, binSize){
  b = binSize*nBases
  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)
  maxt = invnl(yrange[2]+2*db)-b #originally db
  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
  }

  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)
    }
  }

  lags = invnl(centers)-b

  library(pracma)
  ihbas = orth(ihbasis)

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


#Using the getBasis function to create said bases for binsize 0.001
binSize <- 0.001
nBases_history <- 10
nBases_coupling <- 4

histBases <- getBasis(nBases_history, binSize)
couplingBases <- getBasis(nBases_coupling, binSize)

histBasesCorr <- histBases$bas_orth[1:161,]
couplingBasesCorr <- couplingBases$bas_orth[1:161,]

dput(histBasesCorr,paste("historyBases.dd"))
dput(couplingBasesCorr,paste("couplingBases.dd"))
```

A.2 Model matrix and lasso penalized regression

```
#install.packages("glmnet")
library(glmnet)


############# All parameters that may be adjusted
    ####################
lengthPeriod <- 1000000 #length in ms of time we are to analyze
currentExperiment <- 589
############################################################



clu <- read.delim("Experiment2_clu") #Neurons
res <- read.delim("Experiment2_res") #Timestamps

resInMs <- res/20 #Originally in Hz
startTime <- 712098


timeStimStart <- 2130.7*1000 #for ms
timeStimEnd <- 2804.5*1000 #for ms


#All neuronal activity before stimulation
resNew <- resInMs[resInMs<timeStimStart]
#All respective times for neuronal activity before stimulation
cluNew <- clu[resInMs<timeStimStart]


#Removing all zero's at the start
cluFix <- cluNew[ min( which ( cluNew != 0 )):length(resNew)]
#Removing all zero's at the start
resFix <- resNew[min( which ( cluNew != 0 )):length(resNew)]

#Note: the first entry of resFix is 712099 found on row A90912 in
    the original file



############################################################



spikeMat <- matrix(0, ncol = lengthPeriod, nrow = 13)#nrow
    iterates all neurons (0, 2-13 in experiment 589)


```

```
#We now have to fill in the spikes from resNew in spikeMat at the
    correct places
i <- 1 #initiating while loop
while(round(resFix[i])-startTime <= lengthPeriod){ #continue till
    you reach desired lengthPeriod
  if(cluFix[i] != 0){ #If there is a spike, put assign at correct
    place in spikeMat
    whichNeuron <- cluFix[i]
    spikeMat[(whichNeuron),(round(resFix[i])-startTime)] <- 1
    i <- i+1
  } else if(cluFix[i] == 0){ #Add 0 in spikeMat
    spikeMat[1,(round(resFix[i])-startTime)] <- (spikeMat[1,(round
    (resFix[i])-startTime)]) + 1
    i <- i+1
  }
}
print("done while loop")



########## Retrieving previously calculated bases ##########
historyBases <- dget("historyBases.dd")
couplingBases <- dget("couplingBases.dd")

responseGone <- 161
#Reversed of the bases as we are to go back in time
revHist <- apply(t(historyBases), 1, rev)
revCoup <- apply(t(couplingBases), 1, rev)


### Loop iterating through all neurons for given experiment ###
whichNeurons <- c(2,3,4,5,6,7,8,9,10,11,12,13) #For experiment 589

iteration <- 1
for(i in whichNeurons){

  print("###################### Working on neuron
    ####################")
  print(i)


  focusNeuron <- whichNeurons[iteration] #The neuron we are
    testing
  iteration <- iteration + 1

  colNumb <- 10+(4*(dim(spikeMat)[1]-2))+1 #10 histoy + 4*
    neighbourNeuron not 0 + 1 for 0-spikes
  neighbourNeurons <- (1:dim(spikeMat)[1])[-1][-(focusNeuron-1)] #
    Neighbouring neurons to analyzed neuron
```

```
86
87
88  thisY <- spikeMat[focusNeuron,(responseGone+1):lengthPeriod]
89
90
91  #Components that are to be part of the design matrix,
      preallocating
92  thisXc <- matrix(0, nrow = (lengthPeriod-responseGone), ncol =
      (4*(dim(spikeMat)[1]-2))) #Coupling, 4*#neighbour, not 0
93  thisXh <- matrix(0, nrow = (lengthPeriod-responseGone), ncol =
      10) #History
94  thisXMUA <- matrix(0, nrow = (lengthPeriod-responseGone), ncol =
       1) #MUA
95  print("done preallocating")
96
97
98
99  #History effects
100 allY <- spikeMat[focusNeuron,1:lengthPeriod]
101 for (i in 1:(lengthPeriod-responseGone)){
102   thisXh[nrow <- i,] <- matrix(allY[(responseGone+i-1):i], nrow
      = 1)%*%revHist
103 }
104 print("done history")
105
106
107
108 #Coupling effects
109 counter <- 1
110 for (k in neighbourNeurons) {
111     allY <- spikeMat[k,1:lengthPeriod]
112     for (i in 1:(lengthPeriod-responseGone)){
113     thisXc[nrow <- i, ncol <- (1+4*(counter-1)):(4*counter)] <-
      matrix(allY[(responseGone+i-1):i], nrow = 1)%*%revCoup
114   }
115   counter <- counter + 1
116 }
117 print("done coupling")
118
119
120
121 #MUA
122 thisXMUA[, ncol <- 1] <- spikeMat[1, (responseGone+1):
      lengthPeriod]
123   print("done MUA")
124
125
126
127 #Combine, first 10 cols is history, rest except last is coupling
```

```
    , last is MUA
128   thisX <- cbind(thisXh, thisXc, thisXMUA) #first 10 cols is
        history, then all except last coupling, last MUA
129     print("done combining")
130
131
132
133   #Setting column names
134   colNames <- c(paste("H", 1:10, sep = "."), paste("C", rep(
        neighbourNeurons, each = 4), 1:4, sep = "."), "MUA")
135     colnames(thisX) = colNames
136
137   timeSet <- 1:(lengthPeriod-responseGone)
138   x <- thisX[timeSet,]
139   dput(x, paste("designMatExp", currentExperiment, "Neuron",
        focusNeuron,".dd", sep = ""))
140   y <- thisY[timeSet]
141   dput(y, paste("responseVecExp", currentExperiment, "Neuron",
        focusNeuron,".dd", sep = ""))
142 #Note: It might be clever to not save all x and y as the x-matrix
      is a rather large file!(most design matrices were
      approximately 0.6GB)
143
144
145
146   ############### Fitting models ################
147
148   #Redundant
149   fit <- glm(y~x, family = binomial)
150   print("done fit")
151   print(summary(fit))
152   dput(summary(fit)$coefficients, paste("glmCoeffsExp",
        currentExperiment, "Neuron", focusNeuron, ".dd", sep = ""))
153
154
155   ####This is the lasso penalized model giving our coefficients
        ####
156   fitNew <- cv.glmnet(x, y, family = "binomial", alpha = 1)
157   pdf(paste("binDevExp", currentExperiment, "Neuron", focusNeuron,
        sep = "")) #Place the plot as pdf in a file
158   plot(fitNew)
159   dev.off()
160   print(fitNew$lambda.min)
161   dput(coef(fitNew, s = "lambda.min"), paste("cv.glmnetCoeffsExp",
        currentExperiment, "Neuron", focusNeuron, ".dd", sep = ""))
162   print(coef(fitNew))
163   print("done fitNew")
164
165
```

```
166
167   #newFit <- glmnet(thisX[timeSet,], thisY[timeSet], family = "
        binomial", alpha = 1)
168   pdf(paste("shrinkExp", currentExperiment, "Neuron", focusNeuron,
        sep = ""))
169   plot(fitNew$glmnet.fit, xvar = "lambda")
170   abline(v=log(fitNew$lambda.1se))
171   abline(v=log(fitNew$lambda.min))
172   dev.off()
173   print("done newFit")
174
175
176
177   fit1se <- glmnet(x, y, family = "binomial", alpha = 1, lambda =
        fitNew$lambda.1se)
178   coef(fit1se)
179   dput(coef(fit1se), paste("glmnetCoeffsExp", currentExperiment, "
        Neuron", focusNeuron, ".dd", sep = ""))
180   print("done fit1se")
181
182 }
```

A.3 Multi-sample splitting

```
#install.packages("hdi")
#install.packages("glmnet")
library(hdi)
library(glmnet)


########What are we currently analyzing#############
currentExperiment <- 589
chosenB <- 500
#####################################################


whichNeurons <- c(12,7,8,9,10) #For experiment 589, c
    (2,3,4,5,6,7,8,9,10,11,12,13)
iteration <- 1


for (k in whichNeurons){

  focusNeuron <- whichNeurons[iteration]
  iteration <- iteration+1

  binSize <- 0.001

  x <- dget(paste("designMatExp", currentExperiment, "Neuron",
    focusNeuron,".dd", sep = ""))
  y <- dget(paste("responseVecExp", currentExperiment, "Neuron",
    focusNeuron,".dd", sep = ""))


  #### Function for minimum  ####
  lasso.cv.min <- function (x, y, nfolds = 10, grouped = nrow(x) >
     3 * nfolds, ...)
  {
    fit.cv <- cv.glmnet(x, y, nfolds = nfolds, grouped = grouped,
    ...)
    sel <- predict(fit.cv, type = "nonzero", s = fit.cv$lambda.min
    )
    sel[[1]]
  }


  ###### Performing the multiSplit #####
  multiSplit <- multi.split(x, y, B = chosenB, ci = FALSE ,
    classical.fit = glm.pval, model.selector = lasso.cv, args.
    model.selector = list(family = "binomial"), verbose = TRUE)

```

```
40   ## Saving the interesting parts of the multisplit (pval) ##
41   dput(multiSplit, paste("multSplitExp", currentExperiment, "
       Neuron", focusNeuron,".dd", sep = ""))
42   dput(multiSplit$pval.corr, paste("multSplitPvalCorrExp",
       currentExperiment, "Neuron", focusNeuron, "PvalCorr" ,".dd",
       sep = ""))
43 }
```