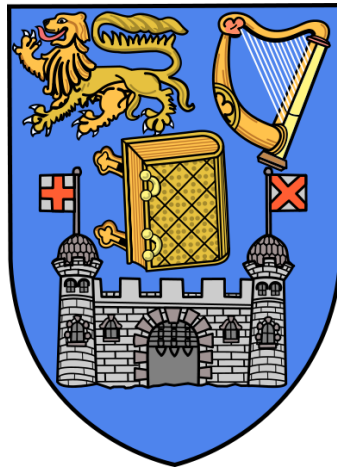# Spike Train Analysis

## Trinity College Dublin

Cathal Cooney

A thesis submitted for the degree of Doctor of Philosophy.

May 2015

# Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed,

———————————————————————

Cathal Cooney

29th May 2015

# Acknowledgements

# Summary

This thesis focuses on the study of spike trains, the information carrying signals conveyed by neurons in the nervous system.

Spiking data from songbirds from [Narayan et al., 2006] was used prominently in this thesis. Multi-unit data for Chapter Three was simulated using a model network from [Houghton and Sen, 2008].

In Chapter Two, several clustering algorithms are introduced. The eigenvalue algorithm of [Newman, 2006a] to cluster networks was used prominently. This algorithm was combined with a measure called the incremental mutual information [Singh and Lesica, 2010] to cluster modules of information in simulated networks of neurons.

Chapter Three deals mainly with two distance measures, the SPIKE distance [Kreuz et al., 2013] and the ISI distance [Kreuz et al., 2007]. These distance measures were extended from single-unit to multi-unit measures.

In Chapter Four a simple neuron model is proposed, based on the idea of sparse coding in the brain. The expected firing rate based on the model was calculated, and then the ISI distribution was calculated from the firing rate. The calculated distribution was tested against the empirical data using the Kolmogorov-Smirnov test for goodness-of-fit [Massey Jr, 1951].

# Contents

# Chapter 1

# Introduction

This thesis presents a number of novel approaches to the analysis of spike trains, the electrical signals which carry information throughout the brain. It has long been noted that different stimuli influence the rate at which neurons fire [Knight, 1972], however, more recently it has been observed that stimuli can be encoded by the temporal precision of signals [Hopkins and Bass, 1981; Engel et al., 1992].

The advancement of computer processing power has led to increased analysis of the highly temporal data sets found in neuroscience. Computational neuroscience is a rapidly growing field of study which looks to apply the expertise of other branches of science and engineering to mine and model neuroscience data. As a subset of computational neuroscience, mathematical neuroscience uses many different mathematical concepts to model the behaviour of the brain at many different levels.

The level of neural activity which is of interest in this thesis is one of the bottom-most levels which is studied. Rather than studying communication between different brain regions, spike train analysis focuses on the signals produced by single neurons.

## 1.1 Spike trains

There are many different types of neurons in the brain. In Figure 1.1 there are two such variations, as drawn and documented by Ramón y Cajal [1904] in the late 19th and early 20th century. Pyramidal cells, such as in Figure 1.1 for example, are abundant in the cerebrum cortex, and Purkinje cells are found in the cerebellum. They have differing purposes in the nervous system, but they all accept and emit electrical signals to communicate with other neurons.

Neurons convey information through the nervous system by generating electrical pulses which propagate along nerve fibers. These pulses are called *action potentials* or *spikes*, first observed in [Du Bois-Reymond, 1884].

A typical cortical pyramidal neuron has a resting potential of approximately -70 mV relative to its surroundings, the cell is said to be polarised at this point. The membrane potential of a neuron is changed by current flowing into its synapses from other neurons, most of which make the membrane potential less negative, but the potential tends back towards the resting potential unless it nears a certain threshold. If a neuron is depolarised so that its membrane potential is raised above this threshold, the neuron generates an action potential, a rapid depolarisation and repolarisation of the membrane potential. Such a fluctuation in the membrane potential typically lasts approximately 1 ms. Following the production of a spike, the neuron briefly becomes hyperpolarised, that is, the potential drops to below its resting level. A neuron cannot generate a spike for several milliseconds after a spike. This period, when a spike cannot be fired, is called the absolute refractory period of a neuron. There is also a longer period following a spike in which it is more difficult for the neuron to spike again, which is called the relative refractory period.
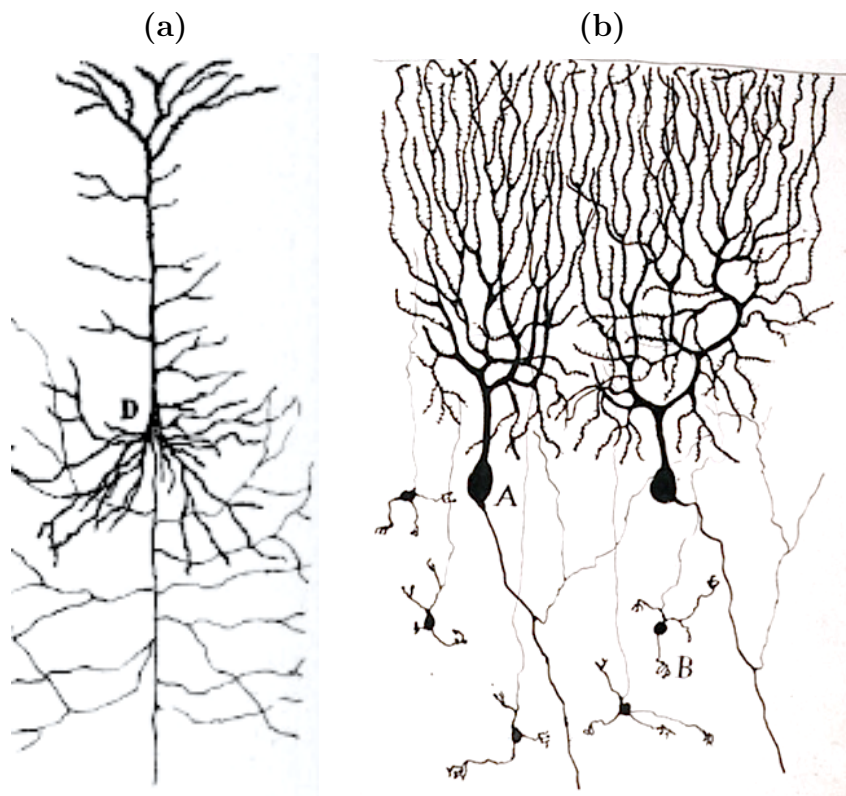
Figure 1.1: Diagrams of neurons as drawn by Ramón y Cajal [1904]. (a) depicts a cortical pyramidal cell . Pyramidal cells are the primary excitatory neurons of the cerebral cortex. Pyramidal cells are thus often modelled as a typical neuron. (b) depicts a Purkinje cell of a pigeon. Purkinje cells are found in the cerebellum in the human brain. They are some of the largest cells in the human brain. With their dense branching dendrites they have much higher connectivity than pyramidal cells.
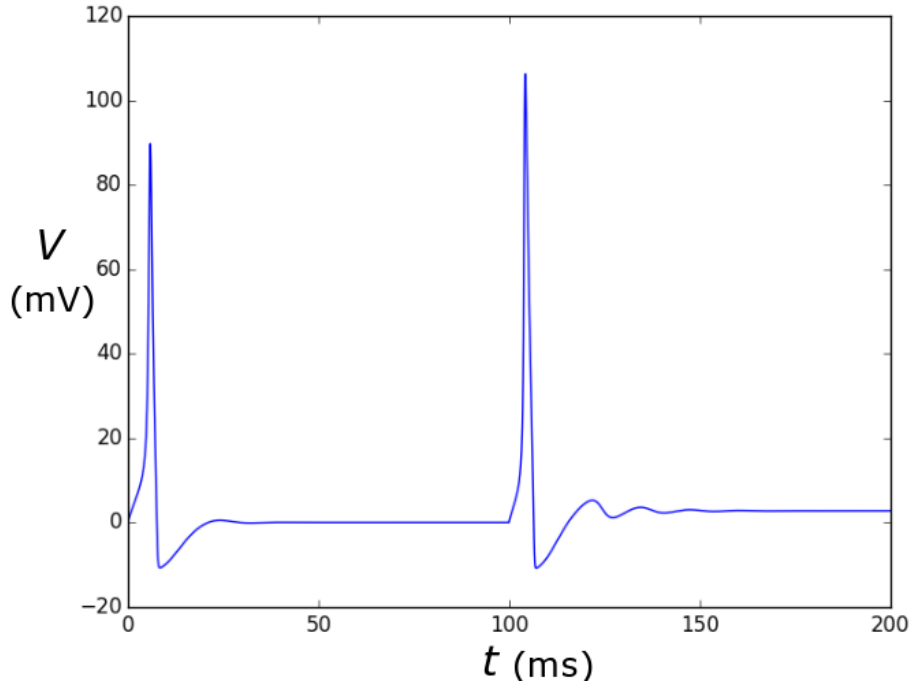
Figure 1.2: The voltage trace of a pair of spikes generated by the Hodgkin-Huxley model. Simulation run using the python package Brian [Goodman and Brette, 2008]

The dynamics of the membrane potential were precisely modelled in [Hodgkin and Huxley, 1952] by modelling the flow of sodium and potassium ions into and out of the neuron. Their model was based on the voltage traces that they recorded from the giant axon in a squid [Hodgkin and Huxley, 1939]. It has proven to be an excellent framework for biophysically accurate neuron models. The Hodgkin-Huxley models, with several currents to depict the flow of ions through the membrane, form the standard against which other neuron models are judged [Izhikevich, 2004].

Spikes are very important to the study of communication in the brain, because they propagate over long distances without attenuation. Other

4

sub-threshold fluctuations attenuate greatly over spaces as short as 1 mm. Therefore, spikes are the way in which neurons communicate with other regions throughout the nervous system. While there is some variation in the duration, amplitude and shape of action potentials, for a single neuron they typically exhibit a characteristic voltage trace as discussed in [Lewicki, 1998]. The spike then can be represented by choosing a particular point of the typical action potential. Thus, the timing of the spikes give much of the information [Bi and Poo, 1998; Bair, 1999]. So, if a neuron spikes $n$ times in a certain trial, then the trial can be described by the timings of the spikes $t_i$. The *spike train* can then be expressed mathematically as:

$$X = x(t) = \sum_{i=1}^{n} \delta(t - t_i). \tag{1.1}$$

The $\delta$ here is the Dirac delta function centred at $t_i$, which has a volume equal to one at the point $t_i$.

## 1.2 Spike train metrics

While observing spike trains from a specific neuron, ideally one would be able to 'decode' them and be able to describe the stimulus which produced them. This problem is very difficult, particularly when looking at neurons *in vivo*, that is, in living animals [Averbeck et al., 2006]. There tends to be a lot of noise from other neurons in the network which can obscure the signal to be reproduced [Hopfield, 1982]. A first step which can be taken towards solving this problem is to try to distinguish which spike trains amongst a collection share the same stimulus. There are several *metrics* in the literature which give a measure of the distance between pairs of spike trains.

In Mathematics a metric space is a space equipped with a scalar function which gives a distance between two elements in the space. The

function is called a metric if it is a scalar function on a set $X$, $d : X \times X \to [0, \infty)$ that has the properties a distance might intuitively be expected to have. That is, distances are non-negative, symmetric and only zero if two elements are the same. The final property is called the triangle inequality, which is a rule that ensures the shortest distance between two points is a straight line.

## 1.2.1   Victor-Purpura metric

In [Victor and Purpura, 1997] a family of metrics are described. All the metrics are a type of metric called 'edit-length' metrics. An edit-length metric is based on a theoretical 'cost' of transforming one point in a space to another via elementary transformations. The principle is that each transformation has a non-zero cost, so the distance between two spike trains can be measured by minimising the cost to transform one spike train, $X$, into another, $Y$. If zero-cost transformations are allowed, then such a measure is a pseudometric rather than a metric.

There are three different permitted fundamental transformations. It is possible to insert a spike at any time, which has a cost of one. Similarly, any spike can be deleted for a cost of one. The third permitted transformation is to move a spike $\Delta t$, which has a cost $q\Delta t$, for some timescale parameter $q \geq 0$. Formally, the Victor-Purpura distance between spike trains is only a metric when the timescale parameter is positive, $q > 0$, but the $q = 0$ case is included in [Victor and Purpura, 1997] to illustrate the boundary conditions of the measure, and so the Victor-Purpura distance is actually a pseudometric.

The insertion and deletion operations may not appear to be influenced by the parameter $q$, but these operations do set a maximum cost of transformation from one spike to another at 2. Since a spike at any time can be deleted at a cost of one, and another inserted at a cost of one, this

6

sets a maximum time-frame for transpositions at $2/q$. Therefore, given two spikes, $t_a$ in $X$ and $t_b$ in $Y$:

$$c(f_i(t_a)) = \begin{cases} q|t_a - t_b| & |t_a - t_b| < 2/q \\ 2 & |t_a - t_b| \geq 2/q \end{cases} \tag{1.2}$$

It follows that as $q$ tends towards zero, it becomes cheaper to move spikes than insert and delete them. In the case when $q = 0$, it costs nothing to move spikes, thus the measure is equal to the difference in the number of spikes in $X$ and $Y$.

Then the Victor-Purpura metric between two spike trains, $X$ and $Y$, is defined to be the minimum over all combinations of fundamental functions which transform $X$ to $Y$. The metric can be written as:

$$d_{\mathrm{VP}}(X, Y) = \min_f \sum_i c\left(f_i(X)\right) \tag{1.3}$$

where $f = f_1 \circ \ldots \circ f_m$ is a function such that $f(X) = Y$.

This is demonstrably a metric. There is a non-zero cost for each transformation, so $d(X, Y) = 0$ only if $X = Y$. Due to the fact that insertion and deletion are symmetric with each other, and transposition is a symmetric operation, it is clear that $d(X, Y) = d(Y, X)$. The triangle equality follows, since if $f$ transforms $X$ to $Y$, and $g$ transforms $Y$ to $Z$, then $h = g \circ f$ must transform $X$ to $Z$, and since the Victor-Purpura metric requires that the minimum cost, $h$ must have cost greater than or equal to $d(X, Z)$.

## 1.2.2  van Rossum metric

The metric proposed in [van Rossum, 2001], is based on the $L^2$ metric on function spaces. The $L^2$ metric between square integrable functions, $f, g$ , is the square root of the integral of the difference between $f$ and $g$

squared, that is:

$$\|f - g\|_2 = \sqrt{\int_0^\infty [f(t) - g(t)]^2 \, dt}.$$  (1.4)

**(a)**



**(b)**



**(c)**
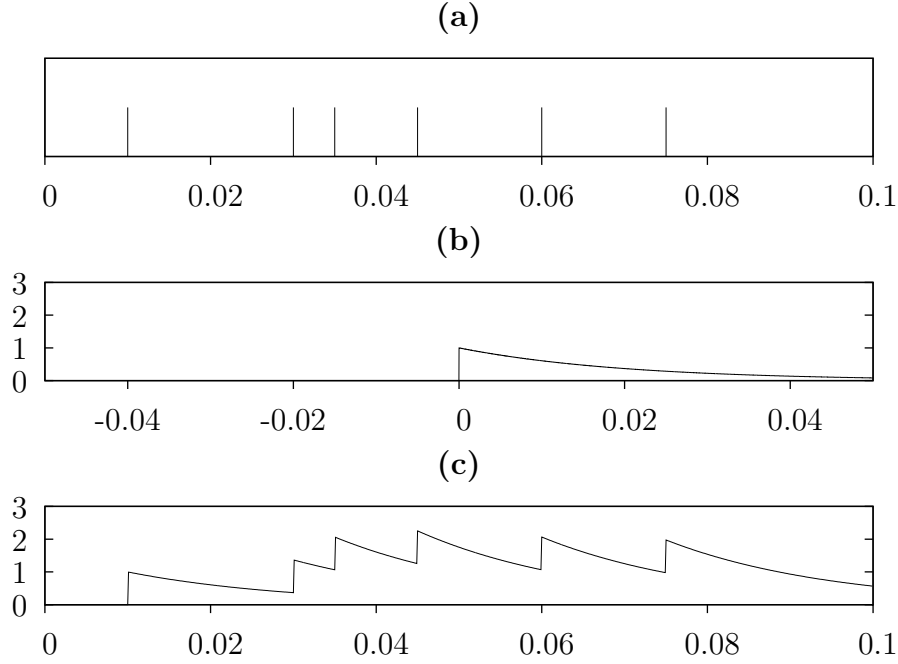


Figure 1.3: Shown above is an example of the filtering process of the van Rossum metric. **(a)** A sample spike train is convolved with **(b)** an exponential filter, where the timescale $\tau = 20$ ms, to get **(c)** the function $u(t)$ .

If spike trains are viewed as sums of Dirac delta functions, then by convolving the spike trains, $X = x(t)$, with a square-integrable kernel, $k(t)$, each spike train can be represented in an $L^2$ function space. A common kernel is the exponential kernel as used in [van Rossum, 2001]:

$$k(t) = \begin{cases} \frac{1}{\tau} e^{-\frac{t}{\tau}}, & t \geq 0 \\ 0, & t < 0 \end{cases}.$$  (1.5)

this function is shown in Figure 1.3.

This kernel has the potential advantage of causality over a Gaussian kernel, and it is very fast to calculate [Houghton and Kreuz, 2012]. Convolving the spike train with the kernel results in a function, $u(t)$, which is $L^2$ measurable:

$$u(t) = (x * k)(t) = \int_0^T x(t - s)k(s)\, ds \qquad (1.6)$$

Figure 1.3 shows the form of the function that is produced by convolving a spike train with the exponential kernel. The metric is then simply the $L_2$ distance between these functions:

$$d_{\mathrm{vR}}(X, Y) = \sqrt{\int_0^T \left[(x * k)(t) - (y * k)(t)\right]^2 dt}. \qquad (1.7)$$

It has been shown that once the correct timescale has been chosen the choice of kernel has little effect on the performance of this metric [Houghton and Victor, 2012].

## 1.3  Information Theory

Information theory has been used widely in computational neuroscience, eg. to compare spike trains [Strong et al., 1998] and to try to determine the information content of spike-trains [Gillespie and Houghton, 2011]. This thesis uses information theory measures such as the incremental mutual information [Singh and Lesica, 2010], in Chapter Two, and the transmitted information, in Chapter Three.

The topic of information theory is centred around the definition of *entropy*, provided by Shannon [1948]. Entropy is a measure of the information content of a random variable $X$, with observations $x \in \mathcal{X}$ and probability distribution $p : \mathcal{X} \to [0, 1]$, it is defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \qquad (1.8)$$

Entropy is measured in *bits*, so the entropy of a random variable can be viewed as the minimum number of binary bits required to efficiently code the random variable.

To measure the entropy of a spike train a probability distribution has to be calculated. In [Strong et al., 1998] a method, which is illustrated in Figure 1.4, is described to determine this distribution. Each spike train is binned into a discrete vector of ones and zeros, which indicate whether there was a spike in that time bin. Typically a time scale less than ten millisecond in length is chosen for the bin-size. A larger time-scale, $T$, is chosen to constitute the events in the probability space, which are then 'words' of ones and zeros. The probability distribution is then calculated empirically from the occurrences of the words throughout the spike train. The entropy can be calculated from the resulting probability distribution.
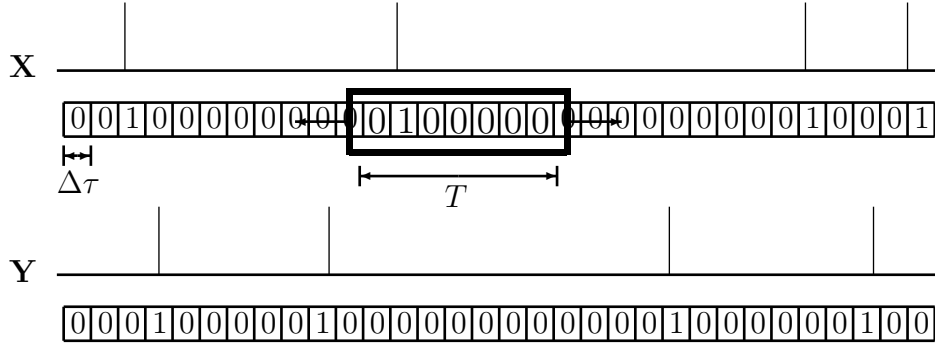


Figure 1.4: The method developed in [Strong et al., 1998] for calculating the entropy and mutual information of spike trains. The spike train is sampled at a rate $\Delta\tau$, typically on the order of $2-5$ ms. The spike train is binned, and each time-step registers only whether there is a spike in that bin or not, putting a value of one or zero in the bin accordingly. On another, larger time-scale, $T$, words of ones and zeros of length $(T/\Delta\tau)$ are counted. These words form the probability distribution for the entropy.

The conditional entropy is a measure of how much information is in a random variable $X$, given that the result of another random variable, $Y$, is known. It is defined:

$$H(X|Y) = \sum_{y \in \mathcal{Y}} p(y) H(X|Y = y) \qquad (1.9)$$

That is, for each value, $y$, of $Y$, the entropy of $X$ is calculated for that set value, then weighted by the probability of that value. This can then be calculated, using Bayes' theorem:

$$H(X|Y) = - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x)}{p(x,y)} \qquad (1.10)$$

The information content of a random variable cannot increase given knowledge of another variable, so it is always true that the conditional entropy, $H(X|Y)$, is less than the entropy of the variable itself, $H(X)$. The difference between the two values can be viewed as the shared information of the two variables. The *mutual information* of two variable is defined as:

$$I(X;Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \left( \frac{p(x,y)}{p(x)p(y)} \right) \qquad (1.11)$$

which can equivalently be expressed as:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \qquad (1.12)$$

If two variables are independent, then the joint distribution of $X$ and $Y$ would be equal to the product of the marginal distributions, and so they would have zero mutual information.

## 1.4   Data sets

The spike train metrics in this thesis were tested on data sets to suit the experiment that was undertaken. While an effort was made to use real electrophysiological data, some experiments required the use of simulated data.

### 1.4.1 Zebra finch data

Single neuron spike train metrics, in Chapters Two and Four of this thesis, were tested on the data set used in [Narayan et al., 2006]. The data set was obtained by inserting an electrode into the auditory forebrain of anaesthetised zebra finches and recording the neuronal responses to multiple presentations of conspecific song stimuli. The fact that the birds were anaesthetised means that there would be minimal feedback to the neurons in the forebrain from other brain functions.

There were recordings from 24 such cells. Each cell was presented with 20 separate songs ten times each, as shown in Figure 1.5, with a tone to signify the start of the recording precisely one second before the onset of the stimulus.

Zebra finch data is of particular use in Chapter Four, as there is evidence of feature recognition in responses to natural songs in the zebra finch forebrain [Sen et al., 2001]. Chapter Four introduces a simple model of firing which would be consistent with sparse coding. Data sets with highly temporal features are thus important in testing the model.

### 1.4.2 Multi-unit data

The multi-unit distance measures in chapter three were tested on the same simulated test data as in [Houghton and Sen, 2008]. Two Poisson neurons form a receptive field and are each connected to two leaky integrate-and-fire neurons (LIF) with relative strength $a$ and $1 - a$; $a$ is called the mixing parameter of the simulation. When $a = 0$ each receptive neuron is connected to a single LIF neuron, and for $a = 0.5$, each LIF neuron receives input equally from each of the receptive neurons.

The data consisted of five different stimuli, each presented 20 times, for a total of 100 responses. These responses are clustered according to the bootstrapped procedure used in [Victor and Purpura, 1996]. A
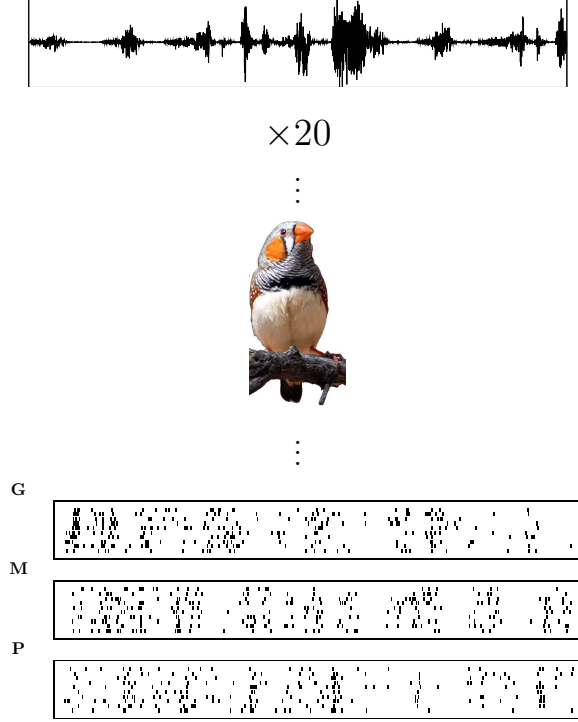
Figure 1.5: The data used was collected for [Narayan et al., 2006]. A total of 20 songs were played to anaesthetised zebra finches, and the responses were recorded in the auditory forebrain. Each song was presented ten times. Responses of three different cells to the same song are shown here.

confusion matrix is calculated whose entries $N_{ij}$ represent the number of responses from stimulus $i$ which are closest, on average, to the responses from stimulus $j$.

The transmitted information, $h$, is then calculated from the confusion matrix, $N$.

$$h = \frac{1}{n} \sum_{ij} N_{ij} \left( \ln N_{ij} - \ln \sum_i N_{ij} - \ln \sum_j N_{ij} + \ln n \right). \qquad (1.13)$$

The transmitted information has maximum value ln(5), as there are five stimuli in this test.

13

While it may be preferable to test metrics on biological data in most cases, it is crucial to compare the performance of the multi-unit metrics with a controlled mixing parameter. The data presented here allows an analysis of the multi-unit metrics' performance as the data varies from independent input to identical input.

Responses (pairs of spike trains)

Integrate and
fire neurons

$1 - a$

Poisson spiking
background $\quad a$

$a \quad$ Poisson spiking
background

Receptive
field neurons

Stimuli (pairs of rate functions)
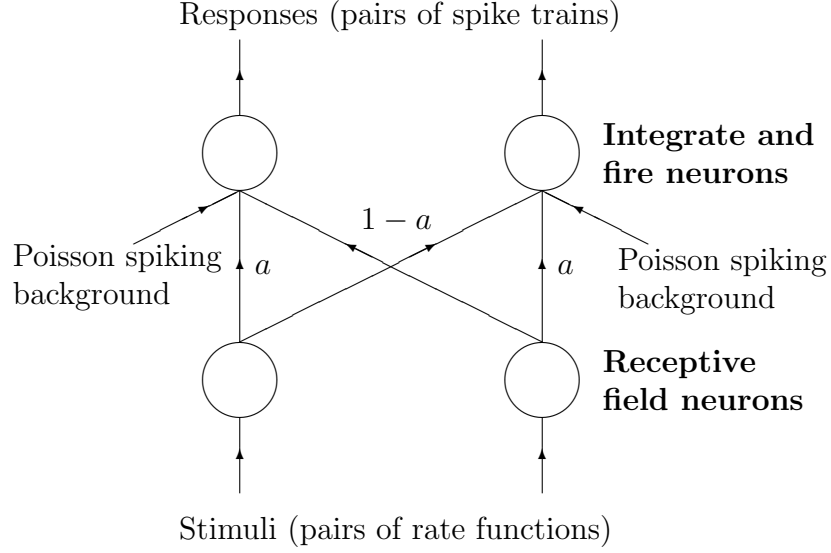
Figure 1.6: This is the schematic for the data set used to test the multi-unit metrics in this thesis. It is the same data set that was used in [Houghton and Sen, 2008]. The receptive field neurons fire Poisson spike trains according to the rate function input they receive. These Poisson spike trains feed into the leaky integrate and fire (LIF) neurons from which the response is measured. The synaptic strengths are parametrised by a mixing parameter $a$. When $a$ is zero, the LIF neurons receive independent input; when $a = 0.5$, the input should be identical.

# Chapter 2

# Clustering methods in spike train analysis

This chapter approaches neuroscience from the point of view that there are clearly functional connections between neurons in the same region of the brain and that these connections could provide a map of information flow in the brain. Complex network theory, a branch of mathematics closely related to graph theory, is used frequently in computational neuroscience.

Complex networks often exhibit a community structure, and so there are many algorithms to find the best clustering of a given network [Karypis et al., 2000; Newman and Girvan, 2004; Newman, 2006a]. This chapter introduces the clustering algorithm developed by Newman [2006a], and suggests some new algorithms based on the modularity measure [Newman and Girvan, 2004]. These new algorithms are based on complexity algorithms, genetic algorithm and simulated annealing, which require nothing more than a 'goodness-of-fit' function. Complexity algorithms often produce interesting results, as they do not take a pre-determined path through the solution space.

Newman's algorithm [Newman, 2006a] is then used to cluster neuron

responses dynamically, where the number of stimuli need not be known. A network of responses is formed by comparing the van Rossum distances between the spike trains, and then choosing a distance threshold where any distance less than the threshold forms a link. By clustering this network using the eigenvalue algorithm of Newman [2006b], it is not necessary to specify the number of stimuli in advance, unlike the commonly used methods, $k$-means and $k$-medoids.

## 2.1 Introduction to networks

Complex network theory differs from graph theory in that it focuses on the traits that networks tend to have when they evolve in real life, rather than focusing on purely random networks. The main difference between these would be the connectivity profile of the network; in graph theory connections are usually spread somewhat uniformly throughout the graph, whereas in Network Theory it is expected that communities would form [Newman, 2010].

A mathematical network is simply a collection of nodes and links between these nodes. A *node* is simply a point in the graph, which is typically discrete and labelled. A *link* is a pair of nodes which indicates a connection between the two nodes. The *degree* of a node is defined to be the number of links incident to the node. An *undirected network* with $n$ nodes can be completely described by its adjacency matrix $\mathbf{A}$ where $A_{ij} = 1$ if nodes $i$ and $j$ are connected, and is equal to zero otherwise. In this case $A_{ij} = A_{ji}$ since two nodes are either connected or not. If matrix entries $A_{ij}$ are permitted to take values other than one and zero then that is called a *weighted* network. A network is called a *directed* network if $A_{ij} = 1$ whenever there is an arrow pointing from node $j$ to node $i$, and the adjacency matrix is not necessarily symmetric. Undirected networks which are unweighted form the bulk of the literature, and so the theory is

richer in this case. A reason for this could be that networks such as friend networks [Zachary, 1977] would typically not be directed as friendship is typically mutual.

A particularly interesting part of network theory that could be useful for neuroscience is clustering, and algorithms to maximise the community structure of the clusterings. To maximise the community structure, a measure of such structure is required.

## 2.2  Modularity

The *modularity* is a measure of a given clustering of a network, which gives a value to the community structure of the clustering [Newman and Girvan, 2004]. The modularity is defined to measure how much more prominent intra-cluster links are than inter-cluster links in a given clustering. The modularity then, as a measure of the given clustering, measures how much community structure is seen in the network. It was introduced by Newman and Girvan [2004] in one form, then Newman [2006a,b] derived another form using the configuration model for graphs [Bender and Canfield, 1978; Bollobás, 1980].

The modularity is a measure of a clustering of a network that assesses how much more community structure there is in the network than there would be in a random network with similar properties. In [Newman, 2006a,b] the modularity is derived as follows in this section. The modularity is defined to be the difference between actual links within clusters and the expected number of links. Summing over all nodes within a cluster, $g$:

$$\text{(Links in } g) - \text{(Expected links in } g) = \sum_{i,j \in g} A_{ij} - \sum_{i,j \in g} P_{ij} \qquad (2.1)$$

where $P_{ij}$ is the probability of a link between nodes $i$ and $j$. Then, if $g_i$ is the community to which node $i$ belongs, the modularity $Q$ can be
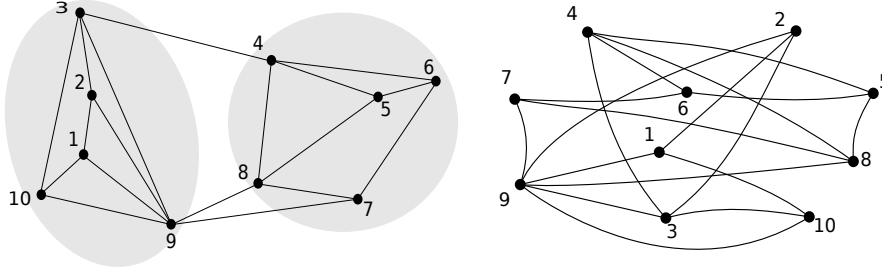
Figure 2.1: The advantage to clustering a network correctly can be seen here. Shown are two diagrams which have the same nodes and links, and so form the same network, but they look vastly different because the diagram on the left has been clustered to maximise modularity.

described:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - P_{ij}]\delta(g_i, g_j) \tag{2.2}$$

where $\delta$ is the Kronecker delta, so $\delta(g_i, g_j) = 1$ if nodes $i$ and $j$ are in the same cluster, and zero otherwise. $m$ is the total number of links in the network, and so $1/2m$ is just a normalising factor. The modularity of a cluster is then the number of edges within a cluster minus the expected number of edges.

Newman [2006a,b] calculates the probability of a link between nodes in the configuration model. In undirected networks, clearly the probabilities should be symmetric, $P_{ij} = P_{ji}$. Then, the expected number of links across the whole network should equal the total number of links. That is, $\sum_{i,j} [A_{ij} - P_{ij}] = 0$, and,

$$\sum_{i,j} P_{ij} = \sum_{i,j} A_{ij} = 2m, \tag{2.3}$$

where $m$, as before, is the total number of edges in the network, so if the

degree of node $i$ is $k_i$, then

$$m = \frac{1}{2} \sum_i k_i = \frac{1}{2} \sum_{i,j} A_{ij}. \tag{2.4}$$

At this point, there is some choice as to how similar the random network, that the calculated probabilities are based on, should be to the network itself. One could suppose that each node in the random network has degree equal to the average degree of the network, but this ignores local structure in the network. Instead, it is supposed that the random network should keep the same degree for all nodes, and so the network can be treated as an instance of the *configuration model* [Bender and Canfield, 1978; Bollobás, 1980]. So, the expected degree of each vertex is equal to the actual degree of the vertex:

$$\sum_j P_{ij} = k_i. \tag{2.5}$$

If it is supposed that beyond the degree distribution the edges are placed at random, then the probability of two nodes connecting depends on their degrees. Supposing the probabilities for each end of a single edge is independent, which is a reasonable assumption in a large network [Newman, 2010], where all the degrees are small relative to the total number of edges, $P_{ij} = f(k_i)f(k_j)$ for some function $f$ on their degrees. Then, by equation 2.5:

$$\sum_{j=1}^{n} P_{ij} = \sum_{j=1}^{n} f(k_i)f(k_j) = f(k_i) \left[ \sum_{j=1}^{n} f(k_j) \right] = k_i, \tag{2.6}$$

so $f(k_i) = Ck_i$, for some constant $C$, and we get

$$2m = \sum_{i,j} P_{ij} = C^2 \sum_{i,j} k_i k_j = (2mC)^2, \tag{2.7}$$

so, $C = 1/\sqrt{2m}$ which gives the probability $P_{ij}$ as

$$P_{ij} = \frac{k_i k_j}{2m}, \tag{2.8}$$

21

and the modularity is

$$Q = \frac{1}{2m} \sum_{i,j} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(g_i, g_j). \tag{2.9}$$

The modularity gives values between $-1/2$ and 1 for any clustering. Since it is known that not dividing the network into separate clusters gives a modularity of zero, this gives us a lower bound for "good" clusterings.

The benefit of finding a good clustering can be seen in figure 2.1; a good clustering can reveal what the nature of the network is, rather than it being a mess of nodes and links. While the modularity itself is a measure of a given clustering of a network, the maximum modularity is a property of the network itself, which can tell much about the inherent community structure of the network. To actually determine what clustering will give the maximum modularity an exhaustive search would be required, which becomes unfeasible for networks of even a moderate size, as the number of potential clusterings is huge. Therefore, clustering algorithms are needed to maximise the modularity of the network.

## 2.3  Newman's eigenvalue algorithm

In [Newman, 2006b,a] Newman noted that if it was supposed that the network was split into two clusters, then the modularity could be redefined in terms of a quadratic form. Defining a vector $s$, which keeps track of the split,

$$s_i = \begin{cases} 1 & \text{if node } i \text{ in cluster 1} \\ -1 & \text{if node } i \text{ in cluster 2} \end{cases} \tag{2.10}$$

These vectors $s_i$ then contain the exact same information as the $2 \times n$ matrix $\delta(g_i, g_j)$, so $\delta(g_i, g_j)$ can be factored:

$$\delta(g_i, g_j) = \frac{1}{2} (s_i s_j + 1) \tag{2.11}$$

22

Then, the modularity is redefined as:

$$
\begin{aligned}
Q &= \frac{1}{2m} \sum_{i,j} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \frac{1}{2} (s_i s_j + 1) \\
&= \frac{1}{4m} \sum_{i,j} \left( A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j,
\end{aligned}
\tag{2.12}
$$

Let $\mathbf{B}$ be a matrix where $B_{ij} = A_{ij} - k_i k_j / 2m$, called the *modularity matrix*, then equation 2.12 becomes:

$$
Q = \frac{1}{4m} \mathbf{s^t B s}
\tag{2.13}
$$

Now, since $\mathbf{B}$ is an $n \times n$ symmetric matrix, it has $n$ real eigenvalues $\lambda_i$, with corresponding eigenvectors $u_i$. Ordering the $\lambda_i$ so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, then write

$$
\mathbf{s} = a_1 \mathbf{u_1} + \ldots + a_n \mathbf{u_n}.
\tag{2.14}
$$

The equation for the modularity becomes

$$
\begin{aligned}
Q &= \frac{1}{4m} \sum_i a_i \mathbf{u}_i^t \mathbf{B} \sum_j a_j \mathbf{u}_j \\
&= \frac{1}{4m} \sum_i \lambda_i (\mathbf{u}_i^t . \mathbf{s})^2
\end{aligned}
\tag{2.15}
$$

so, the positivity of the modularity depends completely on the $\lambda_i$, in particular the most positive eigenvalue $\lambda_1$. This means that to maximise the modularity the "split vector" $\mathbf{s}$ ought to be as close to the first eigenvector $\mathbf{u}_1$ as possible. To do this, choose $\mathbf{s}$ as follows:

$$
s_i = \begin{cases} 1 & \text{if } u_{1_i} > 0 \\ -1 & \text{if } u_{1_i} \leq 0 \end{cases}
\tag{2.16}
$$

This gives a clustering of the network into two smaller clusters. Of course, this split is not strictly in the direction of $\lambda_1$, but it is the best possible estimate, given two clusters. In fact, it has been noted from well-known networks, such as Zachary's karate club [Zachary, 1977], where friendships within a karate club were graphed prior to the club splitting,

that the absolute value of the $i$th entry of $\mathbf{u}_1$ gives a good indication of the "strength" of the membership of node $i$ to its cluster [Newman, 2006b]. That is, an entry $u_1, i$ with absolute value close to one can be observed to be a key intra-cluster node or hub. In the case of Zachary's karate club these intra-cluster hubs were the different instructors who split the club into two clubs.

If the modularity of the split is not positive, then reject the split and say that the best clustering is to leave the network as it is. Similarly, if the first eigenvalue $\lambda_1 = 0$ then say that the best clustering is to leave the network undivided, as it is known that the modularity matrix $\mathbf{B}$ always has a zero eigenvalue, with eigenvector $\mathbf{u} = (1, 1, \ldots, 1)$, which corresponds to no split of the network.

With the network split into two smaller clusters, the next question is how to split the network further. Newman [Newman, 2006b] recommends looking at the clusters one-by-one and splitting them until it gives no benefit to the modularity of the overall clustering. One could naively just look at the adjacency matrix of the cluster itself, and form the corresponding modularity matrix, but this ignores the connectivity of the overall network. That method could inadvertently split the subgraph into communities which would make sense within the cluster, but would ignore the connections from outside the cluster. Instead, the original modularity matrix $\mathbf{B}$ is used to calculate what difference a split of the subgraph would make to the modularity.

Given a cluster of nodes $g$, within the network, the *modularity contribution* $\Delta Q$ of a split of $g$ is:

$$\Delta Q = \frac{1}{2m} \left[ \frac{1}{2} \sum_{i,j \in g} B_{ij}(s_i s_j + 1) - \sum_{i,j \in g} B_{ij} \right]. \qquad (2.17)$$

This is the term in the modularity of the network that a split $\mathbf{s}$ of $g$ would change, minus the modularity of leaving the subgraph $g$ whole. Thus, a

"subgraph modularity matrix" $\mathbf{B}^{(g)}$ can be defined as:

$$\begin{aligned}
\Delta Q &= \frac{1}{4m}\left[\sum_{i,j\in g} B_{ij}s_i s_j - \sum_{i,j\in g} B_{ij}\right] \\
&= \frac{1}{4m}\sum_{i,j\in g}\left[B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}\right]s_i s_j \\
&= \frac{1}{4m}\mathbf{s}^t\mathbf{B}^{(g)}\mathbf{s}
\end{aligned} \tag{2.18}$$

and so $\mathbf{B}^{(g)}$ takes the value

$$B_{ij}^{(g)} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik} \tag{2.19}$$

for labels $i, j$ of nodes in the cluster $g$.

The same approach as before is used, the most positive eigenvalue of $\mathbf{B}^{(g)}$ determines the favoured split $\mathbf{s}$ to maximise $\Delta Q$. $\mathbf{B}^{(g)}$ still has the property that each of its rows sum to zero, so there still is a zero-eigenvalue that represents no split of the cluster. This can now be the stopping criterion for the algorithm; if the most positive eigenvalue is the zero-eigenvalue, then that subgraph is indivisible and stop splitting. It should be noted, however, that as the most positive eigenvalue gets smaller, relative to its negative eigenvalues, there may be no benefit to the modularity of accepting such a split. In this case check, by calculating $\Delta Q$ for the proposed split, to see if it is worth continuing splitting the subgraph. If $\Delta Q \leq 0$ then stop, otherwise continue splitting the graph. The algorithm is summarised in Algorithm 1.

This eigenvalue method for maximising the modularity has shown remarkably good results [Newman, 2006b], despite the apparent drawback of always splitting the network/subgraph in two parts. It is possible to split the network into more than two parts initially by using the first $m$ eigenvalues and using the $i$th entry of the eigenvectors as coordinates in $\mathbf{R}^m$ [Humphries, 2011], but this then requires a choice of number of clusters. This may also require calculating many more eigenvalues of the

**Algorithm 1** This is Newman's eigenvalue algorithm for maximising the modularity of a network.

  Calculate the modularity matrix **B**, where $B_{ij} = A_{ij} - k_i k_j / 2m$

  Find the most positive eigenvalue $\lambda_1$ and its eigenvector $\mathbf{u}_1$

  **if** $\lambda_1 = 0$ **then**

   Stop the algorithm with no split in the network.

  **else**

   Split network such that node $i$ in first group if $u_{1_i} > 0$ and in second group otherwise.

  **Ensure:**    Modularity of split $> 0$

   **end if**

   **for all** subnetworks $g$ **do**

    Calculate $\mathbf{B}^{(g)}$, where $B_{ij}^{(g)} = B_{ij} - \delta_{ij} \sum_{k \in g} B_{ik}$

    Find the most positive eigenvalue $\lambda_1^{(g)}$ of $\mathbf{B}^{(g)}$ and its eigenvector $\mathbf{u}_1^{(g)}$.

    **if** $\lambda_1^{(g)} = 0$ **then**

     Subnetwork can split no further.

    **else**

     Split network such that node $i$ in first group if $u_{1_i}^{(g)} > 0$ and in second group otherwise

  **Ensure:**      $\Delta Q > 0$ for split of subnetwork $g$.

     **end if**

   **end for**

large matrix **B**, which can be computationally expensive, rather than calculating a single eigenvalue for smaller and smaller matrices as above.

## 2.4 New network clustering algorithms

Using the definition of modularity as defined above in equation 2.9 to measure the community structure of given clusterings, two new algorithms were developed in the course of this thesis. They both use complexity algorithms, which can often find novel solutions to difficult problems, since they do not make assumptions about the nature of the solutions.

### 2.4.1 Genetic algorithm

Genetic algorithms have become widely used to solve complex optimisation problems where it is difficult to calculate the gradient of the objective function. The algorithm is based on the process of Darwinian evolution where successful genes remain in the gene pool and unsuccessful genes die out. There have been several clustering algorithms which use the genetic algorithm as their framework, such as [Pizzuti, 2008]. In this proposed algorithm the genes are simply the clusterings of a given network and the modularity is then the objective function.

The simplicity of a typical genetic algorithm is that it typically only needs a fitness/objective function to evaluate the fitness of solutions. A genetic algorithm is described in Algorithm 2.

To define a genetic algorithm, it is important to define what a solution is, and how the genes crossover and mutate from generation to generation. In this case, a solution is any clustering of the network. A fitness function is a function of solutions which should typically be non-negative. This can be attained by setting the fitness of any clustering with negative

**Algorithm 2** An example of a generic genetic algorithm.

Generate a random population of $n$ genes $g_i$, set generation to zero.

**while** Generation $< N$ **do**

    **for all** Genes $g_i$ **do**

        Calculate the fitness of $g_i$.

    **end for**

    Breed new population:

    **for** $i \in \{1, \ldots, n\}$. **do**

        Choose parents according to goodness-of-fit.

        Generate new gene $\tilde{g}_i$ by crossover of parent genes.

        Mutate $\tilde{g}_i$ with probability $\epsilon$.

    **end for**

    **for all** $g_i$ **do**

        $g_i \leftarrow \tilde{g}_i$

    **end for**

**end while**

modularity to zero, but since the lower bound is sharp it is better to use the modularity plus $1/2$ as the fitness function, $f$. Then a clustering $g_i$ is chosen as a parent with probability $f(g_i)/(\sum_j f(g_j))$.

It is important to maintain aspects of the "parent" genes when breeding the new genes; the number of clusters is a very important aspect of a clustering, so any algorithm must be careful not to simply crossover clusterings by taking their intersection as that would typically increase the number of clusters. The upper bound for the number of clusters should be set to the maximum of the numbers of clusters of the parents, or by setting $n_c = (n_i f(g_i) + n_j f(g_j)) / (f(g_i) + f(g_j))$.

Crossover is defined in this algorithm by randomly choosing a single node in the network according to the degree distribution, and then randomly choosing the cluster which contains the node in one of the parents. Remove all nodes in the chosen cluster from the selection pool, and continue the process. If the number of clusters reaches the predefined maximum number of clusters and some nodes remain, then the remaining nodes are placed in the closest cluster to them (by path distance). Mutation is defined by randomly moving a small percentage, approximately 15%, of nodes between clusters in approximately 10% of clusterings.

This algorithm is computationally very expensive for small networks, but scales much better than Newman's eigenvalue algorithm.

## 2.4.2   Simulated annealing

Another algorithm from complexity theory which has previously been used for finding community structure in networks is *simulated annealing.* Simulated annealing is an algorithm based on the technique of annealing in metallurgy; annealing happens when a metal is heated and then cooled slowly to increase the size of the crystals in the metal, as the metal finds the crystal structure which requires lowest energy as it cools slowly.

Simulated annealing mimics this process to find a solution that minimises an *energy* function, by introducing a *temperature* of sorts to the system. At each temperature level, there is a small change to the solution, and the new energy level is calculated.

The probability of accepting the change to the solution is then determined by an *acceptance probability function*, $P(E(s), E(s'), T)$, which depends on the current energy $E(s)$, the potential new energy $E(s')$ and the temperature $T$. The basic rule of the algorithm is that any change of state that reduces the energy of the system is automatically accepted, and a change of state that increases the energy is accepted with a probability that decreases as $T \to 0$.

The acceptance probability function used also has the property that small increases in energy are more likely to be accepted than big increases in energy. The acceptance probability is then a decreasing function of the change in energy $\Delta E = E(s') - E(s)$, which decreases faster as the temperature lowers. An exponential function satisfies these criteria:

$$P(E(s), E(s'), T) = \begin{cases} 1 & E(s') - E(s) < 0 \\ \exp\left(-\frac{E(s') - E(s)}{T}\right) & E(s') - E(s) \geq 0 \end{cases} \qquad (2.20)$$

The solution space for this algorithm was the space of clusterings of a given network $N$. The "energy" function was the modularity with the sign reversed.

It is important for simulated annealing that all moves are small. With this in mind, the move in this clustering algorithm is simply to select a node at random and move it to either any existing cluster or to form a new cluster from it. From observations, this algorithm did not appear to be very discriminating when required to split a network to more than two or three clusters, but it did seem to find "stable" solutions. For example, in Zachary's karate club network [Zachary, 1977] it found the correct split of the network, rather than the clustering which resulted in a slightly higher modularity.

## 2.5  $k$-medoids clustering

By defining a metric on spike trains, there is a way to tell how 'close' different responses are, but there are no co-ordinates in the metric space which would help with clustering responses. Provided that the number of stimuli is known beforehand, the standard method of clustering spike-trains is an analog to $k$-means clustering which does not require calculating a mean solution. [Julienne and Houghton, 2013] introduces a method for computing a 'mean spike train', so $k$-means clustering could now be used. However, here the process of $k$-medoids clustering is described.
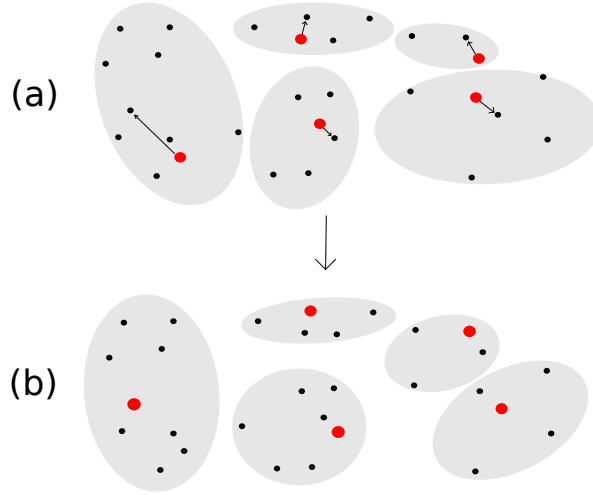


Figure 2.2: An example of the first step in k-medoids clustering: (a) Medoids are initially selected at random and the first clustering puts each point in the cluster of the closest medoid. Then we choose the point in the cluster that will give the lowest total distance to other points, and choose that as the new medoid. (b) Finally, we re-cluster each point to go in the cluster of the closest medoid to it.

For $k$-means clustering, $k$ random points are selected in the space in which clustering is occurring, then each point in the space would be in

the cluster of the closest of these $k$ points to itself. Then the mean of all the points in a cluster is calculated to get a new "centre" for the cluster and re-cluster each point to the closest of these $k$ means. This process is repeated until the means are stable, which gives a clustering of the points into $k$ clusters.

The $k$-medoids algorithm begins by choosing a pre-determined number, $k$, of points in the data, called medoids, and then each other point of the data is placed in the cluster of the closest, or least dissimilar), medoid to it. Then, for each cluster, each point in the cluster is assessed, and the point with the least overall distance is chosen as the new medoid. This step is shown in Figure 2.2. These steps are repeated until there is no change in the medoids. This method suits for spike-trains, as the idea of a 'mean' spike train is not well understood, although, using the van Rossum metric, the average function of Julienne and Houghton [2013] is a possible alternative.

There are some downsides to using $k$-medoids, that could possibly be improved by using network methods. The primary disadvantage is that the number of clusters must be selected in advance, and hence implies prior knowledge of the number of different stimuli that are presented in a data set. This is addressed in the next section.

## 2.6   Clustering responses with modularity

Here it is proposed that current methods of sorting responses could potentially be improved by using the modularity clustering algorithm. An advantage to such a dynamic method would be the ability to cluster responses without foreknowledge of the stimuli. This suggests that future dynamic sorting algorithms may be possible. Since the algorithm determines when to stop itself, it is simply be run to completion. Such a method could perhaps be used for many different data sets, when the

number of stimuli was not known, to determine the different responses.

The data set introduced in the introduction from [Narayan et al., 2006] was used, which featured spike trains from anaesthetised adult male zebra finches as they listened to conspecific songs.

A distance matrix was calculated by taking the van Rossum metric distances between each pair of responses for each neuron, where the parameter $\tau$ in the metric was chosen to be 12.8 ms, as this had previously been determined to be a good parameter for the data set used [Houghton, 2009]. Then, given a threshold value, $\mu$, the adjacency matrix for a network was calculated as:

$$A_{ij}^{\tau} = \begin{cases} 1 & \text{if } d(i,j) < \mu \\ 0 & \text{otherwise} \end{cases}. \qquad (2.21)$$

Once the adjacency matrix was calculated, the algorithm was run to maximise modularity for the network. The maximum van Rossum distance, once normalised, between any two trains was one, so the algorithm was run for different threshold values $\mu$ between zero and one incrementing $\mu$ by steps of size 0.001.

As seen in Figure 2.3, the profile of the graph of the maximum modularity versus the threshold looked promising, as for each cell the modularity peaked as $\mu$ varied, but unfortunately the clusters of responses bore little resemblance to the stimuli, and usually had too few clusters. Despite there being clear peaks in the modularity graphs, the maximised modularity consistently outperforms the modularity of the correct clustering for every value of the threshold $\mu$. This seems to imply that network clustering methods are probably not very good for sorting stimuli.

The modularity maximisation of these networks unfortunately cannot cluster the responses by their stimuli. A similar method was used in [Humphries, 2011], where all of the positive eigenvalues were used to cluster responses, . This method is rather computationally expensive, and uses even the very small positive eigenvalues, which themselves may
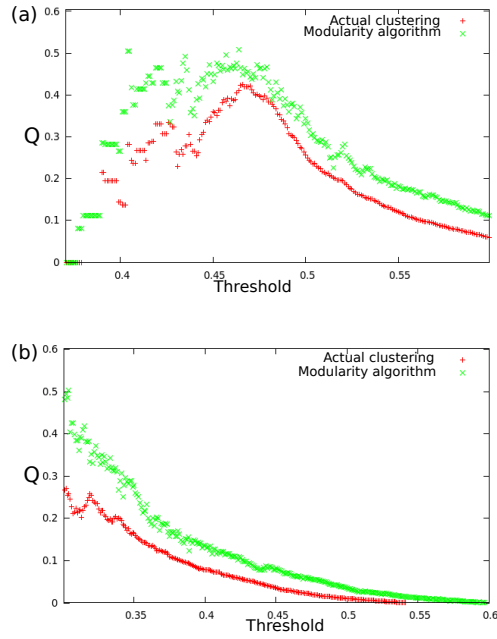
Figure 2.3: Above is the graph of the modularity versus threshold values in two typical cells from the data. The green crosses represent the modularity of the correct clustering of stimuli; the red plusses represent the maximum modularity, according to the algorithm, of the network.

have little effect on the positivity of modularity. This motivated the attempt to use the clustering method of Newman [2006a], but it did not perform very well.

The problem with the above method appears to be that there is no natural network formed by the distance matrix of responses to stimuli. The next section introduces a simulated network of neurons, so that there is a definite network structure to investigate.

## 2.7 Discussion

In this chapter, complex networks were introduced, along with a concept of goodness-of-fit for a specific clustering in the modularity. The eigen-

value algorithm introduced by Newman [2006a] was introduced in detail, as it has become the benchmark for clustering algorithms.

Two new algorithms were introduced to try to find a clustering that would maximise the modularity of a network. Neither produced results better than the standard methods, so were used sparingly. The simulated annealing algorithm appeared to not necessarily maximise the modularity, but seemed to find the highest 'stable' modularity.

Network methods did not cluster stimuli well, perhaps because the network structure was not natural. This may rule out network clustering methods as a tool for discriminating between responses, but [Humphries, 2011] seemed to get results with a different data set, so it is not conclusive. The simulated annealing algorithm introduce in this chapter could potentially provide interesting clusterings of stimuli, but that was not explored in the work for this thesis.

The dynamic response clustering appeared not to work due to a lack of network structure amongst the responses. The next chapter thus focuses on a simulated network of neurons with known network structure. This appears to be the optimal way to use network clustering in neuroscience.

# Chapter 3

# Mapping information flow in a simulated network of neurons

In this chapter, a process is proposed for a way to map how information flows through a network of neurons. The process is proposed due to the observation that given an inter-spike interval (ISI) distribution which arises from a neuron having a refractoriness after each spike, such as the gamma function or the inverted normal distribution, then a very small group of neurons, say 20 to 30, is sufficient for the collection of neurons together to produce an exponential distribution of ISIs - which would be the ISI distribution of a Poisson rate process. Due to this observation, it is proposed that there should be small clusters of neurons that work together to provide a background 'rate' to other similar groups of neurons.

There are different ways to form the networks of many neurons, but of primary interest are neurons whose firing seems to improve the probability of other neurons firing. A method of particular interest is the method of *Incremental Mutual Information* of Singh and Lesica [Singh and Lesica, 2010], which tries to reduce the uncertainty of a neuron as much as possible before checking whether another neuron influences it.

Any network formed from this measure would have to be directed, so a method to turn a directed network into an undirected network is needed. Newman describes a very neat way to do this in [Newman, 2010], where two nodes are related in the undirected network by how many commons nodes they point to in the directed network. This relation called the *bibliographic coupling* of two nodes. In a brain network, this could give a 'map' of information flow.

## 3.1   The bibliographic network

A directed network is defined as a collection of nodes and directed links between those nodes, represented by ordered pairs of nodes. As mentioned towards the beginning of this chapter, the adjacency matrix $\mathbf{A}$ of a directed network is then defined by:

$$A_{ij} = \begin{cases} 1 & \text{if there is a link from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The *bibliographic coupling* of two nodes, $i$ and $j$, in an unweighted directed network is defined to be the number of common nodes that are pointed to by nodes $i$ and $j$. Since an unweighted network typically has entries equal to one or zero, this coupling is calculated as:

$$B_{ij} = \sum_k A_{ki} A_{kj} \tag{3.2}$$

Thus the matrix $\mathbf{B}$ of bibliographic couplings is: $\mathbf{B} = \mathbf{A}^T \mathbf{A}$, a common symmetrisation of a non-symmetric matrix. The diagonal elements $B_{ii}$ are the number of nodes that node $i$ points to, or the *out degree* of node $i$.

The bibliographic network can then be defined in a couple of ways. It could be defined that each pair of nodes $i$ and $j$ with non-zero bibliographic coupling have a link, but this throws away a lot of information

from the original directed network. It is better to view the bibliographic network as a weighted network where each link $(i, j)$ has weight equal to the corresponding entry in the bibliographic matrix, $B_{ij}$, but without any self-links; so the bibliographic network has adjacency matrix equal to the bibliographic matrix with the diagonal entries set to zero.

## 3.2   Incremental mutual information

Incremental Mutual Information (IMI) is a measure defined by Singh and Lesica [Singh and Lesica, 2010] which is used to determine whether a pair of time-series have influence upon each other, for a given time-delay. It is a similar measure to transfer entropy [Schreiber, 2000], but it also uses future information to further eliminate any correlations due to noise.

Given two spike-trains $X$ and $Y$, a time $t$, and a time-delay $\delta$, the contribution to the IMI at time $t$ is defined to be:

$$\Delta I_{XY}[\delta] = H\left(X(t)|Z_\delta(t)\right) - H\left(X(t)|Z_\delta(t), Y(t-\delta)\right) \qquad (3.3)$$

where $Z_\delta$ is a vector containing information about the future and past of both spike-trains $X$ and $Y$, with a delay of $\delta$ added to $Y$:

$$Z_\delta(t) = (X_p(t), X_f(t), Y_p(t-\delta), Y_f(t-\delta)) \qquad (3.4)$$

This is calculated in the typical manner of calculating mutual information between spike-trains introduced in [Strong et al., 1998]. The spike-trains are discretised according to a time-scale which is unlikely to contain more than one spike, for example 2 ms which is approximately the absolute refractory period of most neurons. That is, the spike trains are split into vectors of bins which either contain a spike or do not, and as such are allocated either one or zero. The probability space is then calculated for each combination of "words" of ones and zeros.

The downside to this method is that this is a very data-hungry calculation, which limits the amount of 'past' and 'future' steps used, as

for $\Omega$ past and future steps, there are $2^{4\Omega+2}$ possible states. This means that calculating the IMI with two steps forward and back at each time step requires on the order of $2^{15}$ data points to fully populate the sample space.
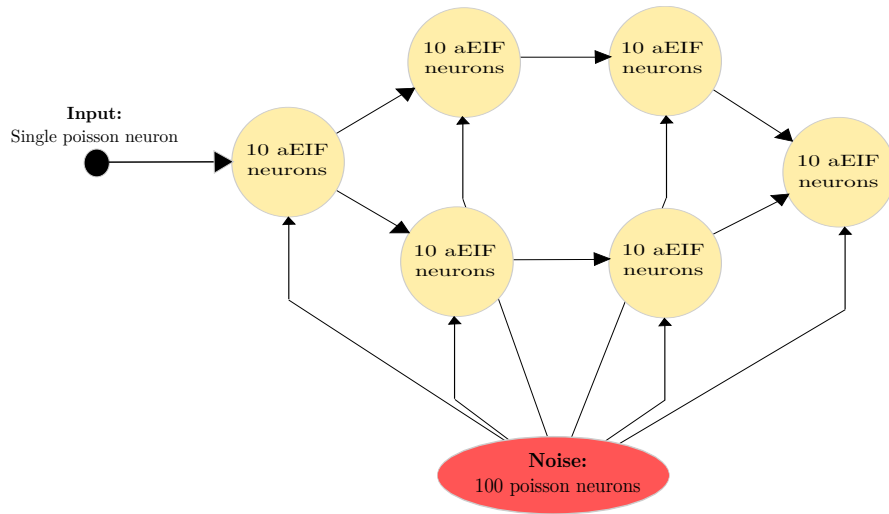
## 3.3   Numerical testing



Figure 3.1: The network model which was used to test the effectiveness of the incremental mutual information was based on the schematic shown here. All simulated neurons in the recurrent layer were aEIF neurons. The connection probability between two nodes at random was 0.1, connections along arrows in the diagram occurred with probability 0.8, and connections from the noise layer to the recurrent layer occurred with probability 0.2.

IMI requires very long data sets to calculate accurately, which are difficult to record in animals that are awake. Due to the scarcity of large simultaneous recordings for long periods of time, the proposed information mapping was instead carried out on a model network. The network

was designed to be small, for ease of calculations, but still had features of note - a divergence and a convergence of groups. The network was randomly generated based on the schematic in Figure 3.1. The neuron-model used was the adaptive exponential integrate and fire neuron-model of [Brette and Gerstner, 2005], as it strikes a good balance between ease of computation [Hopfield and Herz, 1995], and biological accuracy [Hodgkin and Huxley, 1952]. The network was simulated using the python package Brian [Goodman and Brette, 2008], and each simulation ran for four minutes. The amplitude of the spikes from the Poisson neurons in the noise layer was varied from 1 mV to 17 mV to investigate how well the IMI dealt with noise.
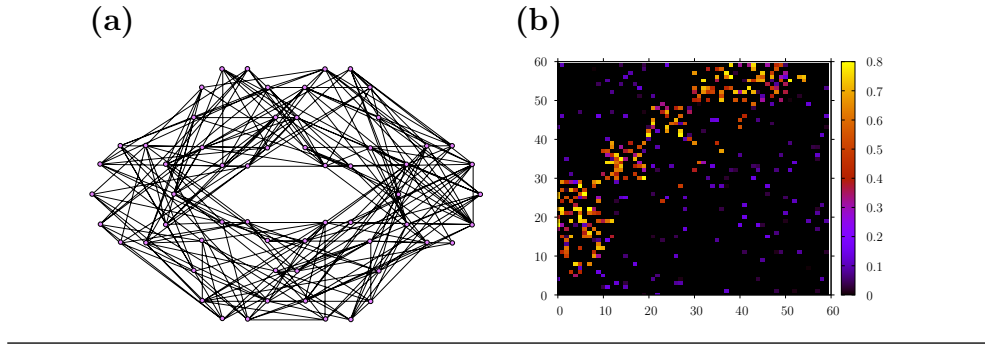


Figure 3.2: **(a)** A typical visualisation of the directed network, very difficult to see what sort of modules exist, despite the layout. **(b)** A heat map of the adjacency matrix of the recurrent layer of aEIF neurons, again there are very few clusters around the diagonal here, which would indicate a module in the network.

The proposed method to map the modules of neurons was to initially calculate the peak IMI between each pair of neurons, and set that equal to the non-diagonal elements of an adjacency matrix of a weighted directed

41

network:

$$A_{ij} = \begin{cases} \max_\delta \Delta I_{ij}[\delta] & \text{if } \delta > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (3.5)$$
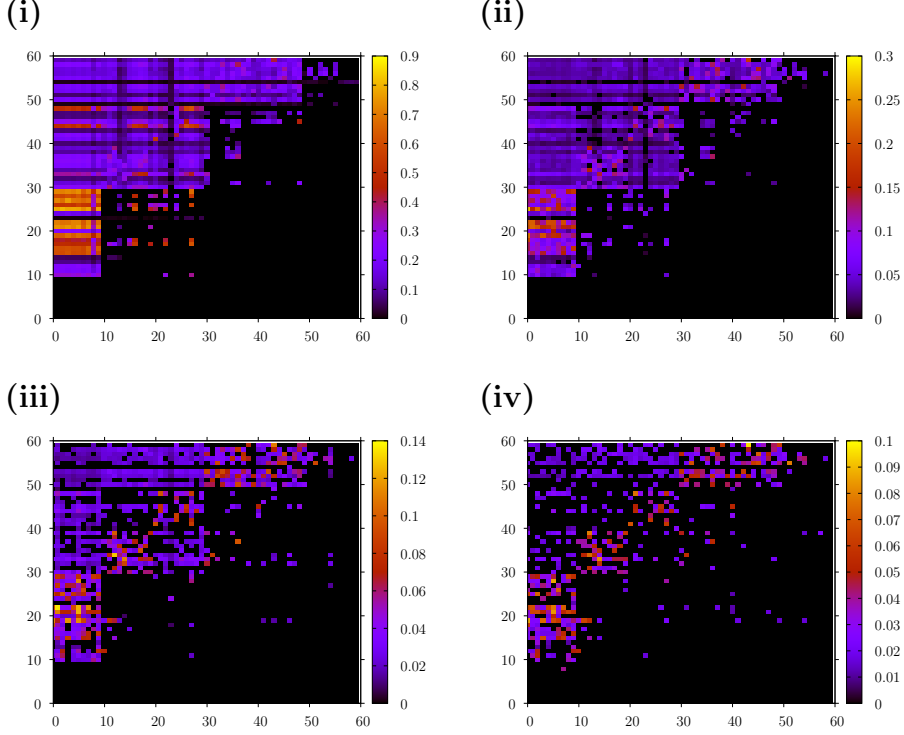
**(i)**



**(ii)**



**(iii)**



**(iv)**



Figure 3.3: Above are shown the peak IMI values between neurons, calculated from their spikes trains. The adjacency matrix is directed so that the peak IMI dictates the direction of influence of the neurons. The amplitude of the individual neurons in the Poisson noise layer increased at each step **(i)** 3 mV to **(iv)**15 mV in steps of 4 mV. Note that the IMI actually becomes more discriminating as the noise level increases.

The absolute value of the IMI was quite small, particularly as the noise level increased, but the important information was in the discriminability of the measure. Therefore, the non-zero IMI values were collected and the median was calculated. This median, $\tau_I$ was then set as the threshold for the network. The new adjacency matrix was then

calculated as:

$$A_{ij,\tau_I} = \begin{cases} 1 & \text{if } \max_\delta \Delta I_{ij}[\delta] > \tau_I \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

As can be seen in Figure 3.3, the IMI actually appears to improve in discriminability as the noise level increases. Accounting for noise was the reason for conditioning for the future along with the past [Singh and Lesica, 2010], as opposed to just conditioning for the past as Schreiber [2000] does with the Transfer Entropy measure. This means that the IMI could be a very good tool to measure influence between neurons in simultaneous recordings, but it does require a lot of data before it is useful, and it has a formidable calculation time.

The next step in the process is to calculate the bibliographic coupling from the thresholded adjacency matrix, $\mathbf{A}_{\tau_I}$, calculated in equation 3.6. This then gives a new network, which is undirected, and hence can be clustered using common methods such as [Newman and Girvan, 2004; Newman, 2006b].
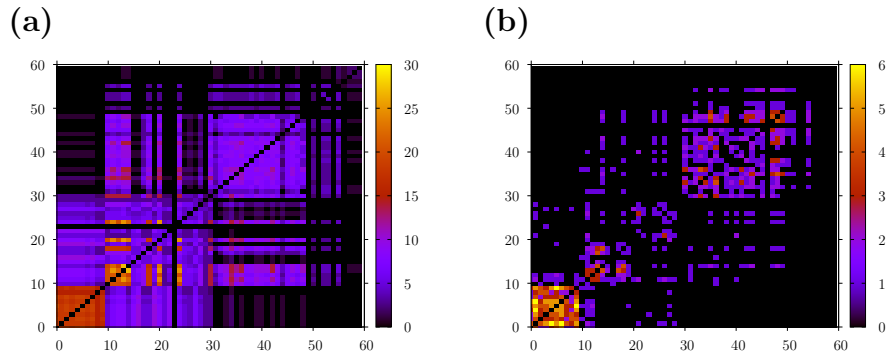


Figure 3.4: In this figure the thresholded bibliographic network is shown from the IMI network when: **(a)** the noise level is very low, that is 1 mV from each Poisson noise neuron, and **(b)** the noise level is much higher, 17mV from each Poisson neuron. The modular structure of the network is showing a lot more when the noise level is high.

In figure 3.4, the bibliographic network is shown to roughly group the first three groups together, but the fourth and fifth are also clustered together. This is because they both influence the sixth group, so they have high bibliographic coupling. It appears that the bibliographic coupling can detect a divergence of groups in a network, but cannot detect a convergence.

In order to correctly cluster the convergent groups towards the end of the model network, another measure needs to be introduced, the cocitation. This coupling is the inverse concept to the bibliographic coupling, as it is the number of common nodes pointing at both nodes. That is:

$$C_{ij} = \sum_k A_{ik} A_{jk} \tag{3.7}$$

The matrix is then $\mathbf{C} = \mathbf{B}\mathbf{B}^T$. This is another very common symmetrisation of matrices.

After running the whole process again with the cocitation instead of the bibliographic coupling, it is clear that the cocitation does indeed detect convergences in the groups of neurons, this can be seen in figure 3.5.

## 3.4 Discussion

The incremental mutual information proved to be a remarkably useful tool in noisy networks, but it requires very large data sets to calculate, these can be very difficult to record in vivo. The length of data sets required may rule out the IMI as a useful measure of real world data.

The bibliographic coupling and the cocitation both gave interesting clusterings of the network, but combining them to consistently return a provided network was a difficult task, which was unfortunately not completed in the course of this work.
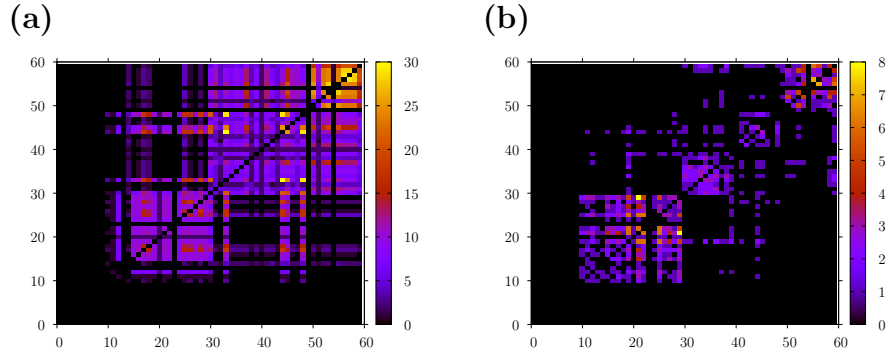
Figure 3.5: In this figure the thresholded cocitation network is shown from the IMI network when: **(a)** the noise level is very low, that is 1 mV from each Poisson noise neuron, and **(b)** the noise level is much higher, 17mV from each Poisson neuron. The modular structure of the network is showing a lot more when the noise level is high.

Perhaps attempting to map the information flow in a network is beyond current data capabilities, but there should be some consistent way to compare simultaneous recordings of many neurons. Previous multi-unit spike train metrics have had both time-scale and population parameters which needed to be tuned. By extending the time-scale free distance measures developed by Kreuz et al. [2007, 2013], this will reduce the need for at least one of the parameters.

# Chapter 4

# Multi-unit spike train metrics

In this chapter, two parameter-free spike train distances are introduced, and multi-unit extensions are then suggested. The ISI distance [Kreuz et al., 2007] was introduced as a parameter-free measure of spike train synchrony. That is, it compares the rates of two spike trains at any point in time. Similarly, the SPIKE distance [Kreuz et al., 2013] is a time-local parameter-free distance measure between spike-trains, which bears some resemblance to edit-length measures such as the Victor-Purpura metric [Victor and Purpura, 1997]. These time-local measures can then be integrated over the length of the spike-trains to give parameter-free distance measures between spike trains.

These multi-unit extensions are then tested on the data used in [Houghton and Sen, 2008] to determine their suitability as multi-unit measures. In testing these extensions, it is observed that there are different ways to consider how to code for a population of neurons which act together.

## 4.1   Parameter-free spike train distances

Two distances have been introduced by Kreuz et al. [2007, 2013] recently, which do not require a time-scale parameter such as the $q$ in the Victor-Purpura metric [Victor and Purpura, 1997] and the $\tau$ in the van Rossum

metric [van Rossum, 2001]. These distance measures are introduced below, and then several multi-unit extensions are proposed.

### 4.1.1 ISI distance

In [Kreuz et al., 2007] a new type of measure was introduced. The ISI distance is a time-local measure of spike train synchrony. The ISI distance is defined for any $t$, $0 < t < T$, in the trial. The distance is a local comparison of the estimated firing rates of spike trains $X$ and $Y$.

For any time $t$ in the trial, the firing rate of spike train $X$ is estimated by the reciprocal of the current inter-spike interval in $X$, and similarly for $Y$:

$$r_x(t) = \frac{1}{I_x(t)}, \qquad r_y(t) = \frac{1}{I_y(t)} \tag{4.1}$$

Typically one rate will be bigger than the other rate. Dividing by that rate a similarity measure, with values between zero and one, for the spike trains can be calculated:

$$s_{ISI}(X,Y)(t) = \begin{cases} r_x(t)/r_y(t) & , r_y(t) \geq r_x(t) \\ r_y(t)/r_x(t) & , r_x(t) > r_y(t) \end{cases} \tag{4.2}$$

Since the rates are estimated by the ISIs, and $1/a \geq 1/b \implies a \leq b$, this can be rewritten as:

$$s_{ISI}(X,Y)(t) = \begin{cases} I_y(t)/I_x(t) & , I_x(t) \geq I_y(t) \\ I_x(t)/I_y(t) & , I_y(t) > I_x(t) \end{cases} \tag{4.3}$$

This similarity measure is always between zero and one, so to change it to a distance measure between spike trains, all that had to be done was to subtract it from one. Then the ISI distance, as defined in [Kreuz et al., 2007, 2009b], is:

$$d_{ISI}(X,Y)(t) = \begin{cases} 1 - \frac{I_y(t)}{I_x(t)} & , I_x(t) \geq I_y(t) \\ 1 - \frac{I_x(t)}{I_y(t)} & , I_y(t) > I_x(t) \end{cases} \tag{4.4}$$
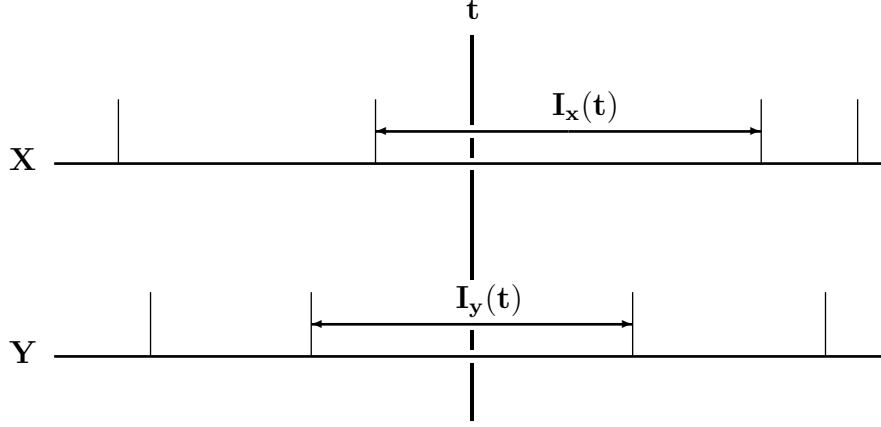
Figure 4.1: Shown here is an example pair of spike trains $X$ and $Y$. At each point $t$ in the trial there are respective intervals $I_x(t)$ and $I_y(t)$. By taking the reciprocal of $I_x(t)$ and $I_y(t)$, an estimator for the firing rate of $X$ and $Y$ at time $t$ is calculated.

The ISI distance is zero when the inter-spike intervals of $X$ and $Y$ are the same, and tends towards one as the ratio between them increases. It is worth noting that, unlike the van Rossum metric, the ISI distance does not require a timescale parameter, and thus can be used without tuning for an optimal parameter.

The ISI distance was found to have a simpler form, which benefits the extension work of Chapter Three of this thesis, as it shows the nature of the ISI distance to be that of an $L_1$ metric:

$$d_{ISI}(X,Y)(t) = \frac{|I_x(t) - I_y(t)|}{\max\{I_x(t), I_y(t)\}} \tag{4.5}$$

This time-local measure can be integrated over the length of the trial to give a metric on full spike trains.

$$d_{ISI}(X,Y) = \frac{1}{T}\int_0^T d_{ISI}(X,Y)(t)\, dt = \frac{1}{T}\int_0^T \frac{|I_x(t) - I_y(t)|}{\max\{I_x(t), I_y(t)\}}\, dt \tag{4.6}$$

### 4.1.2 SPIKE distance

In [Kreuz et al., 2011, 2013] another distance measure was proposed which does not require a time-scale parameter and which can also be used as a time-local measure of spike train synchrony. This measure is similar in ways to the metric of Victor and Purpura [1997] as it is a temporal distance measure rather than a rate metric like the ISI distance.

At each time $t$, there are four 'corner' spikes, the preceding and following spikes in both $X$ and $Y$. These spikes are denoted by $t_p^x, t_f^x, t_p^y$ and $t_f^y$, where $p$ stands for preceding and $f$ for following. Each of these spikes then has an associated 'gap' value, as shown in figure 4.2. This gap, $\Delta t_i^x$, is simply the shortest distance to any spike in the other spike train:

$$\Delta t_i^x = \min_j |t_i^x - t_j^y| \tag{4.7}$$

These gaps are similar to the translation costs in the Victor-Purpura metric, but they do not have an upper bound, or indeed a timescale parameter. Rather than introducing a somewhat arbitrary parameter such as $q$, the gaps are scaled according to the firing rate of the spike train at that point in time.

The firing rate is estimated as before with the ISI distance above, and is just the reciprocal of the current inter-spike interval. To get a continuous function of time, the distance measure interpolates smoothly between spikes in each spike train. So, each spike train must then be summed separately, for $X$:

$$S_X(X, Y)(t) = \frac{(t - t_p^x)\Delta t_f^x + (t_f^x - t)\Delta t_p^x}{I_x(t)} \tag{4.8}$$

and for $Y$:

$$S_Y(X, Y)(t) = \frac{(t - t_p^y)\Delta t_f^y + (t_f^y - t)\Delta t_p^y}{I_y(t)} \tag{4.9}$$

where the inter-spike interval value in the denominator is squared to normalise the measure and keep the distance between zero and one.
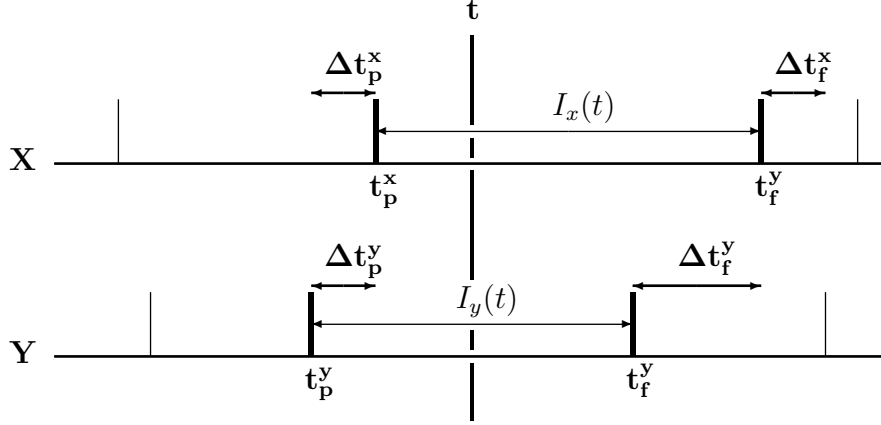
Figure 4.2: Shown here is the same pair of spike trains $X$ and $Y$ as in Figure 4.1, but here the measures relevant for the SPIKE distance [Kreuz et al., 2013] are highlighted. At each point $t$ in the trial there is a preceding and a following spike in each spike train, called $t_p^x, t_f^x, t_p^y, t_f^y$ respectively. At each of these, so called, 'corner' spikes a 'gap' is calculated. The gap for each spike is the shortest distance to a spike in the other spike train.

These distances are then multiplied by the inter-spike interval of the other spike train, and divided by the square of the average of the interstice intervals to get a symmetric distance measure of local spike times, called the SPIKE distance [Kreuz et al., 2013]:

$$S(X,Y)(t) = \frac{I_y(t)S_X(X,Y)(t) + I_y(t)S_Y(X,Y)(t)}{2[I_x(t) + I_y(t)]^2} \qquad (4.10)$$

As above with the ISI distance, the SPIKE distance can be integrated over the course of a trial to get a distance measure between spike trains.

In the study of spike-train metrics, often a multi-unit measure is desired, as many data sets record several neurons at once. A multi-unit measure is a distance between collections of labelled spike-trains. The SPIKE distance in [Kreuz et al., 2011] does not lend itself to a simple

multi-unit extension, so instead a simpler version of the SPIKE distance is extended. This alternative has the added advantage that the measure between two full spike trains can be calculated in a fraction of the time of the integral of the SPIKE distance.

## 4.2   Alternative SPIKE distance

With a view to extending the SPIKE distance to a multi-unit distance measure, it is useful to review the definition of the SPIKE measure provided in [Kreuz et al., 2011]: Given two spike-trains $X$ and $Y$, where $x = \{t_1^x, \ldots, t_n^x\}$ and $y = \{t_1^y, \ldots, t_m^y\}$, where $t_1^x, \ldots, t_n^x, t_1^y, \ldots, t_m^y$ are the spike times. The SPIKE distance in [Kreuz et al., 2011] has the nice property that the distance is bounded such that it is always between zero and one. Unfortunately, to achieve a natural extension to a multi-unit measure, this property is sacrificed. A simpler version of the distance, without the strict upper bound of one, is used.

The simpler distance is very quick and easy to calculate. A gap is associated with each spike; this is the distance to the nearest spike in the other spike-train. The total distance between two spike trains is then simply the sum of these gaps.

The original SPIKE distance included additional normalisation factors that have been omitted here. This simplifies the measure between complete spike trains greatly. The simpler measure is not strictly a metric between spike trains, as it fails the triangle inequality, but the standard SPIKE distance does also. This measure is certainly symmetric and non-negative, and the triangle inequality holds due the $L^1$ metric.

To form the time-local distance, a weighted sum of the gaps for the corner spikes is made, that is, the spikes preceding and following the time of interest in each of the two spike trains. The weighting is chosen so that the integral of the time-local function is just the sum of the gaps.

First, the gaps are calculated for each spike in the two spike-trains. This is simply the nearest spike in the other spike train:

$$\Delta t_i^x = \min_i(|t_i^x - t_i^y|) \tag{4.11}$$

At each time instant, these is a unique set of two corner spikes for each spike train: the preceding and following spikes from each spike train, which are labelled $t_P^x(t), t_F^x(t), t_P^y(t)$ and $t_F^y(t)$; these are, respectively, the preceding and following spikes in spike-train $X$ and the preceding and following spikes in spike-train $Y$.

For each spike-train, $W \in \{X, Y\}$, a time-local distance is then calculated using the associated gap of the four corner spikes for each spike-train, $w = x, y$:

$$s_w(t) = \frac{\lambda_P(t)\Delta t_P^w(t) + \lambda_F(t)\Delta t_F^w(t)}{I^w(t)} \tag{4.12}$$

where

$$\begin{aligned} \lambda_F^w(t) &= \frac{t - t_P^w(t)}{I^w(t)} \\ \lambda_P^w(t) &= \frac{t_F^w(t) - t}{I^w(t)} \end{aligned} \tag{4.13}$$

and $I^w(t)$ is the size of the interspike interval in which $t$ is contained:

$$I^w(t) = t_F^w(t) - t_P^w(t). \tag{4.14}$$

Now, the time-local distance for each neuron is added to give the overall time-local distance:

$$s(t) = s_x(t) + s_y(t). \tag{4.15}$$

This simplified time-local SPIKE distance has the advantage that its integral is simply the sum of the gaps of each spike; that is:

$$\int_0^T s(t)\, dt = \sum_w \sum_i \Delta t_i^w. \tag{4.16}$$

In practice this simplified version of the SPIKE distance, which is given the name Simple-SPIKE for the remainder of this thesis, produces similar time profiles to the version described in [Kreuz et al., 2011].

## 4.3 Extensions to multi-unit recordings

As recording methods have improved, there are now more and more data sets recorded by multi-probe recordings. These recordings can isolate many separate neurons simultaneously, but it is not clear how exactly these collections of spike-trains should be compared to each other. Typically, there are two extreme cases for multi-unit recordings, and a population parameter is varied to find the best mix.

The first extreme case would be where each neuron is carrying a completely independent signal to all the other neurons. In this case, each spike-train from a specific neuron should only be compared to other spike-trains from the same neuron. This is called a *labelled line* (LL) code. For a LL code, a spike train labelled $i$ will only be compared to other spike trains with the same label, and then the distances are summed over all the labels.

The second extreme case would be that all the neurons complement each other and carry a signal together as a community. In this case, it would not make sense to only compare spike trains with the same label; it would be expected that the signal would be carried by the whole population. This is called a *summed population* (SP) code. Typically for a SP code, the spike trains are all pooled together and they are then treated as one large spike train.

In [Houghton and Sen, 2008], the van Rossum metric which was introduced in Chapter One was extended to the multi-unit case, and in [Aronov et al., 2003] the Victor-Purpura metric was extended. A multi-unit distance is defined to be between two sets of spike-trains; $\mathbf{X} = \{X_i\}$ and $\mathbf{Y} = \{Y_i\}$, where $i \in 1 \dots N$ and the index $i$ labels the neuron.

The single-unit distance, Simple-SPIKE, from the previous section is extended to a multi-unit distance. All that is required to calculate is the gap associated with each spike in each set of spike trains. By including

the possibility that the nearest spike may belong to a different neuron in the other set, this is achieved. However there must a distance penalty associated with changing from one neuron to the other. This penalty is similar to the costs in edit-length metrics such as Victor-Purpura [Victor and Purpura, 1997].

In other words, it is necessary to introduce a parameter $k$, which quantifies a fictional distance between spikes fired in different cells. The size of this distance quantifies the importance of the labelling of the neurons, and varying $k$ interpolates smoothly from $k = 0$, a summed population (SP) code, to $k$ large, a labeled line (LL) code. While theoretically the LL code occurs as $k \to \infty$, in practice a LL code occurs when $k$ is on the order of the largest ISI in the trial. The gaps can then be calculated as follows:

$$\Delta t_\alpha^{x_i}(k) = \min_{\beta, j} \left( |t_\alpha^{x_i} - t_\beta^{y_j}| + k \left[ 1 - \delta(i, j) \right] \right) \tag{4.17}$$

where $\delta(i, j)$ is the Kroneker delta, which is one if $i$ and $j$ match and zero otherwise. Hence, if the spikes have the same label in $x$ and $y$ there is no added distance, otherwise $k$ is added to the time difference between the spikes. This is illustrated in Figure 4.3

The time-local distance measure is then defined as before, using the corner spikes normalised by the interspike-interval:

$$s_{\mathbf{X}}(t) = \sum_{\mathbf{x}_i \in \mathbf{X}} \frac{\lambda_{\mathrm{P}}(t) \Delta t_{\mathrm{P}}^{\mathbf{x}_i}(t) + \lambda_{\mathrm{F}}(t) \Delta t_{\mathrm{F}}^{\mathbf{x}_i}(t)}{I^{\mathbf{x}_i}(t)} \tag{4.18}$$

where

$$\lambda_{\mathrm{F}}^{\mathbf{x}_i}(t) = \frac{t - t_{\mathrm{P}}^{\mathbf{x}_i}(t)}{I^{\mathbf{x}_i}(t)} \tag{4.19}$$

and

$$\lambda_{\mathrm{P}}^{\mathbf{x}_i}(t) = \frac{t_{\mathrm{F}}^{\mathbf{x}_i}(t) - t}{I^{\mathbf{x}_i}(t)} \tag{4.20}$$

and $I^{\mathbf{x}_i}(t)$ is the length of the interval in the spike-train $\mathbf{x}_i$ containing $t$,
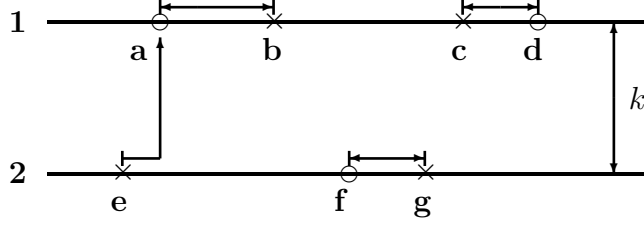
Figure 4.3: An example of how to calculate the gaps between spikes. If the crosses represent spikes from $\mathbf{X}$ and circles represent spikes from $\mathbf{Y}$, then in the picture above, the distance $k$ is simply the cost of relabelling a spike. Spike $\mathbf{e}$ in $X_2$ is much closer to spike $\mathbf{a}$ in $Y_1$ than $\mathbf{f}$ in $Y_2$, so it pays the cost $k$ to calculate its gap, $\Delta t_e^{x_2}$, according to spike $\mathbf{a}$ rather than spike $\mathbf{f}$.

$I^{\mathbf{x}_i}(t) = t_{\mathrm{F}}^{\mathbf{x}_i}(t) - t_{\mathrm{P}}^{\mathbf{x}_i}(t)$. As before:

$$s(t) = s_{\mathbf{X}}(t) + s_{\mathbf{Y}}(t) \tag{4.21}$$

Once more, this distance measure has the nice property that if the time-local measure is integrated over the course of the trial it equals the sum of the gaps of each spike:

$$\int_0^T s(t)\, dt = \sum_{\mathbf{W}} \sum_{\mathbf{w}_i \in \mathbf{W}} \sum_{\alpha} \Delta t_\alpha^{\mathbf{w}_i} \tag{4.22}$$

While this extension seems like the most natural extension of the Simple-SPIKE distance measure, it does not have a clear boundary for when it becomes an LL code. It appears that the distance stabilises at approximately twice the reciprocal of the rate, $2/r$, but this is not a precise bound. If $2/r$ is the upper bound, then that would give a nice symmetry to the upper bound for the translation of spikes in the Victor-Purpura metric, as was mentioned in the introduction. The penalty

parameter $k$ is difficult to compare to a more conventional population parameter which would vary between zero and one.

Motivated by [Houghton and Sen, 2008], another variant on the SPIKE extension was created. Houghton and Sen [2008] used geometry to compare the rate functions by rotating the vectors from perpendicular to parallel. Since the SPIKE metric is an $L^1$ metric, any rotations must be along the $(n-1)$-simplex in $\mathbb{R}^n$.

A simplex is a convex mathematical object in $\mathbb{R}^n$, which is defined as all points $(a_1, \ldots, a_n)$, such that $\sum_i a_i = 1$, and all $a_i \geq 0$. The 1-simplex is a line in $\mathbb{R}^2$, the 2-simplex is a triangle in $\mathbb{R}^3$ and the 3-simplex is a tetrahedron in $\mathbb{R}^4$.

If each label $i$ is initially positioned at the $i$th unit vector, $(0, \ldots, 1, \ldots, 0)$, then all the labels should travel along the simplex and meet at the centroid, $(1/n, \ldots, 1/n)$.

A way to use this geometry could be by dividing the distances between labels $i$ and $j$ by the $j$th coordinate of label $i$ on the simplex. That is:

$$\Delta t_\alpha^{x_i}(p) = \begin{cases} \min_\beta |t_\alpha^{x_i} - t_\beta^{x_i}| & , p = 0 \\ \min_{j,\beta} \left\{ \frac{n}{n-(n-1)p} |t_\alpha^{x_i} - t_\beta^{x_i}|, \frac{n}{p} |t_\alpha^{x_i} - t_\beta^{x_j}| \right\} & , 0 < p \leq 1 \end{cases}$$
(4.23)

While this equation seems completely different to equation 4.17 above, with a cost of $k$, this can actually be compared to it easily. If the distance between labels is kept constant, then this equation becomes:

$$\Delta t_\alpha^{x_i}(p) = \begin{cases} \min_\beta |t_\alpha^{x_i} - t_\beta^{x_i}| & , p = 0 \\ \min_{j,\beta} \left\{ |t_\alpha^{x_i} - t_\beta^{x_i}|, \left(1 + \frac{n-pn}{p}\right) |t_\alpha^{x_i} - t_\beta^{x_j}| \right\} & , 0 < p \leq 1 \end{cases}$$
(4.24)

So, there is an equivalence between $k$ and $p$, as follows:

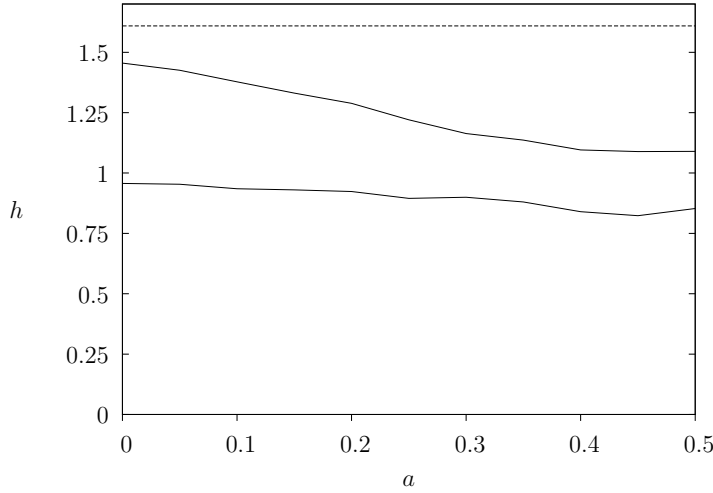$$k = \frac{n - pn}{n|t_\alpha^{x_i} - t_\beta^{x_j}|}$$
(4.25)

Figure 4.4: This figure shows the performance of the multi-unit Simple-SPIKE extension in identifying stimuli correctly in the data set described above. The plot shows the maximum and minimum values of the transmitted information for each value of the mixing parameter $a$, averaged over 20 trials. This compares favourably with Figure 3 in [Houghton and Sen, 2008].

### 4.3.1 Results

The new extended measures were tested on the [Houghton and Sen, 2008] test data as described in Chapter One. Two Poisson neurons form a receptive field and are each connected to two leaky integrate-and-fire neurons (LIF) with relative strength $a$ and $1 - a$. Hence, for $a = 0$ each receptive neuron is connected to a single LIF neuron, and for $a = 0.5$, each LIF neuron receives input equally from each of the receptive neurons.

In figures 4.4 and 4.6 the maximum and minimum values of transmitted information, $h$, are then plotted against the mixing parameter $a$, for the additive and multiplicative distances respectively. This allows com-
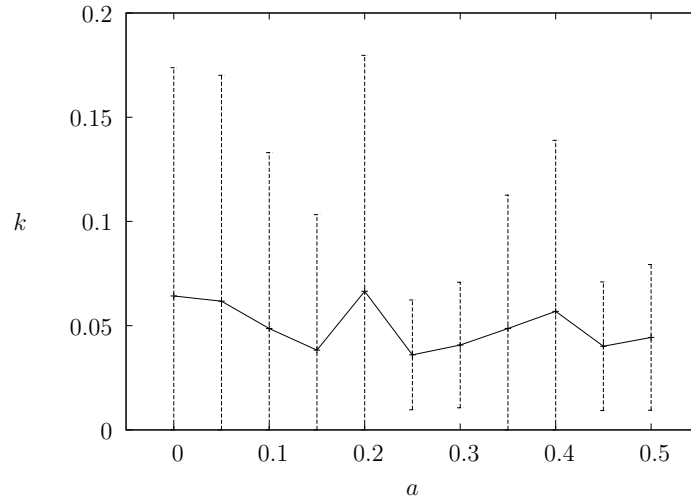
Figure 4.5: This figure shows the value for the penalty cost $k$ for the multi-unit Simple-SPIKE distance which gave the maximum transmitted information for each value of $a$. There does not appear to be any trend towards LL or SP in either direction, the best value was always a mix of the two.

parison with the multi-unit van Rossum and multi-unit Victor-Purpura metrics which are plotted in [Houghton and Sen, 2008]. In figures 4.5 and 4.7 the optimal values of population parameters are plotted against the mixing parameter $a$, to give a visual representation of how well each measure mixes according to the population parameters.
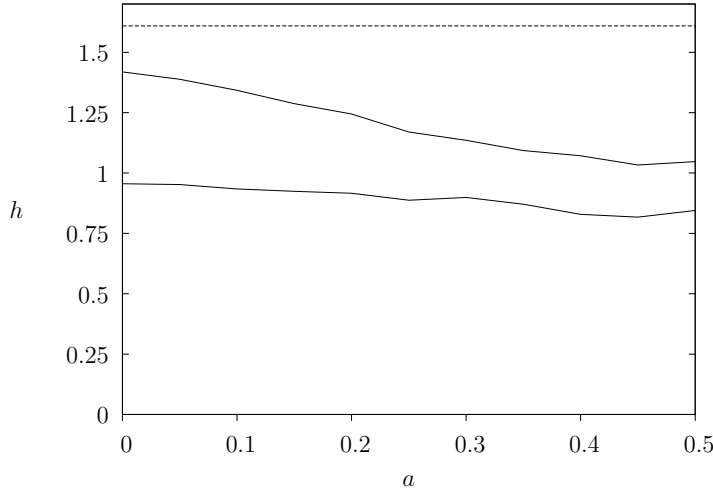
Figure 4.6: This figure shows the performance of the multiplicative multi-unit Simple-SPIKE extension, using the gaps from equation 4.23, in identifying stimuli correctly in the data set described above. The plot shows the maximum and minimum values of the transmitted information for each value of the mixing parameter $a$, averaged over 20 trials.

## 4.4 Multi-Unit ISI distance

### 4.4.1 Single unit recordings

With a view to extending the ISI distance to a multi-unit distance measure, it is useful to review the previous definition of the ISI distance between two spike trains $\mathbf{x}$ and $\mathbf{y}$ . The ISI distance can be thought of as a rate-comparison, where the inter-spike interval (ISI) is used as a proxy for the firing rate. As with all the time-local distance measures, it is defined as the integral over time of a local distance function $s(t)$.

Given two spike-trains $\mathbf{x}$ and $\mathbf{y}$, then $I^{\mathbf{x}}(t)$, $I^{\mathbf{y}}(t)$ are the inter-spike intervals at time $t$, for the spike-trains $\mathbf{x}$ and $\mathbf{y}$. The time-local distance
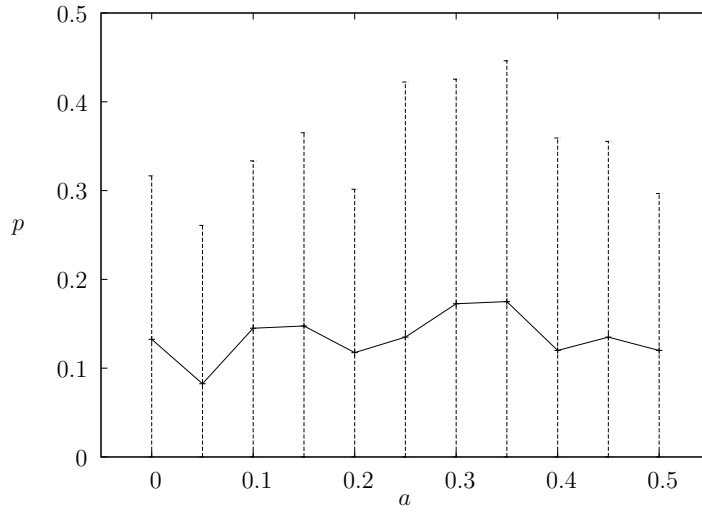
Figure 4.7: This figure shows the value for the population parameter $p$ for the multiplicative multi-unit Simple-SPIKE distance which gave the maximum transmitted information for each value of $a$. There does not appear to be any trend towards LL or SP in either direction.

function is then defined in Kreuz et al. (2007) as:

$$
s(t) = \begin{cases} 1 - I^{\mathbf{x}}(t)/I^{\mathbf{y}}(t) & \text{if } I^{\mathbf{x}}(t) \leq I^{\mathbf{y}}(t) \\ 1 - I^{\mathbf{y}}(t)/I^{\mathbf{x}}(t) & \text{otherwise} \end{cases}
\tag{4.26}
$$

From the point of view of generalising to more than one neuron, it is convenient to rewrite this as:

$$
s(t) = \frac{|I^{\mathbf{x}}(t) - I^{\mathbf{y}}(t)|}{\max(I^{\mathbf{x}}(t), I^{\mathbf{y}}(t))}
\tag{4.27}
$$

## 4.4.2 Initial extension to multi-unit recordings

In the multi-unit case a distance is defined to be between two multi-unit recordings. Thus, rather than two spike-trains there are two sets of spike-trains; $\mathbf{X} = \{\mathbf{x}_i\}$ and $\mathbf{Y} = \{\mathbf{y}_i\}$, where $i \in 1 \ldots N$ and the index $i$ is labelling the neuron.

61

The ISI-distance is a rate-based metric, so here a similar approach to multi-unit recordings as was used successfully for the van Rossum metric, another metric calculated using estimated rates, is used. That approach, outlined in [Houghton and Sen, 2008], replaces the rate with a rate-vector in an $N$-dimensional space. To do this, each neuron is assigned a unit vector describing its direction in the rate-space. At a given time, this is multiplied by the estimated rate of the neuron to give a rate-vector. The population rate-vector is then the sum of individual rate-vectors for each of the neurons in the population.

An $L^1$ norm, $|I^{\mathbf{x}}(t) - I^{\mathbf{y}}(t)|$, appears in the numerator of $s(t)$ in the single-unit case, so the natural extension of this idea to the multi-unit ISI-distance involves an $L^1$-structure on the $N$-dimensional space, which is the ISI-space in this case. In practice this means the individual vectors are unit vectors under the $L^1$-norm. For example for $N = 2$, one vector may be $(1, 0)$ and the other one would be $(1 - \alpha, \alpha)$. The corresponding individual neuron ISI-vectors would then be:

$$\mathbf{I}^{\mathbf{x}_1}(t) = \begin{pmatrix} I^{\mathbf{x}_1}(t) \\ 0 \end{pmatrix} \tag{4.28}$$

$$\mathbf{I}^{\mathbf{x}_2}(t) = \begin{pmatrix} (1 - \alpha)I^{\mathbf{x}_2}(t) \\ \alpha I^{\mathbf{x}_2}(t) \end{pmatrix}. \tag{4.29}$$

The overall population ISI-vector $\mathbf{I}^{\mathbf{x}}(t) = \mathbf{I}^{\mathbf{x}_1}(t) + \mathbf{I}^{\mathbf{x}_2}(t)$:

$$\mathbf{I}^{\mathbf{X}}(t) = \begin{pmatrix} I^{\mathbf{x}_1}(t) + I^{\mathbf{x}_2}(t) - \alpha I^{\mathbf{x}_2}(t) \\ \alpha I^{\mathbf{x}_2}(t) \end{pmatrix} \tag{4.30}$$

It remains to extend the definition of $s(t)$ to ISI-vectors. It is proposed here that this should be:

$$s(t) = \frac{\|\mathbf{I}^{\mathbf{X}}(t) - \mathbf{I}^{\mathbf{Y}}(t)\|_1}{\sum_i \max(I_i^X, I_i^Y)} \tag{4.31}$$

where $I_i^X$, $I_i^Y$ are the $i$th components of $\mathbf{I}^{\mathbf{X}}$ and $\mathbf{I}^{\mathbf{Y}}$ respectively, and:

$$\|\mathbf{I}^{\mathbf{X}}(t) - \mathbf{I}^{\mathbf{Y}}(t)\|_1 = \sum_i |I_i^X - I_i^Y| \tag{4.32}$$

One important property of any multi-unit distance measure is that it interpolates from the summed population (SP) code to the labelled line (LL) code . In the case of the labelled line code, it is necessary to consider what the result should be, for example in the Victor-Purpura metric the LL code is the sum of the distances of the individual neurons, whereas in the case of the van Rossum metric, the LL code is the Pythagorean sum of the distances. Here the LL code corresponds to when the vectors are all perpendicular, this is $\alpha = 1$ in the $N = 2$ example above. Up to an overall $L^1$ rotation, this means that the individual ISI vectors will have a single non-zero component, and $s(t)$ is given by:

$$s(t) = \frac{\sum_i |I^{\mathbf{x}_i}(t) - I^{\mathbf{y}_i}(t)|}{\sum_i \max(I^{\mathbf{x}_i}, I^{\mathbf{y}_i})} \tag{4.33}$$

Given the rational structure of the ISI-distance, this seems to be the natural structure for the LL code. Conversely, when all the vectors are parallel, the result is the SP code:

$$s(t) = \frac{|\sum_i I^{\mathbf{x}_i}(t) - \sum_i I^{\mathbf{y}_i}(t)|}{\max(\sum_i I^{\mathbf{x}_i}(t), \sum_i I^{\mathbf{y}_i}(t))} \tag{4.34}$$

in which, effectively, the ISIs are averaged across the population before being compared in the distance measure.

### 4.4.3 Results

This multi-unit extension of the ISI distance is tested on the same data that was used above to test the multi-unit SPIKE distance. The ISI extension does not cluster the responses as well as the SPIKE extension when the mixing parameter $a$ is low, but it clusters the responses very well when the signals are more mixed.

While the extension proposed above was derived logically from the single-unit case, it does not agree with the standard definition of a summed-population code, that is, it is possible to find a population of
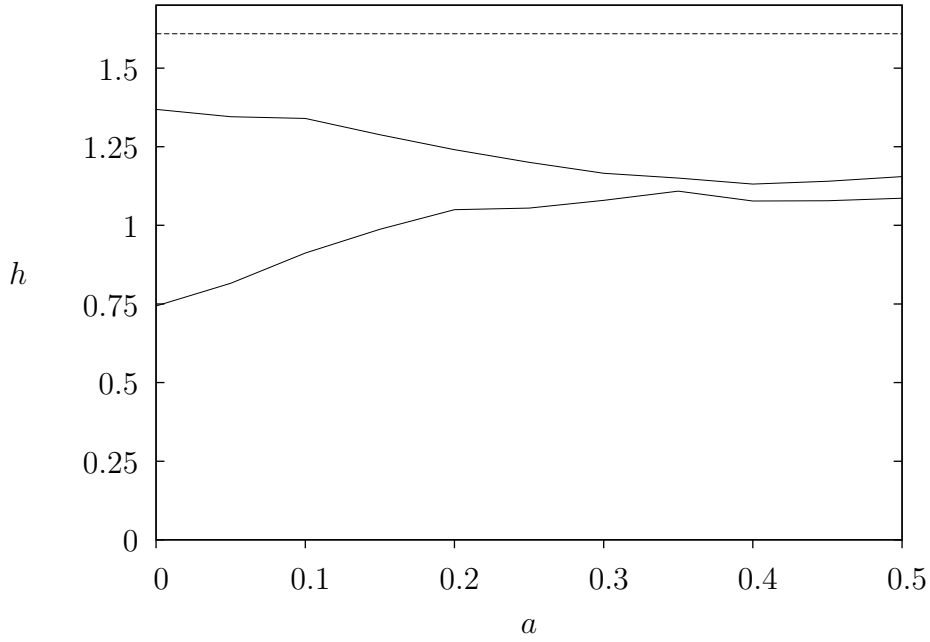
Figure 4.8: The dashed line at the top corresponds to the maximum possible transmitted information of log(5). Then the upper of the two remaining lines is the average of the maximum transmitted information as $\alpha$ is varied from zero to a half, and the bottom line is the minimum transmitted information over $\alpha$.

neurons which have spiked at exactly the same time, but where the average inter-spike intervals are not equal at a given time.

### 4.4.4 Alternative extensions to the multi-unit case

Upon comparing the above extension to the previous extension to the multi-unit case by Kreuz et al. [2009b], it became clear that there was a fundamental difference in how each distance measure viewed the concept of a rate-based metric.

The extension proposed in [Kreuz et al., 2009b] interpolates between the average of the individual ISI distances and the ISI distance of the two

"population neurons", that is, treating the individual spike times from the entire population as though they were all from the same neuron. This is done with a "population parameter" $p$, which runs from 0 (SP) to 1 (LL). This is given by:

$$s_p(t) = (1-p)\left(\frac{|I^{\mathbf{x}}(t) - I^{\mathbf{y}}(t)|}{\max(I^{\mathbf{x}}, I^{\mathbf{y}})}\right) + p\left(\frac{\sum_i |I^{\mathbf{x}_i}(t) - I^{\mathbf{y}_i}(t)|}{\sum_i \max(I^{\mathbf{x}_i}, I^{\mathbf{y}_i})}\right), \quad (4.35)$$

with notation as before, and $I^{\mathbf{x}}(t)$ is the interval in $\mathbf{x}$ at time $t$, where the population is viewed as a single neuron.

The extension proposed above can be compared to the extension in [Kreuz et al., 2009a] by replacing the ISI of the "population neuron" with the average of the ISIs across the population of neurons. This leads to the following equation:

$$s_a(t) = (1-p)\left(\frac{|\sum_i I^{\mathbf{x}_i}(t) - \sum_i I^{\mathbf{y}_i}(t)|}{\max(\sum_i I^{\mathbf{x}_i}(t), \sum_i I^{\mathbf{y}_i}(t))}\right) + p\left(\frac{\sum_i |I^{\mathbf{x}_i}(t) - I^{\mathbf{y}_i}(t)|}{\sum_i \max(I^{\mathbf{x}_i}, I^{\mathbf{y}_i})}\right),$$
$$(4.36)$$

.

Figure 4.9 shows the different behaviour of the two extensions. The 'average' extension depends heavily on the chosen population parameter when there is not much mixing of inputs, and does not depend on the parameter as much when the input is well-mixed. On the other hand, the 'population' extension behaves just like the previous extensions in [Aronov et al., 2003; Houghton and Sen, 2008].

There is a fundamental difference in how each of these distance measures view what a SP metric should be. The example from [Kreuz et al., 2009b] assumes that there is information being carried by the frequency of arrival of spikes across the population, regardless of origin, and the second measure using the average ISI assumes that each neuron is essentially carrying the same message, with an inherent noise due to the biological constraints of neurons. Each argument has its own merits, and it could be useful to have both of these measures, as in equations 4.35 and 4.36, depending on the focus of the experiment.
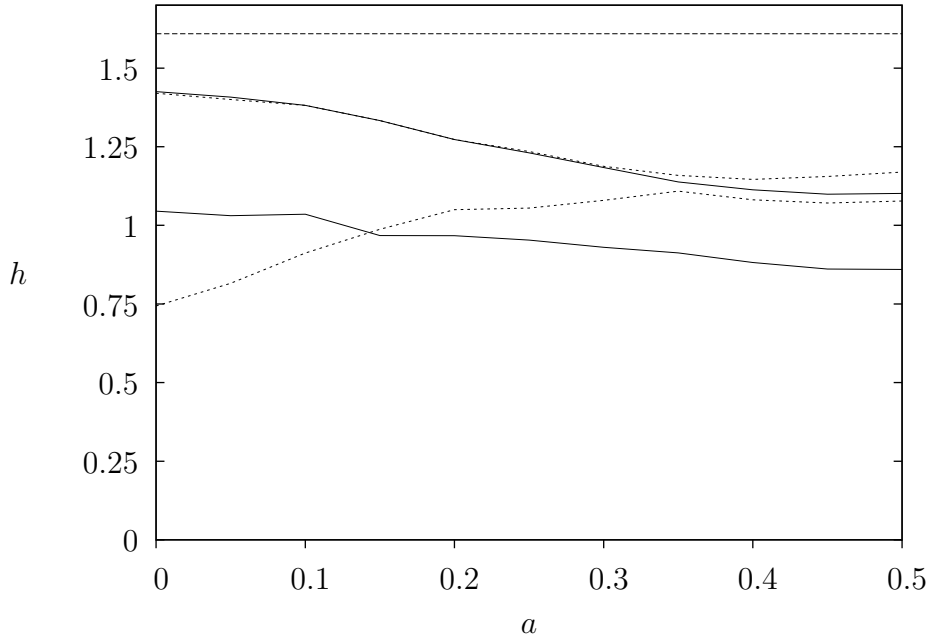
Figure 4.9: The lines, as in figure 4.8 correspond to the maximum and the minimum for: $s_p$, the 'population' extension, in the solid lines; and $s_a$, the 'average' extension, in the broken lines. The 'average' extension has a large difference between the maximum transmitted information and minimum when there is minimal mixing of input, and small difference when the inputs are fully mixed. The 'population' extension, however, performs exactly like previous extensions.

## 4.5 Adaptive ISI distances

The downside to the measures introduced above is that there must be parameters to vary between the SP and LL cases, and thus far there is no way to choose a parameter, which often leads to a grid search to find the optimal parameter. A goal of this work is to find a way for the data itself to choose its own population parameter. The SPIKE and ISI distances for single neurons are both parameter-free, so it would be ideal
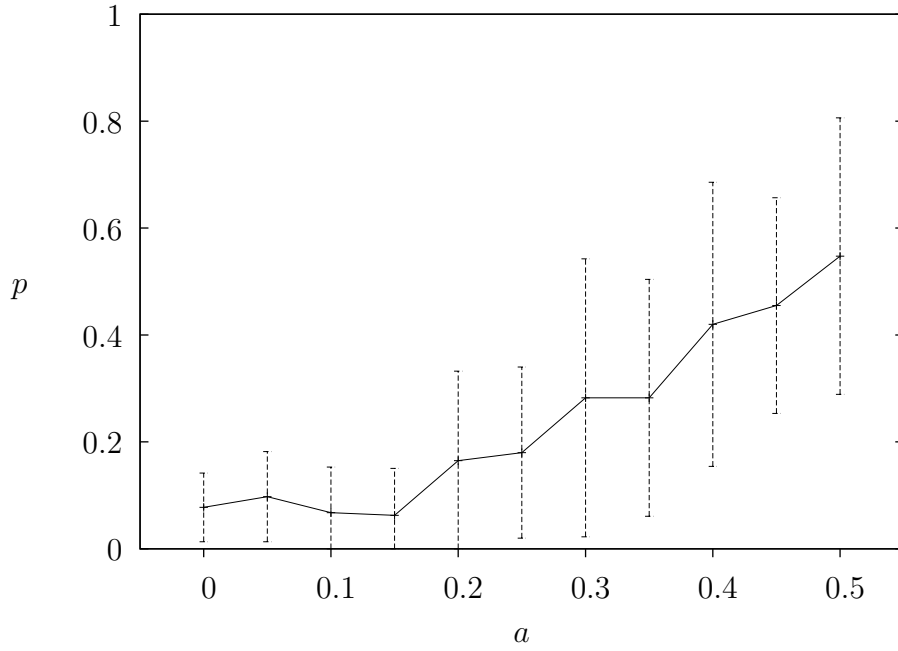
66

Figure 4.10: This figure shows the optimal value for the population parameter $p$ for $s_a$, the 'average' extension, as the mixing parameter $a$ is varied from zero to a half, with the errorbars showing the standard deviation.

if an extension could be found that was also parameter-free.

Since the ISI distance is a distance based on the firing rate of the neurons, the assumption was made that if the rates of the individual neurons were reasonably similar throughout the two populations, then the correct code to use would be an SP code, whereas if they were very different, then a LL code should be used.

It is important to understand what ISI distance one would expect, on average, between two trains which exhibited the same firing rate. [Mulansky et al., 2015] calculates this expected distance. If two Poisson spike trains have firing rate $\lambda$, then the probability of being in an inter-
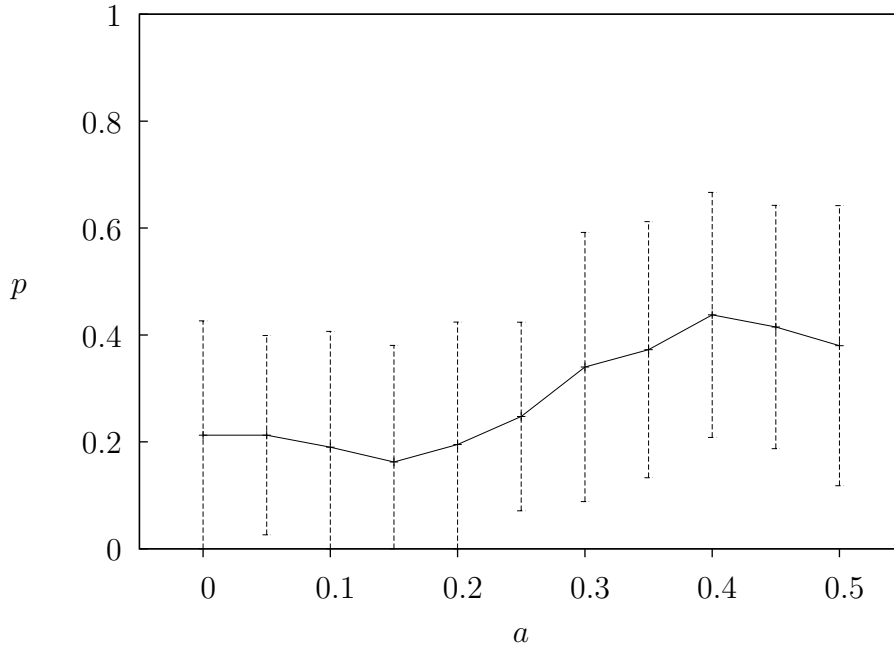
Figure 4.11: This figure shows the optimal value for the population parameter $p$ for $s_p$, the 'population' extension, as the mixing parameter $a$ is varied from zero to a half, with the errorbars showing the standard deviation.

spike interval of size $x$ at an arbitrary time $t$ would be:

$$p(x) = \lambda^2 x e^{-\lambda x}, \tag{4.37}$$

then the expected ISI distance between two Poisson process with constant rate $\lambda$ is:

$$\mathbf{E}[d_{ISI}(t)] = \lambda^4 \int_0^\infty \left( \int_0^\infty xy e^{-\lambda(x+y)} \, dy \right) dx \tag{4.38}$$

The $dy$ integral must be split into two integrals, $y < x$ and $y > x$, which gives:

$$\mathbf{E}[d_{ISI}(t)] = \lambda^4 \int_0^\infty \left( \left[ \int_0^x dy + \int_x^\infty dy \right] xy e^{-\lambda(x+y)} \right) dx = \frac{1}{4} + \frac{1}{4} \tag{4.39}$$

So, if two Poisson neurons have the same rate, then the expected ISI distance between them is 0.5.

In fact, given two Poisson neurons with a ratio $r$ between their firing rates, the expected ISI distance works out very nicely:

$$\mathbf{E}[d_{ISI}(t)] = \frac{1}{(1+r)^2} + \frac{1}{(1+1/r)^2} \qquad (4.40)$$

This then led to an adaptive distance where $p$ at a given moment in time was set to either one or zero, depending on the average distance between spike trains in the same trial. As it can be observed from previous figures, such as 4.10 and 4.11, that the best population parameter for the ISI distance is typically less than a half, $p < 0.5$, another adaptive distance was proposed which took population parameters $p = 0$ or $p = 0.5$ instead. These adaptive measures are summed up in Figure 4.12.

The zero-one adaptive ISI distance clearly does not work well for the 'population' distance, which suggests that the 'average' distance is more representative for rate-based measures. However, the 'population' distance performs very well with the zero-half adaptive distance, perhaps due to the small number of mixed stimuli.

## 4.6   Discussion

The different multi-unit extensions introduced in this chapter combine to give an interesting picture of multi-unit extensions. It is not clear precisely what an SP code should look like, as it has to be considered whether the distance measure is rate-based, like the ISI distance is, or based more on temporal precision, like the SPIKE distance. It appears that an average of the population should be used in the case of rate-based measures, but a pooled population has been the standard for multi-unit extensions up to this point.

The adaptive measures developed all chose the population parameter at the specific point in time, so it is odd to compare their transmitted information to the maximum transmitted information depending on a
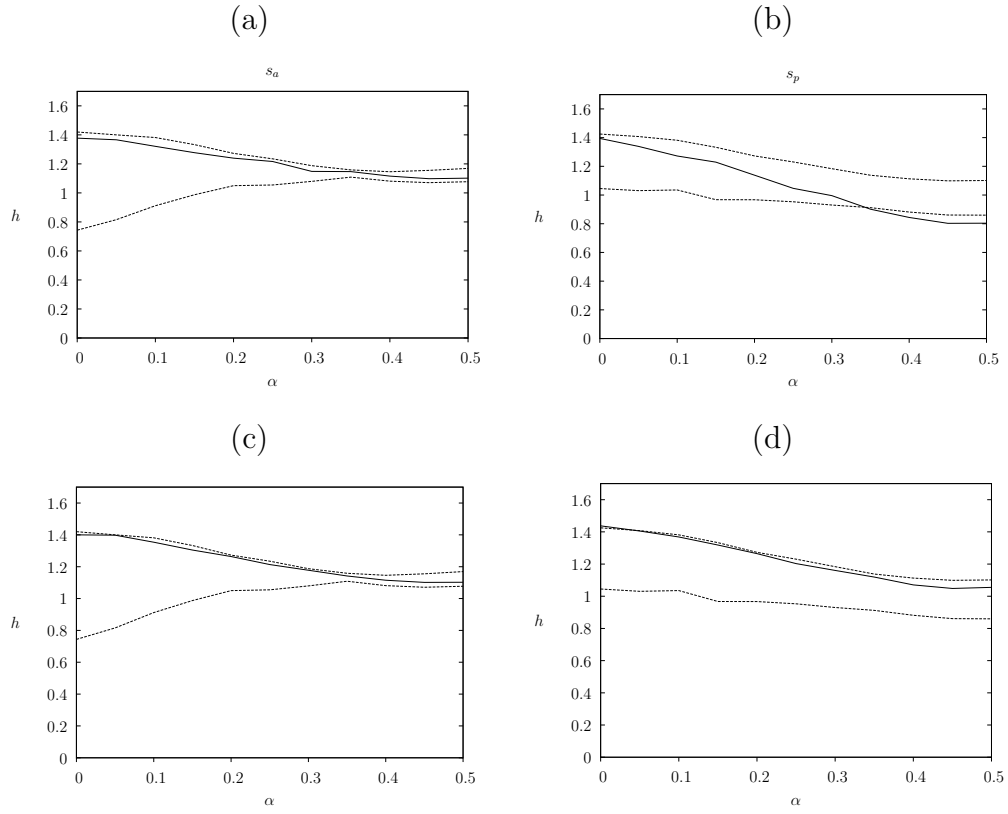
Figure 4.12: Above is the performance of the adaptive multi-unit ISI metrics. (a) $s_a$ with population parameter $p$ at each instant set to one if $s(t) < 0.5$ and set to zero otherwise. (b) $s_p$ with population parameter $p$ at each instant set to one if $s(t) < 0.5$ and set to zero otherwise. (c) $s_a$ with population parameter $p$ at each instant set to 0.5 if $s(t) < 0.5$ and set to zero otherwise. (d) $s_p$ with population parameter $p$ at each instant set to 0.5 if $s(t) < 0.5$ and set to zero otherwise.

static population parameter. The local population parameter was chosen as it can then be used for instantaneous spike train synchrony. There is definitely potential to choose population parameter for a trial as a whole, but it is beyond the scope of this thesis.

To improve the adaptive measures, it would be beneficial to understand the model of the background firing rate. In the next chapter a simple model is introduced, which could in future be used to improve the adaptive measures above.

# Chapter 5

# A simple neuron model

Sparse coding has been observed in neurons in the visual tract [Olshausen and Field, 2004], and it is suspected that it should occur in the auditory tract to pick out features of songs or speech. In this chapter a simple model is proposed for such neurons. This simplified model consists of a Poisson process which is either in an excited or depressed state, with two associated firing rates. If it is assumed that a spike train has been generated by this model then the bayesian firing rate is calculated.

Such a firing rate leads to a very specific ISI distribution, and so the zebra finch data used in chapter two is used to confirm whether or not such a firing rate is likely. It is found that sparse coding is likely to occur in the auditory forebrain of songbirds.

## 5.1   Neuron model

It is unlikely that a neuron is ever truly "off", so it is assumed that a neuron has a base-firing state that will be referred to as the "off-state". For the sake of simplicity, it is assumed that a neuron in such an "off-state" would have a constant firing rate, $\lambda_d$. Correspondingly, for sparse-coding a neuron must have a higher firing-rate when the feature that it codes for is present; this firing rate, $\lambda_u$, is also taken to be constant.

This idea is simplified to the extreme case, where a neuron is either in its up-state and has a high firing-rate, or it is "off". The model is treated as a Poisson process, for ease of calculation.

The data that is simulated has an 'up rate' $\lambda_u$ and a 'down rate' $\lambda_d$. For the initial trials, $\lambda_d = 0$. When in the down-state, the state switches 'up' with expected frequency $u$, and when in the up-state it switches 'down' with expected frequency $d$. Thus, the average background rate, $r$, of the inhomogeneous Poisson process can be calculated as:

$$r = \frac{\frac{\lambda_u}{d}}{\frac{1}{u} + \frac{1}{d}} = \frac{u\,\lambda_u}{u + d} \tag{5.1}$$

since the expected time for being in the up-state is simply $1/d$ and the expected time in the down-state is $1/u$.
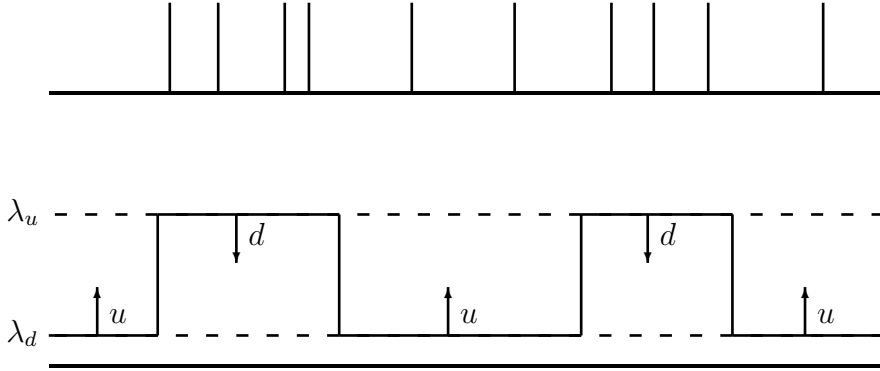


Figure 5.1: An illustration of the model firing rate, and an associated output spike train. The process has two states, an up-state and a down-state, and has Poisson transition rates: $u$ from down to up, and $d$ from up to down. Each state has an associated Poisson firing rate: $\lambda_u$ when up and $\lambda_d$ when down.

## 5.2   Estimating the firing rate $r(t)$

With the model for the firing rate as above, it is possible to explicitly calculate the probability of being in the up-state for the time following a spike. In the initial setting, where $\lambda_d$ is set to zero, then it is known that when there is a spike that the rate is in the up-state, so at the time of spiking $t_0$ the probability $p(t_0)$ of being in the up-state is equal to one.

For the estimate of the rate, $\tilde{r}(t)$, this is reflected by setting $\tilde{r}(t_0) = \lambda_u$. The probability is reset to one every time there is a spike, so it is only required to calculate the probability of being in the up-state given that there has been no spike since the time, $t_0$, of the last spike.

It is possible, using Bayes' Theorem, to calculate a first approximation of the probability of being in the up-state at a time $t + \Delta t$, given a spike at time $t = t_0$ for small $\Delta t$.

Let $X =$ up at $t = t_0 + \Delta t$, $Y =$ no spike since $t = t_0$ and $Z =$ spike at $t = t_0$. Then,

$$P(X|Y|Z) = \frac{P(Y|X,Z)P(X|Z)}{P(Y|Z)} = \frac{(1 - \lambda_u\Delta t)(1 - d\Delta t)}{1 - r\Delta t} \qquad (5.2)$$

Then, it is possible to calculate the first approximation to the probability of being up at time $t$, by letting $t_0 = 0$, $\Delta t = t/n$.

$$P(\text{up at } t|Y, Z) = \lim_{n \to \infty} \prod_{k=1}^{n} \frac{(1 - \lambda_u t/n)(1 - dt/n)}{1 - rt/n} \qquad (5.3)$$

in the limit as $n$ goes to infinity all terms smaller than $1/n$ are disregarded, so:

$$P(\text{up at } t|Y, Z) = \lim_{n \to \infty} \prod_{k=1}^{n} \left(1 + t\frac{r - \lambda_u - d}{n}\right)$$
$$= \lim_{n \to \infty} \left(1 + t\frac{r - \lambda_u - d}{n}\right)^n \qquad (5.4)$$

This is the famous limit form of calculating the exponential function, so:

$$P(\text{up at } t|Y, Z) = \exp[(r - \lambda_u - d)] \qquad (5.5)$$

Recalling from equation 5.1 that $\lambda_u = r(u+d)/u$:

$$p = e^{-\frac{d(r+u)}{u}(t-t_0)} \qquad (5.6)$$

Comparing this to the exponential kernel in the Van Rossum metric [van Rossum, 2001], then:

$$\tau = \frac{u}{d(r+u)} \qquad (5.7)$$

This is clearly just the first approximation, since it does not take into account the possibility of the model switching into the down-state and back up. To calculate the full probability, let $y(t) = P(\text{up at } t | Y, Z)$, with $Y, Z$ as in equation 5.2 above. Then:

$$y(t+h) = y(t)P(\text{staying in up-state}|\text{up at t}|Y, Z)$$
$$+ (1 - y(t))P(\text{switching up}|\text{down at t}|Y, Z)$$

$$= y(t)\left(\frac{(1-dh)(1-\lambda_u h)}{1-rh}\right) + (1 - y(t))\left(\frac{(uh)(1)}{1-rh}\right) \qquad (5.8)$$

$$= y(t)\left(1 + h(r - d - \lambda_u - u)\right) + uh$$

This leads to the following ordinary differential equation:

$$\frac{dy}{dt}(t) = (r - d - u - \lambda_u)\, y(t) + u \qquad (5.9)$$

This ODE has the general solution:

$$y(t) = Ce^{(r-d-u-\lambda_u)(t-t_0)} + \frac{u}{u+d+\lambda_u - r} \qquad (5.10)$$

Since $y(t_0) = 1$:

$$y(t) = \frac{d + \lambda_u - r}{d + u + \lambda_u - r}e^{(r-d-u-\lambda_u)(t-t_0)} + \frac{u}{u + d + \lambda_u - r} \qquad (5.11)$$

Letting $\lambda_u = r(u+d)/u$, this becomes:

$$y(t) = \frac{d(u+r)}{u^2 + d(u+r)}e^{-\frac{u^2+d(u+r)}{u}(t-t_0)} + \frac{u^2}{u^2 + d(u+r)} \qquad (5.12)$$

76

This solution was tested against a standard exponential model, which was minimised over $\tau$ with a downhill simplex method [Nelder and Mead, 1965]. Downhill simplex is a non-parametric minimisation algorithm that searches the solution space through reflection and stretching of a simplex.

Figure 5.2 shows how the ratio of the predicted value for $\tau$ over the optimised value rises towards one as the value for $u$ increases. This suggests that, while this method is clearly not completely correct, that it is at least a reasonable approximation.

## 5.3   Markov Process

The transition between the up and down states is defined only by the switching rates, and not the rate history. In this way, the rate process is a continuous-time Markov chain (CTMC). That is, the future of the process does not depend on the past at all, merely whether the model is
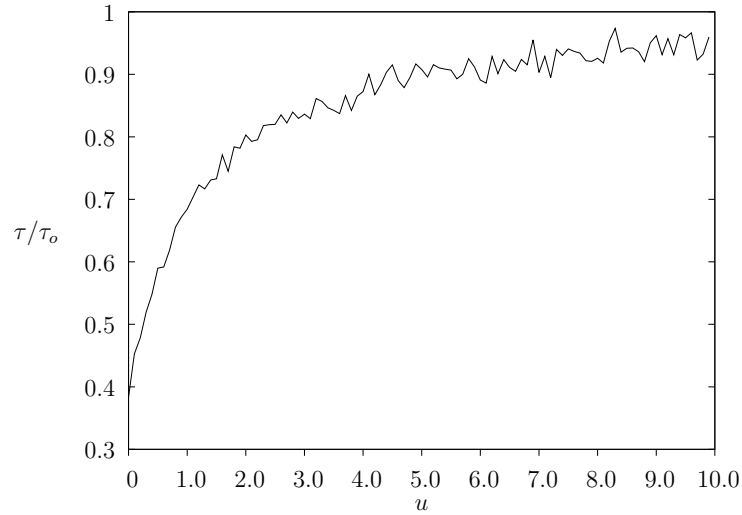


Figure 5.2: This figure shows the ratio of the predicted value for $\tau$ in an exponential model over the empirical optimised value $\tau_o$.

currently in the up-state or the down-state.

By the theory of Markov Chains, this process can be described by the simultaneous differential equations:

$$\begin{cases} p_u'(t) = -dp_u(t) + dp_d(t) \\ p_d'(t) = up_u(t) - up_d(t) \end{cases} \tag{5.13}$$

This is simply the linear ODE:

$$P'(t) = P(t)Q \tag{5.14}$$

where

$$Q = \begin{pmatrix} -d & d \\ u & -u \end{pmatrix} \tag{5.15}$$

Then the solution to the matrix $P(t)$ is simply the exponential of the transition matrix, $Q$:

$$P(t) = e^{tQ}. \tag{5.16}$$

The solution to the Markov chain of rates is then:

$$P(t) = e^{tQ} = \begin{pmatrix} \frac{u}{u+d} + \frac{d}{u+d}e^{-(u+d)t} & \frac{d}{u+d} - \frac{d}{u+d}e^{-(u+d)t} \\ \frac{u}{u+d} - \frac{u}{u+d}e^{-(u+d)t} & \frac{d}{u+d} + \frac{u}{u+d}e^{-(u+d)t} \end{pmatrix} \tag{5.17}$$

In order to calculate the probability of being in the up-state at a time $t$ after a spike at $t = 0$, it is necessary to introduce another state to the Markov chain. At any time $t$ between two spikes, $t = 0$ and $t = t_1$ say, there is implicit knowledge that there was a spike at time $t = 0$, and that there has not been a spike since. If the spiking state is treated as the end of the particular Markov chain, as a default is in financial mathematics [Kijima and Komoribayashi, 1998], then, provided there has been no spike since $t = 0$, it suffices to calculate the exponential of the transition matrix $Q_s$ below:

$$Q_s = \begin{pmatrix} -d - \lambda_u & d & \lambda_u \\ u & -u & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{5.18}$$

78

The third state of the Markov chain described by the transition matrix $Q_s$ is an "absorbing" state, the state of spiking, from which it is impossible to return to either the up or the down state.
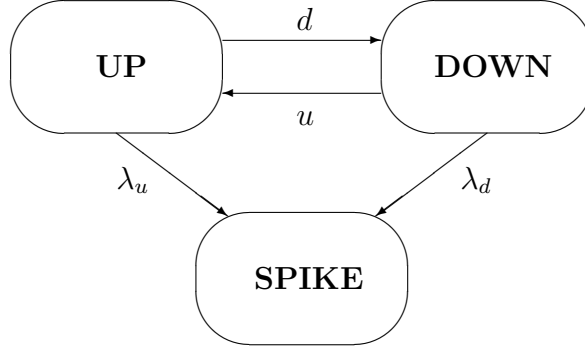


Figure 5.3: An illustration of the state space of the Markov chain at any point in time between two spikes.

The solution to the spiking Markov chain is:

$$P(t) = \begin{pmatrix} Ae^{-\alpha t} + (1-A)e^{-\beta t} & Be^{-\alpha t} - Be^{-\beta t} & p_{13}(t) \\ Ce^{-\alpha t} - Ce^{-\beta t} & (1-A)e^{-\alpha t} + Ae^{-\beta t} & p_{23}(t) \\ 0 & 0 & 1 \end{pmatrix} \quad (5.19)$$

where $-\alpha, -\beta$ are the roots to the characteristic polynomial of $Q_s$, and $p_{13}(t), p_{23}(t)$ are such that their rows sum to one.

$$\begin{aligned} -\alpha &= \frac{-(u+d+\lambda_u) + \sqrt{(u-d-\lambda_u)^2 + 4ud}}{2} \\ -\beta &= \frac{-(u+d+\lambda_u) - \sqrt{(u-d-\lambda_u)^2 + 4ud}}{2} \end{aligned} \quad (5.20)$$

Since $u, d, \lambda_u > 0$, these probabilities are simply double exponentials, and there are no trigonometric terms.

Specifically:

$$A = \frac{\left(u - d - \lambda_u + \sqrt{(u - d - \lambda_u)^2 + 4ud}\right)}{2\sqrt{(u - d - \lambda_u)^2 + 4ud}}$$

$$B = \frac{d}{\sqrt{(u - d - \lambda_u)^2 + 4ud}} \tag{5.21}$$

$$C = \frac{u}{\sqrt{(u - d - \lambda_u)^2 + 4ud}}$$

It can be easily confirmed that $A, B, C > 0$, which is necessary to ensure that all the probabilities are between 0 and 1.

If the model is changed such that there is a non-zero spiking probability, $\lambda_d > 0$, in the down-state, then the transition matrix $Q_s$ becomes:

$$Q_s = \begin{pmatrix} -d - \lambda_u & d & \lambda_u \\ u & -u - \lambda_d & \lambda_d \\ 0 & 0 & 0 \end{pmatrix} \tag{5.22}$$

The Markov chain with this transition matrix has the solution:

$$P_s(t) = \begin{pmatrix} Ae^{-\alpha t} + (1 - A)e^{-\beta t} & B\left(e^{-\alpha t} - e^{-\beta t}\right) & p_{13}(t) \\ C\left(e^{-\alpha t} - e^{-\beta t}\right) & (1 - A)e^{-\alpha t} + Ae^{-\beta t} & p_{23}(t) \\ 0 & 0 & 1 \end{pmatrix} \tag{5.23}$$

where $-\alpha, -\beta$ are once again the roots of the characteristic polynomial of $Q_s$.

$$-\alpha = \frac{-(u + d + \lambda_u + \lambda_d) + \sqrt{(u - d + \lambda_d - \lambda_u)^2 + 4ud}}{2}$$

$$-\beta = \frac{-(u + d + \lambda_u + \lambda_d) - \sqrt{(u - d + \lambda_d - \lambda_u)^2 + 4ud}}{2} \tag{5.24}$$

again, $u, d > 0$ means that the probabilities above are all double exponentials, with no trigonometric terms.

$$A = \frac{u - d + \lambda_d - \lambda_u + \sqrt{(u - d + \lambda_d - \lambda_u)^2 + 4ud}}{2\sqrt{(u - d + \lambda_d - \lambda_u)^2 + 4ud}}$$

$$B = \frac{d}{\sqrt{(u - d + \lambda_d - \lambda_u)^2 + 4ud}} \tag{5.25}$$

$$C = \frac{u}{\sqrt{(u - d + \lambda_d - \lambda_u)^2 + 4ud}}$$

To simplify notation, set:

$$\Delta = \lambda_u - \lambda_d$$
$$\gamma = \sqrt{(u - d - \Delta)^2 + 4ud} \tag{5.26}$$

Thus:

$$A = \frac{u - d - \Delta + \gamma}{2\gamma}, \ B = \frac{d}{\gamma}, \ C = \frac{u}{\gamma} \tag{5.27}$$

### 5.3.1   Estimating the rate function $r(t)$

These probabilities can now be used to calculate the estimated rate function, $r(t)$, based on the timing of spikes. Assuming that the probability of the last spike is known, then the rate function between any two spikes is estimated by this Markov model.

If the probability vector at the time of the last spike is $\mathbf{p_0} = (p_0, 1 - p_0, 0)$, then the probabilities at a time $t$ after the spike are:

$$\mathbf{p(t)} = \mathbf{p_0}e^{Q_s} = (p_u(t), p_d(t), p_s(t)) \tag{5.28}$$

where

$$p_u(t) = (p_0(A - C) + C)e^{-\alpha t} + (p_0(1 - A + C) - C)e^{-\beta t}$$
$$p_d(t) = (p_0(A + B - 1) + 1 - A)e^{-\alpha t} + (p_0(-A - B) + A)e^{-\beta t} \tag{5.29}$$
$$p_s(t) = 1 - p_u(t) - p_d(t)$$

The probabilities $p_u(t)$, $p_d(t)$ represent the probability of being in the up/down state and not spiking. The probability required to calculate the rate function, $r(t)$, between spikes is the probability of being in the up-state given that there has not been a spike. This is because it is known that there has been no spike since the last spike. Therefore, by Bayes' Theorem, the probability $p(t)$ is just:

$$p(t) = \frac{p_u(t)}{p_u(t) + p_d(t)} \tag{5.30}$$

so, the rate function becomes:

$$r(t) = \Delta p(t) + \lambda_d = \frac{A_0 e^{-\alpha t} + B_0 e^{-\beta t}}{C_0 e^{-\alpha t} + D_0 e^{\beta t}} \qquad (5.31)$$

where:

$$A_0 = \Delta(p_0(-u - d - \Delta + \gamma) + 2u) + \lambda_d C_0$$

$$B_0 = \Delta(p_0(u + d + \Delta + \gamma) - 2u) + \lambda_d D_0$$

$$C_0 = p_0(-2\Delta) + u + d + \Delta + \gamma \qquad (5.32)$$

$$D_0 = p_0(2\Delta) - u - d - \Delta + \gamma$$

Since $-\beta < -\alpha$, this can be written as:

$$r(t) = \frac{A_0 + B_0 e^{-\gamma t}}{C_0 + D_0 e^{-\gamma t}} \qquad (5.33)$$

so the asymptotic value of $r(t)$ is equal to the fraction $A_0/C_0$, so this value should not depend on $p_0$. Allowing $\delta = u + d + \Delta$, then $\delta^2 - 4u\Delta = \gamma^2$ which simplifies the calculation.

$A_0 = 2\Delta u + p_0\Delta(\gamma - \delta) + \lambda_d C_0$, and $C_0 = \gamma + \delta - 2\Delta p_0$:

$$\Delta p_0 = -\frac{C_0 - \gamma - \delta}{2} \qquad (5.34)$$

then

$$A_0 = 2\Delta u - \frac{1}{2}C_0(\gamma - \delta - 2\lambda_d) + \frac{1}{2}(\gamma + \delta)(\gamma - \delta) \qquad (5.35)$$

Therefore:

$$\frac{A_0}{C_0} = \lambda_d + \frac{1}{2}(\delta - \gamma) = \alpha \qquad (5.36)$$

similarly

$$\frac{B_0}{D_0} = \lambda_d + \frac{1}{2}(\gamma + \delta) = \beta \qquad (5.37)$$

This can be viewed as the asymptotic value of the rate as time goes to $-\infty$.

So the rate function $r(t)$ simplifies to:

$$r(t) = \frac{\alpha C_0 + \beta D_0 e^{-\gamma t}}{C_o + D_0 e^{-\gamma t}} = \frac{\alpha C_0 e^{-\alpha t} + \beta D_0 e^{-\beta t}}{C_o e^{-\alpha t} + D_0 e^{-\beta t}} \qquad (5.38)$$

In fact, comparing $C_0$ to $\beta$ and $D_0$ to $\alpha$, get:

$$C_0 = 2(\beta - \lambda_d - p_0\Delta),$$
$$D_0 = 2(-\alpha + \lambda_d + p_0\Delta) \tag{5.39}$$

Remembering that $p_0$ is just the probability of being in the up-state at $t = 0$, then:

$$\Delta p_0 + \lambda_d = r(0), \tag{5.40}$$

that is, the firing rate at $t = 0$. Then equation 5.38, the firing rate at time $t$, simplifies to:

$$r(t) = \frac{\alpha(\beta - r(0))e^{-\alpha t} + \beta(r(0) - \alpha)e^{-\beta t}}{(\beta - r(0))e^{-\alpha t} + (r(0) - \alpha)e^{-\beta t}} \tag{5.41}$$

## 5.3.2 Change in probability when a spike arrives

In the previous section, the rate function, $r(t)$, and the probability density function of the ISI distribution, $p_{ISI}(t)$, were calculated, given that the probability of being in the up-state at the time of the last spike, $p_0$, is known. However, it is necessary to re-evaluate the probability at the time a spike arrives.

If the probability of being in the up-state before the spike is $p_u(t)$, then the presence of a spike provides information on the state of the model. By Bayes' Theorem:

$$P(\text{up-state}|\text{spike}) = \frac{P(\text{spike}|\text{up-state})P(\text{up-state})}{P(\text{spike})}$$
$$= \frac{\lambda_u p_u(t)}{\Delta p_u(t) + \lambda_d} \tag{5.42}$$

Thus:

$$p_u(t) \to \frac{\lambda_u p_u(t)}{\Delta p(t) + \lambda_d} \tag{5.43}$$

so the rate function becomes:

$$r(t) \to (\lambda_u - \lambda_d)\frac{\lambda_u p_u(t)}{\Delta p_u(t) + \lambda_d} + \lambda_d$$
$$= \frac{\lambda_u^2 p_u(t) + \lambda_d^2(1 - p_u(t))}{\lambda_u p_u(t) + \lambda_d(1 - p_u(t))} \tag{5.44}$$

Equivalently, this can be written as:

$$r(t) \rightarrow (\lambda_u + \lambda_d) - \frac{\lambda_u \lambda_d}{r(t)} \tag{5.45}$$



Figure 5.4: An example of the estimated rate, $r(t)$, for a bimodal Poisson spike train. For this example, $u = d = 5$, $\lambda_u = 50$ and $\lambda_d = 5$. In this example, the initial rate was set randomly between $\alpha$ and $\beta$.

### 5.3.3 Calculating the ISI distribution

When dealing with spike-train data, it is very useful to know the inter-spike interval (ISI) distribution, as this can be observed much easier than the firing rate of a neuron.

The probability density function of the ISI distribution of an inhomogeneous Poisson process, with rate function $r(t)$, is:

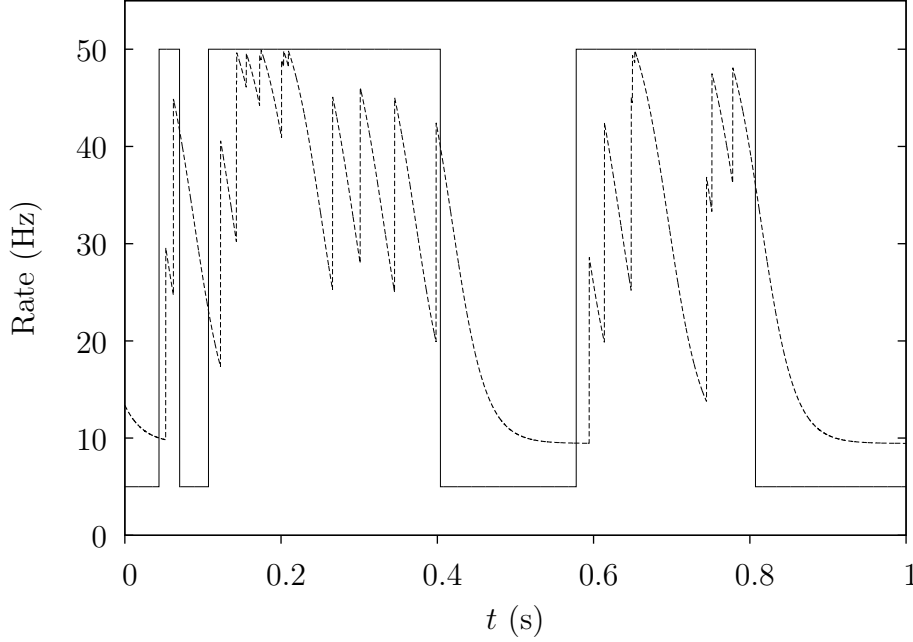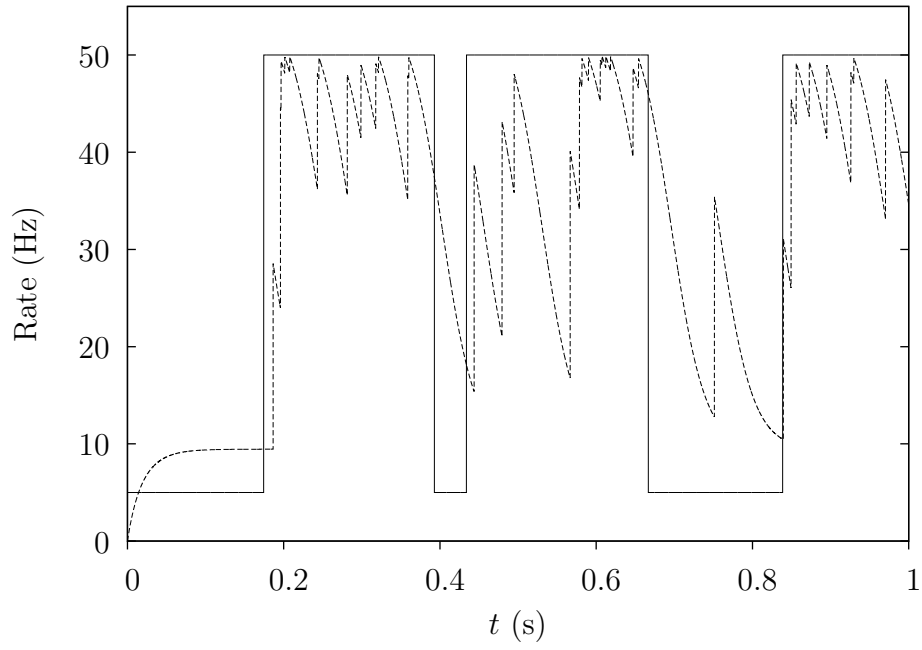$$p_{ISI}(t) = r(t) e^{-\int_0^t r(s)\, ds} \tag{5.46}$$

Figure 5.5: An example of the estimated rate, $r(t)$, for a bimodal Poisson spike train. For this example, $u = d = 5$, $\lambda_u = 50$ and $\lambda_d = 5$. In this example, the initial rate was set to zero and it subsequently set itself.

Now it is necessary to integrate the rate function from 0 to $t$:

$$\int_0^t r(s)\,ds = \int_0^t \frac{\alpha C_0 e^{-\alpha s} + \beta D_0 e^{-\beta s}}{C_0 e^{-\alpha s} + D_0 e^{\beta s}}\,ds$$
$$= \frac{\alpha^2 C_0 D_0 - \beta^2 C_0 D_0}{(\alpha - \beta)C_0 D_0}t + \frac{\beta C_0 D_0 - \alpha C_0 D_0}{(\alpha - \beta)C_0 D_0}\log\left(\frac{C_0 e^{\beta t} + D_0 e^{\alpha t}}{C_0 + D_0}\right)$$

$$(5.47)$$

Since $C_0$ and $D_0$ are not equal to zero, the fractions in the above equation become:

$$\frac{\alpha^2 C_0 D_0 - \beta^2 C_0 D_0}{(\alpha - \beta)C_0 D_0} = \alpha + \beta,$$
$$\frac{\beta C_0 D_0 - \alpha A_0 D_0}{(\alpha - \beta)C_0 D_0} = -1$$

$$(5.48)$$

Thus equation 5.47 becomes:

$$\int_0^t r(s)\,ds = (\alpha + \beta)t - \log\left(\frac{C_0 e^{\beta t} + D_0 e^{\alpha t}}{C_0 + D_0}\right)$$
$$= \log\left(\frac{C_0 + D_0}{C_0 e^{-\alpha t} + D_0 e^{-\beta t}}\right)$$

$$(5.49)$$

Now $p_{ISI}(t)$ becomes:

$$p_{ISI}(t) = \frac{A_0 e^{-\alpha t} + B_0 e^{-\beta t}}{C_0 e^{-\alpha t} + D_0 e^{-\beta t}} e^{-\log\left(\frac{C_0 + D_0}{C_0 e^{-\alpha t} + D_0 e^{-\beta t}}\right)}$$
$$= \frac{A_0}{C_0 + D_0} e^{-\alpha t} + \frac{B_0}{C_0 + D_0} e^{-\beta t}$$

$$(5.50)$$

This is the probability density function (pdf) of a hyper-exponential distribution, $H_2$, which has typical probability density function:

$$f(x) = \rho \lambda_1 e^{-\lambda_1 x} + (1 - \rho)e^{-\lambda_2 x}$$

$$(5.51)$$

Clearly, then $\lambda_1, \lambda_2$ which are calculated experimentally correspond to $\alpha, \beta$. If $\lambda_1 = \alpha$ and $\lambda_2 = \beta$, then it is possible to derive the terms $C_0$ and $D_0$ from equation 5.38 for the probabilistic rate function:

$$C_0 = 2p(\lambda_2 - \lambda_1),\ D_0 = 2(1 - p)(\lambda_2 - \lambda_1)$$

$$(5.52)$$

86

## 5.4  Testing on data

The expectation of a hyperexponential ISI distribution allows for testing on real data. The data used in this report is the data used in [Narayan et al., 2006]. An electrode was placed in L1 of anaesthetised zebra finches, and the birds were presented with 20 natural stimuli ten times each. As the model does not account for the refractory period of neurons, a well-documented feature of neurons [Olshausen and Field, 2004], the ten presentations of each stimulus are pooled to counter the impact of the refractory period [Berry and Meister, 1998].

The inter-spike intervals (ISIs) were then aggregated across 16 of the 20 stimuli, to give a representation of a typical ISI distribution for each neuron. The hyperexponential distribution, $H_2$, was trained on the ISI distribution of the 16 stimuli by minimising the statistic $D_n$ of the Kolmogorov-Smirnov test [Massey Jr, 1951] by downhill-simplex search. Then the remaining four stimuli were aggregated across the ten trials to form a test ISI distribution upon which the calculated minimum hyperexponential distribution was tested using the Kolmogorov-Smirnov test for goodness-of-fit.

The Kolmogorov-Smirnov (KS) test is a nonparametric goodness-of-fit test introduced in [Massey Jr, 1951], based on results from Kolmogorov [1933] and Smirnoff [1939]. It is used to compare a model probability distribution to a sample. The KS statistic is the $d_\infty$ metric between the cumulative distribution function of the model distribution and the empirical distribution function of the data.

$$D_n = \sup_x \left| \hat{F}_n(x) - F(x) \right| \tag{5.53}$$

where $\hat{F}_n(x)$ is the empirical distribution function of the data set of $n$ points and $F(x)$ is the cumulative distribution function of the proposed probability distribution.
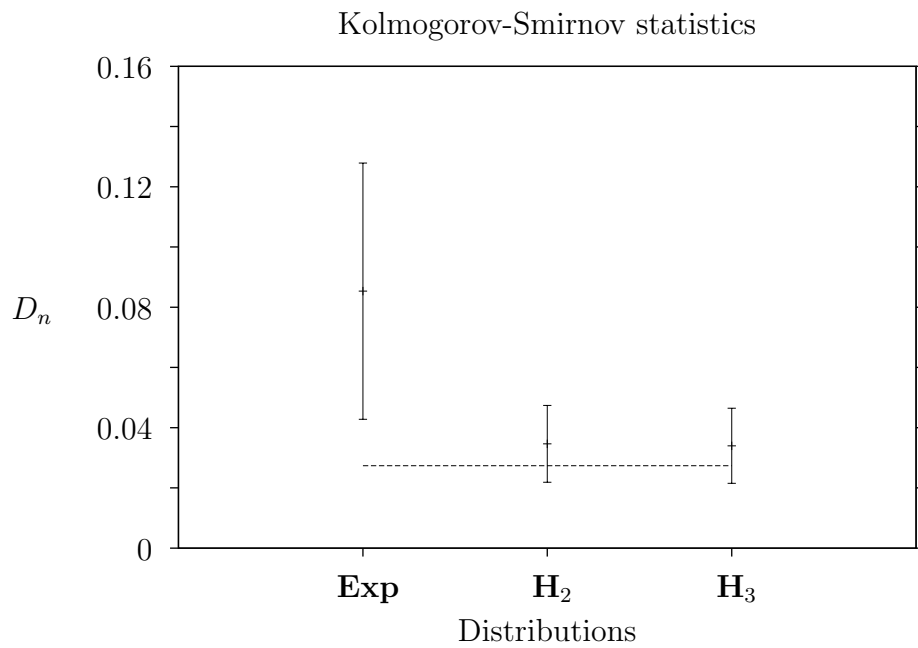
Figure 5.6: The Kolmogorov-Smirnov test statistic $D_n$ for (a) the exponential distribution, (b) the hyperexponential distribution with two modes, and (c) the hyperexponential distribution with three modes. The mean of the $p < 0.05$ critical value is indicated by the broken line.

Then the ISI distributions were tested against the hyperexponential distribution with both two and three modes ($H_2$ and $H_3$). As the exponential distribution is a special case of the hyperexponential distribution, and $H_2$ is a special case of $H_3$, it is clear that there should be an improvement with each step. What was observed, as shown in Figure 5.6, was that the $H_2$ distribution was a drastic improvement on the exponential distribution, but the improvement of $H_3$ over $H_2$ was insignificant.

At the $p < 0.05$ significance level, only three out of 24 neurons were not rejected by the Kolmogorov-Smirnov test for both the $H_2$ and the $H_3$ distributions. This could be due to the number of data points, which would make the KS test extremely rigorous. It is also notable that this model is simply a first approximation to a neuron model; the KS statistic serves as a metric to determine goodness-of-fit rather than fitting the exact ISI distribution.

This seems to demonstrate that there is a bimodality of firing rates in neurons in the auditory forebrain of zebra finches. This may suggest a sparse coding of features in the auditory forebrain of songbirds.

## 5.5    Discussion

The model above was motivated by the fact that sparse coding has been observed in other areas of the brain. It has been speculated that sparse coding should be present in the auditory forebrain, which leads to the use of spectral-temporal receptive fields (STRFs) for calculating the firing rates of neurons [Theunissen et al., 2000]. This chapter provides some further evidence of sparse coding, as it is not likely that the $H_2$ distribution would outperform the exponential to such a degree without the existence of two distinct firing-rates.

The simple model proposed here does not completely model the behaviour of a neuron, it is just based on the typical background rate, and

so it is not clear how this analysis would be used to estimate rates directly from stimuli. A perceptron approach to calculate a background STRF was investigated, but did not produce any significant results.

# Chapter 6

# Conclusion

This thesis was concerned with using several different methods in the field of spike-train analysis. These methods were expanded upon, and then used to inform about the behaviour of individual neurons.

In chapter two of this thesis, clustering methods of network theory were used to attempt to address several problems.

Initially, there was an attempt to cluster responses by forming a network based on the distance between them, using the van Rossum metric. The clustering algorithms did not separate responses into the correct stimuli particularly well. This is likely due to the unnatural structure of the network. Spike train metrics are affected by the noise in the system, and so a responses that is 'close' may not have a significantly different metric value to one that is not. A study of baseline expected values of spike train metrics could lead to a better representative network of responses.

Chapter three introduced an information-theory measure which attempts to determine causal influence between two neurons based on their spike trains. The incremental mutual information of [Singh and Lesica, 2010] proved to be a very useful tool to actually determine direction of influence between neurons in a network. It would, however, be very dif-

ficult to collect data sets that are long enough to fully determine the probability spaces. This is always a problem with any information theory measure, due to the fact that there are no assumptions made about the prior probability distributions.

The bibliographic coupling and cocitation did cluster the network into cooperative groups, but could not completely accurately describe the diagram of the model network. There may be a way to combine the two couplings to do so, but any method of combining the maximum or the mean of the couplings did not prove to work for the model used in this work.

In chapter four, several extensions of distance measures were provided to the SPIKE and the ISI distances from [Kreuz et al., 2007, 2013]. These extensions proved to be comparable to the multi-unit extensions to the Victor-Purpura metric [Aronov et al., 2003] and the van Rossum metric [Houghton and Sen, 2008].

The multi-unit ISI extensions lead to a different understanding of the summed population codes. An adaptive ISI distance measure was developed, which would not require either a time-scale parameter nor a population parameter. This sort of distance should be very useful in comparing instantaneous trials. However, the adaptive distance was not based on any specific neuron model. This motivated the study of a simple background rate which chapter five was centred on.

The firing rate model introduced was very simplistic, but it was based on the fact that neurons are often feature-specific. That assumption suggested that there should be a bimodality in the firing rates of a neuron.

The bimodal firing rate suggest that the ISI distribution of the neuron should be a hyperexponential distribution, with two time-scales, rather than an exponential distribution. When this distribution was tested on the data, it significantly out-performed the exponential distribution, and adding a third mode did not improve results. Thus, it seems as though

certain neurons should have this 'on-off' firing rate, specifically neurons in the auditory forebrain of some birds.

# Bibliography

Aronov, D., Reich, D. S., Mechler, F., and Victor, J. (2003). Neural coding of spatial phase in v1 of the macaque monkey. *Journal of Neurophysiology*, 89:3304–3327.

Averbeck, B. B., Latham, P. E., and Pouget, A. (2006). Neural correlations, population coding and computation. *Nat Rev Neurosci*, 7(5):358–366.

Bair, W. (1999). Spike timing in the mammalian visual system. *Current Opinion in Neurobiology*.

Bender, E. A. and Canfield, E. R. (1978). The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307.

Berry, M. J. and Meister, M. (1998). Refractoriness and neural precision. *The Journal of Neuroscience*, 18(6):2200–2211.

Bi, G. and Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18:10464–10472.

Bollobás, B. (1980). A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316.

Brette, R. and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94:3637–3642.

Du Bois-Reymond, E. H. (1884). *Untersuchungen über thierische elektricität: bd., 1. abth*, volume 2. G. Reimer.

Engel, A. K., König, P., Kreiter, A. K., Schillen, T. B., and Singer, W. (1992). Temporal coding in the visual cortex: new vistas on integration in the nervous system. *Trends in neurosciences*, 15(6):218–226.

Gillespie, J. and Houghton, C. (2011). A metric space approach to the information capacity of spike trains. *Journal of Computational Neuroscience*, 30(1):201–209.

Goodman, D. and Brette, R. (2008). Brian: a simulator for spiking neural networks in python. *Frontiers in neuroinformatics*, 2.

Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544.

Hodgkin, A. L. and Huxley, A. F. (1939). Action potentials recorded from inside a nerve fibre. *Nature*, 144(3651):710–711.

Hopfield, J. and Herz, A. (1995). Rapid local synchronization of action potentials: Toward computation with coupled integrate-and-fire neurons. *Proceedings of the National Academy of Sciences*, 92:6655–6662.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Hopkins, C. D. and Bass, A. H. (1981). Temporal coding of species recognition signals in an electric fish. *Science*, 212(4490):85–87.

Houghton, C. (2009). Studying spike trains using a van rossum metric with a synapses-like filter. *Journal of Computational Neuroscience*, 26(1):149–155.

Houghton, C. and Kreuz, T. (2012). On the efficient calculation of van rossum distances. *Network*, 23:48–58.

Houghton, C. and Sen, K. (2008). A new multi-neuron spike-train metric. *Neural Computation*, 20(6):1495–1511.

Houghton, C. and Victor, J. (2012). Spike rates and spike metrics. In Kriegeskorte, N. and Kreiman, G., editors, *Visual Population Codes: Toward a Common Multivariate Framework for Cell Recording and Functional Imaging*, chapter 8, pages 391–416. MIT Press, Cambridge, MA.

Humphries, M. D. (2011). Spike-train communities: Finding groups of similar spike trains. *Journal of Neuroscience*, 31:2321–2336.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 15(5):1063–1070.

Julienne, H. and Houghton, C. (2013). A simple algorithm for averaging spike trains. *The Journal of Mathematical Neuroscience*, 3(1):1–14.

Karypis, G., Han, E., and Kumar, V. (2000). Multilevel refinement for hierarchical clustering.

Kijima, M. and Komoribayashi, K. (1998). A markov chain model for valuing credit risk derivatives. *The Journal of Derivatives*, 6(1):97–108.

Knight, B. W. (1972). The relationship between the firing rate of a single neuron and the level of activity in a population of neurons experimental

evidence for resonant enhancement in the population response. *The Journal of general physiology*, 59(6):767–778.

Kolmogorov, A. N. (1933). Sulla determinazione empirica di una legge di distribuzione. *Giorn. Ist. Ital. Attuari*, 4:1–11.

Kreuz, T., Chicarro, D., and Andrzejak, R. G. (2009a). Measuring spike train synchrony between neuronal populations. Presented at CNS2009.

Kreuz, T., Chicharro, D., Andrzejak, R. G., Haas, J. S., and Abarbanel, H. D. (2009b). Measuring multiple spike train synchrony. *Journal of Neuroscience Methods*, 183(2):287–299.

Kreuz, T., Chicharro, D., Greschner, M., and Andrzejak, R. G. (2011). Time-resolved and time-scale adaptive measures of spike train synchrony. *Journal of Neuroscience Methods*, 195(1):92 – 106.

Kreuz, T., Chicharro, D., Houghton, C., Andrzeja, R. G., and Mormann, F. (2013). Monitoring spike train synchrony. *Journal of Neurophysiology*, 109:1457–1472.

Kreuz, T., Haas, J. S., Morelli, A., Abarbanel, H. D. I., and Politi, A. (2007). Measuring spike train synchrony. *Journal of Neuroscience Methods*, 165(1):151 – 161.

Lewicki, M. (1998). A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computational Neural Systems*, 9(4):R53–R78.

Massey Jr, F. J. (1951). The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78.

Mulansky, M., Bozanic, N., Sburlea, A., and Kreuz, T. (2015). Spike train synchrony: Measures and algorithms. *IEEE Trans. Neural Networks (under review)*.

Narayan, R., Graña, G., and Sen, K. (2006). Distinct time scales in cortical discrimination of natural sounds in songbirds. *Journal of Neurophysiology*, 96:252–258.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4):308–313.

Newman, M. E. J. (2006a). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104.

Newman, M. E. J. (2006b). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582.

Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review*, 69:026113.

Olshausen, B. A. and Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487.

Pizzuti, C. (2008). Ga-net: A genetic algorithm for community detection in social networks. In *Parallel Problem Solving from Nature–PPSN X*, pages 1081–1090. Springer.

Ramón y Cajal, S. (1904). *Textura del Sistema Nervioso del Hombre y de los Vertebrados*, volume 2. Madrid.

Schreiber, T. (2000). Measuring information transfer. *Phys. Rev. Lett.*, 85:461–464.

Sen, K., Theunissen, F. E., and Doupe, A. J. (2001). Feature analysis of natural sounds in the songbird auditory forebrain. *Journal of Neurophysiology*, 86:1445–1458.

Shannon, C. E. (1948). A mathematical thoery of communication. *Bell Systems Technical Journal*, 27:379–423,623–656.

Singh, A. and Lesica, N. A. (2010). Incremental mutual information: A new method for characterizing the stregth and dynamics of connections in neuronal circuits. *PLOS Computational Biology*, 6(12).

Smirnoff, N. (1939). Sur les écarts de la courbe de distribution empirique. *Matematicheskii Sbornik*, 48(1):3–26.

Strong, S. P., Koberle, R., de Ruyter van Steveninck, R. R., and Bialek, W. (1998). Entropy and information in neural spike trains. *Physics Review Letters*, 80(1):197–200.

Theunissen, F., Sen, K., and Doupe, A. J. (2000). Spectral-temporal receptive fields of nonlinear auditory neurons obtained using natural sounds. *Journal of Neuroscience*, 20(6):2315–2331.

van Rossum, M. (2001). A novel spike distance. *Neural Computation*, 13:751–763.

Victor, J. D. and Purpura, K. P. (1996). Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of Neurophysiology*, 76(2):1310–1326.

Victor, J. D. and Purpura, K. P. (1997). Metric-space analysis of spike trains: theory, algorithms and application. *Network: Computation in Neural Systems*, 8(2):127–164.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473.