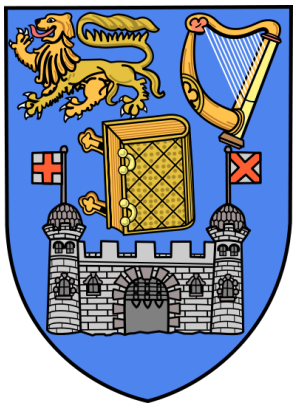
Spike Train Analysis

Trinity College Dublin



Cathal Cooney

2015-1-31

Contents

Chapter 1

Introduction

1.1 Spike Trains

Neurons convey information through the nervous system by generating electric pulses which propagate along nerve fibers. These pulses are called action potentials or spikes [Dayan and Abbott, 2001].

A neuron has a resting potential of approximately -70mV relative to its surroundings, and we say that the cell is polarised at this point. The membrane potential of a neuron becomes less negative with current flowing into it, but will tend back towards its resting potential unless the membrane potential 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. An action potential, or spike, is a 100mV fluctuation in the membrane potential which lasts approximately 1ms. Following the production of a spike the neuron briefly becomes hyperpolarised, and as such cannot generate a spike for the next couple of milliseconds. This period,

when a spike cannot be fired, is called the absolute refractory period of a neuron. Since the neuron becomes hyperpolarised by the spike, there is also a period in which it is "more difficult" for the neuron to spike again, this is called the relative refractory period of a neuron.

Spikes are very important because they propagate over long distances without attenuation, and so are the way in which neurons communicate with one another throughout the nervous system. The timing of spikes is of particular importance; for example, in motor neurons when a number of spikes happen close together it leads to movement of the muscle, the more spikes that happen together, the faster the muscle twitches. While there is some variation in the duration, amplitude and shape of action potentials, we can largely treat them as identical processes, where the timing is the only consideration. So, if a neuron spikes n times in a certain trial, then the trial can be described by the times of the spikes t_i. Then we can describe the *spike train* mathematically as:

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

The δ here is the Dirac delta, which essentially gives a spike of volume one at the point t_i. It can be very difficult to differentiate between different spike trains, and so we have *metrics* to tell them apart.

1.1.1 Spike Train Metrics

If we look at spike trains from a specific neuron, we would like to be able to "decode" them and be able to describe the stimulus which evoked them. Unfortunately, this seems like a very difficult problem, so we first settle for

trying to distinguish which spike trains amongst a collection were evoked by the same stimulus. To do this, there are several *metrics* in the literature which tell us the "distance" between two spike trains.

In Mathematics a metric space is a space where we can always tell the "distance" between two elements, whether or not we can give the "coordinates" of the elements in the space. A metric is a function on a set X, $d : X \times X \to [0, \infty)$ that basically follows our intuition for what a distance should be. That is, distances are non-negative, symmetric and only zero if two elements are the same. The Triangle Inequality states that you should never be able to shorten the distance between two points by going through an intermediate point, so it is the notion that the shortest distance between two points is a straight line.

1.1.2 Van Rossum metric

The metric that we primarily use is the metric described by van Rossum in [van Rossum, 2001a], which is simply based on the L^2 metric on function spaces. If we view the spike trains as a collection of Dirac delta functions, then we can get a function by convolving the distribution $s(t)$ as above with a kernel. A common kernel that is used is the exponential kernel

$$k(t) = \begin{cases} \frac{1}{\tau} e^{-\frac{t}{\tau}}, & t \geq 0 \\ 0, & t < 0 \end{cases} . \tag{1.2}$$

This kernel has some advantages, like causality and computability, over a gaussian kernel. So, we get a new function $u(t)$ after convolving the spike

train with the kernel:

$$u(t) = s * k(t) = \int_0^T s(t-s)k(s)\,ds \tag{1.3}$$

Figure **??** above shows the form of the function that we get when we convolve a spike train with the exponential kernel. Once we have these functions for different spike trains, u and v say, then we calculate the distance between them by taking the L^2 metric on the function space, that is:

$$d(u,v) = \sqrt{\int_0^T (u(t) - v(t))^2\,dt} \tag{1.4}$$

This metric has some mathematical advantages over other metrics, such as Victor- Purpura [Victor and Purpura, 1997a], an "edit-length" metric, because there is a lot of material in functional analysis on the L^2 metric. It performs to a similar standard to most other metrics, and remarkably the choice of kernel has little effect on its efficiency.

1.2 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].

The topic of Information Theory is centred around the definition of *entropy*, provided by Shannon in 1948 [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:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{1.5}$$

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.

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, that these connections could possibly illuminate the functional landscape of the brain. This approach suggested that Complex Network Theory, a branch of Mathematics closely related to Graph Theory, could be useful. 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 through 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. An *undirected network* with n nodes can be completely

described by the adjacency matrix A_{ij} 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, and if the matrix is not symmetric; it is called a *directed* network, where $A_{ij} = 1$ if there is an arrow pointing from node j to node i. Binary undirected networks form the bulk of the literature, and so the theory is richer in this case. A reason for this could be that often one can say that two objects, or properties, are related or that they are not, without giving the relation a number to quantify the relation.

A particularly interesting part of Network Theory that could be useful for Neuroscience is clustering, and algorithms to maximise the community structure of the communities. The main focus is on a measure known as the *modularity*.

2.1 Modularity

The *modularity* is a measure of a given clustering of a network, it 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.

Thus, 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, such as degree of nodes. If the probability of a link between nodes i and j is P_{ij} and

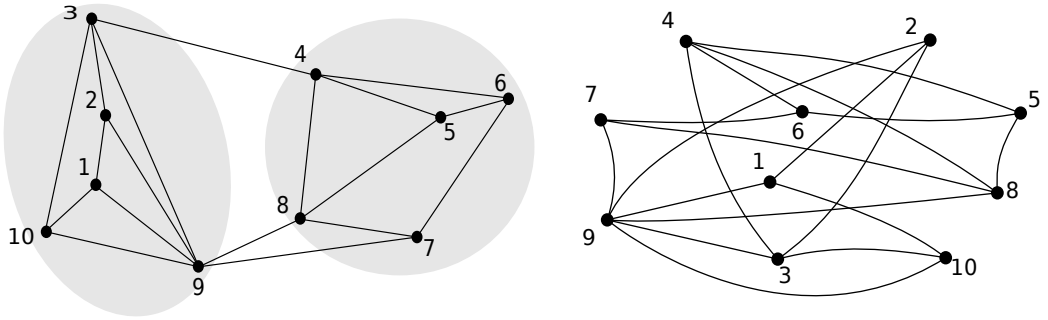


Figure 2.1: The advantage to clustering a network correctly is seen here. Both of these networks have the same nodes and links, and so are the same network, but they look vastly different because the diagram on the left has been clustered to maximise modularity.

g_i is the community to which node i belongs, then it can be described as [Newman, 2006a]:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - P_{ij}] \delta(g_i, g_j) \tag{2.1}$$

where δ 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 $\frac{1}{2m}$ 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.

It needs to be understood what exactly P_{ij} should be. In undirected networks, then clearly $P_{ij} = P_{ji}$. The case of all nodes being in the same cluster is not interesting, as it displays no further community structure, so the modularity is set $Q = 0$ in that case. Therefore, get $\sum_{i,j}[A_{ij} - P_{ij}] = 0$, and,

$$\sum_{i,j} P_{ij} = \sum_{i,j} A_{ij} = 2m, \tag{2.2}$$

where m, as before, is the total number of edges in the network, so if the degree of node i is k_i, then

$$m = \frac{1}{2}\sum_{i} k_i = \frac{1}{2}\sum_{i,j} A_{ij}. \tag{2.3}$$

Beyond this, there is more scope for choice. 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. If, instead, it is supposed that the "random" network should keep the same degree for all nodes, then the network can be treated as an instance of the *configuration model* [Bender and Canfield, 1978, Bollobás, 1980]. So, if the expected degree of each vertex is equal to the actual degree of the vertex, then:

$$\sum_{j} P_{ij} = k_i. \tag{2.4}$$

If it is supposed that beyond the constrained degree distribution that edges are placed at random, then the probability of two nodes connecting is dependent on just 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, get that $P_{ij} = f(k_i)f(k_j)$ for some function f on their degrees. Then, by equation 2.4, get

$$\sum_{j=1}^{n} P_{ij} = f(k_i)\sum_{j=1}^{n} f(k_j) = k_i, \tag{2.5}$$

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

$$2m = \sum_{i,j} P_{ij} = C^2 \sum_{i,j} k_i k_j = (2mC)^2, \tag{2.6}$$

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

$$P_{ij} = \frac{k_i k_j}{2m}, \tag{2.7}$$

and the modularity is

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(g_i, g_j). \tag{2.8}$$

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 a moderate to large size, and so clustering algorithms are needed to maximise the modularity.

2.2 Newman's eigenvalue algorithm

In [Newman, 2006b] 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,

$$s_i = \begin{cases} 1 & \text{if node } i \text{ in cluster 1} \\ -1 & \text{if node } i \text{ in cluster 2} \end{cases} \tag{2.9}$$

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

$$\delta(g_i, g_j) = \frac{1}{2}\left(s_i s_j + 1\right) \tag{2.10}$$

Then, the modularity is redefined as:

$$\begin{aligned} Q &= \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\frac{1}{2}(s_i s_j + 1) \\ &= \frac{1}{4m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)s_i s_j, \end{aligned} \tag{2.11}$$

Let $\mathbf{B}$ be a matrix where $B_{ij} = A_{ij} - k_i k_j/2m$, called the *modularity matrix*, then equation 2.11 becomes:

$$Q = \frac{1}{4m}\mathbf{s^t B s} \tag{2.12}$$

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

$$\mathbf{s} = a_1\mathbf{u_1} + \ldots + a_n\mathbf{u_n}. \tag{2.13}$$

The equation for the modularity becomes

$$\begin{aligned} Q &= \frac{1}{4m}\sum_i a_i \mathbf{u}_i^t \mathbf{B}\sum_j a_j \mathbf{u}_j \\ &= \frac{1}{4m}\sum_i \lambda_i(\mathbf{u_i^t}.\mathbf{s})^2 \end{aligned} \tag{2.14}$$

so, the positivity of the modularity depends completely on the λ_i, in particular the most positive eigenvalue λ_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:

$$s_i = \begin{cases} 1 & \text{if } u_{1_i} > 0 \\ -1 & \text{if } u_{1_i} \leq 0 \end{cases} \tag{2.15}$$

This gives a clustering of the network into two smaller clusters. Of course, this split is not strictly in the direction of λ_1, but it is the best possible estimate[1], given two clusters. 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

[1]In fact, it has been noted from well-known networks, such as Zachary's karate club [Zachary, 1977] that the absolute value of the ith entry of $\mathbf{u}_1$ gives a good indication of the "strength" of the membership of node i to its cluster [Newman, 2006b].

cluster. Instead, the original modularity matrix $\mathbf{B}$ is used to calculate what difference a split of the subgraph would make to the modularity.

Call the cluster of nodes g and the *modularity contribution* ΔQ of a split of g is:

$$\Delta Q = \frac{1}{2m}\left[\frac{1}{2}\sum_{i,j\in g} B_{ij}(s_i s_j + 1) - \sum_{i,j\in g} B_{ij}\right]. \tag{2.16}$$

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:

$$\begin{aligned}
\Delta Q &= \frac{1}{4m}\left[\sum_{i,j\in g} B_{ij}s_i s_j - \sum_{i,j\in g} B_{ij}\right] \\
&= \frac{1}{4m}\sum_{i,j\in g}\left[B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}\right] s_i s_j \\
&= \frac{1}{4m}\mathbf{s^t}\mathbf{B}^{(g)}\mathbf{s}
\end{aligned} \tag{2.17}$$

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

$$B^{(g)}_{ij} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik} \tag{2.18}$$

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

Using the same approach as before and find the most positive eigenvalue of $\mathbf{B}^{(g)}$ to determine the favoured split $\mathbf{s}$ to maximise ΔQ. $\mathbf{B}^{(g)}$ still has the property that each of its rows and 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 say that the subgraph is indivisible and stop splitting. However, it should be noted 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 ΔQ 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], in fact considerably better results than the *betweenness* algorithm of Newman and Girvan [Newman and Girvan, 2004], despite the apparent drawback of always splitting the network/subgraph in two parts. It is also possible to use the first m eigenvalues and use the ith entry of the eigenvectors as coordinates in $\mathbf{R}^m$ [Humphries, 2011], but this then requires a choice of number of clusters, which may not be known from the data. This may also require calculating many more eigenvalues of the large matrix $\mathbf{B}$, which can be computationally expensive.

2.3 New network clustering algorithms

Using the definition of modularity as defined above in equation 2.8 to measure the community structure of given clusterings, two new algorithms were developed in the course of this thesis. They both use complexity algorithms, and so tend to be a lot slower than principled clustering algorithms such as in [Newman, 2006b] [Newman and Girvan, 2004].

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

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

Find the most positive eigenvalue λ_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

2.3.1 Genetic algorithm

Genetic algorithms have become very popular 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 proposed 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 typical genetic algorithm is described in algorithm 2.

Thus, to define a genetic algorithm, it is important to understand what a solution is, and how the genes crossover and mutate from generation to generation. A fitness function in genetic algorithms should typically be non-negative, which can be attained by setting the fitness of any clustering with negative 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. This can be achieved by setting the upper bound for the number of clusters to be the maximum of the numbers of clusters of the parents, or by

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 ϵ.

  end for

  for all g_i **do**

    $g_i \leftarrow \tilde{g}_i$

  end for

end while

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 choose 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 (15%) percentage of nodes between clusters in approximately 10% of clusterings.

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

2.3.2 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 lowest energy state 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. The probability of changing the state of the system 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, as can be seen:

$$P(E(s), E(s'), T) = \begin{cases} 1 & E(s') < E(s) \\ min(e^{-\frac{E(s')-E(s)}{T}}, 1) & E(s') \geq E(s) \end{cases} \tag{2.19}$$

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. This algorithm was not particularly good at splitting a network to more than two or three clusters, but it did seem to find "stable" solutions, eg. in Zachary's karate club network [Zachary, 1977] it found the correct split, rather than the clustering which resulted in a slightly higher modularity.

2.4 k-medoids clustering

By defining a metric on spike trains, we can tell how close different responses are to each other, but there are no "co-ordinates" in the metric space which would help with clustering responses. The standard method of clustering spike-trains is similar to k-means clustering, but since there is no standard

method for computing a "mean spike-train" given a number of spike trains, it must be altered slightly.

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.

There is no mean in the metric space of spike-trains, and so k-means clustering cannot be used. In [Julienne and Houghton, 2013] an *average function* is suggested as a possibility to replace the concept of a 'mean spike-train', however the common method is to use k-medoids clustering instead. The algorithm, k-medoids, begins by choosing k 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 Houghton and Julienne 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.

The next chapter deals with our efforts to address this issue.

2.5 Clustering responses with modularity

We proposed that we could perhaps improve on current methods of sorting responses by using the modularity clustering algorithm. The advantage to such a method would be that we would hopefully not need to know how many different stimuli there were. Since the algorithm determines when to stop itself, it could 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.

We used the data set from [Narayan et al., 2006], which featured spike trains from anesthetized adult male zebra finches as they listened to natural birdsong. Each neuron was played 20 songs ten times each, so we had 200 data points from which to form a network for each neuron.

We formed a network by first taking the Van-Rossum metric distances between each pair of responses, then we chose a threshold value τ for the network and we set

$$A_{ij}^{\tau} = \begin{cases} 1 & \text{if } d(i,j) < \tau \\ 0 & \text{otherwise} \end{cases} . \tag{2.20}$$

Once we have the adjacency matrix for our network, we can run the algorithm to maximise modularity for the network. For the data that we used, each spike train was less than a second long, so the maximum van Rossum distance between any two trains was one, so we ran the algorithm for different threshold values τ between zero and one incrementing τ by steps

of size 0.001.

As we can see in Figure **??**, the profile of the graph of the maximum modularity versus the threshold looked promising, as in nearly all cases it had a clear maximum, but unfortunately the clusters of responses bore little resemblance to the stimuli, and usually had too few clusters.

It appears that the margins are just too small, and there is no natural network, rather a bunch of networks determined by a strict cut-off, so the modularity algorithm cuts some of the correct clusters in two. A recent paper by Humphries [Humphries, 2011] uses all of the positive eigenvalues to cluster the responses with more accuracy. This method is rather computationally expensive, and uses even the very small positive eigenvalues, which tell us very little about the positivity of modularity, so we tried to use Newman's more simple algorithm, despite its obvious drawbacks.

The problem with the above method appears to be that there is no natural network formed by the distance matrix of responses to stimuli.

Due to the fact that the modularity that the algorithm found for the network is higher than the modularity of the correct clustering, it is clear that network methods are not useful to sort stimuli from a single neuron. Since the original goal was to illuminate the functionality of the brain, or at least small areas of the brain, this non-result doesn't matter too much. We want to get simultaneous recordings from many neurons, and form networks which would correspond to different neurons rather than spike trains.

2.6 Mapping information flow in a simulated network of neurons

In this section, 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 hard spiking threshold 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 that seem to *drive* other neurons. That is, neurons whose firing seems to improve the probability of the other neuron firing significantly. A method of particularly 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. He calls this the *bibliographic coupling*

of two nodes. In a brain network, this could give a "map" of information flow.

2.7 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. The adjacency matrix **A** of a directed network is then defined by:

$$A_{ij} = \begin{cases} 1 & \text{if there is a link from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

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 the nodes. Since an unweighted network typically has entries equal to one or zero, this coupling is calculated as:

$$B_{ij} = \sum_k A_{ki} A_{kj} \tag{2.22}$$

Thus the matrix **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}, typically without any self-links; so the

bibliographic network has adjacency matrix equal to the bibliographic matrix with the diagonal entries set to zero.

2.8 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 δ, the contribution to the IMI at time t is defined to be:

$$\Delta I_{XY}[\delta] = H\left(X(t)|Z_\delta(t)\right) - H\left(X(t)|Z_\delta(t), Y(t-\delta)\right) \tag{2.23}$$

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

$$Z_\delta(t) = (X_p(t), X_f(t), Y_p(t-\delta), Y_f(t-\delta)) \tag{2.24}$$

This is calculated in the typical manner of calculating Mutual Information in spike-trains as per Bialek et al. [Strong et al., 1998]. First the spike-trains are discretised according to a time-scale which is unlikely to contain more than one spike, for example $2ms$ is approximately the spiking threshold in 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. Then the probability space is 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 n past and future steps, there are 2^{4n+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.

2.9 Numerical testing

Due to the data-hungry nature of the IMI and the scarcity of simultaneous recordings of action potentials (spikes) 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 worth noting in it - a divergence and a convergence of groups. The network was randomly generated based on the schematic in figure 2.3. 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 being ease of computation [Hopfield and Herz, 1995], and biological accuracy [Hodgkin and Huxley, 1952]. The network was simulated using the python package BRIAN [**CITATION**], and each simulation ran for four minutes. The amplitude of the poisson neurons in the noise layer was varied from 1 mV to 20 mV to investigate how well the IMI dealt with noise.

The proposed method to map the modules of neurons was to initially calculate the peak IMI between each pair of neurons, and determine the non-

diagonal elements of an adjacency matrix of a weighted directed network:

$$A_{ij} = \begin{cases} \max_\delta \Delta I_{ij}[\delta] & \text{if } \delta > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.25}$$

As can be seen in figure 2.5, the IMI actually improves its discriminability as the noise level increases. This should not be a surprise, as this was the reasoning provided by the authors of [Singh and Lesica, 2010] for conditioning for the future along with the past, as opposed to just conditioning for the past as [**CITATION**] 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.

2.10 Bibliographic coupling

The next step in the process is to calculate the bibliographic coupling from the adjacency matrix calculated in equation 2.25. This then gives a new network, which is undirected, and hence can be clustered using common methods such as [Newman, 2006b, Newman and Girvan, 2004].

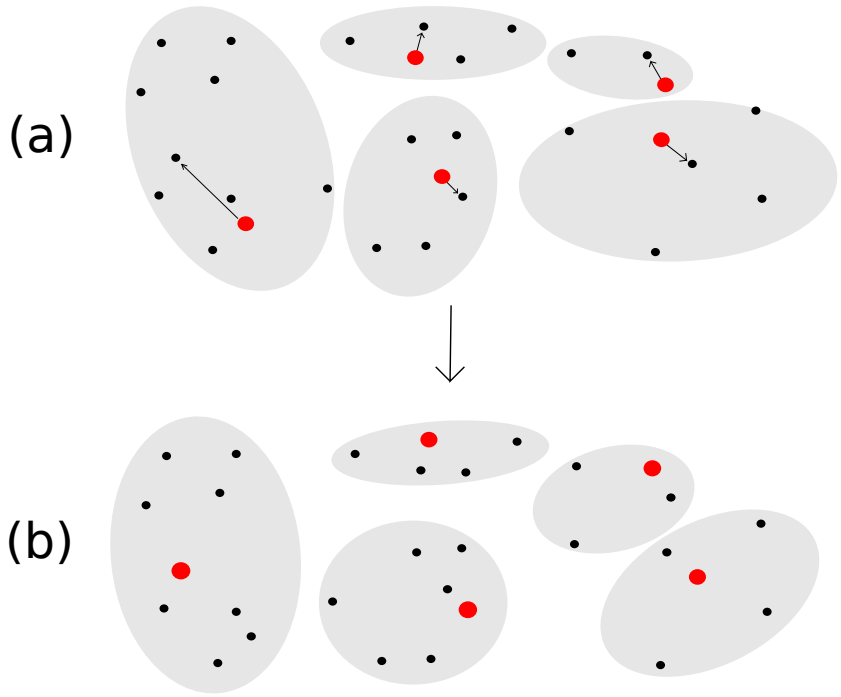


Figure 2.2: An example of the first step in k-medoids clustering: (a) Medoids are initially selected at random and the first clustering puts each point in the cluster of the closest medoid. Then we choose the point in the cluster that will give the lowest total distance to other points, and choose that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

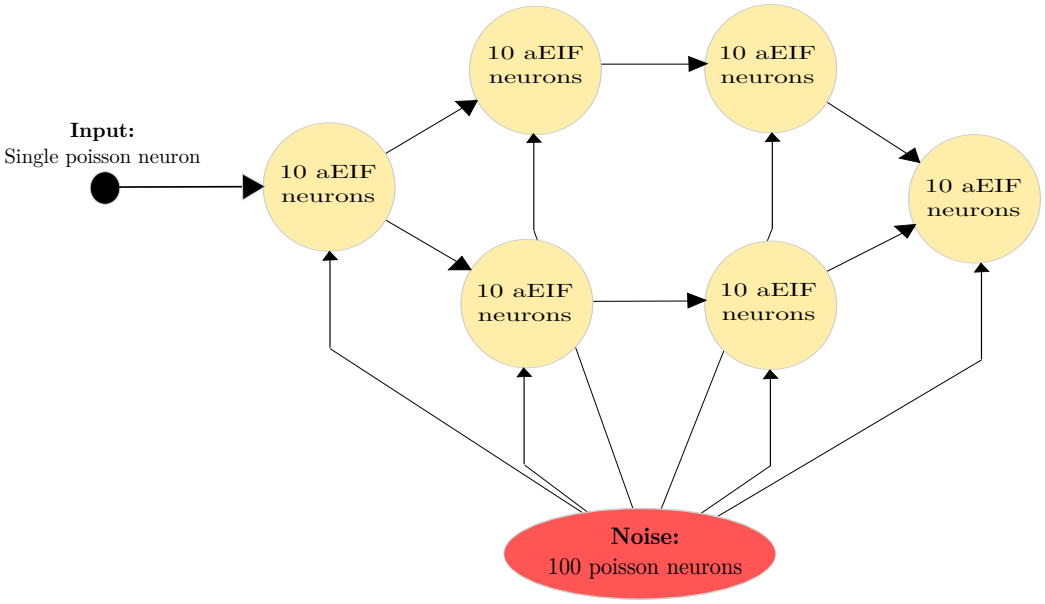


Figure 2.3: The network model was based on the schematic shown here. All simulated neurons in the recurrent layer were aEIF neurons. The connection probability between two nodes at random was 0.1, connections along arrows in the diagram occurred with probability 0.8, and connections from the noise layer to the recurrent layer occurred with probability 0.2.

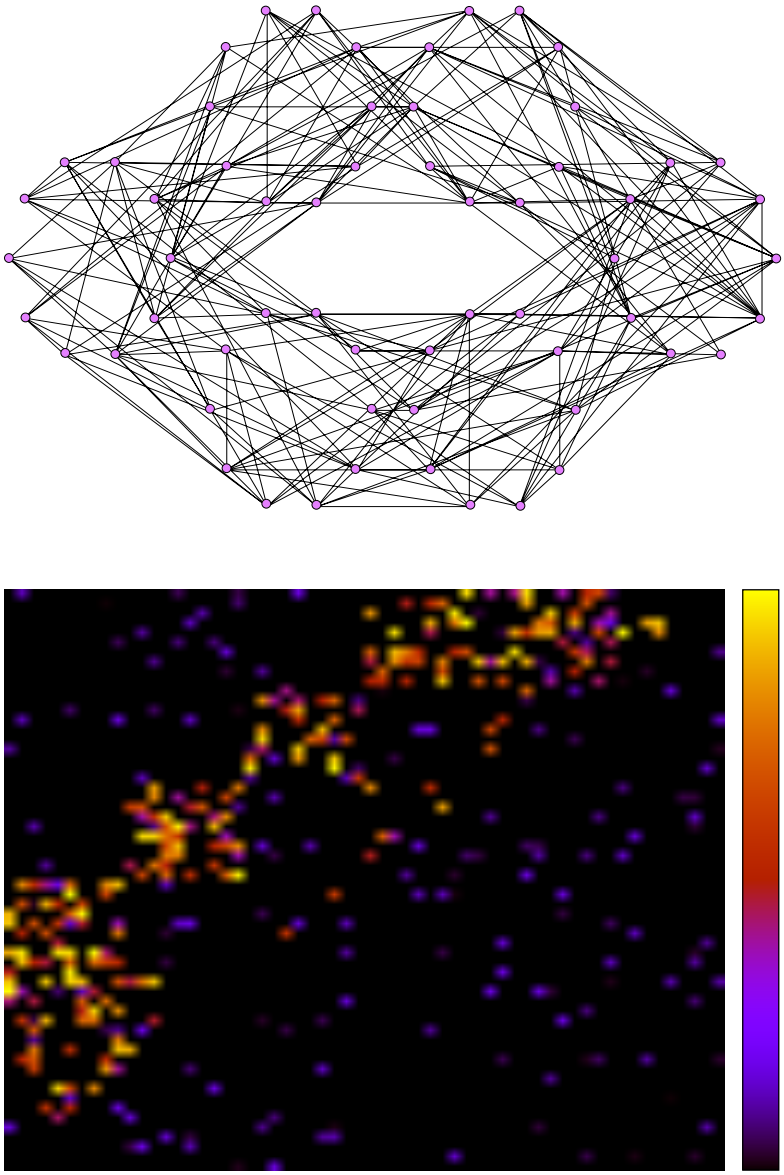


Figure 2.4: A standard representation of the directed network above, and the connection matrix of the recurrent layer of aEIF neurons below.

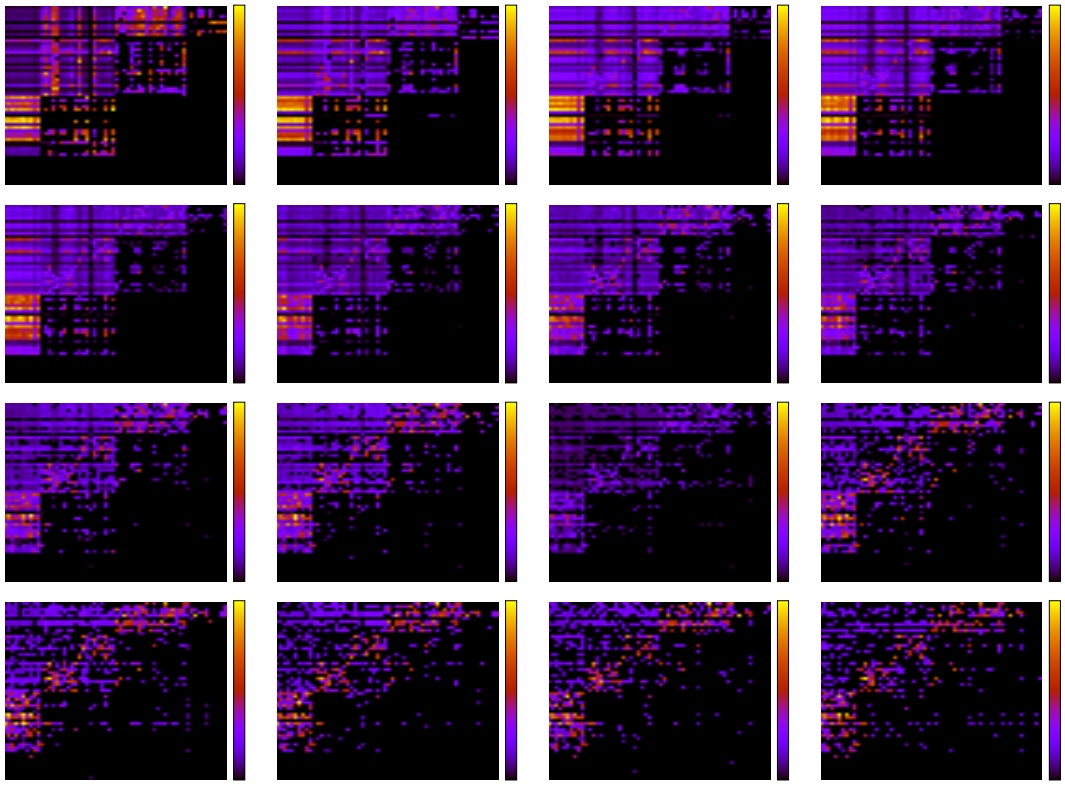


Figure 2.5: The connection matrix calculated from the peak IMI of the neurons, as the amplitude of the individual neurons in the noise layer increased from 1 mV to 16 mV. Note that the IMI actually becomes more discriminating as the noise level increases.

Chapter 3

Spike Train Metrics

The van Rossum [van Rossum, 2001a] and Victor-Purpura [Victor and Purpura, 1996] (VP) metrics introduced in Chapter 1 are the most commonly used metrics in the field of Computational Neuroscience, but they do have some drawbacks. For example, they both have a time-scale parameter which must be chosen according to the cell, τ for the van Rossum metric and q for the VP metric. They also do not have a time-local measure to discriminate between the behaviour of different spike-trains at specific times. The SPIKE and ISI measures introduced in [Kreuz et al., 2007, Kreuz et al., 2011a] are two newer measures that address these concerns.

It is actually possible to develop a time-local version of the VP metric. Such a time-local version must integrate to the VP metric between two spike-trains. This is more complicated than in the case of the SPIKE distance, since it is not just a case of assigning a 'gap' to each spike and interpolating between them. There is a way to assign gaps, but it requires to calculate the VP metric itself on the spike-trains first.

The VP metric is defined as in chapter 1:

$$d_{\mathrm{VP}}(X,Y) = \inf_{f} \sum_{i} c(f_i(X)) \tag{3.1}$$

where $f = f_1 \emptyset f_2 \emptyset \dots \emptyset f_n$ is any transformation that takes spike-train X to spike-train Y, and c is the associated cost function for the elementary transformations $f_i, i \in 1, \dots, n$.

The permitted elementary transformations are insertion of a spike, with a cost of 1; deletion of a spike, with a cost of 1 also; and translation of a spike by a distance δt, with a cost of $q\delta t$. Thus, each spike in X and Y has an associated transformation which has a fixed cost; but there is a lack of symmetry between

The SPIKE distance proposed by Kreuz et al in [Kreuz et al., 2011a] is an instantaneous parameter-free distance measure between spike-trains. By instantaneous measure, it is meant that for each time t there is a time-local distance between two spike-trains. This measure can the be integrated over the length of the spike-trains to give a parameter-free distance measure between spike trains. In the study of spike-train metrics, often a multi-unit measure is desired (to minimise compounding any errors that may result from inefficient spike-sorting?). A multi-unit measure is a distance between collections of labelled spike-trains. The SPIKE distance in [Kreuz et al., 2011a] does not lend itself to a simple multi-unit extension, so instead a simpler version of the SPIKE distance is extended.

3.1 Single-unit recordings

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., 2011b]: 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., 2011b] 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 the sum of these gaps. The original SPIKE distance included additional normalisation factors that have been omitted in simplifying the measure. 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:

$$\Delta t_i^x = min_i(|t_i^x - t_i^y|) \tag{3.2}$$

At each time instant, these is a unique set of four corner spikes: 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, a time-local distance is then calculated using the associated gap of the four corner spikes for each spike-train, $w = x, y$:

$$s_w(t) = \frac{\lambda_P(t)\Delta t_P^w(t) + \lambda_F(t)\Delta t_F^w(t)}{I^w(t)} \tag{3.3}$$

where

$$\lambda_F^w(t) = \frac{t - t_P^w(t)}{I^w(t)},\ \lambda_P^w(t) = \frac{t_F^w(t) - t}{I^w(t)} \tag{3.4}$$

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

$$I^w(t) = t_F^w(t) - t_P^w(t). \tag{3.5}$$

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

$$s(t) = s_x(t) + s_y(t). \tag{3.6}$$

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

$$\int_0^T s(t)\,dt = \sum_w \sum_i \Delta t_i^w. \tag{3.7}$$

In practice this simplified version of SPIKE produces similar time profiles to the version described in [Kreuz et al., 2011b]. The time-local distance profile for two similar spike-trains is given in Figure [NEED TO ADD PICTURE].

3.2 Extension to multi-unit recordings

In the multi-unit case a distance is defined 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 labels the neuron.

Here the single-unit distance above is extended to a multi-unit distance by including the possibility that the nearest spike for one neuron may belong to a different neuron in the other set, however there is 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, 1997b]. 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 $2/\lambda$, where λ is the average frequency of the trials. The gaps can then be calculated as follows:

$$\Delta t_\alpha^{\mathbf{x}_i} = \min_{\beta,j} \left(|t_\alpha^{\mathbf{x}_i} - t_\beta^{\mathbf{y}_j}| + k\,[1 - \delta(i,j)] \right) \tag{3.8}$$

where $\delta(i,j)$ is the Kroneker delta. Hence, if the spikes have the same label in $\mathbf{x}$ and $\mathbf{y}$ there is no added distance, otherwise k is added to the time difference between the spikes. This is illustrated in Figure 3.1:

The time-local distance measure is then defined as before, using the corner

spikes normalised by the inter-spike-interval:

$$s_{\mathbf{X}}(t) = \sum_{\mathbf{x}_i \in \mathbf{X}} \frac{\lambda_{\mathrm{P}}(t)\Delta t_{\mathrm{P}}^{\mathbf{x}_i}(t) + \lambda_{\mathrm{F}}(t)\Delta t_{\mathrm{F}}^{\mathbf{x}_i}(t)}{I^{\mathbf{x}_i}(t)} \tag{3.9}$$

where

$$\lambda_{\mathrm{F}}^{\mathbf{x}_i}(t) = \frac{t - t_{\mathrm{P}}^{\mathbf{x}_i}(t)}{I^{\mathbf{x}_i}(t)} \tag{3.10}$$

and

$$\lambda_{\mathrm{P}}^{\mathbf{x}_i}(t) = \frac{t_{\mathrm{F}}^{\mathbf{x}_i}(t) - t}{I^{\mathbf{x}_i}(t)} \tag{3.11}$$

and $I^{\mathbf{x}_i}(t)$ is the length of the interval in the spike-train $\mathbf{x}_i$ containing t, $I^{\mathbf{x}_i}(t) = t_{\mathrm{F}}^{\mathbf{x}_i}(t) - t_{\mathrm{P}}^{\mathbf{x}_i}(t)$. As before:

$$s(t) = s_{\mathbf{X}}(t) + s_{\mathbf{Y}}(t) \tag{3.12}$$

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:

$$\int_0^T s(t)\, dt = \sum_{\mathbf{W}} \sum_{\mathbf{w}_i \in \mathbf{W}} \sum_{\alpha} \Delta t_{\alpha}^{\mathbf{w}_i} \tag{3.13}$$

3.2.1 Testing on data

The multi-unit distance measure was tested on similar test data to that found in Houghton & Sen [Houghton and Sen, 2008], where two Poisson neurons form a receptive field and are each connected to two leaky integrate-and-fire neurons 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.

3.3 ISI distance

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

$$s(t) = \begin{cases} 1 - I^{\mathbf{x}}(t)/I^{\mathbf{y}}(t) & \text{if } I^{\mathbf{x}}(t) \leq I^{\mathbf{y}}(t) \\ 1 - I^{\mathbf{y}}(t)/I^{\mathbf{x}}(t) & \text{otherwise} \end{cases} \tag{3.14}$$

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

$$s(t) = \frac{|I^{\mathbf{x}}(t) - I^{\mathbf{y}}(t)|}{\max(I^{\mathbf{x}}(t), I^{\mathbf{y}}(t))} \tag{3.15}$$

3.3.2 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.

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) [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; so 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:

$$\mathbf{I}^{\mathbf{x}_1}(t) = \begin{pmatrix} I^{\mathbf{x}_1}(t) \\ 0 \end{pmatrix} \tag{3.16}$$

$$\mathbf{I}^{\mathbf{x}_2}(t) = \begin{pmatrix} (1-\alpha) I^{\mathbf{x}_2}(t) \\ \alpha I^{\mathbf{x}_2}(t) \end{pmatrix}. \tag{3.17}$$

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

$$\mathbf{I}^{\mathbf{X}}(t) = \begin{pmatrix} I^{\mathbf{x}_1}(t) + I^{\mathbf{x}_2}(t) - \alpha I^{\mathbf{x}_2}(t) \\ \alpha I^{\mathbf{x}_2}(t) \end{pmatrix} \tag{3.18}$$

It remains to extend the definition of $s(t)$ to ISI-vectors. It is proposed

here that this should be:

$$s(t) = \frac{\|\mathbf{I}^{\mathbf{X}}(t) - \mathbf{I}^{\mathbf{Y}}(t)\|_1}{\sum_i \max(I_i^X, I_i^Y)} \tag{3.19}$$

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

$$\|\mathbf{I}^{\mathbf{X}}(t) - \mathbf{I}^{\mathbf{Y}}(t)\|_1 = \sum_i |I_i^X - I_i^Y| \tag{3.20}$$

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:

$$s(t) = \frac{\sum_i |I^{\mathbf{x}_i}(t) - I^{\mathbf{y}_i}(t)|}{\sum_i \max(I^{\mathbf{x}_i}, I^{\mathbf{y}_i})} \tag{3.21}$$

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:

$$s(t) = \frac{|\sum_i I^{\mathbf{x}_i}(t) - \sum_i I^{\mathbf{y}_i}(t)|}{\max(\sum_i I^{\mathbf{x}_i}(t), \sum_i I^{\mathbf{y}_i}(t))} \tag{3.22}$$

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

3.3.3 Numerical tests

This multi-unit extension of the ISI distance is tested on the same data that was used to test the multi-unit van Rossum distance in Houghton and Sen (2006). In this simulation, two integrate and fire neurons receive noisy input from two sources. A parameter a determines how much these two inputs are mixed. If $a = 0$ each receives independent input, if $a = 0.5$, each receives an input averaging the two sources. Both neurons also receive shot noise. The two inputs are inhomogeneous Poisson processes, and there are five separate randomly chosen inhomogeneous functions. Each of these are used to drive the neurons 20 times, and the distance function is used to cluster these 100 neurons by stimulus in the usual way. The accuracy of this clustering is evaluated by calculating the Transmitted Information (h) of the clustering. The results are shown in Figures **??** and **??** . Compared to similar graphs in [Houghton and Sen, 2008], the performance is similar to the multi-unit van Rossum metric; but compared to the van Rossum metric, the optimal value of α in the distance measure is less clearly modulated by a.

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

3.3.4 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. [Kreuz et al., 2009], 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., 2009] 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:

$$s(t) = (1-p)\left(\frac{|I^{\mathbf{x}}(t) - I^{\mathbf{y}}(t)|}{\max(I^{\mathbf{x}}, I^{\mathbf{y}})}\right) + p\left(\frac{\sum_i |I^{\mathbf{x}_i}(t) - I^{\mathbf{y}_i}(t)|}{\sum_i \max(I^{\mathbf{x}_i}, I^{\mathbf{y}_i})}\right), \tag{3.23}$$

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 Kreuz extension 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:

$$s(t) = (1-p)\left(\frac{|\sum_i I^{\mathbf{x}_i}(t) - \sum_i I^{\mathbf{y}_i}(t)|}{\max(\sum_i I^{\mathbf{x}_i}(t), \sum_i I^{\mathbf{y}_i}(t))}\right) + p\left(\frac{\sum_i |I^{\mathbf{x}_i}(t) - I^{\mathbf{y}_i}(t)|}{\sum_i \max(I^{\mathbf{x}_i}, I^{\mathbf{y}_i})}\right), \tag{3.24}$$

.

There is a fundamental difference in how each of these distance measures view what a SP metric should be. The Kreuz example believes 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 believes 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 [**I think we should find a few papers where each side of the argument would be given merit by the results**], and it could be useful to have both of these measures, as in equations 3.23 and 3.24, depending on the focus of the experiment.

[**combine the population distance and the average distance graphs into one graph**]

3.4 Adaptive SPIKE & 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 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 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.

The ISI extensions above have a population parameter p that varies from 0 to 1, so a measure of similarity (or dissimilarity) that varies from 0 to 1 would be ideal. The normalised cross-entropy of the ISIs of the two populations is such a measure. Labelling the ISIs as above, the cross-entropy at any point in time t is calculated as:

$$h_{\mathbf{X},\mathbf{Y}}(t) = \frac{1}{\log 2n} \sum_{\mathbf{z}\in\mathbf{x},\mathbf{y}} \sum_i -\frac{I^{z_i}(t)}{\sum_{\mathbf{z}} \sum_i I^{z_i}(t)} \log\left(\frac{I^{z_i}(t)}{\sum_{\mathbf{z}} \sum_i I^{z_i}(t)}\right) \tag{3.25}$$

This is a dissimilarity measure on the ISIs across the two populations. If the ISIs are very similar, then the cross-entropy is close to 1, if they are very dissimilar then it is close to 0. Thus, the cross-entropy is a good candidate for an instantaneous population parameter p, or more accurately $(1-p)$.

Figures **??** and **??** show that this does not work very well, because one

would hope that the adaptive measure would perform on a par with the maximum values of transmitted information from figures **??** and **??**, but it is considerably lower than the max values in both the "average" and "population" ISI measures.

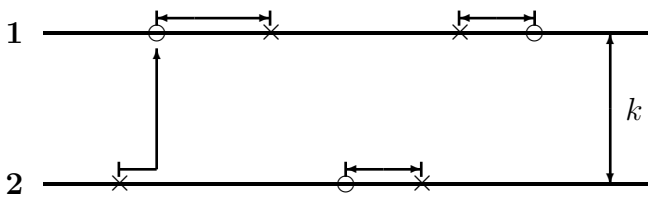


Figure 3.1: An example of how to calculate the gaps between spikes. If the circles represent spikes from $\mathbf{X}$ and crosses represent spikes from $\mathbf{Y}$, then in the picture above, the distance k is simply the cost of relabelling a spike. We also see that the gaps are not necessarily symmetric.

Chapter 4

A simple neuron model

Sparse coding has been observed in neurons in the visual tract [Olshausen and Field, 2004], a simple model is proposed for such neurons.

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, λ_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, λ_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" λ_u and a "down rate" λ_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:

$$r = \frac{\frac{\lambda_u}{d}}{\frac{1}{u} + \frac{1}{d}} = \frac{u\,\lambda_u}{u + d} \tag{4.1}$$

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

4.1 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 λ_d is set to zero, then it is known 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 $\tilde{r}(t_0) = \lambda_u$. Then, the probability is reset 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 Baye's 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 Δt.

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

$$P(X|Y|Z) = \frac{P(Y|X|Z)P(X|Z)}{P(Y|Z)} = \frac{(1 - \lambda_u \Delta t)(1 - d\Delta t)}{1 - r\Delta t} \tag{4.2}$$

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

$$
\begin{aligned}
P(\text{up at } t|Y|Z) &= \lim_{n\to\infty} \prod_{k=1}^{n} \frac{(1-\lambda_u t/n)(1-dt/n)}{1-rt/n} \\
&= \lim_{n\to\infty} \prod_{k=1}^{n} \left(1 + t\frac{r-\lambda_u-d}{n}\right) \\
&= \lim_{n\to\infty} \left(1 + t\frac{r-\lambda_u-d}{n}\right)^n \\
&= e^{(r-\lambda_u-d)t}
\end{aligned}
\tag{4.3}
$$

Recalling from equation 4.1 that $\lambda_u = r(u+d)/u$, get:

$$
p = e^{-\frac{d(r+u)}{u}(t-t_0)} \tag{4.4}
$$

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

$$
\tau = \frac{u}{d(r+u)} \tag{4.5}
$$

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 4.2 above. Then:

$$
\begin{aligned}
y(t+h) &= y(t)P(\text{staying in up-state}|\text{up at t}|Y|Z) \\
&\quad + (1-y(t))P(\text{switching up}|\text{down at t}|Y|Z) \\
&= y(t)\left(\frac{(1-dh)(1-\lambda_u h)}{1-rh}\right) + (1-y(t))\left(\frac{(uh)(1)}{1-rh}\right) \\
&= y(t)\left(1 + h(r-d-\lambda_u-u)\right) + uh
\end{aligned}
\tag{4.6}
$$

This leads to the following ordinary differential equation:

$$y'(t) = (r - d - u - \lambda_u)\, y(t) + u \tag{4.7}$$

This ODE has general solution:

$$y(t) = Ce^{(r-d-u-\lambda_u)(t-t_0)} + \frac{u}{u + d + \lambda_u - r} \tag{4.8}$$

Since $y(t_0) = 1$, get:

$$y(t) = \frac{d + \lambda_u - r}{d + u + \lambda_u - r} e^{(r-d-u-\lambda_u)(t-t_0)} + \frac{u}{u + d + \lambda_u - r} \tag{4.9}$$

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

$$y(t) = \frac{d(u + r)}{u^2 + d(u + r)} e^{-\frac{u^2 + d(u+r)}{u}(t-t_0)} + \frac{u^2}{u^2 + d(u + r)} \tag{4.10}$$

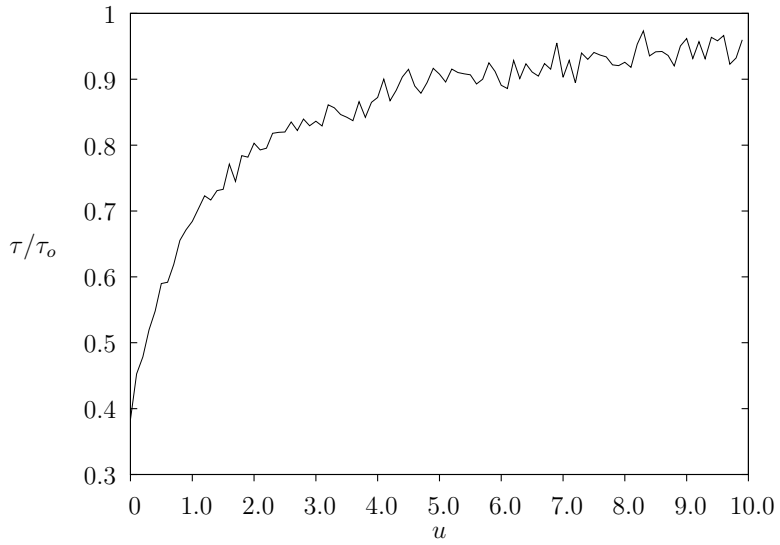


Figure 4.1: This figure shows the ratio of the predicted value for τ in an exponential model over the empirical optimised value τ_o.

This solution was tested against a standard exponential model, which was minimised over τ with a downhill simplex method. Figure 4.1 shows how the ratio of the predicted value for τ 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.

4.2 Markov Process

The switching of the up and down states defines 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 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:

$$\begin{cases} p'_u(t) = -dp_u(t) + dp_d(t) \\ p'_d(t) = up_u(t) - up_d(t) \end{cases} \tag{4.11}$$

This is simply the linear ODE:

$$P'(t) = P(t)Q \tag{4.12}$$

where

$$Q = \begin{pmatrix} -d & d \\ u & -u \end{pmatrix} \tag{4.13}$$

Then the solution to the matrix $P(t)$ is simply the exponential, e^{tQ} of the transition matrix Q.

$$P(t) = e^{tQ} = \begin{pmatrix} \frac{u}{u+d} + \frac{d}{u+d}e^{-(u+d)t} & \frac{d}{u+d} - \frac{d}{u+d}e^{-(u+d)t} \\ \frac{u}{u+d} - \frac{u}{u+d}e^{-(u+d)t} & \frac{d}{u+d} + \frac{u}{u+d}e^{-(u+d)t} \end{pmatrix} \tag{4.14}$$

To calculate the probability of being in the up-state at a time t after a spike at $t = 0$, provided there has been no spike since $t = 0$, it suffices to calculate the exponential of the transition matrix Q_s below:

$$Q_s = \begin{pmatrix} -d - \lambda_u & d & \lambda_u \\ u & -u & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{4.15}$$

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.

The solution to the spiking Markov chain is:

$$P(t) = \begin{pmatrix} Ae^{-\alpha t} + (1-A)e^{-\beta t} & Be^{-\alpha t} - Be^{-\beta t} & 1 - (A+B)e^{-\alpha t} - (1-A-B)e^{-\beta t} \\ Ce^{-\alpha t} - Ce^{-\beta t} & (1-A)e^{-\alpha t} + Ae^{-\beta t} & 1 - (1-A+C)e^{-\alpha t} - (A+C)e^{-\beta t} \\ 0 & 0 & 1 \end{pmatrix} \tag{4.16}$$

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

$$\begin{aligned} -\alpha &= \frac{-(u+d+\lambda_u) + \sqrt{(u-d-\lambda_u)^2 + 4ud}}{2} \\ -\beta &= \frac{-(u+d+\lambda_u) - \sqrt{(u-d-\lambda_u)^2 + 4ud}}{2} \end{aligned} \tag{4.17}$$

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

Specifically:

$$\begin{aligned} A &= \frac{\left(u-d-\lambda_u+\sqrt{(u-d-\lambda_u)^2+4ud}\right)}{2\sqrt{(u-d-\lambda_u)^2+4ud}} \\ B &= \frac{d}{\sqrt{(u-d-\lambda_u)^2+4ud}} \\ C &= \frac{u}{\sqrt{(u-d-\lambda_u)^2+4ud}} \end{aligned} \tag{4.18}$$

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:

$$Q_s = \begin{pmatrix} -d-\lambda_u & d & \lambda_u \\ u & -u-\lambda_d & \lambda_d \\ 0 & 0 & 0 \end{pmatrix} \tag{4.19}$$

This matrix has solution:

$$\begin{pmatrix} Ae^{-\alpha t}+(1-A)e^{-\beta t} & B\left(e^{-\alpha t}-e^{-\beta t}\right) & 1-e^{-\beta t}-(A+B)\left(e^{-\alpha t}-e^{-\beta t}\right) \\ C\left(e^{-\alpha t}-e^{-\beta t}\right) & (1-A)e^{-\alpha t}+Ae^{-\beta t} & 1-e^{-\alpha t}-(C-A)\left(e^{-\alpha t}-e^{-\beta t}\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.20}$$

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

$$\begin{aligned} -\alpha &= \frac{-(u+d+\lambda_u+\lambda_d)+\sqrt{(u-d+\lambda_d-\lambda_u)^2+4ud}}{2} \\ -\beta &= \frac{-(u+d+\lambda_u+\lambda_d)-\sqrt{(u-d+\lambda_d-\lambda_u)^2+4ud}}{2} \end{aligned} \tag{4.21}$$

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

$$
\begin{aligned}
A &= \frac{u-d+\lambda_d-\lambda_u+\sqrt{(u-d+\lambda_d-\lambda_u)^2+4ud}}{2\sqrt{(u-d+\lambda_d-\lambda_u)^2+4ud}} \\
B &= \frac{d}{\sqrt{(u-d+\lambda_d-\lambda_u)^2+4ud}} \\
C &= \frac{u}{\sqrt{(u-d+\lambda_d-\lambda_u)^2+4ud}}
\end{aligned}
\tag{4.22}
$$

To simplify notation, set:

$$
\begin{aligned}
\Delta &= \lambda_u - \lambda_d \\
\gamma &= \sqrt{(u-d-\Delta)^2+4ud}
\end{aligned}
\tag{4.23}
$$

Thus:

$$
A = \frac{u-d-\Delta+\gamma}{2\gamma},\ B = \frac{d}{\gamma},\ C = \frac{u}{\gamma} \tag{4.24}
$$

4.2.1 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:

$$
\mathbf{p(t)} = \mathbf{p_0} e^{Q_s} = (p_u(t), p_d(t), p_s(t)) \tag{4.25}
$$

where

$$
\begin{aligned}
p_u(t) &= (p_0(A-C)+C)e^{-\alpha t} + (p_0(1-A+C)-C)e^{-\beta t} \\
p_d(t) &= (p_0(A+B-1)+1-A)e^{-\alpha t} + (p_0(-A-B)+A)e^{-\beta t} \\
p_s(t) &= 1 - p_u(t) - p_d(t)
\end{aligned}
\tag{4.26}
$$

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 Baye's Theorem, the probability $p(t)$ is just:

$$p(t) = \frac{p_u(t)}{p_u(t) + p_d(t)} \tag{4.27}$$

so, the rate function becomes:

$$r(t) = \Delta p(t) + \lambda_d = \frac{A_0 e^{-\alpha t} + B_0 e^{-\beta t}}{C_0 e^{-\alpha t} + D_0 e^{\beta t}} \tag{4.28}$$

where:

$$\begin{aligned}
A_0 &= \Delta(p_0(-u-d-\Delta+\gamma)+2u) + \lambda_d(p_0(-2\Delta)+u+d+\Delta+\gamma) \\
B_0 &= \Delta(p_0(u+d+\Delta+\gamma)-2u) + \lambda_d(p_0(2\Delta)-u-d-\Delta+\gamma) \\
C_0 &= p_0(-2\Delta)+u+d+\Delta+\gamma \\
D_0 &= p_0(2\Delta)-u-d-\Delta+\gamma
\end{aligned} \tag{4.29}$$

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

$$r(t) = \frac{A_0 + B_0 e^{-\gamma t}}{C_0 + D_0 e^{-\gamma t}} \tag{4.30}$$

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 $C_0 = \gamma + \delta - 2\Delta p_0$:

$$\Delta p_0 = -\frac{C_0 - \gamma - \delta}{2} \tag{4.31}$$

then

$$A_0 = 2\Delta u - \frac{1}{2}C_0(\gamma - \delta - 2\lambda_d) + \frac{1}{2}(\gamma + \delta)(\gamma - \delta) \tag{4.32}$$

Therefore:

$$\frac{A_0}{C_0} = \lambda_d + \frac{1}{2}(\delta - \gamma) = \alpha \tag{4.33}$$

similarly

$$\frac{B_0}{D_0} = \lambda_d + \frac{1}{2}(\gamma + \delta) = \beta \tag{4.34}$$

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

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

$$r(t) = \frac{\alpha C_0 + \beta D_0 e^{-\gamma t}}{C_o + D_0 e^{-\gamma t}} = \frac{\alpha C_0 e^{-\alpha t} + \beta D_0 e^{-\beta t}}{C_o e^{-\alpha t} + D_0 e^{-\beta t}} \tag{4.35}$$

In fact, comparing C_0 to β and D_0 to α, get:

$$C_0 = 2(\beta - \lambda_d - p_0\Delta),\ D_0 = 2(-\alpha + \lambda_d + p_0\Delta) \tag{4.36}$$

which further simplifies equation 4.35 to:

$$r(t) = \frac{\alpha(\beta - \lambda_d - p_0\Delta)e^{-\alpha t} + \beta(-\alpha + \lambda_d + p_0\Delta)e^{-\beta t}}{(\beta - \lambda_d - p_0\Delta)e^{-\alpha t} + (-\alpha + \lambda_d + p_0\Delta)e^{-\beta t}} \tag{4.37}$$

4.2.2 Change in probability when a spike arrives

In the previous section, the rate function, $r(t)$, and the pdf 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 reevaluate 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 Baye's Theorem:

$$\begin{aligned} P(\text{up-state}|\text{spike}) &= \frac{P(\text{spike}|\text{up-state})P(\text{up-state})}{P(\text{spike})} \\ &= \frac{\lambda_u p_u(t)}{\Delta p_u(t) + \lambda_d} \end{aligned} \tag{4.38}$$

Thus:

$$p_u(t) \rightarrow \frac{\lambda_u p_u(t)}{\Delta p(t) + \lambda_d} \tag{4.39}$$

equivalently:

$$\begin{aligned} r(t) \rightarrow &(\lambda_u - \lambda_d)\frac{\lambda_u p_u(t)}{\Delta p_u(t) + \lambda_d} + \lambda_d \\ = &\frac{\lambda_u^2 p_u(t) + \lambda_d^2(1 - p_u(t))}{\lambda_u p_u(t) + \lambda_d(1 - p_u(t))} \end{aligned} \tag{4.40}$$

Equivalently, this can be written as:

$$r(t) \rightarrow (\lambda_u + \lambda_d) - \frac{\lambda_u \lambda_d}{r(t)} \tag{4.41}$$

4.2.3 Calculating the ISI distribution from the estimated rate function

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:

$$p_{ISI}(t) = r(t)e^{-\int_0^t r(s)\,ds} \tag{4.42}$$

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

$$\begin{aligned}\int_0^t r(s)\,ds &= \int_0^t \frac{A_0e^{-\alpha s}+B_0e^{-\beta s}}{C_0e^{-\alpha s}+D_0e^{\beta s}}\,ds \\ &= \frac{\alpha A_0D_0-\beta B_0C_0}{(\alpha-\beta)C_0D_0}t + \frac{B_0C_0-A_0D_0}{(\alpha-\beta)C_0D_0}\log\left(\frac{C_0e^{\beta t}+D_0e^{\alpha t}}{C_0+D_0}\right)\end{aligned} \tag{4.43}$$

Substituting equations 4.29 and 4.23, and noting that $\beta-\alpha=\gamma$, the fractions in the above equation become:

$$\begin{aligned}\frac{\alpha A_0D_0-\beta B_0C_0}{(\alpha-\beta)C_0D_0} &= u+d+\lambda_u+\lambda_d=\alpha+\beta, \\ \frac{B_0C_0-A_0D_0}{(\alpha-\beta)C_0D_0} &= -1\end{aligned} \tag{4.44}$$

Thus equation 4.43 becomes:

$$\begin{aligned}\int_0^t r(s)\,ds &= (\alpha+\beta)t-\log\left(\frac{C_0e^{\beta t}+D_0e^{\alpha t}}{C_0+D_0}\right) \\ &= \log\left(e^{(\alpha+\beta)t}\right)+\log\left(\frac{C_0+D_0}{C_0e^{\beta t}+D_0e^{\alpha t}}\right) \\ &= \log\left(\frac{C_0e^{(\alpha+\beta)t}+D_0e^{(\alpha+\beta)t}}{C_0e^{\beta t}+D_0e^{\alpha t}}\right) \\ &= \log\left(\frac{C_0+D_0}{C_0e^{-\alpha t}+D_0e^{-\beta t}}\right)\end{aligned} \tag{4.45}$$

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

$$\begin{aligned}p_{ISI}(t) &= \frac{A_0e^{-\alpha t}+B_0e^{-\beta t}}{C_0e^{-\alpha t}+D_0e^{-\beta t}}e^{-\log\left(\frac{C_0+D_0}{C_0e^{-\alpha t}+D_0e^{-\beta t}}\right)} \\ &= \frac{A_0e^{-\alpha t}+B_0e^{-\beta t}}{C_0e^{-\alpha t}+D_0e^{-\beta t}}e^{\log\left(\frac{C_0e^{-\alpha t}+D_0e^{-\beta t}}{C_0+D_0}\right)} \\ &= \frac{A_0e^{-\alpha t}+B_0e^{-\beta t}}{C_0e^{-\alpha t}+D_0e^{-\beta t}}\left(\frac{C_0e^{-\alpha t}+D_0e^{-\beta t}}{C_0+D_0}\right) \\ &= \frac{A_0}{C_0+D_0}e^{-\alpha t}+\frac{B_0}{C_0+D_0}e^{-\beta t}\end{aligned} \tag{4.46}$$

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

$$f(x) = \rho\lambda_1 e^{-\lambda_1 x} + (1-\rho)e^{-\lambda_2 x} \tag{4.47}$$

Clearly, then λ_1, λ_2 which are calculated experimentally correspond to α, β. If $\lambda_1 = \alpha$ and $\lambda_2 = \beta$, then it can be calculated that $\rho = p_0$ and thus:

$$\rho\alpha + (1-\rho)\beta = \rho\lambda_u + (1-\rho)\lambda_d \tag{4.48}$$

This gives an expression for ρ in terms of α, β, λ_u and λ_d.

$$\begin{aligned} \rho(\alpha - \beta + \lambda_d - \lambda_u) &= \lambda_d - \beta \\ \rho = \frac{\beta - \alpha + \lambda_u - \lambda_d}{\beta - \lambda_d} &= 2\frac{\gamma + \Delta}{\gamma + \delta} \end{aligned} \tag{4.49}$$

4.3 Testing on data

The previous result allows for testing on real data. The data used in this report is the data used in [Sen et al., 2001]. 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 **CITATION** , the ten presentations of each stimulus are overlaid to counter the impact of the refractory period.

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]. It is used to compare a model probability distribution to a sample. The KS statistic is the d_∞ metric between the cumulative distribution function of the model distribution and the empirical distribution function of the data.

$$D_n = \sup_x \left| \hat{F}_n(x) - F(x) \right| \tag{4.50}$$

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.

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 4.2, was that the H_2 distribution was a drastic improvement on the exponential distribution, but the 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

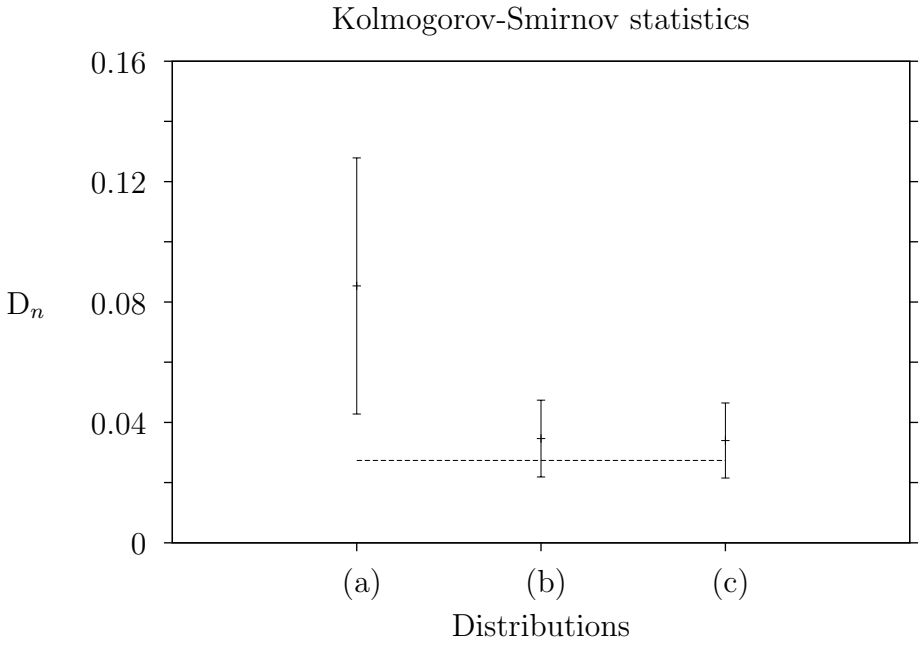


Figure 4.2: The Kolmogorov-Smirnov test statistic D_n for (a) the exponential distribution, (b) the hyperexponential distribution with two modes, and (c) the hyperexponential distribution with three modes. The mean of the $p < 0.05$ critical value is indicated by the broken line.

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.

Chapter 5

Conclusion

more to follow

Bibliography

[Bender and Canfield, 1978] 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.

[Bollobás, 1980] 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 and Gerstner, 2005] 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.

[Dayan and Abbott, 2001] Dayan, P. and Abbott, L. F. (2001). *Theoretical neuroscience*. MIT Press.

[Gillespie and Houghton, 2011] 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.

[Hodgkin and Huxley, 1952] 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.

[Hopfield and Herz, 1995] 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.

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

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

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

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

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

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

[Kreuz et al., 2007] 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.

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

[Narayan et al., 2006] 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.

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

[Newman, 2006b] 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, 2010] Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

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

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

[Pizzuti, 2008] 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.

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

[Sen et al., 2001] 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, 1948] Shannon, C. E. (1948). A mathematical thoery of communication. *Bell Systems Technical Journal*, 27:379–423,623–656.

[Singh and Lesica, 2010] 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).

[Strong et al., 1998] 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.

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

[van Rossum, 2001b] van Rossum, M. (2001b). A novel spike distance. *Neural Computation*, 13(4):751–763.

[Victor and Purpura, 1996] 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 and Purpura, 1997a] Victor, J. D. and Purpura, K. P. (1997a). Metric-space analysis of spike trains: theory, algorithms and application. *Network: Computation in Neural Systems*, 8(2):127–164.

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

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