Spike Sorting based upon Machine Learning Algorithms (SOMA)

PM Horton[†], AU Nicol[‡], KM Kendrick[‡] J.F. Feng[§]
[†]Department of Informatics, Sussex University, Brighton,BN1 9QH, UK
[‡] Laboratory of Cognitive and Behavioural Neuroscience,
The Babraham Institute, Cambridge CB2 4AT, UK
[§] Department Of Computer Science, Warwick University, Coventry, CV2 4AT, UK

Abstract

We have developed a spike sorting method, using a combination of various machine learning algorithms, to analyse electrophysiological data and automatically determine how many neurons are sampled from a single electrode, and discriminate the activities of those individual neurons. Whereas a simple application of a standard unsupervised learning algorithm (Kohonen) would find a known number of clusters, our new approach will find the number of actual clusters (discriminable spikes from individual neurons), and their size, thereby reducing the chance of misclassification. A new pre-processing technique is also discussed. The standard principal component analysis (PCA) may fail to extract the most useful information from this kind of data. Our new approach combines the features acquired using PCA with features describing the geometric shapes (curvature) within the spike waveform. To validate our new spike sorting approach, we have applied it to multi-electrode array datasets acquired from the rat olfactory bulb, and from the sheep infero-temporal cortex, and also from simulated data. The SOMA sofware is available at http://www.cogs.susx.ac.uk/ users/ pmh20.

Keywords: Spike sorting; Neural Networks; Olfactory bulb; Odour; Sheep temporal cortex; pre-processing; Self-organising maps

1 Introduction

After over a century of neurophysiological research we still do not understand the principle by which a stimulus such as an odour, an image or a sound is represented by distributed neural ensembles within the brain. While large numbers of studies have made detailed analyses of response profiles of single cells in isolation, such techniques cannot address holistic issues of how large ensembles of neurones can integrate information both spatially and temporally.

There is little doubt that much of the information processing power of the brain resides in the activities of co-operating and competing networks of neurons (Feng, 2004). If we can unlock the principles whereby information is encoded within these networks as a whole, rather than focusing on the activities of single neurones in isolation, we may come closer to understanding how the brain works. While some progress towards understanding how this is achieved at a gross structural level is being achieved with brain imaging techniques, the only way to provide an understanding at the level of multiple cell-cell interactions in the brain is to record simultaneously from large numbers of cells within a defined system. The two main difficulties in achieving this step have been, firstly, the lack of appropriate hardware to record simultaneously the electrical activity of ensembles of neurones at the single cell level and, secondly, restrictions of current methods in analysing the resulting huge volume of multivariate, high frequency data . In the current paper, we aim to resolve a crucial issue in the second category: spike sorting.

There have been many automatic (software including Klustakwik (Harris, 2002) and BubbleClust (Lipa, 2004)), manual (software including Mclust (Redishi, 2005), Kluster (Hazan, 2004) and Spike2), and semi-automatic methods, i.e. a combination of both, to sort spikes, but none supersedes the rest as each has associated difficulties (Lewicky, 1998).

Normally, the first step in these methods is to find the number of clusters(neurons), where the number formed is dependent on the waveform features chosen to be classified from the outset, within the dataset. Manual methods require vast amounts of user input to achieve this. For example, a number of clusters (template waveforms) corresponding to each of the neurons must be determined manually, which can be highly labour intensive and inefficient. Methods, which are fully automated, rely on a chosen pre-processing method, normally principal component analysis (PCA), which automatically extracts features, thus discriminating the waveforms. A clustering process is then used to find the number of template waveforms, which represent the number of clusters within this feature space. This approach is less time consuming, as the identification of the number of neurons can be automated. However, there are still problems associated with this approach; mainly that most clustering algorithms require a pre-determined number of groups to identify, and PCA does not always produce a number of distinguishable clusters. The semi-automatic methods have a combination of the above problems.

The next step in these methods is to identify, the sets of waveforms that best represent each of the clusters, usually referred to as cluster cutting. These are normally defined either manually by defining the boundaries of each cluster, which again, is very time consuming, or automatically by using a distance measure and finding which spike waveforms are closest to each of the template waveforms (means), resulting in a set of implicit decision boundaries that separate the clusters. Clustering, using this type of automated method, ignores the distribution of the clusters, so that if a subset of a cluster exceeds the implicit boundary, i.e. the subset is closer to the incorrect mean, it will be misclassified. This method will only work if the clusters have a spherical distribution and are well separated, but the likelihood of finding such clusters in real electrophysiological data is minimal, as differential noise levels attributed to the waveforms create differentially distributed clusters.

The technique proposed in this paper has the three following objectives, resolving some of the problems stated above(see also Lewicky, 1998):

1. To form distinguishable clusters (where each represents a biologically plausible firing rate) by evolving the PCA transformation of the waveforms, i.e. adding a number of extra feature components, which describe the geometrical structure of the waveform.

2. To identify, automatically, the number of clusters within the feature space.

3. To reduce the number of waveforms classified to the incorrect groups.

These objectives are dealt with by extending a version of an unsupervised neural network, resulting in a fully automated process which, is a vital consideration when dealing with multiple electrode recordings.

In the first part of our spike sorting methodology, the waveform data is pre-processed using our new combined approach. First, we acquire the PCA components, and if necessary add to this features representing the curvature score for sections of the waveform. The use of additional features, and their number, is dependent on the results produced, i.e. does each cluster formed represent a neuron with an acceptable firing rate.

The next part of the process is associated with finding the number of clusters formed within the feature space. We achieve this by using the feature representations of the waveforms to train a Kohonen neural network. The number of outputs (nodes) used for the network is larger than the expected number of neurons, so that each node will represent a region of the feature space, where a few will correspond to the centres of the clusters. We then extend the approach, by analysing the distribution of the datapoints represented by each node. A node represents a cluster (i.e. the centre) if it corresponds to an area of the feature space where the distribution of datapoints is larger in density.

The final part of our proposed process is to define the waveforms belonging to each cluster. To achieve this and resolve the earlier problems, we propose another extension to the Kohonen process, where sets of nodes are identified, which represent each of the clusters. In each cluster the nodes outside the central node are adapted to sufficiently represent the outer regions of that cluster. Thus a set of implicit boundaries for each cluster is acquired, which, when combined, form a boundary containing the cluster's size and shape. Classification of a spike is then implemented by identifying the node to which it is closest, and then identifying the cluster (central node) to which that node belongs. This resolves the problem of misclassification; if several clusters are in close proximity and one is distributed more widely, a boundary would be defined to encompass this area, thereby reducing the likelihood of waveforms belonging to one cluster, being incorrectly assigned to another.

This new approach is tested using datasets acquired from the rat olfactory bulb and the sheep temporal cortex. We demonstrate that our method out performs others and also show that these processes can be used to extract artifactual waveforms, i.e. clusters, from the data, limiting inaccuracies in the analysis of the results.

2 Spike Sorting Pre-Processing Method

2.1 Feature Extractions

The first stage of our methodology is to transform the spike waveform data into fewer dimensions by extracting the most crucial features, which differentiate the waveforms, thus forming clusters. It has been shown in (Csicsvari et al., 1998) that using Principal Component Analysis(PCA) (a description of this is not described in this paper but can be found in (Jolliffe, 2002; Bishop 1995)) will automatically extract these features and form distinguishable clusters. In Fig.5 (bottom panel), we have shown that this is not always true and can produce results which are not biologically plausible, i.e. one cluster was formed concluding that one neuron, with a firing rate > 200hz, was recorded.

To resolve this, our proposed method adds an identical number of extra components (features) to each of the PCA component sets, where each set corresponds to a waveform. This increases the waveforms feature descriptions, thus breaking the clusters down into smaller ones[1], where each one represents a more clearly defined set of similar waveform shapes. The extra components describe a waveform as a set of curvature values, thus describing the waveform's geometrical structure.

Our proposed pre-processing method is achieved by implementing the following:

1. **The spike waveform data is transformed using PCA:** Any components that contribute to $< 0.5\%$ variance are excluded.
2. **Identify the number of clusters formed and their associated firing rate:** This is achieved by using the clustering method in the next section. Step 3 is not required if the firing rate associated to each cluster is acceptable.
3. **Identify and calculate a minimum number of extra components:** These are added to each of the PCA component sets to form a number of clusters where each has a acceptable firing rate. To achieve this:
   i. Each of the waveforms needs to be smoothed, by using the moving average method.
   ii. A set of curvature scores *curv* is calculated for every spike waveform (Explained in more detail below).
   iii. A minimum number of averages (components) need to be identified, which are calculated using consecutive sets of *curv* scores, that sufficiently describe the curvature features of all the spike waveforms (Explained in more detail below).

[1]This approach can be associated with support vector machine algorithms, where the number of variables are incremented until an acceptable number of clusters has been achieved

2.1.1 Calculating A Set Of Curvature Scores *curv* For A Spike Waveform

A curvature score is calculated for every point t along the spike waveform, where all of the scores form the set *curv*. To calculate a curvature score at point t, we firstly acquire the values of the voltage (V) at $t-1$, t and $t+1$. These values are then used to approximate the first and second derivatives at t, using the central difference approximation. The calculation to obtain the curvature score at t is shown below:

$$\text{curv}(t) = \frac{V''}{(1+V'^2)^{\frac{3}{2}}} \tag{2.1}$$

where V' and V" are the first and second derivatives at point t.

The curvature score for the first and last points of the waveform are not calculated, as they do not have two adjacent points. For example, if there were 48 sampling points we would obtain 46 curvature scores.

2.1.2 Identifying and calculating the extra components

The aim of this stage is to produce a new set of feature descriptions, i.e. appending an identical number of extra components to each of the PCA component sets, which will sufficiently describe the features of the spike waveforms, thus describing their differences. This results in the formation of a number of clusters, where each represents an acceptable firing rate.

To achieve this, firstly, a number of extra components for a spike waveform are calculated by averaging sets, where the number in a set is defined by S, of consecutive *curv* scores. Therefore, each average corresponds to an extra component. Secondly, if the current number of clusters is not acceptable, the number of averages (components) to describe the *curv* sets are increased, thus increasing the curvature description of the spike waveforms. This process divides the clusters, formed from the previous set of components, into smaller ones.

The sequence of steps below describes the process in more detail, where the number of extra components is defined by C:

1. **One extra component is calculated and appended to every PCA component set:** This single component, i.e. $C = 1$, describes the average of all the values in set *curv*, i.e. $S = A$, where A is the number of curvature values in set *curv*. For example, if a is the extra component, a is derived using the following:

   $$a = avg(curv(1...S))$$

2. **Round each new component to a -1 or 1:** When we have a set of curvature components, i.e. averaged curvature scores, we replace each of them with either a -1

or 1. If the curvature average is > 0 a value of 1 is given. If the value is < 0 the value is replaced with a -1. This produces distinguishable clusters.

3. **Identify the number of clusters formed:** We use the clustering process (described in the next section) with this new set of transformed data to identify the number of clusters present, and their associated firing rates.

4. **Replace the previous sets of extra components with a new set, if one of the clusters represents a firing rate exceeding a threshold level:** To create a new set of extra components we implement the following:

   i. The previous set of extra components is replaced with a new set containing one more component, i.e. $C = C + 1$.

   ii. The new set of extra components are then calculated by averaging successive sets of S $curv$ scores, where S is defined by using $S = A/C$. The ith component in the set is then calculated using one of the following:

   $$\text{If } (i == 1) : avg(curv(1...S))$$
   $$\text{Else} : avg(curv((i-1) * S...S * i))$$

   For example, if $C = 4$ and a,b,c,d represent the four components respectively, then each component is calculated using the following:

   $$a = avg(curv(1...S))$$
   $$b = avg(curv(S * 1...S * 2))$$
   $$c = avg(curv(S * 2...S * 3))$$
   $$d = avg(curv(S * 3...S * 4))$$

5. **Repeat steps 2-4:** These steps are continually implemented, until all the clusters formed represent a firing rate below a threshold level.

3 Spike Sorting Clustering Method

3.1 Simulated Data

To be able to explain this methodology we simulated sets of feature components, where a set of 3 components (C_1-C_3) represents a waveform[2]. Throughout the paper we assume that noise attributed to the neuronal recordings, used with our spike sorting process, is gaussian. Therefore, the clusters within the simulated datasets represent gaussian distributions.

To acquire a simulated set containing two clusters, we selected two sets of three values, where each set represents the centre and position of the clusters. 500 feature descriptions,

[2]3 dimensions is only used here for presentation purposes, as this may not be sufficient to describe an individual spike waveform

i.e. 500 C_1-C_3 values, were then created from each of the sets, by adding random gaussian noise to the starting values. For this paper the term datapoint, which is used throughout the paper, refers to the vector of feature components, i.e. C_1-C_3, describing a waveform.

This simulated dataset represents two neuron recordings and is shown in Fig.1 (top left panel). The next sections describe how we identify the number of clusters present within a dataset.

3.2 Kohonen Network

The first stage of the clustering process is to locate, using a Kohonen network, the denser areas of the feature space which represent cluster centres. The Kohonen network will not be discussed in detail here, but can be found in many books and papers including (Kohonen, 1984; Kohonen, 2001).

To begin this stage a number of network outputs are selected, which refer to the number of clusters within the dataset. The assumption is that every output, at the end of the process, will represent a cluster centre. As the number of clusters is not known from the outset, it is impossible to select the precise number of outputs to achieve this assumption. Therefore, we use a number greater than the expected number of clusters, so that a subset of the outputs will represent cluster centres. This subset is identified using a method discussed in the next section, as it is not dealt with using this process. The initial connections to the outputs, i.e. the weight vectors, are then chosen at random from the range of feature component values within the feature space. For the rest of this section we show the outcome of the Kohonen network when used with 9 outputs and trained with the simulated data shown in Fig.1 (top left panel). The figure also shows the beginning of the training process, where the 9 black nodes represent the initial outputs of the network and the region of feature space they represent. For the rest of this paper we refer to a node as an output, where a moved node corresponds to an updated output. The initial positions of each node n, i.e. the weight vector W_n, are chosen at random from the range of C_1-C_3 values in the dataset.

The other features we used in the Kohonen process are:

1. A learning rate that starts at 0.003 and decreases at the start of every epoch.

2. 250 epochs.

3. Euclidean distance measure[3]

4. Fairer competition: The winning node's two adjacent neighbours are also updated but only half the amount. For example nodes 1 and 3 would be updated if the winning node were 2. This is implemented so all the outputs have a chance of representing a portion of the dataset.

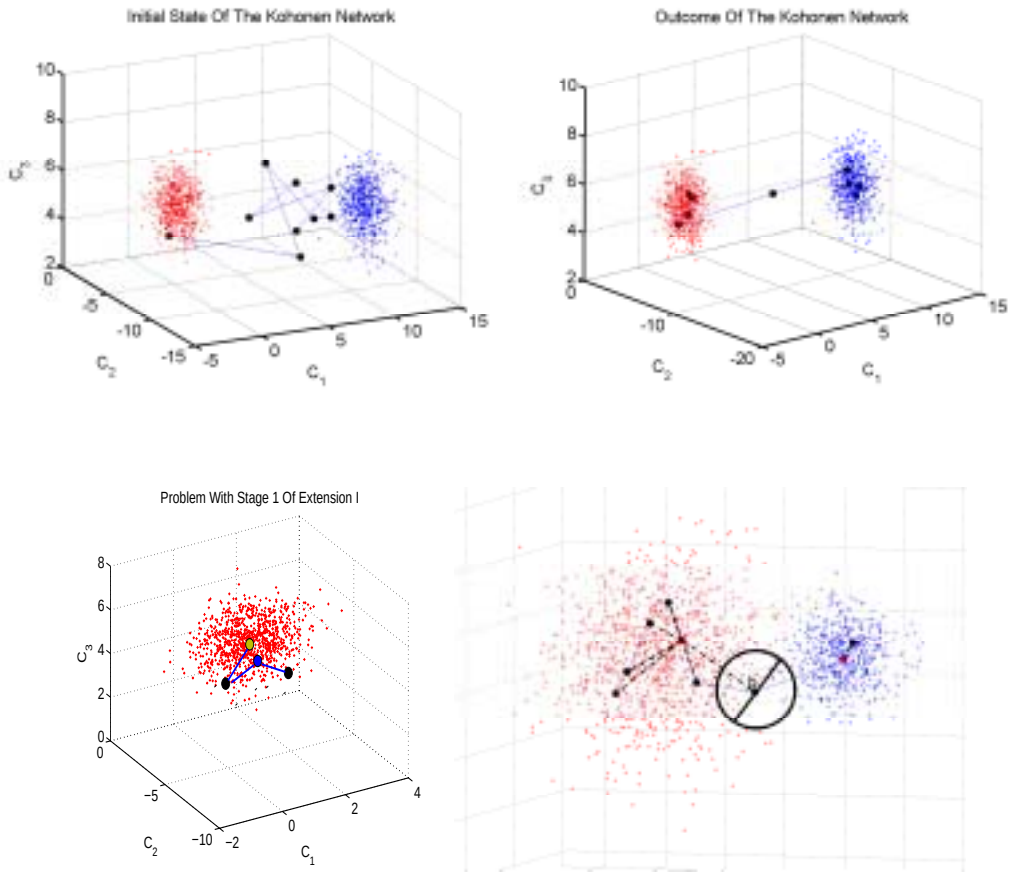

Figure 1: **Top Left:** This shows the simulated data set and the initial state of the Kohonen network. The Black circular markers (nodes) represent the outputs. **Top Right:** This shows the position of the nodes (weight vectors) from using 250 epochs. It also shows all 9 nodes have been updated but 8 of them represent some fraction of the dataset, i.e. within one of the clusters. The lines that connect them represent the node's neighbours. **Bottom Left:** This shows two nodes, yellow and blue, representing the same cluster. The *dens* scores for both are lower compared to their neighbours, thus concluding that they represent different clusters. **Bottom Right:** This shows the data and the results from using the Kohonen process, extension I and II on the second set of simulated data. The red nodes in the middle of the clusters represent the nodes in set R. The nodes connected to the red nodes with black striped lines constitute the associated O sets. The black circular sphere, surrounding node b, and the line through the middle, represent the full and half boundaries respectively.

The final result of the Kohonen network process is shown in Fig.1 (top right panel). All nodes have been updated but 8 of them represent some fraction of the dataset, and 2 of them represent the denser areas, i.e. cluster centres.

The next section describes a method to identify the nodes that represent the centres, thus identifying the number of clusters.

3.3 Extension to Kohonen Network I (Identifying The Number Of Clusters)

This section describes how we extend the Kohonen network approach to identify the number of clusters present within a dataset. This is achieved by analysing and comparing the datapoint distributions that each node represents.

As we firstly use the Kohonen training process, the outputs (nodes) of the network have been adapted to represent different areas of the feature space. Therefore, a subset of the nodes represents the same cluster, where one, and the rest of the subset, represent the centre and the outer area respectively. An example of this is shown in Fig.1 (top right panel). As the density of the datapoins within a cluster decreases from the centre to the outer area and the nodes are distributed to represent various cluster regions, the density represented by each node, compared to its neighbours, would be different. Therefore, we acquire a set of nodes that represent the cluster centres, by identifying the ones that represent a denser area compared to their adjacent neighbours. Consequently, the set compiled can contain nodes that represent the same cluster, so we have to identify and remove them. Hence, we implement two stages to identify a set of central nodes.

The first stage contains two steps to identify a set of nodes, which potentially represent different cluster centres.

- Stage 1

    1. **Calculate the density score for every node:** This score corresponds to the density of the datapoints that a node represents, where a low score represents a high density.
    2. **Identify a set D of possible central nodes:** This is achieved by comparing the density score of every node with its two adjacent neighbours. A node which has a lower score compared to its neighbours would represent a denser area, thus represent a potential cluster centre.

This stage alone may not identify the correct set of central nodes, i.e. two nodes that are in set D may represent the same cluster. To resolve this, a second stage is implemented outlined below:

[3]This is used throughout the paper for all distance related calculations.

- Stage 2

    1. **Verifying the nodes in D represent different clusters:** This is achieved by investigating the density of the area between every two nodes in set D, by moving a new node m, in a series of steps, from one node to another. The changes in the density as node m moves are analysed, to conclude whether both nodes from D represent different clusters. If both nodes are identified as representing the same cluster, one of them is removed from the set.

These stages are discussed in more detail below:

3.3.1 Stage 1: Identify A Set Of Nodes That Represent Potential Cluster Centres

3.3.1.1 Calculating The Density Score For Every Node

To acquire the density level of the area represented by each node, we calculate a score *dens* where a low score represents a high density, i.e. a potential cluster centre, and a high score represents a low density, i.e. the outer area of a cluster. This score is acquired by calculating the average euclidean distance between a node's weight vector and the closest set of datapoints surrounding the node. The calculation for this is shown below:

$$Dens_n = (\sum_{p=1}^{P} M_p)/P \tag{3.1}$$

where M is the set of distances between the node's weight vector and the closest datapoints, P is the number of distances contained in set M, where we used $P = 100$, and n is the node the score is attributed to. Please note, the same datapoint can be used in the calculation of more than one score, as the set of closest datapoints for every node is found by examining the entire dataset.

3.3.1.2 Identify A Set Of Central Nodes

To achieve this we identify the nodes which have a lower *dens* score compared to its two adjacent neighbours, so if

$$(Dens_n < Dens_{(n+1)})\&(Dens_n < Dens_{(n-1)})$$

then node n represents a potential cluster centre.

Once all the nodes have been compared using this condition, we acquire two sets D and ND which contain the nodes that represent the central, and outer areas, respectively.

Using this method on our simulated data, shown in Fig.1 (top right panel), resulted in identifying a set D comprising of two nodes, n_3 and n_7, which have the lowest scores. The *dens* score for all of the nodes are shown in table 1.

n	1	2	3	4	5	6	7	8	9
Dens scores	0.7200	0.6439	0.5988	0.6290	4.6731	0.6538	0.5923	0.6230	0.6478

Table 1: The table shows the nodes n_3 and n_7 represent potential cluster centres. The nodes surrounding these has higher scores, thus represent the cluster outer areas

3.3.1.3 Summary Of The Main Variables In This Section:

1. D=set of nodes that represent potential cluster centres.
2. ND=set of nodes that represent the clusters outer areas.

3.3.2 Stage 2: Verifying The Nodes In Set D Represent Different Clusters

It is possible for the set D to contain many nodes that represent the same cluster. The reason for this is shown in table 3 where two nodes (n_2 and n_4) have lower *dens* scores compared to their neighbours, concluding that they represent the centres of different clusters. This conclusion is not correct, as the nodes actually represent the same cluster, shown in Fig.1 (bottom left panel), where n_2 and n_4 represent the centre and outer area of the cluster respectively. This incorrect result is produced because both neighbours of n_4 (blue node) are further out causing the score of n_4 to be lower than its neighbours. n_2 (yellow node) would be identified as representing the centre of the cluster as it has a lower *dens* score than n_4.

n	1	2	3	4	5
Dens scores	0.7898	0.5678	0.6898	0.6459	0.7061

Table 2: The table shows there are two nodes, n_2 and n_4, that represent potential cluster centres.

To identify whether two nodes represent the same cluster, we analyse the datapoint distribution in the area between both. For example, if we analysed the datapoint distribution in a series of steps starting from n_2 (we start from this node as it represents the lower *dens* score) to n_4, we would find the distribution becomes less dense as we move from the centre

n_2 to the outer area of the cluster n_4. Therefore, if the *dens* score at a number of positions, equally apart, from n_2 to n_4 were calculated, the scores respectively would increase and be $< dens_{n_4}$, thus concluding that both represent the same cluster. An Example set of *dens* scores for four positions between n_2 and n_4, which would produce this conclusion, are shown in table 3 (*dens* scores 1).

Positions	$0(n_2)$	1	2	3	$end(n_4)$
dens scores 1	0.5678	0.5873	0.6068	0.6264	0.6459
dens scores 2	0.5678	0.7874	0.8068	0.7564	0.6459
numd scores 1	50	38	34	28	21
numd scores 2	50	12	0	16	21

Table 3: This table shows the distribution of the datapoints for four consecutive positions between n_2 and n_4, using the *dens* and *numd* scores. Rows 1,3 and 2,4 would conclude that both nodes represent the same or different clusters respectively.

It is also possible to conclude whether two nodes represent different clusters. If we assume n_2 and n_4 represent different clusters, the set of *dens* scores corresponding to the defined positions from n_2 to n_4, would relate to one of the following. The set would either, contain a score $> dens_{n_4}$, i.e. the position is outside the cluster n_4 represents, or, increase for the first few scores, i.e. the first few positions are within the outer area of the cluster n_2 represents, and then decrease, i.e. the remainder of the positions are outside, and within, another cluster. An Example set of *dens* scores for four positions between n_2 and n_4, which would produce this conclusion, are shown in table3(*dens* scores 2). This conclusion could also be made from the second step as the score is $> dens_{n_4}$.

The conclusion, whether two nodes represent the same cluster, produced, using a set of *dens* scores, can be verified using another method. This new method analyses the number of datapoints *numd*, contained in a boundary, surrounding each of the defined positions between the two nodes. The scores in Table 3(*numd* scores 1), where no score is $> numd_{n_4}$ would conclude that n_2 and n_4 represent the same cluster. The scores in Table 3(*numd* scores 2), or, a score is $< numd_{n_4}$, would conclude that both nodes represent different clusters.

By analysing the *dens* and *numd* scores over a number of positions, in an area of datapoints, between two nodes taken from set D, we can identify the nodes within the set that represent different clusters. If two nodes are found to represent the same cluster, the node that has the highest *dens* and lowest *numd* score is removed from the set. Thus, a set of nodes is formed, where each one represents the centre of a cluster.

To achieve this we implement the following:

1. **An extra output (node) m, is added to the Kohonen network, and is used to represent various regions of datapoints in the area between two nodes:** Node m is moved from a start node sn to a destination node dn, i.e. $W_m = W_{dn}$ to

$W_m = W_{sn}$, in a series of equal steps (positions). The $dens_m$ and $numd_m$ scores at every new position (update) are calculated.

2. **Select an** sn **and** dn **node from set** D**:** (The selection process is explained later).

3. **Conclude whether** sn **and** dn **represent the same cluster, by averaging and analysing the** $dens_m$ **and** $numd_m$ **scores over sets of consecutive steps:** The average of these scores are used, i.e. smoothing, to improve reliability. Two procedures are used to average and analyse these scores, where the conclusion from one will be verified by the other:

   i. The procedure *densmjourney* analyses the average $dens_m$ scores and compares them to the $dens_{dn}$ score (described below).

   ii. The procedure *numdmjourney* analyses the average $numd_m$ scores and compares them to the $numd_{dn}$ score (described below).

4. **Form a new set which contains the identified central nodes:**This is achieved by continually implementing steps 2 and 3, until every node in set D is identified as representing, or not, a cluster centre.(Explained later in this section).

In Fig.2, using the data from Fig.1 (top right panel), shows an example of node m (black marker) moving between two nodes (blue markers), i.e sn and dn, taken from set D, which represent two potential cluster centres. The striped red line is the total set of positions m can represent, therefore every position represents a different region of the feature space, i.e. different *dens* and *numd* scores. The number of steps (positions) m uses to make the journey, from one node to another, is chosen, as using all would be computationally inefficient. For this paper 50 steps(positions) were used and we refer to the term, journey, as the number of steps. The figure shows the 10th position (update) of node m, where the left panel shows the closest set of datapoints (connected with a set of lines) used to create the $dens_m$ score. The right panel shows the datapoints that are used to create the $numd_m$ score, i.e. the number of datapoints within the boundary (sphere) surrounding m.

3.3.2.1 The *densmjourney* procedure

This procedure calculates the averages of successive sets of $dens_m$ scores, where the consecutive $dens_m$ scores correspond to the positions node m has represented from sn to dn. These averages are then analysed to conclude whether both nodes represent the same cluster.

To achieve this analysis we implement the following:

1. Calculate the $dens_{dn}$ score.

2. Calculate the $dens_m$ score at every update of m.

3. Average consecutive sets of the $dens_m$ scores (explained below).
4. Compare the averages, using the three conditions (explained below), to conclude whether sn and dn represent the same cluster.

Calculating the $dens_m$ Averages

A set of averages $Densavg_m$ is produced using consecutive sets of $dens_m$ scores, i.e.:

$$Densavg_m(a) = (\sum_{i=1}^{G}(dens_m(i + ((a-1) * G))))/G \tag{3.2}$$

where G is the number of $dens_m$ scores used for the average, a is the averaged scores index, i.e. a=1...T/G and T is the number of steps (positions) to cover the journey. For this paper $G = 5$ and $T = 50$.

Set Of Conditions

Every average in set $Densavg_m$ is compared with the previous one, to identify which one of the conditions, stated below, the set of scores satisfy, concluding whether sn and dn represent the same cluster:

Condition 1: If fulfilled, this states that both nodes represent the same cluster.

The condition shown below would be satisfied if node m has a starting position, i.e. $W_m = W_{sn}$, in the centre of the cluster and moves toward the outer area where dn is positioned, i.e.:

$$((Densavg_m(a-1) < Densavg_m(a))\&(Densavg_m(a) < Dens_{dn})$$

For example, both nodes would be in the same cluster if the movement of m, produced $Densavg_m$ scores of $0.6, 0.7, 0.8, 0.9, 1$ where $Dens_{dn}$=1.1.

Condition 2: If fulfilled, this also states that both nodes represent the same cluster.

The condition below is satisfied, if node m moves over the centre of the cluster, as the starting position of m, i.e. $W_m = W_{sn}$, was not quite centre, and then moves toward the outside of the cluster where dn is positioned, i.e.:

$$(Densavg_m(a-1) > Densavg_m(a))\&(Densavg_m(a) < Dens_{dn}),$$

Node m is moving towards the centre, and then:

$$(Densavg_m(a-1) < Densavg_m(a))\&(Densavg_m(a) < Dens_{dn})$$

Node m has crossed the centre and is moving to the outer area.

For example, both nodes would be in the same cluster, if the movement of m produced $Densavg_m$ scores of $0.5, 0.4, 0.6, 0.7, 0.8$ where $Dens_{dn}$=0.9.

Condition 3: If fulfilled, this states that both nodes represent different clusters.

The condition below is satisfied, if node m moves toward the outer area of the cluster and then moves towards another i.e.:

$$Densavg_m(a-1) < Densavg_m(a)$$

Node m is moving towards the outer area of the cluster, and then:

$$Densavg_m(a-1) > Densavg_m(a)$$

Node m is moving closer, or into, another cluster.

For example, both nodes would be in different clusters if the movement of m, produced $Densavg_m$ scores of $0.5, 0.6, 0.7, 0.65, 0.55$ where $Dens_{dn}$=0.5. Condition 3 is also satisfied if $(Densavg_m(a) > Dens_{dn})$.

3.3.2.2 The procedure *numdmjourney*

This procedure calculates the averages of successive sets of $numd_m$ scores, where the consecutive $numd_m$ scores correspond to the positions node m has represented from sn to dn. These averages are then analysed to conclude whether both nodes represent the same cluster.

To achieve this analysis we implement the following:

1. Calculate the $numd_{dn}$ score(explained below).
2. Calculate the $numd_m$ score at every update of m.
3. Average consecutive sets of the $numd_m$ score (explained below).

4. Compare the averages, using the three conditions (explained below), to conclude whether sn and dn represent the same cluster.

Calculating the $numd$ score

The $numd$ score for a node n, i.e. $numd_n$, is the number of datapoints that are positioned within the boundary rad surrounding n, i.e. the number of datapoints that satisfy $dist\{W_n, x\} < rad$, where x is a datapoint vector and W_n is the node's weight vector.

The boundary rad is the same for both m and dn, and acquired by:

1. Identifying two nodes in set ND, n_1 which is the closest to node sn, and n_2 which is the closest to node dn.
2. The rad value is the smallest distance between these nodes, i.e. $rad = dist\{W_{sn}, W_{n_1}\}/2$ or $rad = dist\{W_{dn}, W_{n_2}\}/2$.

This provides a boundary which is considerable enough to only analyse datapoints which are within the sn and dn clusters, at the start and end of node m's journey respectively. Otherwise, datapoints from other clusters, surrounding these, would be included in the scores and could create an incorrect conclusion.

Calculating the $numd_m$ averages

A set of averages $Numdavg_m$ is produced using consecutive sets of $dens_m$ scores, i.e.:

$$Numdavg_m(a) = (\sum_{i=1}^{G}(numd_m(i + ((a-1) * G))))/G \tag{3.3}$$

where G is the number of $numd_m$ scores used for the average, a is the averaged scores index, i.e. a=1...T/G, and T is the number of steps (positions) to cover the journey. For this paper $G = 5$ and $T = 50$.

Set Of Conditions For $numdmjourney$

Every average in set $Numdavg_m$ is compared with the previous one, to identify which one of the conditions, stated below, the set of scores satisfy, concluding whether sn and dn represent the same cluster:

Condition 1: If fulfilled, this states that both nodes represent the same cluster.

The condition below is satisfied, if node m has a starting position, i.e. $W_m = W_{sn}$, in the centre of the cluster and moves toward the outer area where dn is positioned, i.e.:

$$(Numdavg_m(a-1) > Numdavg_m(a))\&(Numdavg_m(a) > Numd_{dn})$$

For example, both nodes would be in the same cluster if the movement of m, produced $Numdavg_m$ scores of $40, 38, 37, 34, 30$ where $Numd_{dn}$=25.

Condition 2: If fulfilled, this also states that both nodes represent the same cluster.

The condition below is satisfied, if node m moves over the centre of the cluster, as the starting position of m, i.e. $W_m = W_{sn}$, was not quite centre, and then moves toward the outside of the cluster where dn is positioned, i.e.:

$$(Numdavg_m(a-1) < Numdavg_m(a))\&(Numdavg_m(a) > Numd_{dn})$$

Node m is moving towards the centre, and then:

$$(Numdavg_m(a-1) > Numdavg_m(a))\&(Numdavg_m(a) > Numd_{dn})$$

Node m has crossed the centre and is moving to the outer area.

For example, both nodes would be in the same cluster if the movement of m produced $numdavg_m$ scores of $28, 30, 32, 27, 21$ where $Dens_{dn}$=10.

Condition 3: If fulfilled, this states that both nodes represent different clusters.

The condition below is satisfied, if node m moves toward the outer area of the cluster and then moves towards another, i.e.

$$Numdavg_m(a-1) > Numdavg_m(a)$$

Node m is moving towards the outer area of the cluster, and then:

$$Numdavg_m(a-1) < Numdavg_m(a)$$

Node m is moving closer to another cluster.

For example, both nodes would be in different clusters if the movement of m produced $Numdavg_m$ scores of $30, 25, 20, 22, 28$ where $Numd_{dn}$=36. Condition 3 is also satisfied if $(Numdavg_m(a) < Numd_{dn})$.

3.3.2.3 Form A New Set Which Contains The Identified Central Nodes

A new set R is formed from set D, which contains the identified central nodes, i.e. the nodes that represent different clusters. To achieve this we implement the following:

1. Acquire a set SD, by sorting the nodes in D into $dens$ ascending order.

2. A number of nodes, sn and a set of dn, are selected from SD, where sn and every dn node, in turn, are used with the $densmjourney$ and $numdmjourney$ procedures. If the outcome of both is $condition1$ or $condition2$, the node in SD related to dn is tagged, as it represents the same cluster as sn. Selecting the sn and dn nodes, and producing the R set is discussed below:

   i. sn is a node from the SD set, which has the lowest $dens$ score, is not tagged, or, not in the R set. For example, the first sn node to be chosen would be $SD(1)$. Once sn is chosen it is appended to the R set.

   ii. The nodes in SD, which have not been tagged or in the R set, correspond to the set of dn nodes.

3. Step 2 is repeated until all the nodes in SD are either tagged or in the R set.

4. The tagged nodes form a set ND, which represent the cluster outer areas.

As the nodes in SD can be used more than once in this process, a number of rad values (used within the $numdmjourney$ procedure) would have been associated to each node. The lowest rad score for every node is recorded, and used in subsequent analyses throughout the paper.

3.3.3 Summary Of The Main Variables In This Section:

1. R=set of nodes that represent the cluster centres.

2. ND=set of nodes that represent the cluster outer areas.

3.4 Extension to the Kohonen Network II (Reducing Misclassification)

This section describes a further extension to the Kohonen network, which reduces the number of waveforms incorrectly classified. The typical classification process involves classifying each of the datapoints (waveforms) to the closest central node, from set R, thus forming implicit boundaries between the nodes, which separate the clusters. Therefore, a subset of datapoints which, are closer to the incorrect central node, i.e. pass the implicit boundary, would be misclassified. This can happen when one cluster has a wider distribution area than another, shown in Fig.3 (bottom right panel). A subset of the datapoints from the left cluster would be misclassified as they are closer to the right central node, i.e. they have passed the implicit boundary (black line) formed by the two central nodes.

To resolve this misclassification problem, we use the nodes in both sets R and ND in the classification process. Thus, every cluster is represented by a set of nodes O, which is a subset of ND, and a node from R, where both are positioned within the outer region and centre of the cluster respectively. Therefore, if a datapoint is closer to a node in O it is classified to the corresponding node in set R, thus reducing misclassification, i.e. the implicit boundaries are formed from the sets O improving the separation between the clusters. For the example in Fig.3 (bottom right panel), if a number of nodes represented the larger cluster's outer area, these areas would be associated to the central node, thus a greater subset of the cluster would be correctly classified, i.e. the implicit boundary is moved further towards the cluster's outer edge. To produce a further reduction, the nodes in the O sets are moved further into the outer regions of the cluster to sufficiently represent them, i.e. the implicit boundaries are moved to produce an optimal separation.

To achieve a reduction in misclassification we implement the following:

1. **The Identification Process: Associate a set of nodes O, acquired from ND, to every node in R, where a set O and the corresponding node in R represent the same cluster.** This is achieved by using the $densmjourney$ and $numdmjourney$ procedures outlined in extension I, where sn will be a node from R and dn will be a node from ND. If the outcome using these two procedures is $condition1$ or $condition2$ then dn is within the same cluster as sn and appended to the corresponding O set.

The reduction in misclassifications, when using these identified nodes within the classification process, may still be insufficient as using the standard Kohonen method in the first stage, occasionally, does not distribute the nodes sufficiently enough to maximise the representation of each cluster, thus subsets of clusters can still be misclassified. Therefore, the nodes in each set O are moved, further into, and around, the outer area of the cluster they represent, reducing the chance of misclassification further.

To achieve this, we use a second set of epochs (for this paper the extra number was 250) and for every epoch we implement the following:

2. **The Pushing Process: At the start of every epoch, a node is identified from every set O, which is closest to the corresponding node in set R. The nodes are then pushed further into the outer area of their associated cluster.** Each node's new position, i.e. the position the node will be pushed to, is analysed using a set of conditions to verify that the new position is not outside the cluster or within another. If the verification process proves this, the node is moved.

3. **The Dispersing Process: For the rest of the epoch, the nodes in each set O are dispersed around the cluster they represent, using the Kohonen training process.** To ensure that the nodes reside within the outer area of the cluster, they cannot be updated using (i.e. move towards) datapoints (a) within the centre of the cluster, or (b) outside of the cluster.

These two processes have been designed to slowly disperse the nodes in each set O. This prevents subsets of the nodes in O being trapped in representing one area of the cluster, where there are still unrepresented areas.

To visualise this extension, we show an example of the pushing and dispersing stages graphically in Fig.3 (top and middle panels), where the central and connected outer nodes represent the R and O sets respectively. The nodes in the O sets have been pushed and dispersed, over a number of epochs, to represent, maximally, the outer area of the clusters.

Once these stages are complete, we classify each of the datapoints using the new classification process described below:

4. **The Classification Process: Classify each datapoint to a node in set R.** The closest node to the datapoint is identified using the euclidean distance measure. The identified node is either from a set O, so the datapoint is classified to the corresponding node in R, or set R, so the datapoint is classified to the identified node.

The first three processes are explained in more detail below:

3.4.1 The Identification Process

To identify a set of nodes O for every node in R, we have to:

1. **Identify the node in R, each of the nodes in ND is closest to, thus forming a set of closest nodes Q for every node in R:** This is implemented as there is a greater chance a node from Q, and the corresponding node in R, represent the same cluster.

2. **Verify that a node in R and its corresponding Q set represent the same cluster:** This is achieved using the procedures *densmjourney* and *numdmjourney* to verify the nodes *sn*, which is a node from R, and *dn*, which is a node from the corresponding Q set, represent the same cluster, so:

   i. If the outcome of both procedures is condition 1 or 2 then *dn* is appended to the corresponding O set.

   ii. If condition 3 is satisfied by *dn*, it is tested against the other nodes in R. If every test results in satisfying *condition*3, i.e. *dn* is identified as not belonging to any cluster, the node is ignored.

   Each node in R and the corresponding Q set are verified using this process.

To derive the *dens* and *numd* score for *dn*, we use the datapoints that are positioned within the cluster that both *sn* and *dn* potentially represent, i.e. the datapoints between the two nodes. This is to prohibit the use of datapoints from other clusters, i.e. behind *dn*, which may produce incorrect conclusions. The modifications to both *numdmjourney* and *densmjourney* to achieve this are described below:

3.4.1.1 Changes To The Procedure *numdmjourney*

The datapoints used to derive $numd_{dn}$ depends on where they are positioned within the boundary (*rad*) surrounding *dn*(node from Q), with respect to *sn* (corresponding central node from R) and *dn*. The spherical boundary, i.e. *rad*, surrounding *dn* is implicitly divided through the middle so that one half of the boundary faces *sn*, i.e. the division is through *dn* at a right angle to *sn*. The half sphere facing *sn* contains the datapoints to derive the score. The datapoints in the other half, i.e. behind *dn*, could be positioned within another cluster, so are ignored.

An example of this is shown in Fig.1 (bottom right panel), where a spherical boundary (black circle), i.e. *rad*, surrounding node *b* has a division line through it. The datapoints contained in the half circle facing the central node *sn* (red marker) would be used. If the full boundary was used (as in the previous version of *numdmjourney*) the datapoints from the right cluster would also be included, which could result in an incorrect conclusion, i.e. if $numdavg_m(a) < numd_{dn}$ was satisfied as node m moves from *sn* to *dn*(*b*), would imply *sn* and *dn*(*b*) represent different clusters.

As *dn* moves around the cluster (part of a later process), the division used to form the half sphere changes with respect to *sn*. Therefore, the datapoints used are always within the cluster represented by *sn*.

To identify if a datapoint is positioned within the half sphere boundary of *dn* facing *sn*, we implement the following:

1. **Calculate an angle C which identifies if a datapoint x is positioned within sn and dn:** The angle C is calculated using W_{sn}, W_{dn} and x(datapoint vector), so that if $C < 90$, x is positioned between the node dn and sn.
2. **Identify whether the datapoint x is also within the specified boundary rad:** i.e. $(dist\{W_{dn}, x\} < rad)$.

Calculating Angle C

To find angle C we need to rearrange the cosine law (shown below):

$$c^2 = a^2 + b^2 - 2ab * CosC \tag{3.4}$$

where $a = dist\{W_{dn}, x\}, b = dist\{W_{sn}, W_{dn}\}, c = dist\{W_{sn}, x\}$.

Calculating the $numd_{dn}$ score

The $numd_{dn}$ is now the number of datapoints that satisfy

$$(dist\{W_{dn}, x\} < rad)\&(C < 90)$$

where rad is the lowest value associated to the central node (sn) in extension I ($numdmjourney$ procedure) during Stage 2.

3.4.1.2 Changes To The Procedure *densmjourney*

The $dens_{dn}$ score is calculated using 3.1, where the closest datapoints that derive the score also have to be within the sn and dn nodes, i.e. $C < 90$, where the process to calculate C is in the previous section.

3.4.2 The Pushing Process

At the start of every epoch a node is chosen from every set O and pushed away from its corresponding central node, contained in R, further into the outer region of their associated cluster.

The pushing process is achieved by associating a new boundary, unrelated to the rad values, to each of the central nodes in R and to each of the nodes in sets O. Therefore, if the boundaries from a node in set O and the corresponding central node overlap, the node from O is pushed away. The nodes in set R are no longer moved so they remain within the cluster centres, however, their boundaries increase over time pushing the nodes in O further out.

An example of this is shown in Fig.4(bottom panel), where the black circles surrounding the central node and a node from set O, b, represent the boundaries. The central node's boundary would be increased, thus pushing node b further out.

The pushing process is implemented in this way, as in a later stage the nodes in every set O are dispersed around the outer regions of their associated cluster. This is achieved by using the Kohonen training process with a limited set of datapoints, where a subset of the restricted datapoints is contained within the boundaries surrounding the nodes in R. As the boundaries increase, the restricted number of datapoints also increase, preventing the nodes moving towards the centre, thus their movement is retained within the outer area.

As clusters can be in a variety of shapes and sizes some nodes in O may need to be pushed out further than others. For example, if a cluster represented an elongated elliptical shape, there are two axes that define this, one longer than the other. To represent the longer (elongated) sections of the cluster, some nodes would have to be pushed further out compared to the nodes representing the smaller sections. Therefore, one boundary associated to the central node, to push out all the nodes in O would not be sufficient. The boundary would either be too small to represent the larger sections, i.e. the nodes are not pushed out far enough, or too large to represent the smaller sections, i.e. the nodes have been pushed out of the cluster or into the larger sections. To resolve this, a set of boundaries is associated to each node in R, where each one in the set is associated to a node in the corresponding O set. The nodes in a O set, which are associated to smaller and larger boundaries, would represent smaller and larger sections of the cluster respectively.

To achieve this pushing process we:

1. **Associate a set of boundary values** *bound* **to each central node** r **in** R**, where each value in** *bound* **is associated to a node in** r**'s corresponding** O **set:**

    i. Each central node r is associated to a set $bound_r$ containing O_n boundary values, where O_n is the number of nodes within the corresponding O set.

    ii. All of the values in a $bound_r$ set are initially the same and are calculated using $dist\{W_r, W_g\}/2$, at the beginning of extension II. W_g is the weight vector (node), from the O set associated to r, closest to node r.

2. **Associate a constant boundary value** *bnd* **to each node in the** O **sets:** A value of 1 was given to all nodes.

3. **A node is pushed from every set** O**:** At the start of every epoch, the node closest to the corresponding node in set R in each set O is identified and pushed out:

    i. Every identified node's boundary is increased, unless the node satisfies a condition (explained below), to overlap the boundary of the corresponding central node. Therefore, the identified nodes with overlapping boundaries are pushed out (explained below) to a position, where the boundaries of both no longer overlap.

3.4.2.1 Identifying and Pushing A Node From An O Set.

We firstly identify a node g from set O, which is closest to r (the corresponding node from set R). The following steps are then implemented if no condition, explained in the next section, is satisfied:

1. The boundary for r associated with g is increased using:

$$bound_{rg} = dist\{W_r, W_g\} - bnd_g + \kappa \tag{3.5}$$

   where κ adds a small amount to the boundary, which was 0.01 for this paper, and bnd_g is the boundary for node g.

2. Increasing $bound_{rg}$ then satisfies the condition:

$$dist\{W_r, W_g\} < (bound_{rg} + bnd_g)$$

3. So node g's weight vector is updated, i.e. pushed out, using:

$$W_g = W_g - \lambda_2 * (W_r - W_g) \tag{3.6}$$

   where λ_2 is the amount to push the node, which was 0.1 for this paper.

3.4.2.2 Verifying A Node's Push Position

When a node is chosen to be pushed, we verify that the node will not be moved out, or into another, cluster. To achieve this, **node g is not moved, i.e. $bound_{rg}$ is not increased, when a condition below is satisfied.**

The first two conditions analyses the $numd_g$ score at every push, i.e. the number of datapoints within the boundary surrounding node g. As node g moves to the outside of the cluster it will represent an area with a sparser distribution, therefore, the $numd_g$ score would decrease with every push. The final condition analyses the position node g will be pushed to, i.e. is the new position within another cluster?

Calculating the $numd$ score for every push

The $numd_g$ score is the number of datapoints that satisfy:

$$(dist\{W_g, x\} < rad_r)\&(C < 90)$$

where:

1. x is a vector of a datapoint.

2. The rad values associated to the R nodes, i.e. the lowest values identified during extension I Stage 2, are also associated to the nodes in their corresponding O sets. Therefore, the rad_r value corresponds to the rad value associated with the O set to which g belongs.

3. C is calculated using the rearranged form of equation 3.4, where $a = dist\{W_g, W_x\}, b = dist\{W_r, W_g\}, c = dist\{W_r, W_x\}$.

Set of conditions

Condition 1: This condition identifies whether node g currently represents an area near the edge of the cluster: This is achieved by analysing the current $numd_g$ score to identify whether the node represents an acceptable number of datapoints, i.e. does the node represent an area near the edge of the cluster? This can be identified in two ways:

1. If $(numd_g < \alpha_g)$ is satisfied, the node currently represents an acceptable area of the cluster.

    i. α_g is calculated using the $numd_g$ value of the first push and dividing it by a reasonable number.

2. If $(numd_g == 0)$ is satisfied, the node represents an empty region of the feature space, i.e. it is outside the cluster.

Condition 1 would be satisfied when the clusters are clearly separated, however, if they are in close proximity, and slightly overlapping, the outer edge of a cluster may not be identified. Therefore, condition 1 would fail to identify whether a node has moved into another cluster. The two conditions below resolve this:

Condition 2: This condition identifies whether node g, over a number of pushes, is gradually moving further into another cluster: This is achieved by analysing the average $numd_g$ and $dens_g$ scores.

To achieve this:

1. The scores $dens_g$ and $numd_g$ are calculated whenever the node is pushed. $dens_g$ is calculated using equation 3.1, where the score is derived using datapoints that satisfy $C <= 90$, where C is calculated using the above.

2. The $numd_g$ and $dens_g$ scores for every two pushes are averaged and appended to the sets nav_g and dav_g, where nav_g and dav_g represent the $numd_g$ and $dens_g$ averages respectively.

3. The current average (p) in sets dav_g and nav_g is compared with the previous $(p-1)$, to identify whether node g has moved into another cluster. The condition below is satisfied when the node is continually pushed into a denser area, so if:

$$(davg_g(p-1) > davg_g(p))\&(navg_g(p-1) < navg_g(p))$$

node g is moved to a previous position which did not satisfy this condition, i.e. the node is placed into the correct cluster, and is no longer pushed.

Condition 3: This condition analyses whether the position, node g will be pushed to, is within another cluster: The new position vector np is examined with the other cluster node sets, i.e. is np positioned between another cluster's central node f and a node from its associated O set.

To do this we implement the following:

1. Identify a set of nodes Y that contains a node from every set O, except for the set g is from, which is closest to the new position np.

2. Each of the nodes in Y are analysed with their corresponding central node from R. The condition below is satisfied when np is positioned between them, i.e. np is positioned within another cluster if:

$$dist\{W_f, np\} < dist\{W_f, W_h\}$$

where W_h is the weight vector (node) from set Y and W_f is the weight vector of the corresponding central node from set R.

Conditions 2 and 3 are both used to negate the disadvantages associated to each one when used alone. Condition 3 can be satisfied earlier than condition 2, but is dependent on the speed of the nodes, in the O sets, reaching the outer areas. Therefore, the nodes from one cluster could be pushed deep into another before condition 3 is satisfied, hence the need for condition 2 which prevents this.

3.4.3 The Dispersing Process

For the rest of the epoch, the nodes in the O sets are used in the Kohonen training process, so they move to, i.e. represent, unrepresented regions of their associated cluster. Thus, the nodes are dispersed around the outer area of their respective cluster.

The standard training process used is slightly modified:

1. Fairer competition is no longer used.
2. A winning node can only be from one of the O sets.
3. A restriction on the data used in the training process is implemented, preventing the nodes from moving into the centre or outside the cluster (Explained below).

3.4.3.1 Restricting the data used in the training process

The Kohonen training process is used with the dataset, so that a winning (closest) node is identified for every datapoint.

The winning node d is not updated using a datapoint x that is either:

1. Within the boundary of the corresponding central node r, i.e. $bound_{rd}$
2. Behind the winning node d with respect to r, i.e. if $C > 90$, where C is calculated by rearranging the equation from 3.4, and $a = dist\{W_d, x\}, b = dist\{W_r, W_d\}, c = dist\{W_r, x\}$

These restrictions on the data prevent the winning node moving towards the centre or outside the cluster, thus, the node's movement is retained within the cluster's outer area. Therefore, a winning node d is moved, i.e. updated, when the datapoint is positioned between the two described boundaries, so if x satisfies:

$$(dist\{W_r, x\} >= bound_{rd})\&(C <= 90)$$

the weight vector W_d is updated using x.

Fig. 4 (top right panel) shows an example of the data restrictions for node b(representing the left cluster) where the small circle surrounding the central node, i.e. $bound_{rb}$, and line (x), at a 90 degree angle to r, represents the boundaries node b cannot pass. The area within the circle and behind the line show the datapoints, which are disallowed to update node b, as the node would either move towards the centre or outside the cluster. These boundaries change over time. The boundary $bound_{rb}$ increases so node b is continually pushed further out. The second boundary, the line through node b, also changes with respect to r. This

defines different sets of datapoints positioned behind b, as the node moves around the cluster. For example, if b moved to the right side of the cluster, the second line (y) would represent the new boundary, which defines a new subset of restricted datapoints.

4 Results Generated From Using The Spike Sorting Process On Simulated Data

To test the efficiency of our process, we created simulated datasets containing clusters of varying number, size, shape and position. A high proportion of the datasets were correctly classified, regardless of the cluster configurations. A few of the datasets contained clusters which were specifically configured to test that our process would classify a higher percentage of these datasets correctly, than when other standard clustering methods were used. The results of using the spike sorting process on two of the datasets, containing 2 and 3 clusters respectively, are discussed below.

The spike sorting process was used on these two datasets, where the correct number of central nodes and the corresponding outer node sets O were identified. The number of outputs (nodes) used to find the number and size of the clusters was 10. The number used is arbitrary but must exceed the expected number of clusters. The classification process was then used with all 10, and then the central, nodes. The results of both were then compared, as using the central nodes is the equivalent to using other methods such as K-means, which would prove that our process is more efficient than other clustering techniques.

The process was firstly used on a dataset containing 3 clusters, where all of them varied in size. The classification process classified 100% of the dataset correctly when used with all the nodes, shown in Fig.4 (top left panel). This result was also produced when only using the central nodes. The results show that due to the clusters representing small areas and were not in close proximity, the central nodes produced a implicit boundary that sufficiently separated the clusters.

The next dataset used, shown in Fig.3 (bottom panel, right), contained two clusters which slightly overlap, where one has a wider distribution area than the other. The classification process classified 85% of the dataset correctly, when only the two identified central nodes were used. This result increased to 98% when all 9 nodes were used. The central nodes produced a lower score as a subset of the larger cluster was closer to the incorrect central node, i.e. the implicit boundary (black line) formed between the two nodes, shown in Fig. 3 (bottom right panel), was insufficient at separating the clusters. When all 9 nodes were used, the outer nodes O, which were distributed to sufficiently represent the size and shape of the clusters, aided the classification process by associating the outer areas of the cluster to the central nodes, i.e. the nodes in sets O formed the implicit boundaries, thus optimally separating the clusters. The optimal separation between the two clusters is shown in Fig. 3, (bottom left panel) where the black line (implicit boundary) has been positioned to produce

a better separation, compared to where the black line is, when only two central nodes are used (bottom panel, right).

For the rest of this section we investigate the efficiency of the spike sorting process when the number of outputs (nodes) used is varied. The assumption is that the number of nodes needs to be increased to classify a high proportion of the datasets correctly when they contain either larger clusters, or clusters which are in close proximity. To test this assumption we used the process several times on a number of datasets, increasing the number of nodes each time the process is repeated on a dataset. Firstly, we used a number of datasets that contained the same number of clusters, where the size and position were varied in each set. The spike sorting process was used with the same number of nodes (the number used was 4 times the number of clusters) on each dataset, where we assume that the number of correct classifications is high when the datasets contain similarly sized clusters, but decreases when some of the clusters become larger. This assumption was incorrect; as an average of 98% of datapoints in each dataset were correctly classified, no increase in nodes was required. This was due to the nodes at the beginning of the process being appropriately distributed, i.e. more nodes are associated to larger clusters or equaled out amongst similar sized ones. This aids in reducing misclassification of spikes (datapoints), especially for larger clusters as these have a higher chance of exceeding implicit boundaries when represented by a low number of nodes. Examples of 10 nodes' starting positions are shown in Fig.3 (top left panel), where a higher percentage of nodes represents the larger cluster compared to the smaller one, and in Fig.4 (top left panel) where an equal number of nodes represents each of the clusters, due to their similarity in distribution. Secondly, we used similar datasets as above, but the clusters within them were positioned in close proximity to one another. It was found that the number of nodes had to be increased when the clusters were positioned closer together. This enhanced their separation, and hence increased the number of correct classifications. The results also showed that the more nodes used, the better the separation.

5 Applications To Experimental Data

5.1 Datasets

Electrophysiological data were acquired from two animal systems, the olfactory bulb (OB) of anaesthetized rats, and the inferotemporal cortex of awake behaving sheep. All experimental procedures involving animals were conducted in strict accordance with the Animals (Scientific Procedures) Act, 1986. Rats were anaesthetized (25% urethane, 1500mg/kg) and fixed in a stereotaxic frame. A craniotomy was performed and the left eye enucleated to expose the left OB and allow a 30 channel MEA (6x5 electrodes) to be positioned laterally in the mitral cell layer of the region. Throughout the recordings, humidified air was supplied to the rat via a mask over the nose. Sheep, under halothane (fluothane) anaesthesia, were implanted with two chronic 64-channel MEAs in the right and left inferotemporal cortices respectively. Recordings were made from the unanaesthetised sheep while they performed

an operant discrimination task in which different pairs of sheep faces were presented and a correct panel-press response elicited a food reward. For both preparations, individual electrodes were fabricated from tungsten wires (125μ diam.) sharpened to a $< 1\mu$ tip and insulated with epoxylite. Electrode impedances were 200W. Neuronal activity (spikes) was sampled extracellularly from each electrode.

Spikes were extracted from the continuous electrophysiological trace sampled at each electrode when the signal exceeded a given threshold (2 * background noise). The spikes were collected by sampling the signal from 0.4 preceding to 1.25ms after the threshold was crossed. For each channel of data, the width of the spikes collected was identical throughout the period of recordings.

Each spike collected comprised 48 sample points (t1-t48), the voltage crossing the spike-triggering threshold at $t = 11$. An example of a spike waveform can be seen in Fig.6 (bottom panel).

5.1.1 Noise Extraction

Electrical noise is almost invariably incorporated into electrophysiological data. This often originates from electrical equipment, e.g. from electrical actuators for delivering stimuli. Such signals may exceed the spike-triggering threshold and may resemble spikes in their form and amplitude. As such they are often difficult to exclude from electrophysiological datasets. Here we introduce a process for extracting waveforms that have been caused by noise, i.e. artifacts, from the spike waveform datasets, so that they do not compromise the results of subsequent analyses.

To achieve this extraction we need to have a manually selected set of noise waveforms that are constructed for each channel of data, and then implement the following:

1. Preprocess the spike waveform dataset, and then use the clustering stage to identify the number of cells contained within the feature space.

2. Identify the waveforms sets, which are associated to each cluster and acquire the average spike waveform from each, using the raw data. A set of weights is then obtained, containing the noise waveforms and averaged spike waveforms.

3. Transform the spike waveform dataset and the set of weights, using our pre-processing method. The full set of unmodified *curv* scores is added to each of the waveforms and weight PCA transformations, i.e. they are not averaged or changed to a -1 or 1.

4. Each waveform is then compared to each of the weights to determine to which it most closely conforms. Those conforming most closely to a noise waveform weight are rejected from the data.

An example set of noise waveforms are shown in Fig. 5 (top left panel).

5.2 Application To Recordings From The Rat Olfactory Bulb

The data used in this section are from single electrode recordings of the activity in the mitral cell layer of the OB when no odour stimulus was being presented. The approximate firing rate of mitral cells is known to be 10-50 spikes/sec, and this range was incorporated into our pre-processing stage. The initial number of outputs (nodes) used to find the number and size of the clusters is 30, unless otherwise stated. This number is arbitrary but needs to exceed the maximum expected number of sampled cells. Misclassification of the waveforms is reduced by using a large number of nodes, but using an excessive number of nodes in the process is computationally expensive.

5.2.1 Dataset Of Known Number Of Cells

The first dataset used to test the process contained an already known number of mitral cells. This dataset was formed by combining two sets of single cell data. The objective of using this data was to determine whether the results acquired from the process, corresponded to the information known, i.e. the number of cells, and the number of spikes recorded from each. The number of nodes used in the process was 9 and from our pre-processing method it was found that PCA alone was adequate to form clusters with firing rates in the range of 10-50hz. The PCA method ignored any components contributing less than 0.5% of the variance, and extracted 11 feature components describing the waveforms. Fig. 5 (top right panel) shows two clusters formed using PCA. The nodes were distributed so that they represented as much of each cluster as possible, as can be seen in Fig. 5(top right panel). The resulting classification process was very accurate; 100% of the waveforms were classified correctly. In Fig.6 (top panels left and right), the set of waveforms attributed to each group is shown; the lower panel shows the averaged waveform shape for each group. The process identified the correct number of neurons and associated the correct number of spikes to each.

5.2.2 Dataset Of Unknown Number Of Cells

Here we test the performance of the process when used to analyse data sampled from an unknown number of neurons. Using our method, we were able to extract a number of differently shaped waveform groups, each group representing a neuron with a biologically plausible firing rate. With our pre-processing method the data was first transformed. PCA alone could not generate a set of clusters with acceptable firing rates, forming only a single cluster with an extrapolated firing rate exceeding 200 spikes/sec, an unfeasibly high rate for a mitral cell to sustain. This is shown in Fig.5 (bottom panel) where there is a single cluster represented by all 30 nodes. To resolve this problem our pre-processing method was used further, and an additional 4 curvature components were added to the PCA features. This resulted in six clusters being found, each with an acceptable firing rate (see Fig.7, panels a-f).

From visual inspection of Fig.7(panel g) we can see that the sorted spikes in different groups have distinctive waveforms (curvatures) and may be treated as spikes from different

neurons. For example, the green spike rises much earlier than the red one. However, as we have mentioned before, an application of traditional methods fails to sort spikes because the curvature information is absent. Our novel approach presented here, on the other hand, first maps the input space into a higher dimensional space with additional information on curvatures, and subsequently has a more robust classifying algorithm. All these factors ensure that the noisy spike data are correctly sorted.

5.3 Application to Recordings from Sheep Temporal Cortex

The spike sorting process was similarly successful when applied to spike data acquired from the sheep temporal cortex. An example is shown in Fig.8 (panels a-f) where the activities of six neurons, and their firing rates, were identified.

Again, from (Panel g) we see that geometrically the sorted spikes are clearly distinguishable. When compared to the data from rats, we see that more detailed geometrical structures are picked up by our algorithm.

Having sorted the spikes, the data from both systems (rat olfactory bulb and sheep temporal cortex) were analyzed and compared across experimental variables (not presented here), enabling us to elaborate the information flow in the recorded area. For example, which cell fires first in response to stimulus presentation (an odour, a face or an object); what stimulus properties elicit the response; is a response modified by behavioural conditions; does the right hemisphere or the left hemisphere respond to the stimulus first; what are the response latencies? For details, we refer the reader to our further publications (Nicol et al., 2005; Tate et al., 2005).

6 Discussion

A comprehensive understanding of the principles employed by the brain in processing information requires sampling the activity of many individual neurons in neural systems. We have presented here a novel approach combining a number of machine learning techniques to identify the number of cells recorded by an electrode and the corresponding firing rates as accurately as possible. The pre-processing technique presented outperforms others that use PCA alone. This method was able to extract separable clusters within the feature space, whereas PCA could not. We have also shown that by extending a common unsupervised neural network we can find the actual number of clusters within the feature space, and create an implicit boundary, which surrounds any cluster, regardless of shape and size. This produces an optimal technique for separating the clusters, thereby reducing the number misclassified spike waveforms.

The process of extracting the number and size of clusters is largely automated, requiring minimal user input. This is a vital consideration when recordings are made simultaneously across multielectrode arrays. The process is also computationally efficient. By using a minimum number of variables to describe each waveform, and requiring the neural network

to find the number of neurons in a subset of the data, the memory needed to process the data is minimised. Thus the speed of the spike sorting process is optimised. These techniques are equally applicable can be applied to many other types of data where clustering of multiple variables is important, including microarray data and functional brain imaging.

In the current paper we have successfully developed the critical technique of effectively sorting the activities of individual neurons from multiple neuron spike trains. The techniques presented here have lead to interesting and important findings, as reported elsewhere (Nicol et al., 2005; Tate et al., 2005; Christen et al, submitted). Our data allows us to look at the combined activity of many individual neurons, revealing much more of the information processing functions of neuronal networks than can be achieved by studying the activities of individual neurons.

7 Acknowledgement

J.F. was partially supported by grants from UK EPSRC (GR/R54569), (GR/S20574), and (GR/S30443).

8 References

Bishop C. Neural Networks For Pattern Recognition. Clarendon Press 1995.

Christen M, Nicol AU, Kendrick KM, Ott T, Stoop R. Stabilization not synchronization: stable neuron clusters signal odour presentation. (submitted to Neuron)

Csicsvari J, Hirase H, Czurko A, Buzsaki G. Reliability and state dependence of pyramidal cell-interneuron synapses in the hippocampus: an ensemble approach in the behaving rat.Neuron 1998;21:179189.

Feng, JF. Computational Neuroscience: A Comprehensive Approach Chapman and Hall/CRC Press, 2004.

Harris K. Klustakwik Software. ($http://klustakwik.sourceforge.net/$).2002

Hazab L. Kluster Software (G. Buzsaki's Lab). ($http://klusters.sourceforge.net/UserManual/index.html$).2004

Jolliffe IT. Principal Component Analysis. Springer-Verlag New York Inc 2002.

Kohonen T, Self-Organizing and Associative Memory, New York: Springer-Verlag 1984.

Kohonen T. Self-Organizing Maps. Springer-Verlag Berlin and Heidelberg GmbH and Co.K 2001.

Lewickiy M. A review of methods for spike sorting: the detection and classification of neural action potentials.Network: Comput. Neural Syst 1998;R53R78

Lipa P. BubbleClust Software. ($http://www.neuralynx.com/$). 2004

Nicol A, Magnusson M, Segonds-Pichon A, Tate A, Feng J, Kendrick K. Local and global encoding of odor stimuli by olfactory bulb neural networks. Annual Meeting of Neuroscience (Oral presentation) 2005.

Redishi A. MClust Software. ($http://www.cbc.umn.edu/\ redish/mclust$).2005
Tate A, Nicol A, Fischer H, Segonds-Pichon A, Feng J, Magnusson M, Kendrick K. Lateralised local and global encoding of face stimuli by neural networks in the temporal cortex. Annual Meeting of Neuroscience(Oral presentation) 2005.

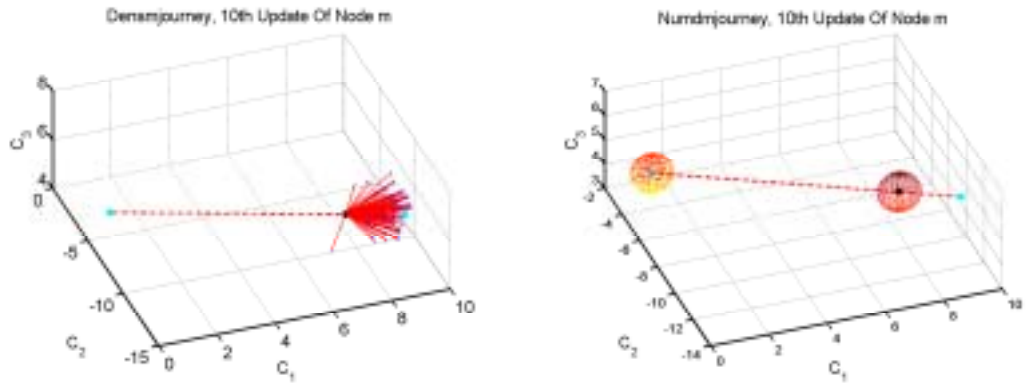

Figure 2: This shows the 10th update of node m (black marker), where the red striped line is the total journey m will make between two central nodes from D, sn (right blue marker) to dn (left blue marker). All of the datapoints for each cluster were removed, so the movement of node m can be clearly seen. **Left:** This shows the closest 100 datapoints to node m used to calculate $dens_m$. **Right:** This shows the number of datapoints within the spherical boundary surrounding m, which derives the score $numd_m$. The sphere surrounding the light blue marker, represents the boundary around dn.

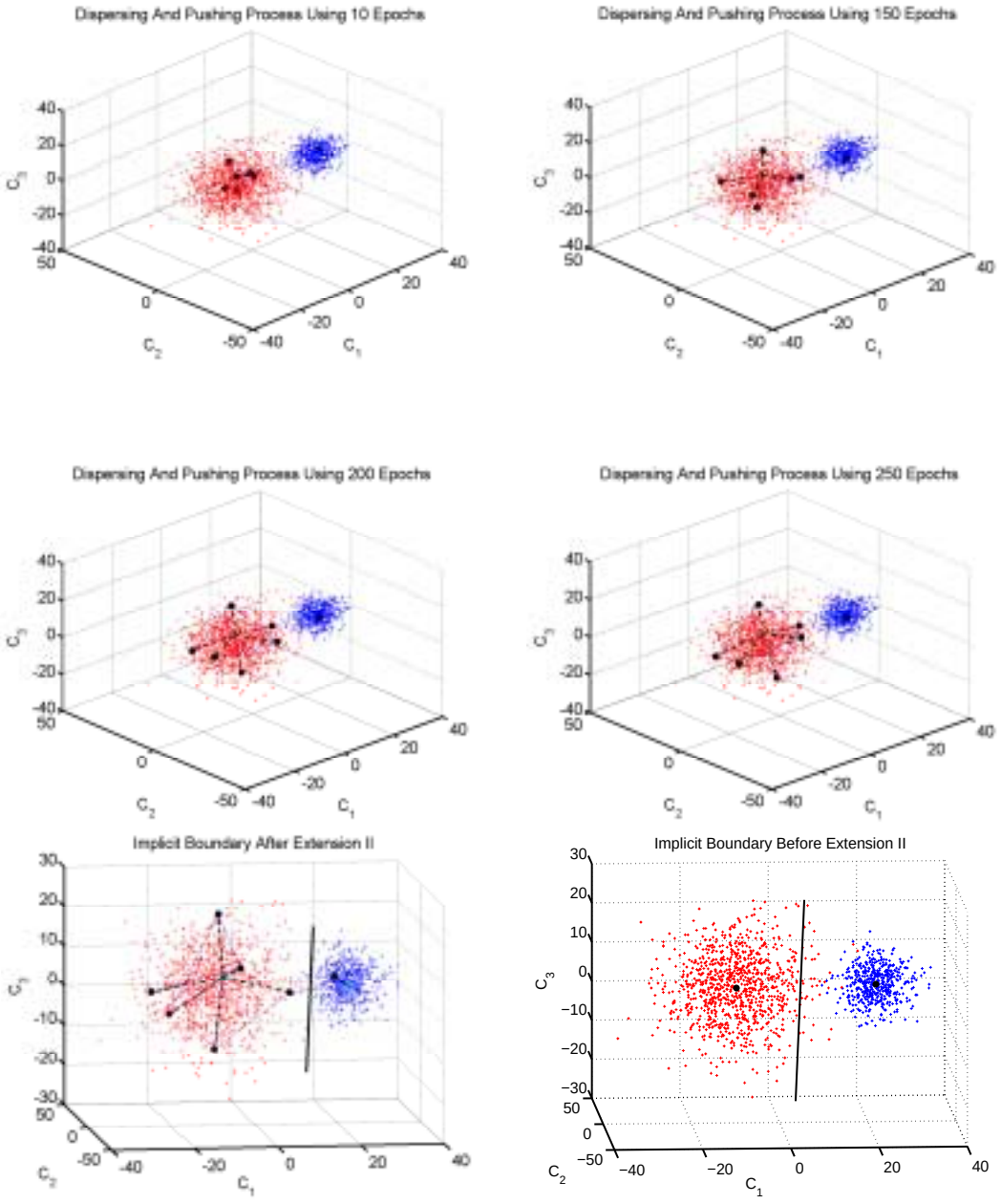

Figure 3: **Top two panels, From top right to bottom left:** These show the results of the dispersing stage over a number of epochs. It shows that the neighbours from each cluster have been progressively pushed out as the number of epochs used increases, thus representing more of the cluster. **Bottom panel, Left:** This shows the end result of the process, where the implicit boundary (black line) has been moved further to provide a more optimal separation. This increases the number of waveforms that are correctly classified, compared to only using stage 1 of the clustering process shown **Bottom Panel, Right**.

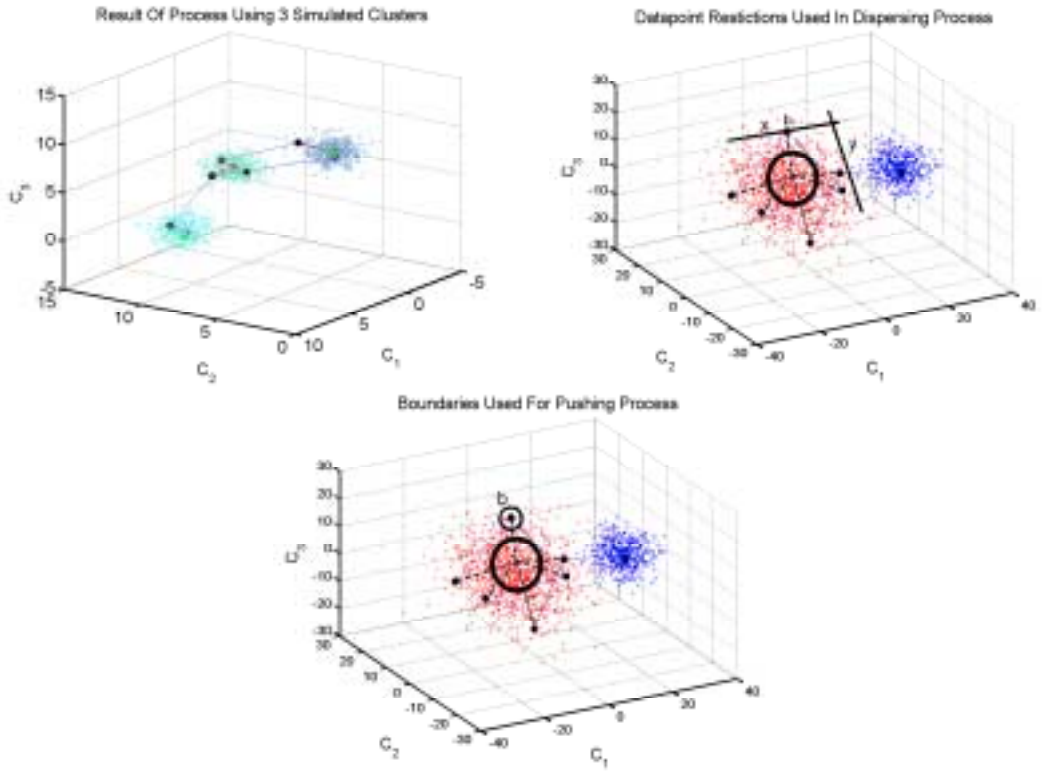

Figure 4: **Top Left:**, This shows the end result of the process when 10 nodes were used. It shows that all of the nodes have been used to represent either the centre, or the outside area, of the clusters. **Top Right:** This shows which datapoints node b (from set O) can move towards, i.e. between the boundaries, line (x) and the circle surrounding the corresponding central node ($bound_{rb}$). Line (y) represents another boundary if node b moved to that side of the cluster. **Bottom:** This shows the boundaries used in the pushing process for node b and the central node. If they overlap node b would be pushed out.

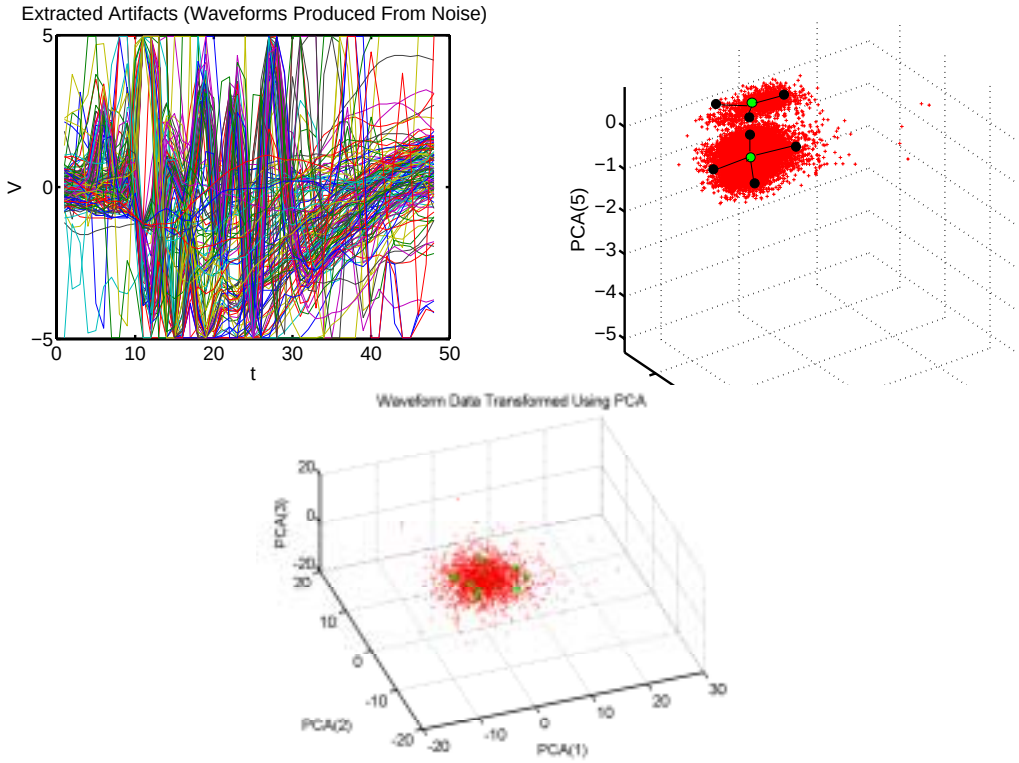

Figure 5: **Top Left:** An example set of artifacts extracted from the rat olfactory bulb dataset. **Top Right:** This shows the results of using PCA for the pre-processing stage, where two clusters were formed. 9 nodes were used in the clustering process, where the middle nodes (green) represent the set R, and the nodes on the outside represent the associated sets O. **Bottom:** This shows the transformation of the spike waveform data using PCA. It is possible to see that only one cluster was formed.

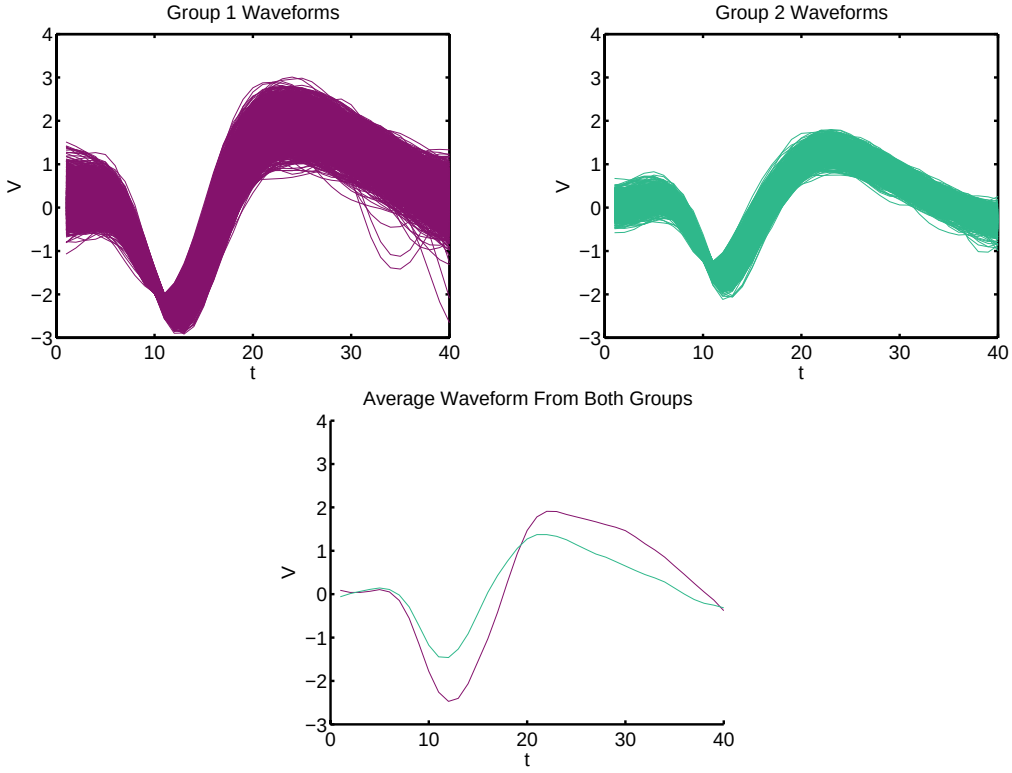

Figure 6: **Top Left and Right:** These show the spike waveforms attributed to the cluster groups 1 and 2 respectively. **Bottom:** This shows the average waveform from both groups

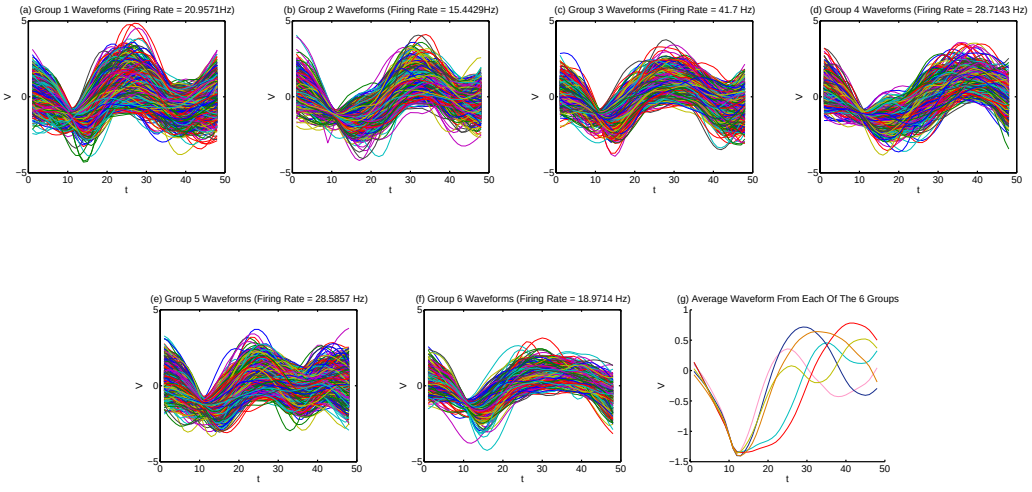

Figure 7: **Panel a-f:** In each panel, spikes generated by a single neuron are overlayed. **Panel g:** shows the average waveform of spikes for each neuron.

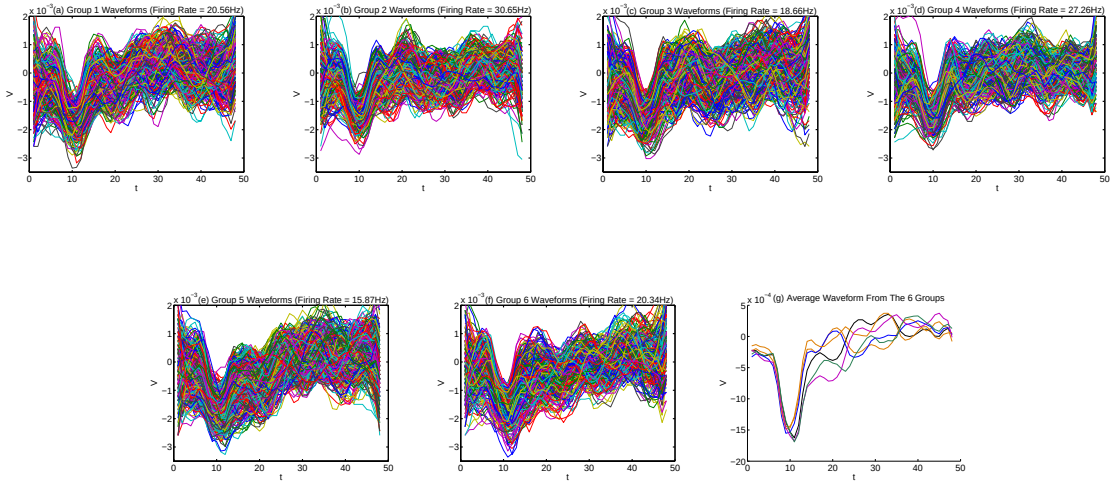

Figure 8: **Panel a-f:** In each panel, spikes generated by a single neuron are overlayed. **Panel g:** shows the average waveform of spikes for each neuron.