The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

Neuron type classification based on extracellular signals – project report

Project number: 18-1-1-1695

Authors:

- Michal Altmark
- David Gertskin

Supervised by:

- Dr. Eran Stark – Sackler faculty of medicine, TAU university

Table of Content

Abstract

Brain function depends on the activity and interaction of different neurons. In this project, we created a classifier that receives as input the spikes of one unit, and determines the distinct cell type of that unit. The classification is based on extracellular data from the brain.

The analysis is carried in an unsupervised setup, and in a supervised setup. The supervised classification is possible due the ability to identify cells of a specific type in a freely-moving mice based on optogenetic manipulations.

This project is innovative because it allows determining, with high confidence, the class (cell type) of a given unit recorded extracellular even in the lack of optogenetic manipulations in the recorded animals. This allows generating a "lookup table", or dictionary, for cell type classification only from extracellular data. Furthermore, it allows similar research on other mammals in which optogenetic manipulations are impossible. This research eventually will help to better understand functions and interactions in the brain, furthering our understanding of cognitive functions and brain diseases such as Epilepsy or Parkinson that stem from abnormalities in neuronal dynamics.

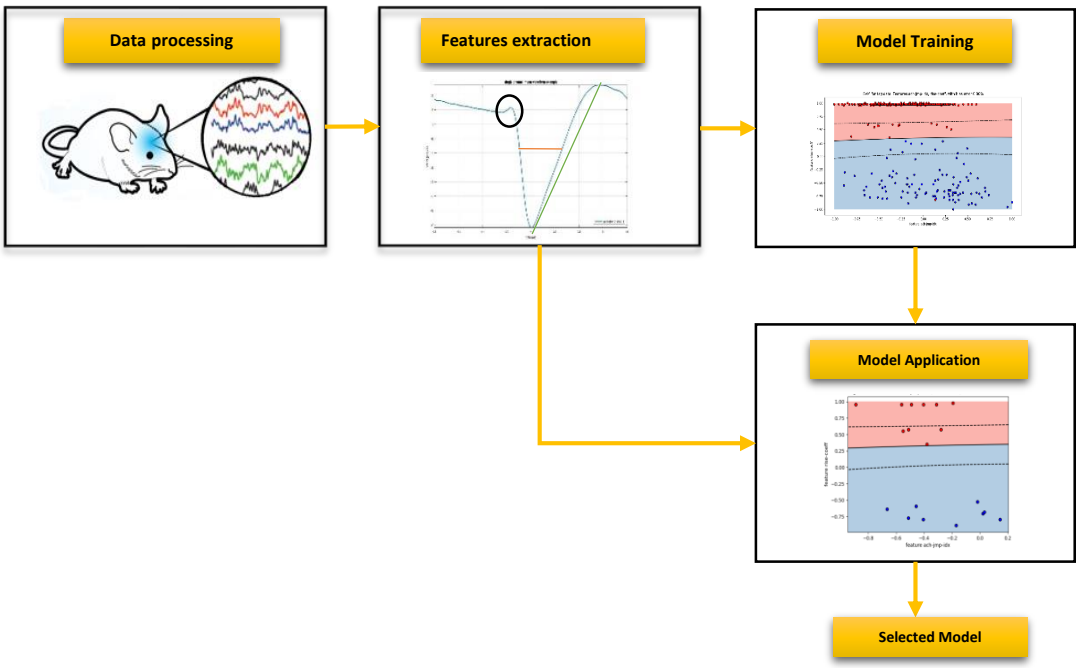

Figure 1: The project layout.

Experimental Procedure

Raw data acquisition

The data for this project have been gathered in experiments in which extracellular voltages were measured from multiple sites in the brain of freely-moving mice. Recordings were done using a multi-shank silicon electrode array, which is a device that can measure voltage in a varying number of channels [ref - Tools for Probing Local Circuits: Buzsaki at el, 2006, NEURON]. This array is placed inside a mouse brain, and then connected to a signal conditioning system (amplification and filtering) and a data acquisition system (DAQ).

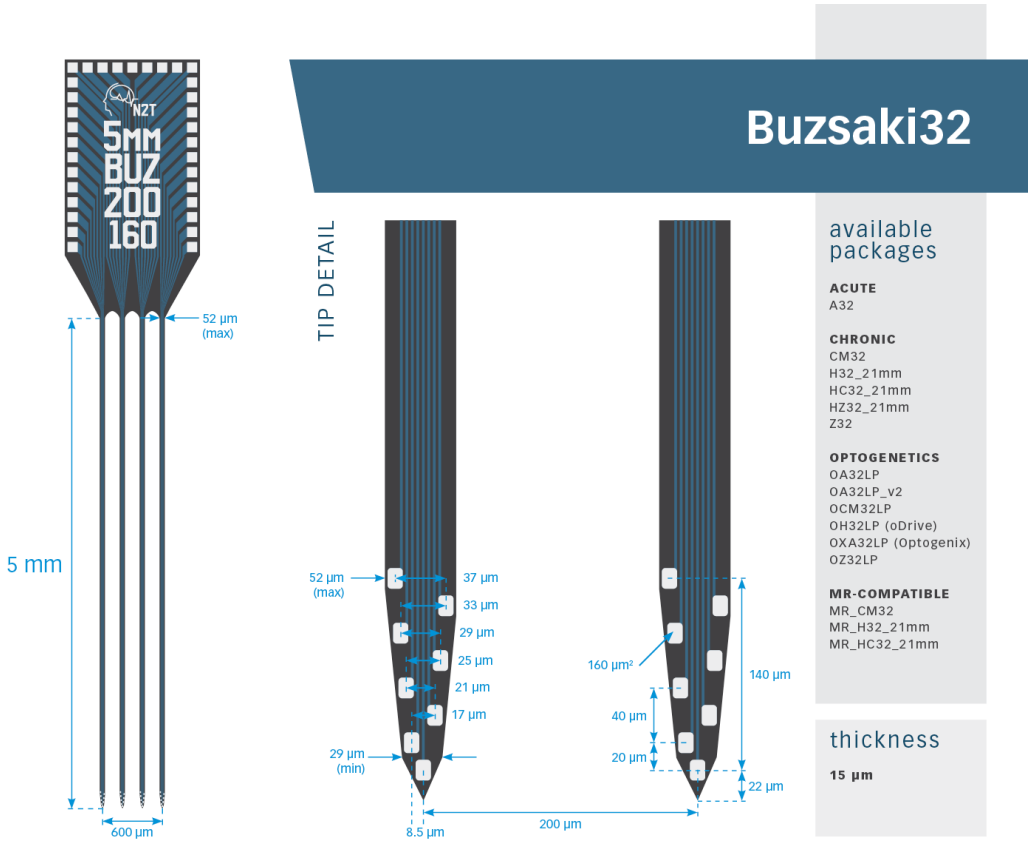

Figure 2: A four-shank/32-channel probe.

In **Figure 2** we show the layout of the probes used to record the extracellular signals from the brain. One can see on the left of the figure that the probe consists of 4 shanks, separated one from another by $200\ \mu m$. Each shank has eight measuring sites (electrodes) placed on its tip. Those sites, placed, on the shank tip allow the measurement to take into account spatial information.

Using these devices, the extracellular voltages was measured in a spatially distributed manner, whereby different channels correspond to the different locations of the voltage measuring points.

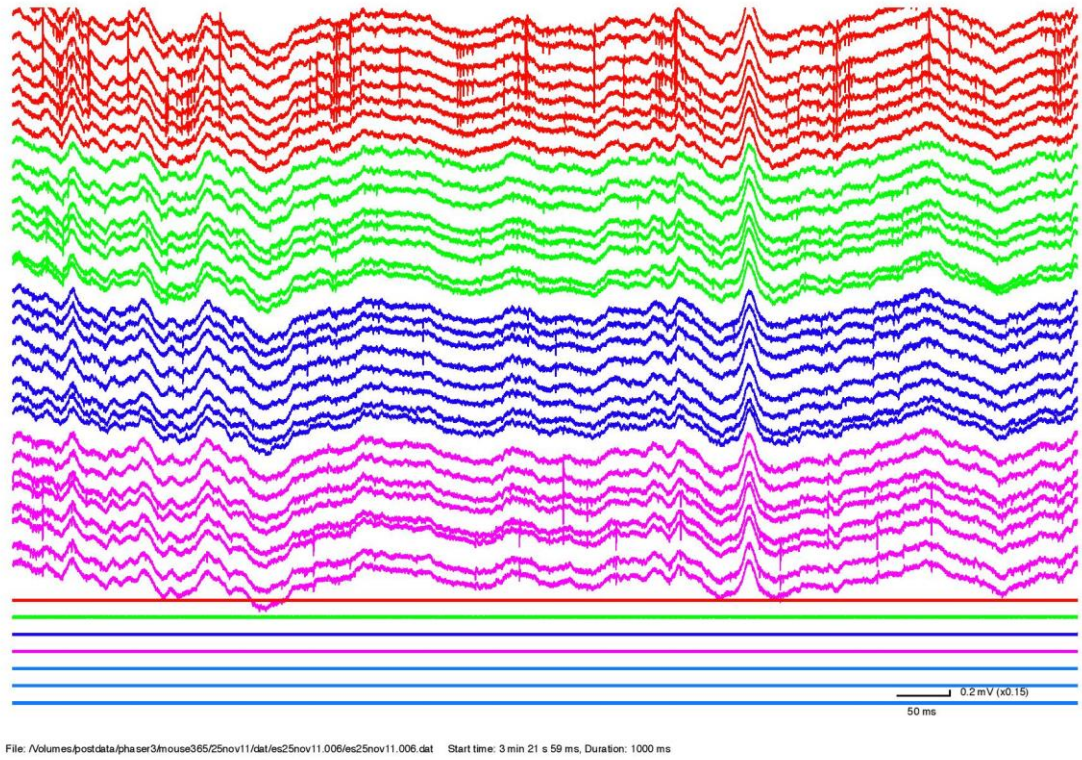

Figure 3: Example multi-channel extracellular recordings from the hippocampal CA1 pyramidal layer of a freely-moving mouse. Data recorded using a 4-shank/32-channel probe (Figure 1). Data from each shank (8 channels) have the same color. Individual spikes are seen as sharp downwards deflections.

In **Figure 3** we present a sample of the raw data acquired in the hippocampus of a freely-moving mouse using multiple shanks. Each color in the figure corresponds to a different shank. Each shank includes eight channels of voltage measurement (recording sites).

Spike extraction

From the raw data using a semi-automatic mechanism, spike times and waveforms have been extracted and sorted. This process is carried out shank-wise, and proceeds in two steps. The first step, "spike detection", is completely automatic and results in a set of 32 time-samples of voltage for each channel in the shank. The sampling rate is 20 kHz. Thus, for an eight-site shank, each spike in the raw data is represented by 8*32 voltage values (32 per channel; stored in the *.spk.* file) and a single time value (the temporal occurrence of the 16^{th} voltage sample; stored in the *.res.* file).

The second step is "spike sorting", and consists of automatic and manual steps. At the end of these steps, the spikes detected in the first step are "sorted": each multi-channel spike is assigned a single integer representing a unit (stored as an integer in the *.clu.* file). Thus, multiple spikes assigned the same clu number are treated as if fired by the same putative neuron.

For a shank in which N spikes have been detected, the spike detection and sorting scheme results in three files [REFERENCE – HAZAN ET AL., 2006, JOURNAL OF NEUROSCIENCE METHODS].

- *.res.# – this is a list of times (measured in samples), yielding a vector of length N. For each spike, the corresponding entry is the number of counts (samples, each 1/20 kHz, i.e. 50 micro-seconds) that elapsed since the start of the measurement

until the 16th sample in the spike waveform. In order to convert samples to time in seconds, each entry should be divided by the sampling rate.

- *.clu.# – This is a list of length N+1. The first entry holds the number of clusters the spikes are divided to. The N other entries are the clu numbers assigned to each of the N spikes.
- *.spk.# – A list of length N * 32 * 8. This is a binary file which is organized continuously: sample1 of channel1 of spike1, then sample1 of channel2 of spike1, and so on; then sample2 (of all channels of spike1); and so on until sample32 of the last channel of spike1; then the rest of the spikes in the same manner. Thus, for each spike, there are a total of 32*8 samples of voltage (32 samples for each of the eight channels in the shank). These numbers are stored as signed 16 bit integers (int16).

Note that clu values from different shanks are unrelated, since they are spatially distant (a single cell can be recorded by an electrode up to 50 micro-meters away, whereas the probe shanks are 200 micro-meters apart; **Figure 1**). Therefore, the underlying assumption is that each clu corresponds to a distinct spiking source (neuronal cell).

Optogenetic Tagging

The experiments that yielded the data were carried out with genetically-modified fixed mice [REF – STARK ET AL. 2013, NEURON], in which the recording apparatus was a modified type of silicon probe called "diode probes" [REF – STARK ET AL., 2012, JOURNAL OF NEUROPHYSIOLOGY]. The diode-probe were used in genetically-modified mice in which an exogenous, blue-light sensitive cation channel protein (ChR2; REF – BOYDEN ET AL., 2005, NATURE NEUROSCIENCE) was expressed in a specific type of cells – fast spiking parvalbumin-immunoreactive (PV) cells. Thus, when blue light generated by an LED was routed via a fiber to the tip of a single shank of the diode-probe, PV cells were activated at the local vicinity of the optic fiber.

In order to tag each cluster (clu), the spiking statistics during the stimulus were assessed. For instance, for clu recorded in PV-mice, if the result was enhanced spiking during localized illumination (in a manner that exceeds a chance level, defined as a probability of $P<0.001$, Poisson test), then the clu was tagged as an optically activated PV cell, denoted as "act" below.

The possible tags are:

- exc – means that the clu group is tagged as excitatory (e.g. pyramidal cell).
- inh – means that the clu group is tagged as inhibitory (i.e. PV or another inhibitory interneuron).
- act – means that the clu group is tagged as lightly activated (i.e. PV).
- null – no tagging

Those tags are organized in the sPV file described in the next (Data Structure) section.

We expect the "exc" units to be distinctly different from the "inh" units. Indeed, zero units were tagged as both "exc" and "inh". PV cells are considered inhibitory cells, and the "inh" and "act" tags may overlap. However, PV cells can also induce spikes in other cells [REF – STARK ET AL., 2013, NEURON] and thus "exc" and "act" may overlap; however, this is expected to be a rare occurrence. For that we symbolize:

- red – units that were tagged as exc.
- blue – units that were tagged as inh or act.

Given this dataset, our first goal in this project was to extract features from the .res and .spk files, such that we could automatically classify an untagged unit (clu) as either red or blue with low error. Our second goal was to divide the tagged units (those colored “red” and “blue”) into separate, more refined clusters.

Out of a total of 1063 units in the available database (comprised of 48 recording sessions), the number of tagged units goes as follows:

- inh – 21 (1.9% of all the units)
- act – 106 (9.9% of all the units)
- exc – 424 (39.8% of all the units)
- red – 424 (39.8% of all the units)
- blue – 114 (10.7% of all the units)
- untagged – 529 (49.7% of all the units)

Therefore, about half of the units were tagged and half untagged. Note that 13 units were tagged as both “inh” and “act”, and four units were tagged as both “exc” and “act”.

Data Structure

As noted above, the data were collected on different sessions of measurement (either distinct days from the same animal, or from different animals). Each session had a distinct name. Using the session name it was possible to access the corresponding .res, .spk and .clu files for each shank. A second part of the data was the sPV data structure mentioned above. This is basically a summary of the tagging process for 1063 of the units. Our work in this project focused solely on these 1063 units.

This structure holds the following fields:

- filename – that distinct session name for each unit.
- shankclu– the shank and clu values for each unit.
- exc, inh, act – the tags (described above) for each of the units.

This structure lists only units that exhibited satisfactory isolation from other units in the parameter space in which those where separated [REF - STARK ET AL., 2013, NEURON]

In order to obtain the tags, the stimulated times were considered. In our work we wanted to access the normal voltage (waveform) and timing behavior of the spikes. Indeed, the periods in which light was applied have been marked so that they could be treated separately. Here, we excluded these periods. Those periods are found in .stm files, unique for each session.

We extended the sPV data structure by adding two fields, each having one entry per unit. These are (1) the single channel mean waveform and (2) the ACH of each unit.

Mean waveform calculation

The mean waveform is an attribute of each unit. It was calculated for each entry in the sPV file. Using the session name and the shank and clu fields, the corresponding .spk and .clu files have been fetched.

The next step was to obtain the indices of the valid spikes to average over. A valid spike is a spike (1) whose time of occurrence is not during a stimulus and (2) its clu value is the same as the clu of the entry in the sPV file. The .spk entries corresponding to those indices were upsampled by a factor of 8 and averaged. This process resulted in a 2D matrix with 256 by 8 elements (256: 32 samples, upsampled by 8; 8: 8 recording sites/shank). The upsampling is done using the FFT method.

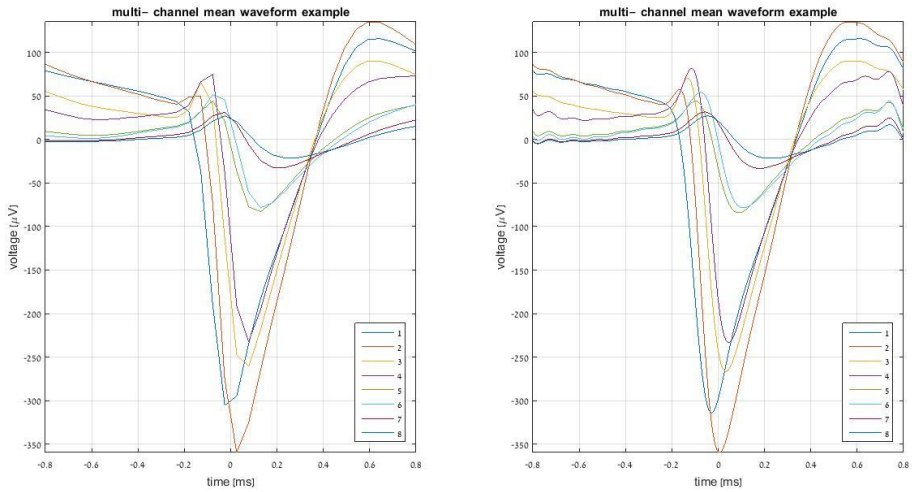

Figure 4: example of the upsampling process on the multichannel voltage

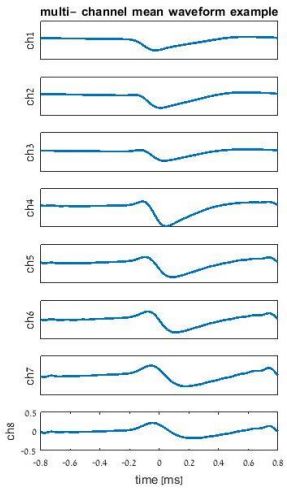

Figure 5: the same plot when we take into account the physical arrangement of the probe. Vertical axis units are μV and horizontal is ms. Spike description: Filename – 'es25nov11_3', shank – 1, clu–7.

From the upsampled multichannel mean waveforms, we extracted the single most informative waveform using the heuristic decision that it would be the channel that exhibits the highest trough-to-peak value. For example, for the clu group in **Figure 3** the mean would be:

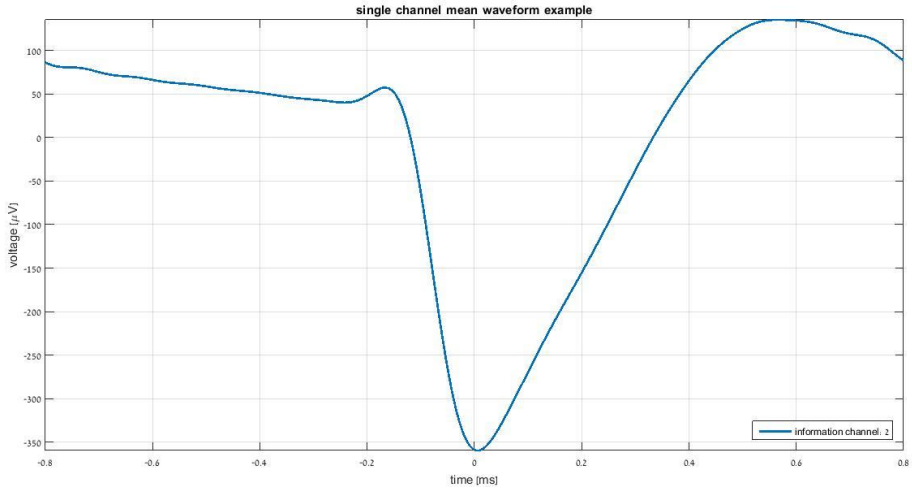

***Figure 5**: the single channel mean extracted from the multichannel mean in **Figure 3**. Spike description: Filename – 'es25nov11_3', shank – 1, clu–7. Using the described heuristic the chosen channel for this spike is 2*

This heuristic is a design choice and other methods may be considered. For instance, we can perform an SNR (signal to noise ratio) assessment for each channel, and choose the channel with the largest SNR. This assessment can be held by defining SNR=mean/variance, and calculating the sample estimate for the variance over the spk entries that were used in the mean waveform calculation. Another way could have been to estimate the center of mass of the 8 mean waveforms, then take the channel closest to this center of mass.

Autocorrelation (ACH) calculation

The autocorrelation of a random process X is defined is as follows:

$$\mathrm{R}_{XX}(\tau) = \mathbb{E}[X(t) \cdot X(t-\tau)]$$

In order to estimate the ACH of the spike train of a single unit, which is a point process rather than a continuous time process, we created a histogram in which each bin counts the number of occurrences of a pair of spikes separated by a specific time difference (with or without another spike between them). For example, for one-millisecond bins, if a spiking event happened at $t_1 = 5\ ms$ and also at $t_2 = 6\ ms$, then the $\tau = 1\ ms$ bin is updated by one. From its definition, the ACH is an even function.

In our dataset, the input to the ACH calculation scheme was the .res file at the valid indices (as described for the mean calculation – those pointing at a single unit in the lack of illumination).

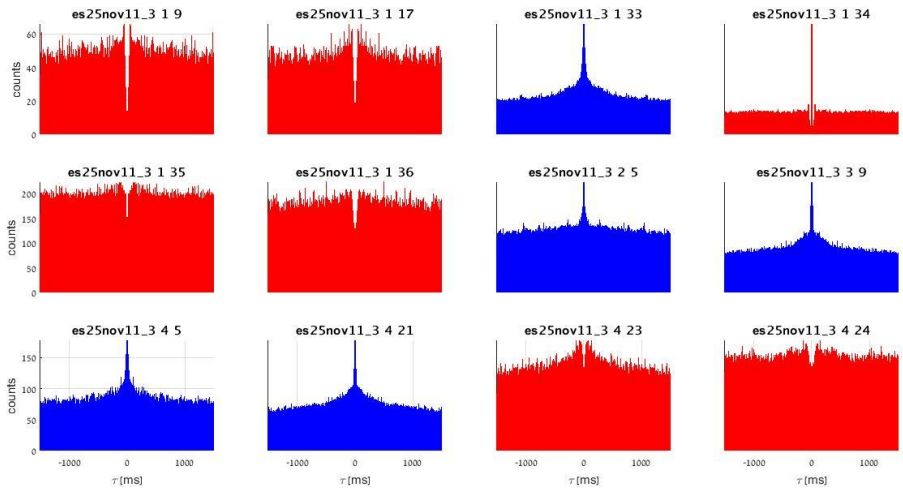

***Figure 6**: Autocorrelation examples. Red means that the tag of the unit is "exc" (excitatory), and blue means that the unit was tagged as inhibitory ("inh") or lightly activated ("act"). The title holds the ACH's filename, shank, and clu values.*

An important feature of the ACH, that stems from the refractory period after the spike is fired, is the zero band around $\tau = 0$. This can be seen by observing the close range ACH plots the units shown in **Figure 6** (see **Figure 7** below).

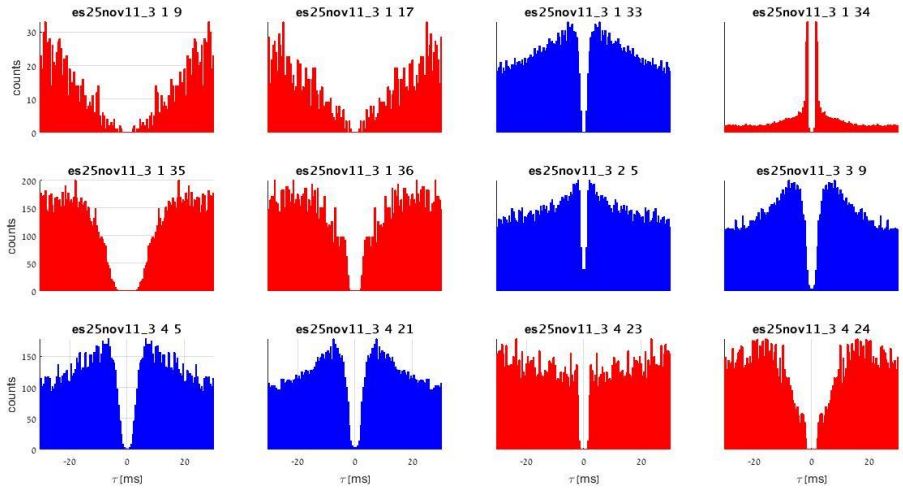

***Figure 7**: Autocorrelation in short range examples. Color code and units are the same as in **Figure 6**. All* ACHs exhibit a zero band around zero.

If we examine a few ACHs more closely (**Figure 8**), we can appreciate some qualitative differences beyond the zero-band stemming from the refractory period.

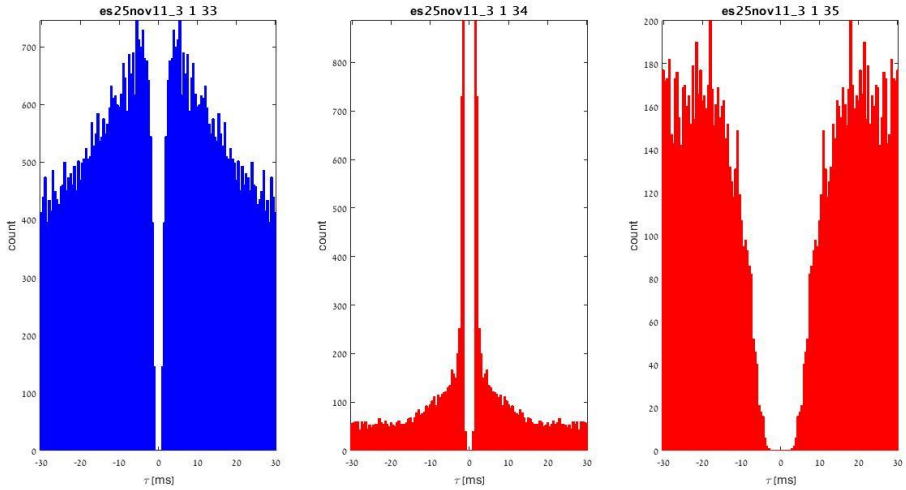

***Figure 8**: autocorrelation in short range examples.*

In **Figure 8**, the right most ACH corresponds to a unit that fires, recharges, and then fires again without a particular structure to its spike times. In contrast, the middle curve has a large peak after it refractory period, which corresponds to a large probability for the cell to fire again right after the first spike. This is a cell that fires pairs (or perhaps triplets, see the second, smaller peak around 6 ms) of spikes. The blue unit fires at in a uniform manner in time.

Feature calculation

From the mean waveform, **w**, and the autocorrelation histogram, **ACH**, we derived two types of features: morphological and temporal.

Morphological Features

The morphological features we have derived are:

- fwhm – the full width half maximum.
- trouph2peak – the distance in ms between the trough and the peak.
- Peak2peak – amplitude between the min to max of the spike.
- rise coeff – a parameter that quantifies the position of the peak of the spike.
- get acc – Early post-tough second derivative measure
- break measure – Pre-trough second derivative measure. This tries to quantify the "bump" that is sometimes observed before the trough (see **Figure 9**, unit 1.7 at the top left) .
- max speed – Post-tough first derivative measure
- smile or cry – Far post-tough second derivative measure

These features are based on the mean waveform computed for each unit (putatively, a well-isolated neuron).

First, we visualized the mean spike waveforms (see **Figure 9** for a sample of 12 units), to see recurrent attributes.

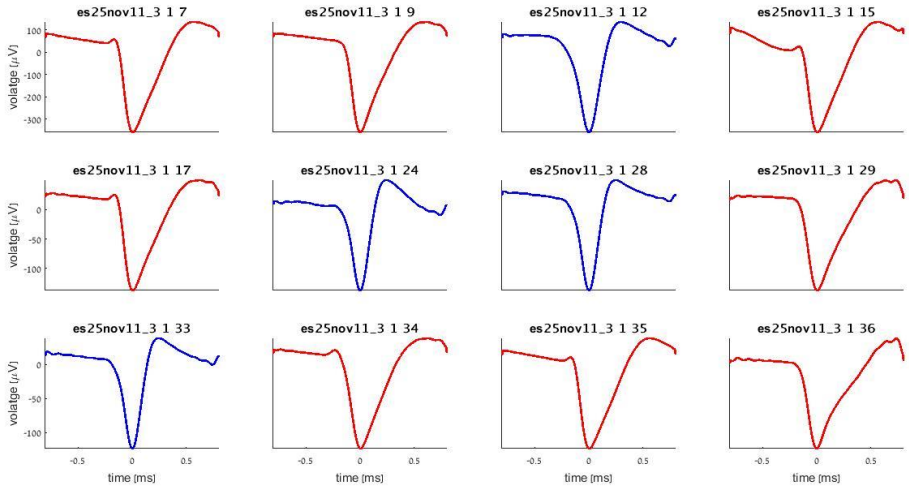

***Figure 9**: Mean waveforms of 12 example units (not the same units as in **Figure 6**).*

Peak2peak

A trivial feature which equals the difference between the minimum and maximum of the mean waveform.

FWHM

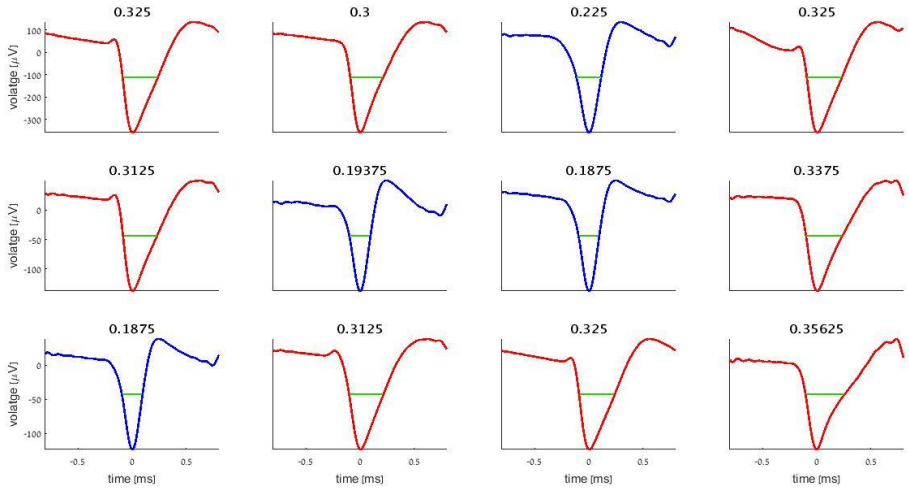

***Figure 10**: FWHM demonstration. Example units and color code are the same as in **Figure 9**. The green lines indicate where the FWHM is with respect to the mean. The titles are the FWHM values in ms.*

FWHM stands for full width at half maximum. In our dataset, it was derived by calculating the trough-to-peak voltage value, and dividing it by 2. The FWHM is the duration in which the signal is below this threshold.

Trough2Peak

This feature is the time it takes for the signal to get from its minimum, the trough, to the first subsequent peak.

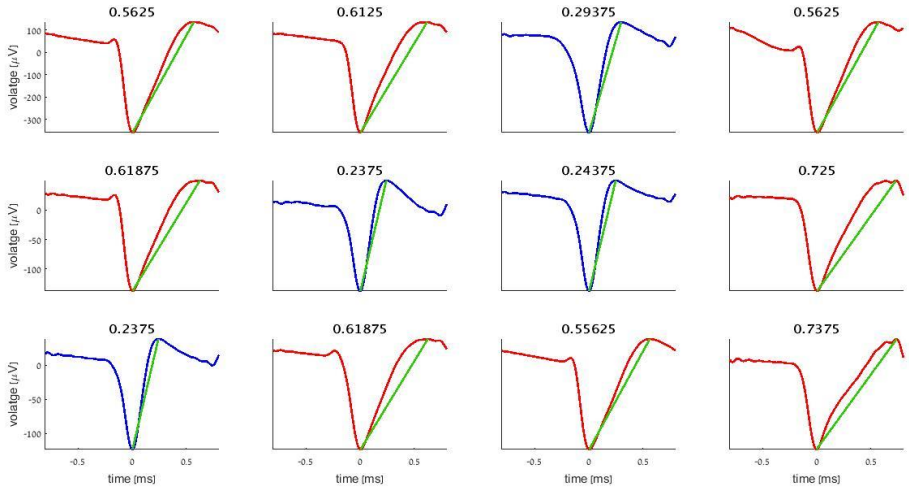

Figure 11: *trough2peak demo. The green line starts from the trough and goes to the subsequent peak. The titles are the trough-2-peak values in ms. The spikes are the same as in* ***Figure 9****.*

Rise coeff

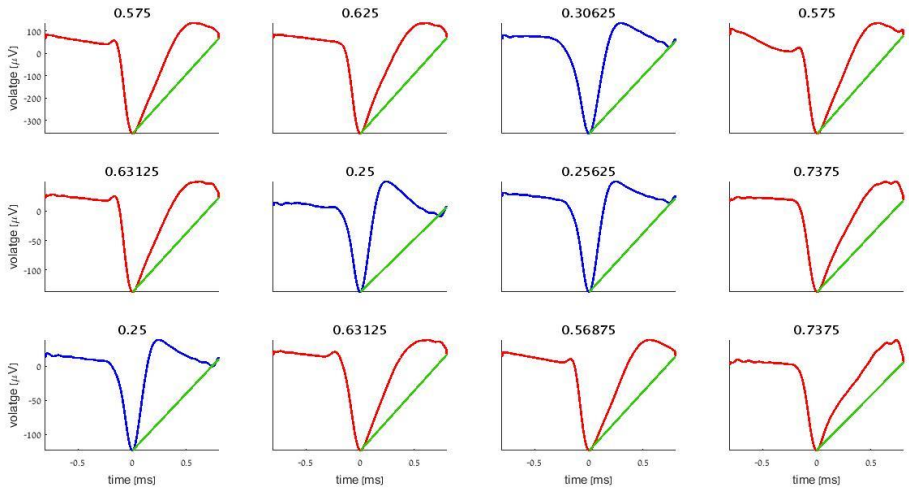

Figure 12*: Rise Coeff demo. The green line passes from the global minimum to the end of signal. The titles are the value of the feature. The spikes are the same as in* ***Figure 9****.*

The green line is just a line passed from the global minimum to the end of signal (sample 256). This line acts as a reference, which helps us find the feature. By subtracting the green line from the spike mean we obtain a 128-sample vector, visualized as the following image:

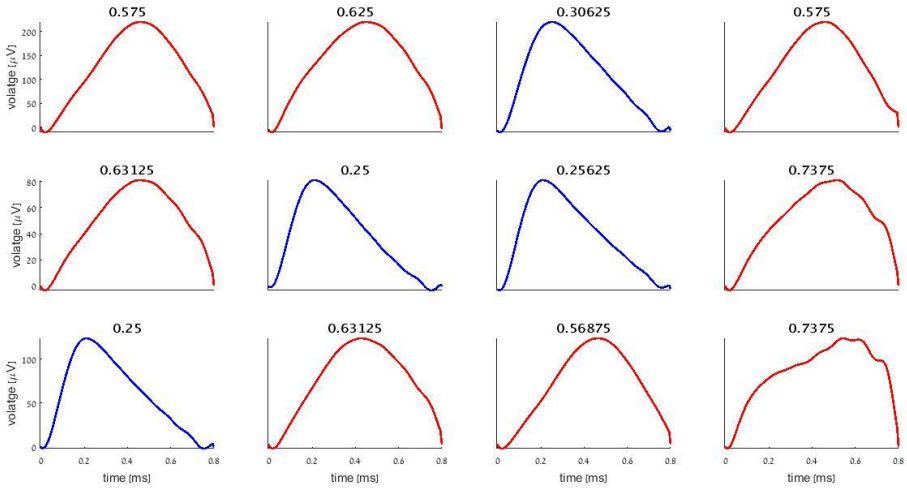

***Figure 13**: the result of subtracting the spike waveform from the green line from **Figure 12**.*

It is clear that the blue (inhibitory) spikes exhibit maxima in a lower index. To quantify this attribute, we took the index of the maximum as the feature.

Retrospectively, after examining all the features, it became clear that this feature is quite similar to the trough2peak, and those features probably correlate. This observation is quantified below, in the "Morphological features redundancy" section.

Break Measure

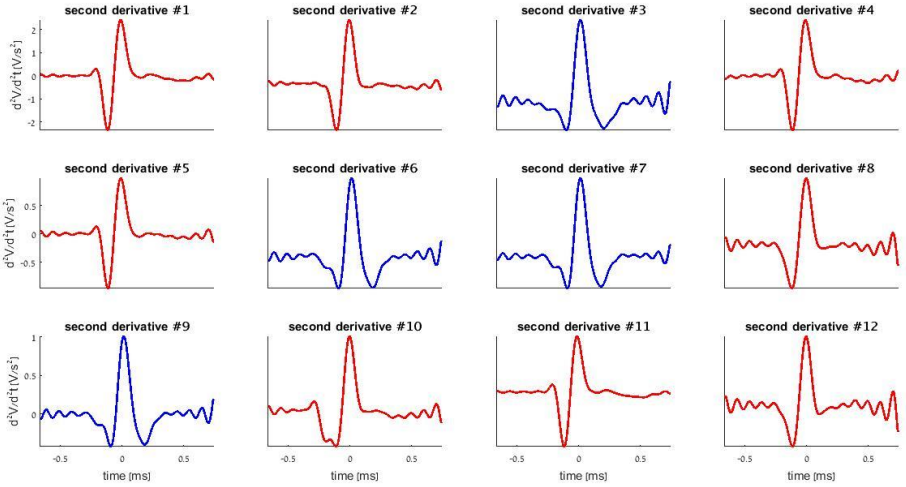

***Figure 14**: second temporal derivatives of example spikes. The spikes are the same as in **Figure 9**.*

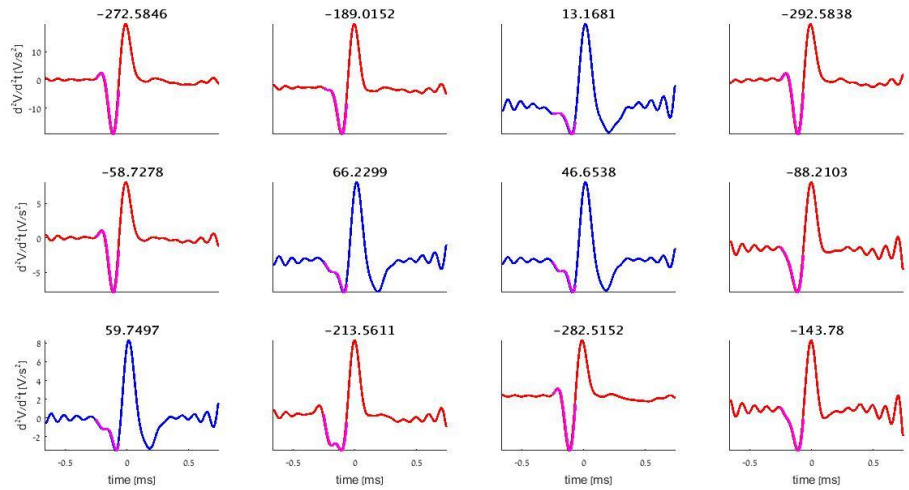

***Figure 15**: The second temporal derivatives (same as in **Figure 17**). Region of interest for the "Break Measure" is shown in purple.*

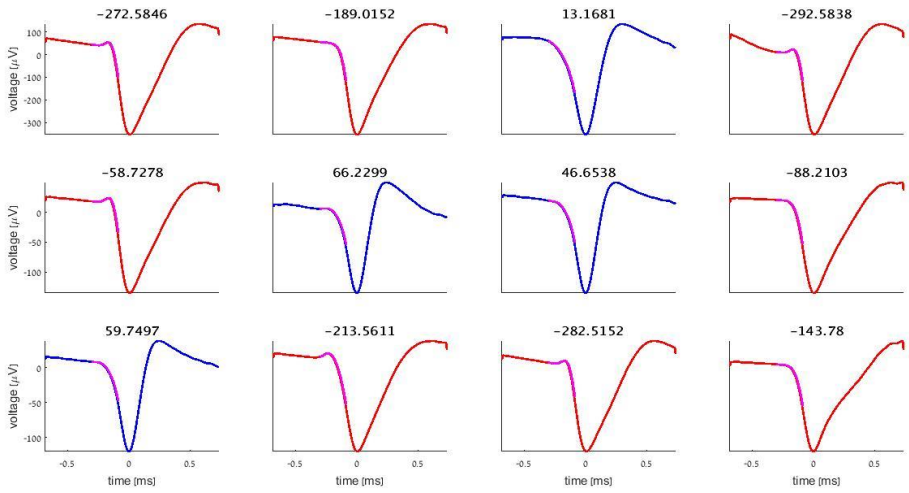

***Figure 16**: Spike waveforms, with the the region of interest (purple) for the "Break Measure". The spikes are the same as in **Figure 9**.*

The next feature was designed to quantify whether there is a "bump" in the waveform, just before the trough. This may correspond to specific ionic currents (e.g. number of gates in the sodium channels and/or additional potassium currents) which may be characteristic of certain neuronal types. Indeed, it is clear (**Figure 19**) that the blue-colored (inhibitory PV cells) waveforms signals do not exhibit this bump, whereas some of the red (excitatory) spikes do. Therefore, this feature can be useful in order to distinguish between different types of excitatory neurons.

The feature extracted here is:

$$\chi = 20\log(\exp(\sum_{\omega \in roi} y''(\omega)) - 1 + \varepsilon) \ ,$$

where ε is as small heuristic number and y'' is the second temporal derivative of the signal. The region of interest (ROI) is the time period before the trough as marked by purple (0.085 ms to 0.3 ms before the trough). We chose this segment to include the "bump" before the trough in a way that will take into account the difference between the signals, i.e. we chose

the purple line so the ROI will result with distinctly different values for different types of signals. In the calculation of this feature the waveform was normalized by dividing it with it sum of squares, $\vec{y} = \frac{\vec{w}}{\|\vec{w}\|^2}$

Get Acc

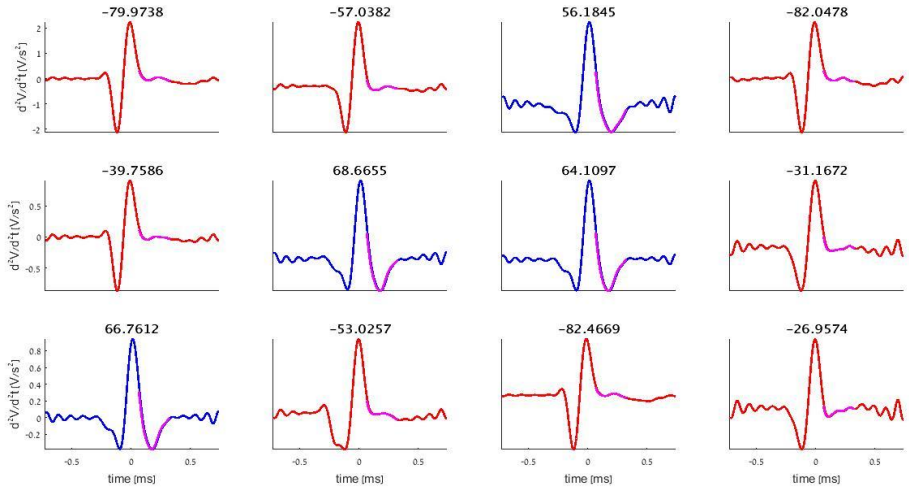

Figure 17 *: the temporal second derivative and the region of interest for the Get Acc feature. Spikes are the same as in* ***Figure 9****.*

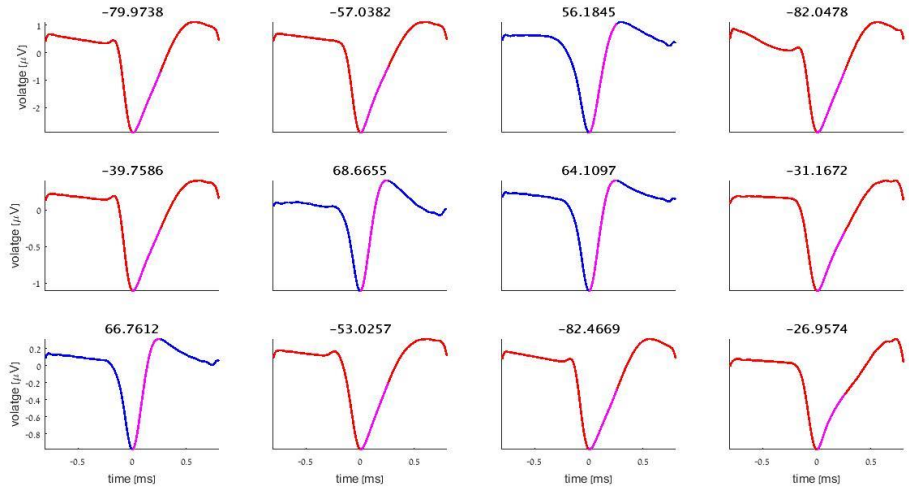

Figure 18*: the spikes, and the region of interest fot the Get Acc feature. Same spikes as in* ***Figure 9****.*

Note that the second derivative of the blue spikes is sunken at the region of interest, as opposed to the red signals (**Figure 20**). The plot is of the second derivative, so the "bump" means that the spike decreases its voltage gaining rate quickly before returning to a somewhat steady rate, that corresponds to a second derivative close to zero. From this we have designed the feature to be:

$$\chi = \sum_{\omega \in roi} \left(\left(10^4 \cdot y(\omega) \right)^2 \right)$$

As one can see this quantity gets a positive score for blue signals, and a negative score for red signals. And is supposed to distinguish between the red and blue groups. The region of

interest (ROI) in this feature is 0.09 to 0.25 ms after the trough. In the calculation of this feature the waveform was normalized by dividing it with it sum of squares, $\vec{y} = \frac{\vec{w}}{\|\vec{w}\|^2}$

When considering the ROI on the raw signal it is clear that for the blue spikes, this feature takes into account the peak, while for the red spikes it does not. Furthermore, we can see that the red signals receive negative score on this feature while the blue ones get positive score. Therefore, we expect this feature to be able to separate the red signals from the blue.

Max Speed

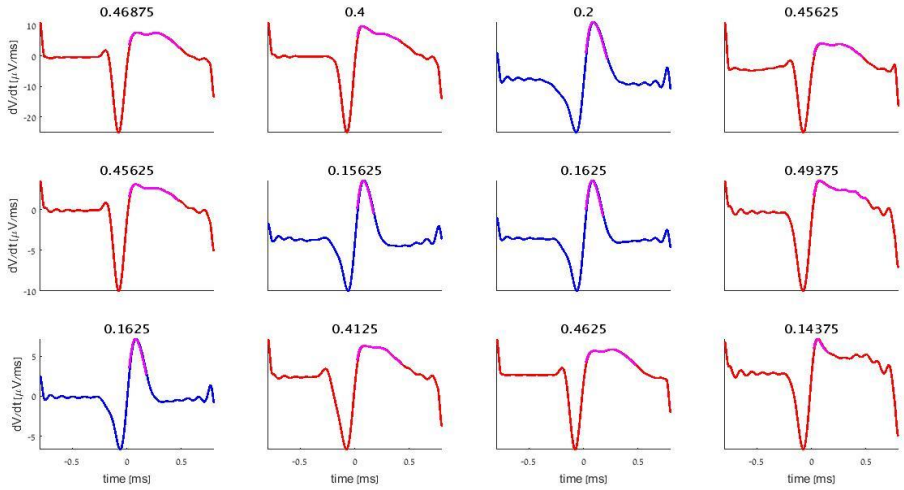

***Figure 19**: First order temporal derivatives of the waveforms. Same units and conventions as in **Figure 9**.*

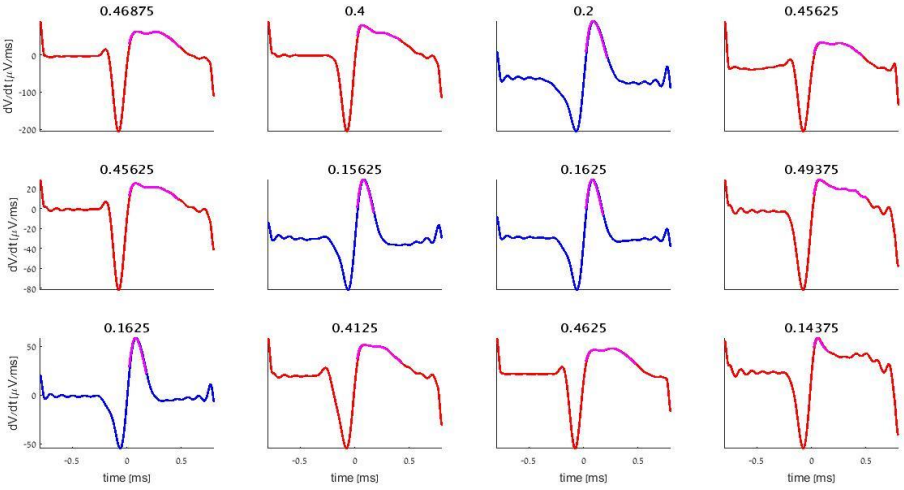

***Figure 20**: The first temporal derivatives (same as in **Figure 14**). The purple line as described below. The numbers above each panel quantify the duration of the post-trough voltage increases (the Max Speed feature).*

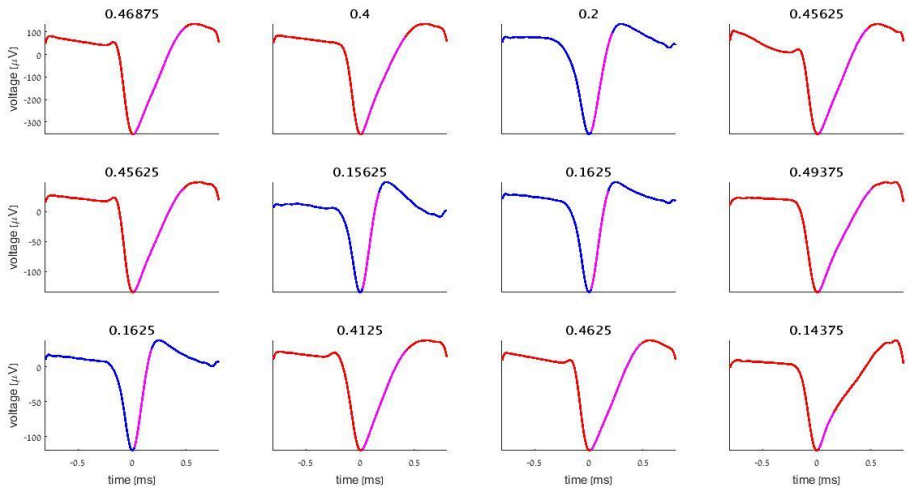

***Figure 21**: The waveforms, with the Max Speed segment highlighted in purple line (same as in **Figure 23**).*

The purple lines in **Figure 21** quantify the duration of the post-trough voltage increases. To measure this duration, we found the segment after the trough that starts with a positive derivative. Due to the upsampling and the temporal derivative operators, this segment always starts at the 131st sample. The Max Speed feature is defined as the duration of the segment that starts in the 131st sample and continues as long as the derivative is above the initial value (the value of the derivative at the 131st sample). It can be seen (**Figures 23 and 24**) that the red spikes typically have longer segments than the blue spikes.

One can take notice in lower right most red spike, and probably but to a lesser degree the second left upper red spike. Those appear to look a little different from the other red spikes, for example because they do not have a "bump" before the trough. The "bump" has been examined and quantified in the "Break Measure" feature. The fact that two distinct features are different for the same two units suggests that these units may be distinct from the other excitatory (red) units.

In these odd red spikes, we observed that the voltage gain rate decreases faster than for the other excitatory units. This effect is more dramatic in the lower right unit.

Smile or Cry

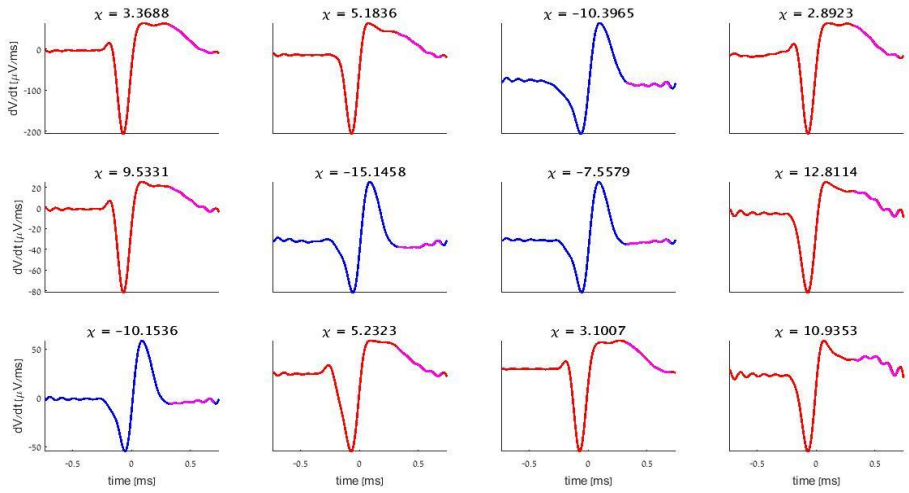

***Figure 22**: the second derivatives. and the relevent ROI*

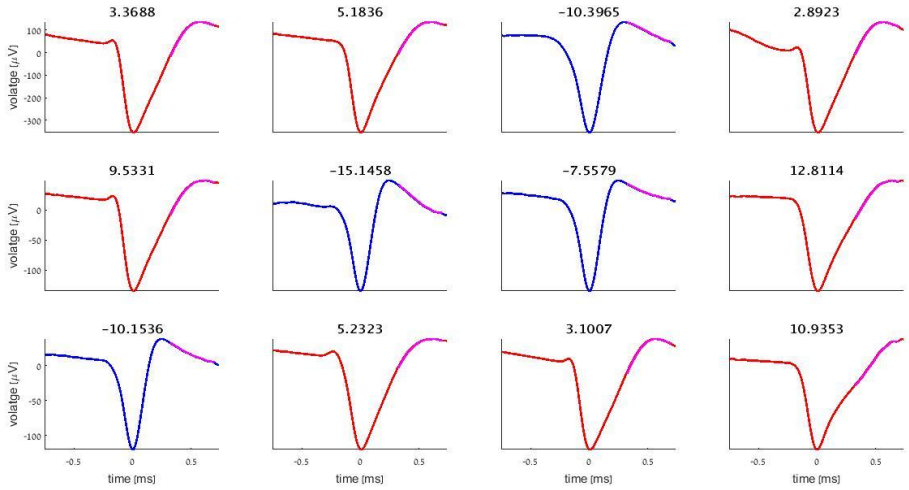

***Figure 23**: the spikes and the relevent ROI*

It can be seen from **Figure 23** that the red waveforms are concave in the purple region, while the blue ones the convex. This means that the second derivative in this region will be negative for the red signals and positive for the blue signals. Hence we defined a "Smile or Cry" feature as:

$$\chi = \exp\left(-\frac{10^5}{|roi|}\sum_{\omega\in roi}\big(y(\omega)\big)\right).$$

The ROI is from 0.3 ms to 0.7 ms from the trough. the ROI was chosen such that the feature will take into account the attribute described, and based on the visualization of the purple lines in **Figures 22** and **23**, and in the calculation of this feature the waveform was normalized by dividing it with it sum of squares, $\vec{y} = \frac{\vec{w}}{\|\vec{w}\|^2}$

Morphological features redundancy

Morphological PDF estimation

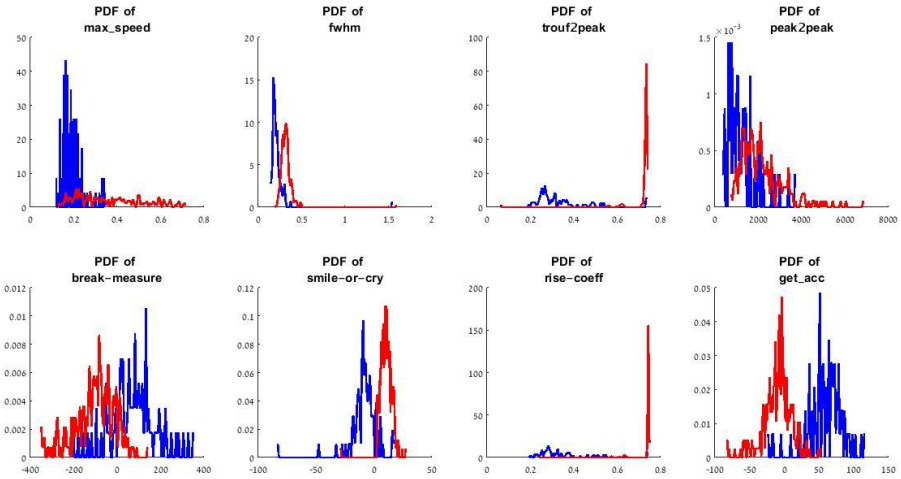

***Figure 24**: Distribution (p.d.f.) of every morphological feature. The blue curves are the normalized histogram for the blue (inhibitory) units, whereas the red curves are for the red units. Each histogram is evaluated using 100 bins, the normalization is to sum to 1.*

The PDF estimation is done by taking the histogram of the values that the features score, for the red units and the blue units separately. In **Figure 24** we can see that the trough2peak and rise-coeff exhibit large margins, while the break-measure histograms overlap heavily.

THE PAIR-WISE SCATTERGRAMS FOR THE MORPHOLOGICAL FEATURES

In order to check for redundancy between the features, we examined the matrix of pair-wise Pearson correlation coefficients. The Pearson correlation coefficient between two vectors is defined as:

$$\rho(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{A_i - \bar{A}}{\sigma_A}\right)\left(\frac{B_i - \bar{B}}{\sigma_B}\right)$$

The Pearson correlation matrix ij-th element is received by calculating the Pearson correlation coefficient between the ith row and jth column. By this definition, $(A,B) = \rho(B,A)$, and thus the correlation matrix is symmetric.

First, we plotted the joint distributions of all pairs of distinct morphological features (8 morphological features, 28 pairs; **Figure 25**).

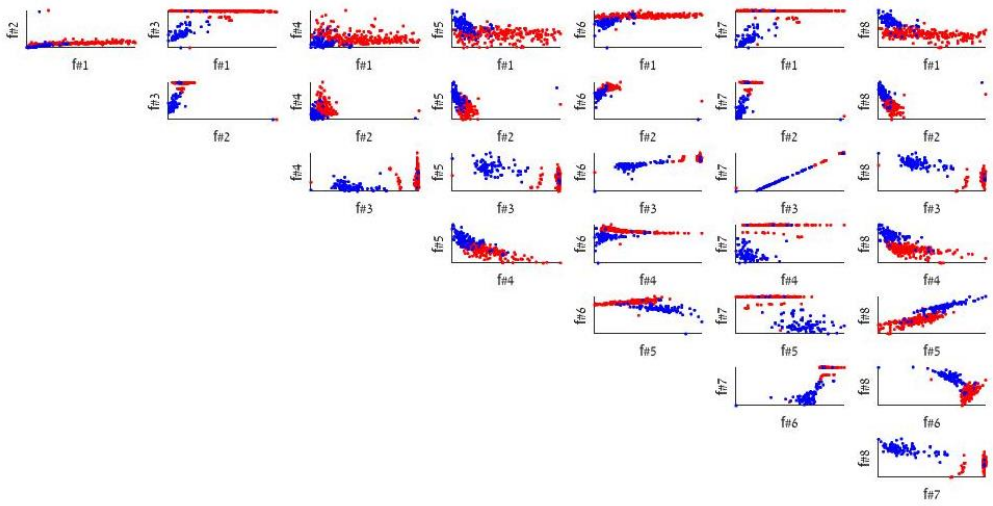

***Figure 25**: Scatter diagrams for pairs of morphological. The axes labels here follows the convention of numbering the features as listed bellow.*

List of morphological features (pertains for **Figure 25**):

1 - 'max_speed'

2 - 'fwhm'

3 - 'trough2peak'

4 - 'peak2peak'

5 -'break-measure'

6 - 'smile-or-cry'

7 – 'rise-coeff'

8 - 'get_acc'

The Correlation matrix is shown in **Table 1**.

	feature 1	feature 2	feature 3	feature 4	feature 5	feature 6	feature 7	feature 8
feature 1	1	0.239	0.444	0.032	-0.368	0.494	0.465	-0.578
feature 2	0.239	1	0.144	-0.055	-0.046	0.024	0.15	-0.137
feature 3	0.444	0.144	1	0.353	-0.578	0.778	**0.992**	-0.753
feature 4	0.032	-0.055	0.353	1	-0.741	0.001	0.356	-0.616
feature 5	-0.368	-0.046	-0.578	-0.741	1	-0.299	-0.576	0.857
feature 6	0.494	0.024	0.778	0.001	-0.299	1	0.786	-0.557
feature 7	0.465	0.15	**0.992**	0.356	-0.576	0.786	1	-0.755
feature 8	-0.578	-0.137	-0.753	-0.616	0.857	-0.557	-0.755	1

***Table 1**: Correlation matrix for the morphological features*

To check whether the various features may provide distinct information, we initially used pair-wise scatter plots (**Figure 25**). It is evident from this figure that the rise-coeff and throupg2peak features (numbers 7 and 3, respectively) are strongly correlated.

REDUNDANCY DISCUSSIO

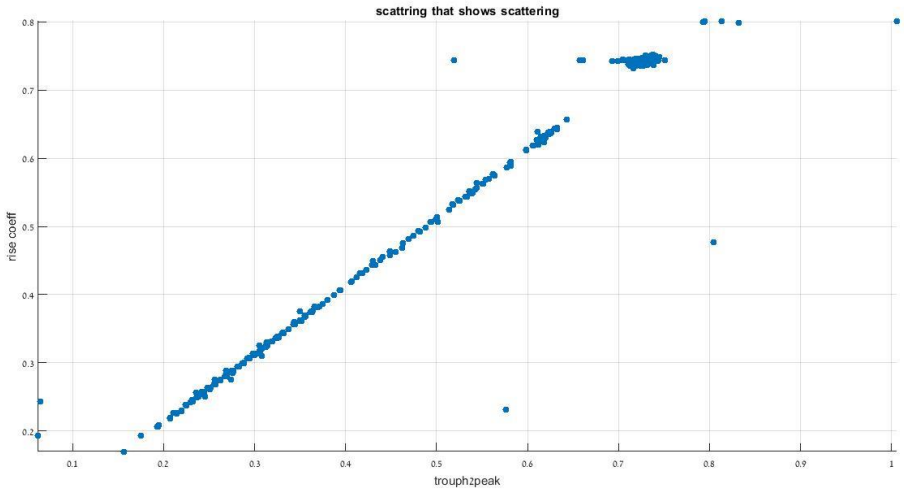

***Figure 26**: Scatter plot of the rise-coeff and trouph2peak for all the clu groups in the sPV file (i.e. 1063 units, as opposed to the data in **Figure 25** which includes only the tagged units).*

Indeed, the rise-coeff and trough2peak features show high linear correlation, i.e. the Pearson correlation coefficient is $= 0.992$. This suggests strong redundancy.

When disregarding one of the redundant features, by discarding the trouph2peak feature, we got the scatter plots shown in **Figure 27**.

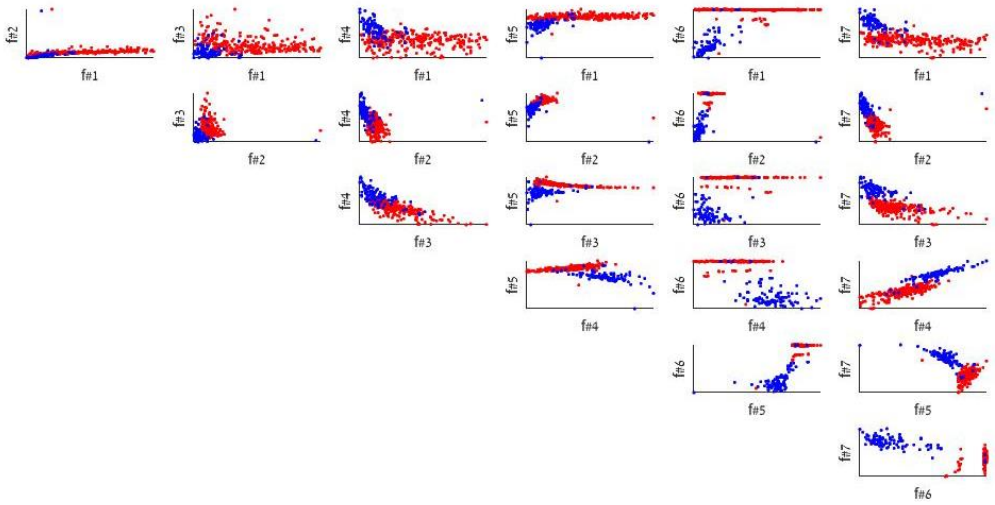

***Figure 27**: Scatter plots for the 7 (out of 8) non-redundant morphological features.*

Figure 27 (confer also **Figure 25**) shows that there are indeed clusters of red vs. blue dots, and a separatrix can be drawn roughly in each combination. A limitation is that when disregarding the fwhm, peak2peakp and t2p features, no two-feature combination exhibits large margin. Another limitation is that we cannot see inner separation inside each cluster except for in the fwhm, peak2peak and t2p features (see pdfs in **Figure 24**). However, it is possible that due to our inability to visualize data properly in e.g. 4 dimensions, a high-dimensional classifier could extract non-redundant information from the additional features. In fact, that is precisely the motivation for exploring a large feature set: each feature by itself may provide a small amount of information, but if it is independent (i.e. not redundant)

when compared to other, more informative features, the overall information will increase and high-dimensional clustering may have a lower generalization error.

Summary

Feature Name	Description	Inclusion In Further Analyses
max_speed	Post-tough first derivative measure	Yes
fwhm	Peak width measure	Yes
peak2peak	Difference of voltage between minimum to maximum	Yes
break-measure	Pre-trough second derivative measure	Yes
smile-or-cry	Far post-tough second derivative measure	Yes
rise-coeff	Variation on the distance of the peak from the trough	Yes
get_acc	Early post-tough second derivative measure	Yes
trough2peak	Distance of the peak from the trough	No

***Table 2**: Morphological features summary*

Temporal Features

In this part we discuss the ACH curve and try to derive from it numerical features that result in a large margin between the red and blue clusters. While trying to do so, we faced a problem that appeared to stem from the fact that many of the units had a small number of samples in the .res file (i.e. only a few spikes). Having a small number of spikes faults since the histogram approach for estimating the ACH curve is very sparse.

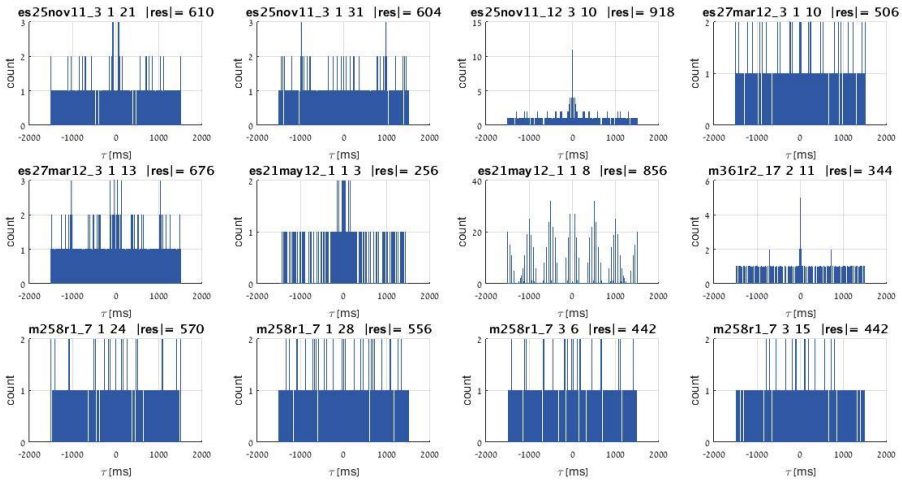

***Figure 28**: Examples of ACH curves of units with a small number of spikes (under 1000).*

As a rule of thumb, we observed that a unit with more than 10,000 spikes results in a "smooth" estimate of the ACH, while units with less than 4,000 spikes exhibited noisy ACHs. Therefore, we only considered temporal feature for units with a sufficient number of samples. The threshold for a satisfactory number of samples was chosen to be 8,000.

The temporal features we have derived are:

- Unif distance index – a measure of how fast the ACH shoots.
- ACH rise time - a measure of the duration of the refractory time.
- ach-jmp-idx – a long term ACH feature.
- psd-features – frequency-domain features (two features)

Processing of the ACH curve

First, ACHs, computed in the range of -1500 ms to +1500 ms in 0.5 ms bins (6001 bins total). Was considered in the close time zone, from -30ms to 30ms. In this zone the ACH were upsampled by a factor of 8, and a CDF was calculated from the r.h.s., i.e. those values for which $\tau \geq 0$ (**Figure 29**). Note that since the ACH is an even function, this CDF completely defines the ACH.

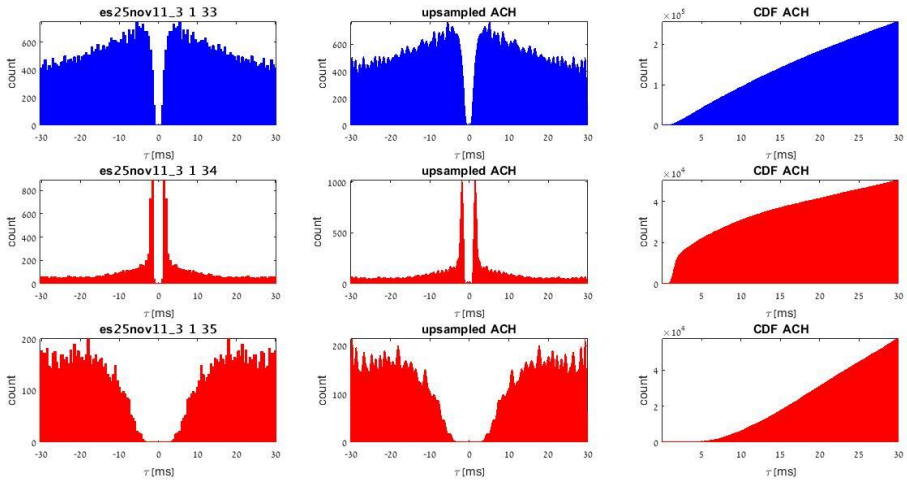

***Figure 29**: ACH preprocessing. Each row corresponds to the same unit. The unit source is discribed in the left column title (same convention as in **Figure 6**).*

Unif distance index

Based on the CDF of the ACH, we derived parameters that accentuate differences between, for instance, a PV cell (**Figure 32**, first row) and excitatory cells (**Figure 32**, other rows). The first heuristic we used is to quantify how similar the CDF is to a uniform distribution. Visually, it was done as shown in **Figure 33**.

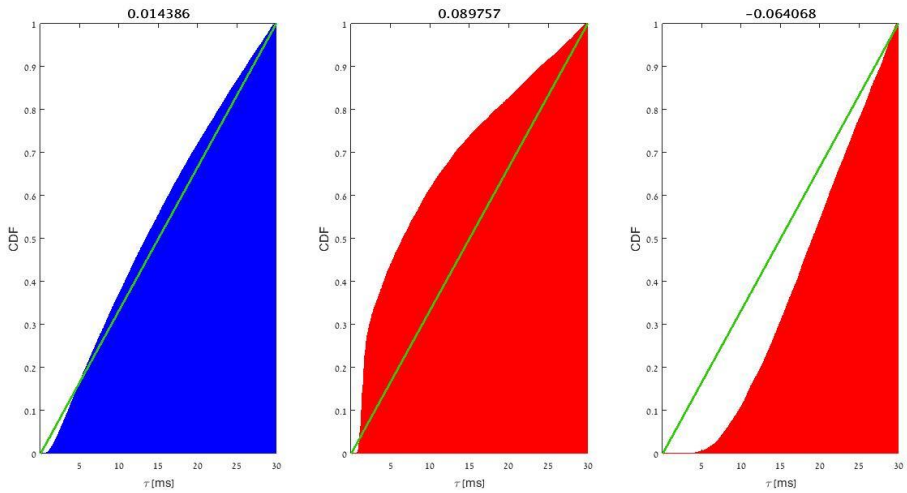

***Figure 30**: unif-dist idx demo. The green line is the CDF of a uniform distribution. Titles show feature values.*

In the visualization (**Figure 33**), we can see how far the CDF is from a uniform CDF (green line). In order to quantify this difference, we calculated:

$$\chi = \sum \frac{q-p}{N},$$

where *q* is the empirical CDF (of the ACH), *p* is the uniform CDF, and N is half the number of samples in the upsampled ACH for which the$\tau \leq 30$(N=968)

One can notice that the red ACHs have larger values – either positive or negative, while the blue ACH has a smaller absolute value. Therefore, we expect to see a blue cluster between two red clusters on this axis.

ACH rise time

The next parameter we examined is the relaxation time near zero. The procedure is depicted in **Figure 29**.

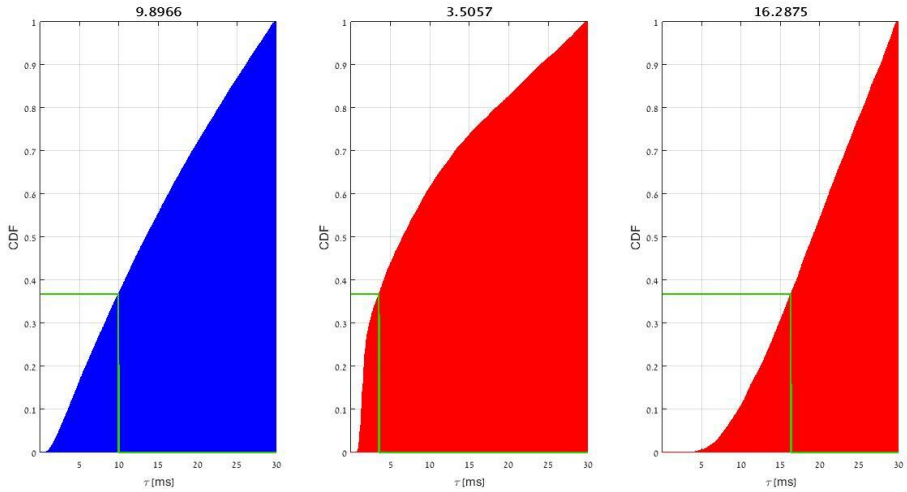

***Figure 31**: ach-risetime demo. The titles are the value of the feature.*

This parameter is the time it takes for the CDF to reach $\frac{1}{e}$ of its final value (computed over a 30 ms time window). Here, similarly to the “Unif distance” feature, the red ACHs exhibit

either small or large values when compared to the blue ACH. From this, one may assume that the ACH rise-time and the Unif-distance features may be correlated, resulting with one of them being redundant. However, over a population of multiple other units, they may also be anti-correlated or independent. Over a sample of 1063 units, the correlation coefficient between this two features was -0.99, indicating that they are strongly anti-correlated. This is further examined in the redundancy analysis of the morphological features

ach-jmp-idx

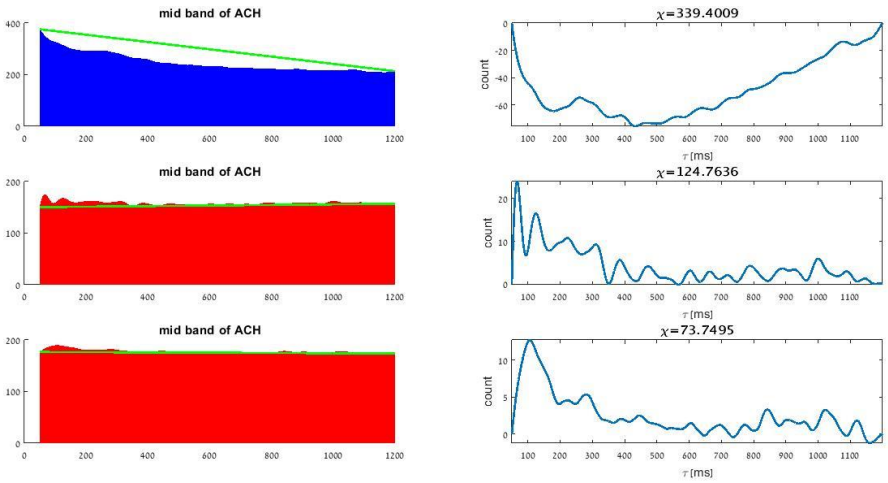

***Figure 32**: Left: ACHs in the midband (50-1200 ms). The green line passes between the first and last samples in this range. Right: the difference between the ACH and green line.*

Here, we examine the long-term auto-correlation, in the band $\tau \in [50,1200]\ ms$. The feature we extracted is the sum of the differences between the long-term ACH y and the long-term linear fit $\bar{y}$, namely:

$$\chi = 100 \log\left(\sqrt{\frac{\sum(y-\bar{y})^2}{5000}}\right)$$

This quantifies the tendency of the unit to oscillate at low frequencies (0.83-20 Hz). As seen in the example (**Figure 32**), the blue ACH scores a higher number than the red ACHs, suggesting that the PV cell is more strongly locked to LFP oscillations in the delta, theta, alpha, and/or beta bands. We chose the numbers 5000 and 100 as a design choice that yielded. Other choices can be made.

psd-features

From the Wiener-Khinchin theorem, the Fourier transform of the ACH is the power spectral density:

$$S_{xx}(f) = \mathcal{F}\{R_{xx}(\tau)\}$$

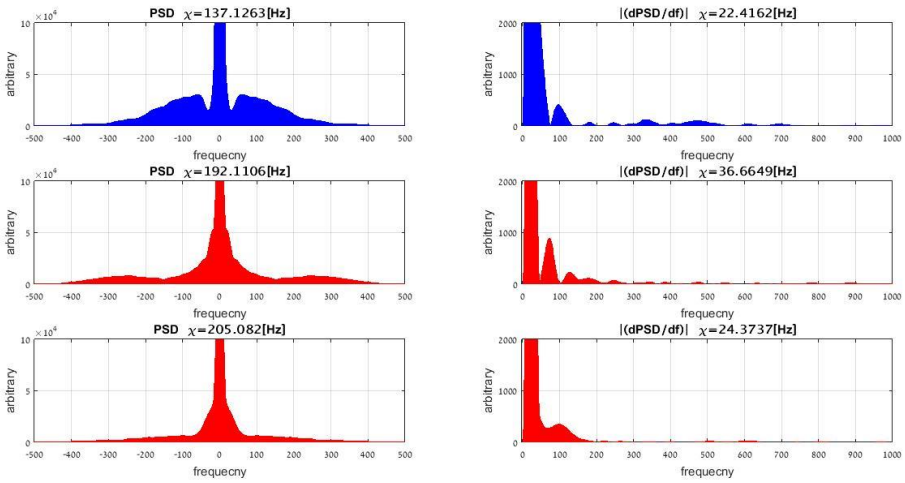

***Figure 33**: psd features demo. Left: there is the PSD of the same units shown in **Figure 32**. Right: absolute value of the frequency derivative of the PSD.*

The features extracted from the PSD are (1) the center of mass of the PSD, and (2) the center of mass of the absolute value of the frequency derivative of the PSD. Both are calculated from the single sided PSD. The reasoning behind the first feature is that the middle red unit exhibits a broader bandwidth, so its center of mass will be in higher frequencies. the scond feature is the center of mass of the derivative of the PSD. We assume that the lower have a derivative more concentrated around zero, while the others are more spread out in the frequency.

Temporal features redundancy

Temporal PDF estimation

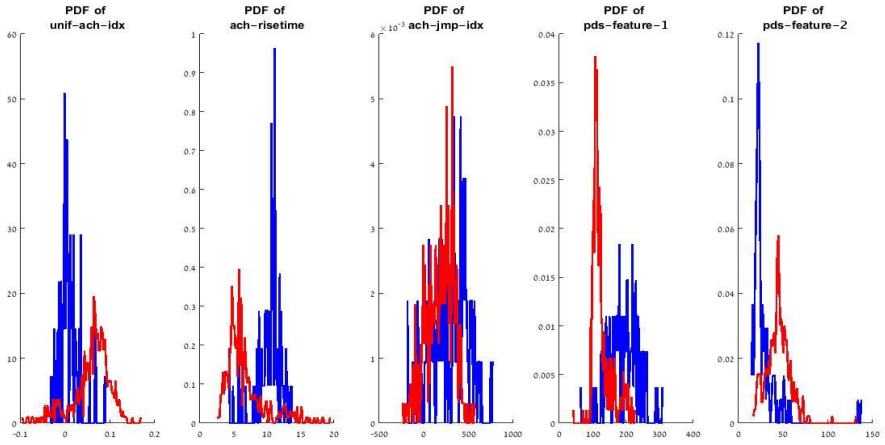

***Figure 34**: Feature histogram for every feature. Same conventions as in **Figure 27**.*

As for the morphological features, the pdf estimation is again the histogram of values that the feature gets, for the red units, or the blue units. Those were plotted on the same axis. Here we see that none of the features exhibits a large margin. Yet it is possible that the combination of the features will provide additional information useful for clustering.

THE PAIR-WISE SCATTERGRAMS FOR THE TEMPORAL FEATURES

We then plotted the joint distributions of all pairs of distinct morphological features (5 temporal features, 10 pairs; **Figure 35**).

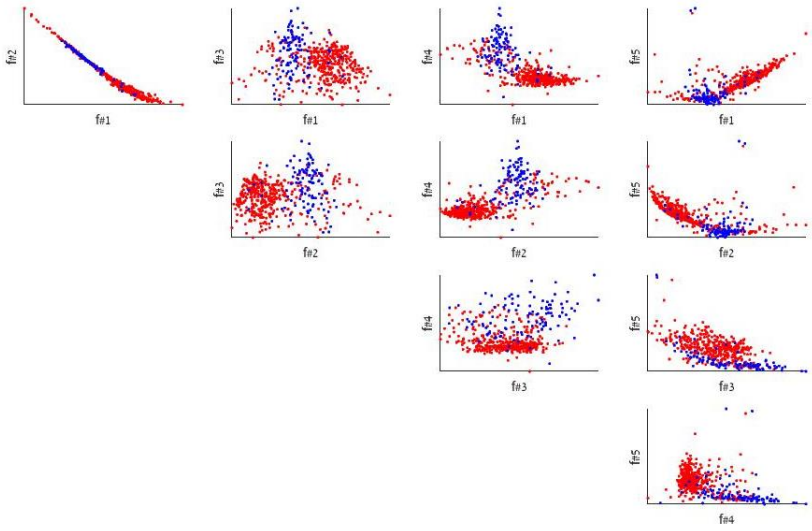

***Figure 35**: Scatter plots for pairs of temporal features. The axes label here follows the convention of numbering the features as listed bellow. The graphs show only clu groups which had more than 8000 spikes.*

These features do not exhibit separation to clusters by a large margin, despite the fact that the ACH-risetime and unif-ach-idx do have a large blue cluster sitting in the middle of the red cluster. The PSD features look like they include the blue cluster inside the red cluster. From these scatter plots, it can be thought that those temporal features will not do well if we will try to classify using them. However, that is only the case for using them in pairs. If they are used in combinations or more than two (or with morphological features), classification may improve considerably.

List of temporal features used in **Figure 35**:

1 –'unif-ach-idx'

2 –'ach-risetime'

3 –'ach-jmp-idx'

4 –'pds-feature-1'

5 –'pds-feature-2'

The Correlation matrix for the temporal features shown in **Table 3**.

	feature 1	feature 2	feature 3	feature 4	feature 5
feature 1	1	**-0.987**	0.039	-0.645	0.242
feature 2	**-0.987**	1	-0.08	0.644	-0.199
feature 3	0.039	-0.08	1	0.049	-0.611
feature 4	-0.645	0.644	0.049	1	-0.155
feature 5	0.242	-0.199	-0.611	-0.155	1

***Table 3**: Correlation matrix for the temporal features.*

REDUNDANCY DISCUSSION

From **Table 3**, we can see that the 'unif-ach-idx' and the 'ach-risetime' features show high anti correlation, suggesting redundancy. Those features exhibit joint scattering as shown in **Figure 36**.

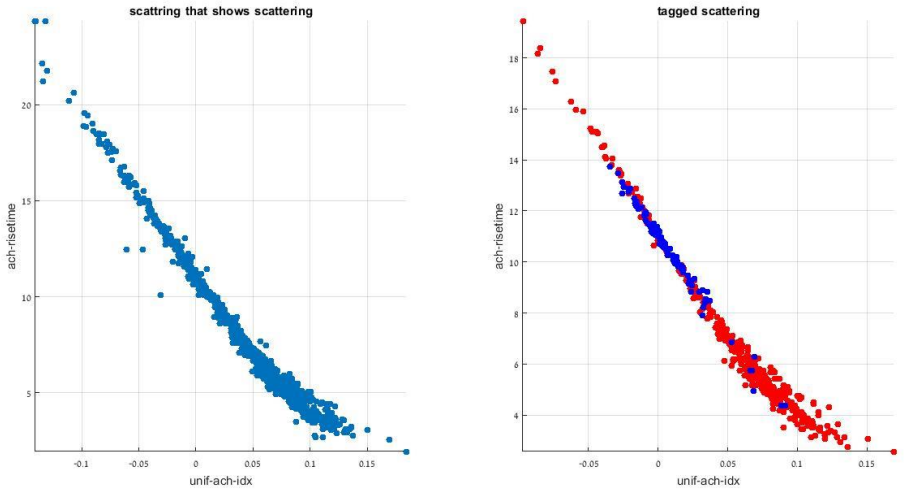

Figure 36: scattring for the highly correlated dimensions.

We chose not to discard this feature, since the data has variance in both axes, and the relationship does not appear completely linear. To further explore the redundancy, we checked how the pair-wise scatter looks when changing the 'ach-risetime' feature to check how much it takes for the ACH's CDF to gain 1% rather than 37% (100/e). This modification yielded the scattering shown in **Figure 37**.

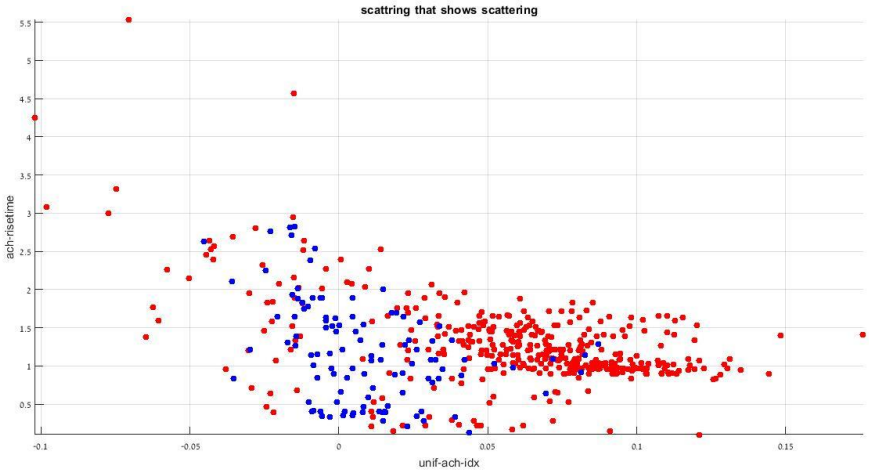

Figure 37: Scatter plot for the unif-ach-idx with the modified ach-risetime feature.

Figure 37 shows that the features are indeed not inherently correlated, and the heuristic modification resulted in a better correlation attribute. This adaptation resulted in a lower Pearson correlation coefficient, $\rho = -0.512$.

After modifying the ach-risetime feature, we obtained the PDFs and pair-wise scatter plots shown in **Figures 38 and 39**.

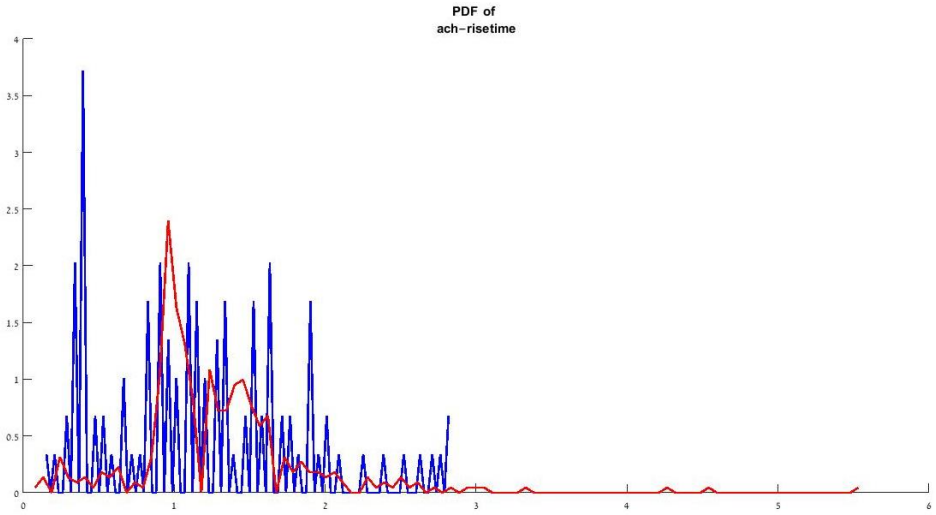

Figure 38: PDF for the 'ach-risetime' feature after modification.

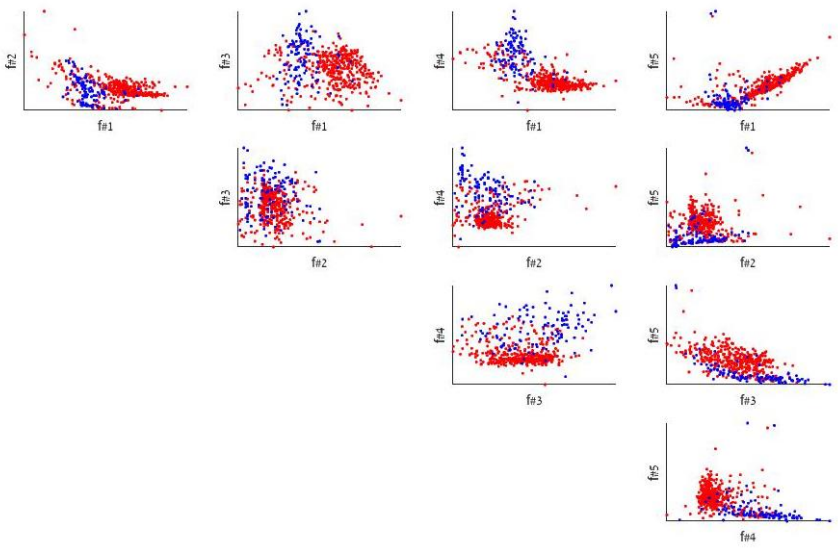

***Figure 39**: Pair-wise scatter plots of the modified 'ach-risetime' feature with all the other temporal features.*

Summary

Feature Name	Description	Inclusion in Further Analyses
'unif-ach-idx'	a measure of how fast the ACH shoots	YES
'ach-risetime'	a measure of the duration of the refractory time.	YES after the fixing
'ach-jmp-idx'	a long term ACH feature.	YES
'pds-feature-1'	frequency-domain feature –PSD center of mass	YES
'pds-feature-2'	frequency-domain features – PSD frequency center of mass	YES

***Table 4**: Temporal features summary*

Redundancy analysis: morphological vs. temporal

The entries for the matrix calculation are ordered as follows:

feature number	Feature name	Feature type
1	unif-ach-idx	temporal
2	ach-risetime	temporal
3	ach-jmp-idx	temporal
4	pds-feature-1	temporal
5	pds-feature-2	temporal
6	max_speed	Morphological
7	fwhm	Morphological
8	peak2peak	Morphological
9	break-measure	Morphological
10	rise-coeff	Morphological
11	smile-or-cry	Morphological
12	get_acc	Morphological

***Table 5**: matrix ordering summary*

	feature 1	feature 2	feature 3	feature 4	feature 5	feature 6	feature 7	feature 8	feature 9	feature 10	feature 11	feature 12
feature 1	1	-0.51215	0.037343	-0.64483	0.241667	0.082544	0.063838	0.324408	-0.26937	0.342172	0.12218	-0.20543
feature 2	-0.51215	1	-0.29396	0.093743	0.321231	0.069413	0.050812	0.037365	-0.09798	0.064884	0.093243	-0.16991
feature 3	0.037343	-0.29396	1	0.050593	-0.6121	-0.04299	-0.13531	0.128546	6.14E-05	-0.25349	-0.26333	0.097897
feature 4	-0.64483	0.093743	0.050593	1	-0.1547	-0.22013	-0.11805	-0.4672	0.452505	-0.54572	-0.30315	0.45563
feature 5	0.241667	0.321231	-0.6121	-0.1547	1	0.109899	0.103103	0.090524	-0.16186	0.301696	0.216901	-0.17454
feature 6	0.082544	0.069413	-0.04299	-0.22013	0.109899	1	0.282239	0.013587	-0.36506	0.482226	0.51305	-0.577
feature 7	0.063838	0.050812	-0.13531	-0.11805	0.103103	0.282239	1	-0.0418	-0.08462	0.197823	0.092571	-0.18547
feature 8	0.324408	0.037365	0.128546	-0.4672	0.090524	0.013587	-0.0418	1	-0.74582	0.370779	0.000414	-0.60227
feature 9	-0.26937	-0.09798	6.14E-05	0.452505	-0.16186	-0.36506	-0.08462	-0.74582	1	-0.61036	-0.33615	0.863198
feature 10	0.342172	0.064884	-0.25349	-0.54572	0.301696	0.482226	0.197823	0.370779	-0.61036	1	0.795249	-0.78167
feature 11	0.12218	0.093243	-0.26333	-0.30315	0.216901	0.51305	0.092571	0.000414	-0.33615	0.795249	1	-0.5816
feature 12	-0.20543	-0.16991	0.097897	0.45563	-0.17454	-0.577	-0.18547	-0.60227	0.863198	-0.78167	-0.5816	1

***Table 6**: Correlation matrix for entire set of 12 features.*

This matrix shows that the features shows small correlation in the fixed features.

Classification process

Supervised Analysis

The supervised approach applies to a setup in which the tags are known. In our case the tagging is chosen to be whether the unit is red or blue. Thus, we cannot generate a model using supervised learning using untagged data. A model generated using supervised learning can be, however, applied to untagged data. Thus, the workflow is as follows

1. Divide the tagged data into two parts: training and test
2. Learn a model using the (tagged) train data
3. Test the model by measuring the generalization error (on the tagged test data)
4. Apply the model to the untagged data

Over all we have 538 tagged units out of 1063 units overall (50.6%). The tagged consist of 424 red units and 114 blue units.

Before classification, the data are organized as follows. The process starts with calculating the features for each unit. These features are organized as a matrix where each column corresponds to a different feature from the set we designed, and each row corresponds to a unit from the sPV file. Thus, the feature matrix is symbolized as $\mathrm{X} \in \mathbb{R}^{538\times12}$. Second, the tags of the units are organized in a column vector, $Y \in \{0,1\}^{538}$, were each entry is 1 if the unit is red, and zero if the unit is blue. Untagged units are not taken into account.

Supervised learning is carried out selecting a model architecture (e.g. SVM, GMM, etc.), and fitting the model parameters to the data. In the context of SVM (see below), this also involves preforming a grid search on several key parameters. As mentioned above, performance is evaluated by measuring the generalization error, obtained by applying the model trained on one part of the tagged data (the training set) on the other part of the data (the test data).

SVM

Support-vector machines (SVMs) are supervised learning models which can be used for both classification or regression challenges. Using the data and tags, an SVM training algorithm results with a model that can assigns predicated tags to new examples. the non-linear SVM returns the classifier that separates the data in a non-linear space by a **margin** that is as wide as possible. Then new data can be projected into that same space and be predicted to have a specific tag based on which side of the margin they fall in.

More precisely the SVM classifier is the separator that has the largest margin[1]. This separator can be achieved using optimization techniques, involving convex optimization and the "Kernel trick", which means a transformation to a high dimensional space, in which a large margin may exist. In such a high dimensional space, classification takes place, and then the inverse transformation is performed. The input for an SVM are X and Y as defined above, and the output is a linear classifier w. Using w, the classification can be made by $sign(y_i \cdot \mathrm{K}(\overrightarrow{x_i}) \cdot \vec{w})$

The parameters of the SVM are C, which stands for the slackness or error penalty. And γ which is the decay coefficient for the 'RBF' (radial basis function) kernel that we symbolize as K. We chose to use SVM using RBF since using it is a non-linear high dimensional model that

[1] TAU ML course: http://ml-intro-2016.wdfiles.com/local--files/course-schedule/ML-lesson-6.pdf

can take into account non linearity in the data, this in contrast to a high dimensional linear SVM.

SVM Practical Use

The SVM classifier was evaluated in the supervised setup using an increasing number of features (dimensions). We expect to see that by increasing the number of features we can improve the score as we take into account different aspects in which the red and blue units are distinct. This expectation will hold if as we increase the number of dimension, only uncorrelated meaningful information is added. We will note that if we introduce noisy data, performance may become worse

For two-dimensional models, all possible pairs formed by drawing two of the 12 non-redundant features were taken into account. The number of possible combinations is $\mathrm{N} = \frac{12 \cdot (12-1)}{2} = 66$. For each combination, the data were split randomly to 60% of training data and 40% of test data. This training process was chosen arbitrarily.
A grid-search on the model parameters was then performed. The model was learned using the training data using different values of the RBF parameters C and γ. The search is done in a logarithmic scale from 10^{-2} to 10^{10} using 13 steps, for C. and from 10^{-9} to 10^{3} using 13 steps, for γ. Then, the generalization error was evaluated using the test data. The process was repeated 10 times for each combination of C and γ; in each repetition, the partitioning of the data into train/test sets was randomized independently of the other repetitions. Then, the C/γ combination that yielded the smallest generalization error was chosen, and the performance (1-generalization error) summarized for each pair of features (Table 7).

	feature 1	feature 2	feature 3	feature 4	feature 5	feature 6	feature 7	feature 8	feature 9	feature 10	feature 11	feature 12
feature 1	0	0.896	0.8901	0.9015	0.898	0.9579	0.9678	0.9248	0.95	0.9916	0.9767	0.9842
feature 2	0	0	0.8287	0.8574	0.8728	0.8861	0.9624	0.8569	0.9114	0.9916	0.9718	0.9767
feature 3	0	0	0	0.8871	0.8649	0.8757	0.95	0.9025	0.9327	0.9946	0.9757	0.9743
feature 4	0	0	0	0	0.8649	0.947	0.9663	0.9109	0.946	0.9911	0.9777	0.9837
feature 5	0	0	0	0	0	0.9302	0.9624	0.9054	0.949	0.9911	0.9807	0.9851
feature 6	0	0	0	0	0	0	0.9639	0.945	0.9356	0.9926	0.9777	0.9767
feature 7	0	0	0	0	0	0	0	0.9708	0.9545	0.9911	0.9832	0.9797
feature 8	0	0	0	0	0	0	0	0	0.9248	0.9876	0.9797	0.9723
feature 9	0	0	0	0	0	0	0	0	0	0.9921	0.9827	0.9792
feature 10	0	0	0	0	0	0	0	0	0	0	0.9916	0.9896
feature 11	0	0	0	0	0	0	0	0	0	0	0	0.9797
feature 12	0	0	0	0	0	0	0	0	0	0	0	0

***Table 7**: the best score for each combination of features.*

For three dimensional models, we proceeded in exactly the same manner as for two-feature models. The number of combinations here is $\mathrm{N} = \binom{12}{3} = \frac{12!}{(12-3)!3!} = 220.$

In the case of four dimensional models, the number of combinations is 495. We decided that evaluating this many times the model, is not useful. Instead, we decided on a heuristic for constructing a combination of 4 features using the results for the 2D models. We selected the 20 top features combination from the 2D models, and created quadruplets from them. We created the quadruplets by going through the pairs, and constructing quadruplets in which there is not a feature that appears twice. The number of quadruplets we tries is $N = 73.$ On those quadruplets we preformed the same process as for the 2D and 3D case.

in **figure 40** we graphed the number of combination versus the dimensions used.

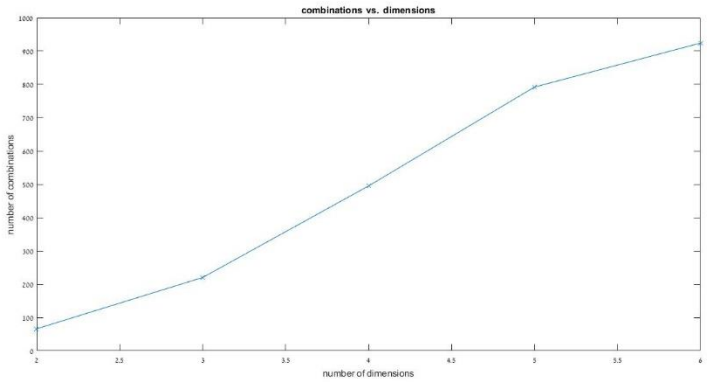

Figure 40: number of combinations of features against the number of dimensions used to evaluate the model.

Five dimensional model: here in a similar manner we took the triplets from the 3D model results, and used them to form quintuplets. We chose two triplets, and added to these triplets three features with all possible pairs $N = 2 \cdot \frac{(12-3)(12-4)}{2} = 72$.

The results for the different models is summarized in the **figure 41:**

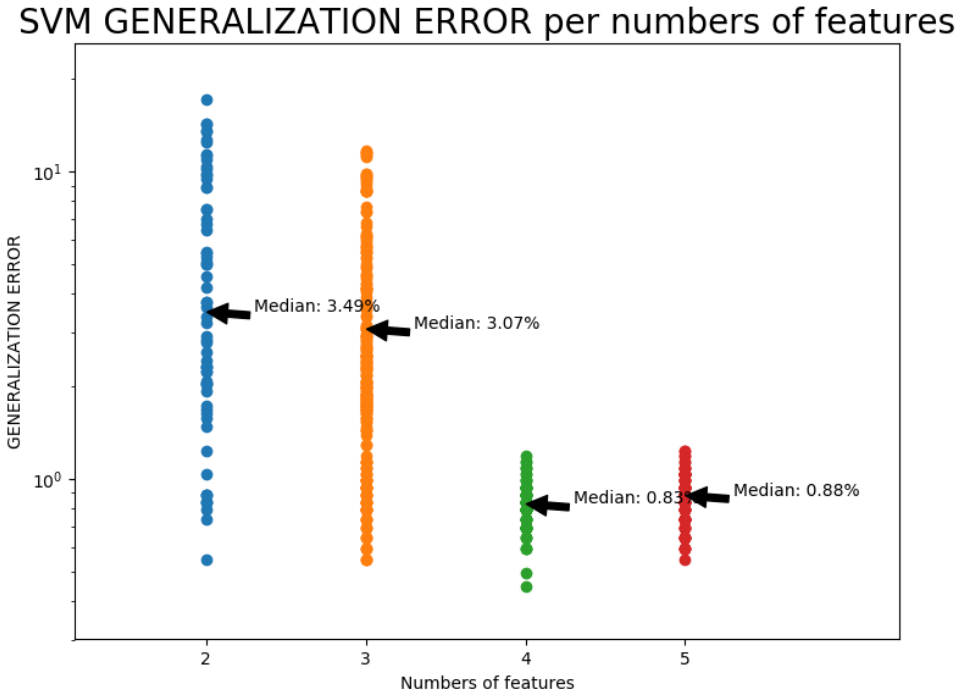

Figure 41: log of 1 minus the score as a function of number of features for the SVM.

In the following table we summarize the results of the SVM evaluation on the best set of parameters:

Best pairs	ach-jmp-idx rise-coeff **Score: 0.9945**	max_speed rise-coeff **Score: 0.9925**	break-measure rise-coeff **Score:0.9920**
Best triplets	rise-coeff ach-jmp-idx get-acc **Score:0.9945**	rise-coeff ach-risetime pds-feature-2 **Score:0.9945**	rise-coeff pds-feature-1 max-speed **Score:0.9945**
Best quadruplets	unif-ach-idx ach-risetime peak2peak smile-or-cry **score: 0.9955**	unif-ach-idx max-speed peak2peak smile-or-cry **Score: 0.9940**	unif-ach-idx peak2peak smile-or-cry get-acc **Score: 0.9940**
Best quintuples	unif-ach-idx ach-risetime max-speed peak2peak smile-or-cry **score:0.9940**	unif-ach-idx ach-jmp-id peak2peak smile-or-cr get-acc **score:0.9940**	

Table 8: features summary for the best SVM fits.

In the following we did is to fix the model. We chose the model with the lowest score and with the smallest number of features. The choice is a model using 2 features, 'ach-jmp-idx' and 'rise-coeff', where using C=1.0 γ =0.1.

The next step is to evaluate the model using the entire tagged data, and to evaluate the intrinsic error (bias error) see **figure 42**.

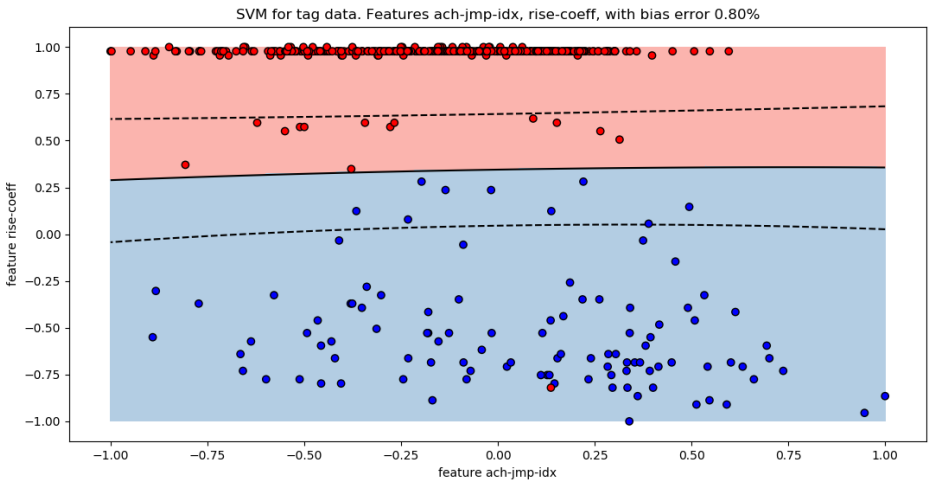

Figure 41: the SVM applied to tagged data. the color code represent the ground true labels of each unit.

Lastly, we apply the model to the untagged data(**Figure 43**). For the untagged data it is impossible to test the performance.

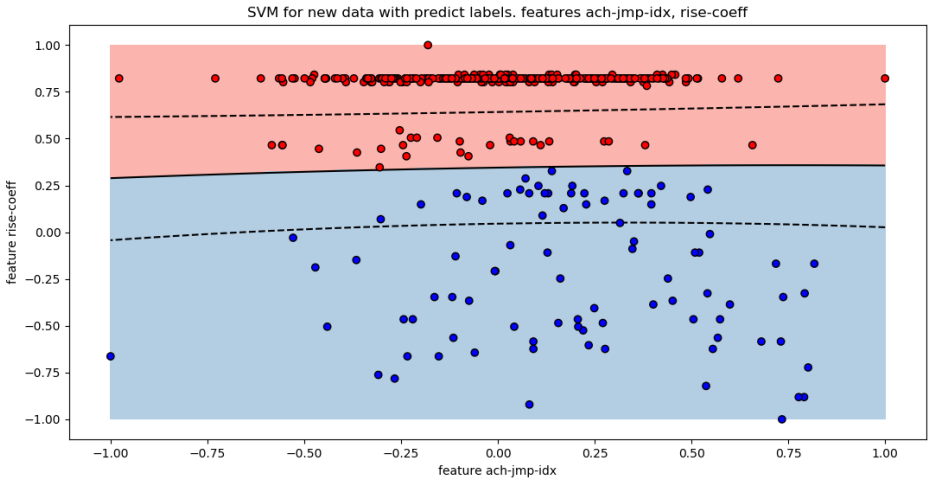

Figure 43: the SVM applied to untagged data. the color code represent the predicted labels of each unit.

Unsupervised Analysis

In this approach, we start the process without using the tags. Thus, we use the full dataset of 1036 units from the beginning. We try to create clusters in the full 12-dimensional feature space, and analyze those clusters using the tags. We expect that the data will cluster in a way that will conserve the separation to groups of red and blue units; this is a sanity-check for the procedure. The goal of the process is to achieve an inner separation of the each of the two tagged sets (red and blue groups), which would correspond to differentiating between different types of excitatory or inhibitory cells. The working assumption here is that mixing of the blue and red units is a non-favorable clustering.

In this setup workflow is as follows:

1. Use the entire data and evaluate a unsupervised model, using a range of model parameters.
2. Use information measures in order to select the model parameters.
3. Use the chosen parameters and evaluate the model using them, this will give us clustering of the data
4. Decide on the cluster tag using the majority of the tagged units, which are included in the cluster.
5. Analyze the error of the clustering using the tagged data.

GMM

A Gaussian Mixture Model (GMM) is a parametric probabilistic model. Rather than identifying clusters by “nearest” centroids, a GMM fits a set of k gaussians to the data. The GMM algorithm estimates Gaussian distributions parameters, namely the mean and the covariance matrix of each cluster. It also weighs each cluster.

A multivariate guassian distribution density is given by:

$$f_X(x|\mu,\Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Where Σ is the covariance matrix of the distribution and μ is its mean.

The Gaussian mixture model is defined by the density:

$$\tilde{f}_X(x|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{N} \pi_k \cdot f_X(x|\mu_k, \Sigma_k)$$

Where π_k is the mixing coefficient of the k[th]. Σ_k and μ_k are the covariance and mean of the k[th] distribution. $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ symbolize the entire parameters of the density.

The model evaluation means fitting the parameters for a mixture of k Gaussians. It is done using EM algorithm Gaussian mixture density[2]. This is an iterative algorithm, for which each iteration has two steps. The E step stands for expectation step, in which the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$, are found. In the M step, the value of those parameters is updated using ML method.

The GMM can be evaluated using four types of covariance modes. **'full'** means the each component of the Gaussian mixture has its own general covariance matrix This means that there are no constrains on $\boldsymbol{\Sigma}$. **'tied'** means that all the components share the same general covariance matrix. This means that we do not put constrains on $\boldsymbol{\Sigma_k}$, but every component will have the same $\boldsymbol{\Sigma_k}$. **'diag'** means that every component has its own diagonal covariance matrix. This means that every component has a different $\boldsymbol{\Sigma}$, which is diagonal. The fourth model is '**spherical**'. In this mode each component has its own single variance means that each entry on the diagonal, in the diagonal covariance matrix $\boldsymbol{\Sigma_k}$ is the same, Yet every component has its own variance. The model complexity is lowest for the 'spherical' mode, and highest for the 'full' mode.

In order to find the optimal number of clusters we used the BIC (Bayesian information criterion). Its definition is: $BIC = \ln(n)\, k - 2\ln(\hat{L})$, where $\hat{L}$ is the likelihood of the model, k is the number of estimated parameters, and n is the number of data points used in the algorithm. This measure takes into account the model complexity and the likelihood of the clustering.

We chose to choose the number of clusters for every covariance model. Thus we k is linear with the number of clusters in the GMM model. In our analysis we substituted k with the number of clusters. A stricter approach of model selection may be applied in further work, is to evaluate k as the sum of all the model parameters. For example in a 2D space in a 'full' covariance mode each gaussian component has 5 parameters (variance in each dimension, the cross-covariance, and mean for every dimension). Using 4 components in this setting results with 20 parameters. Using this method models evaluated in different modes and dimensions can be tested together.

.GMM Practical use

In this approach we will preform GMM in the high dimensional feature space (12 dimensions). We used the BIC in order to find the optimal number of clusters for the GMM.

The GMM was tested in every mode (4 modes), and for a range of cluster numbers. The results are summarized in **figure 44:**

[2]TAU ML course: http://ml-intro-2016.wdfiles.com/local--files/course-schedule/ML-lesson-12.pdf

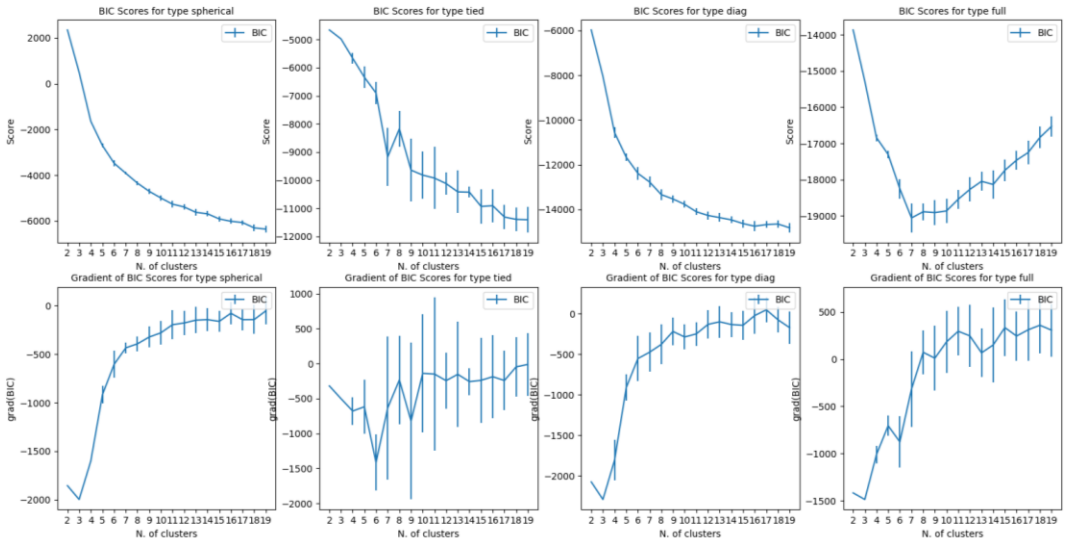

Figure 44: the BIC and BIC gradient for each of the 4 GMM covariance modes.

One can see that as we increase the number of clusters, the BIC decreases for every mode but the 'full' mode. This can indicate that the model reached a state of OVERFITTING. To reduce the over-fitting we will choose the number of clusters from which the gradient becomes steady. This is sometimes called the ELBOW method. For each mode, we chose the optimal number of clusters and evaluated using this number the model 100 times.

For each iteration we had calculated an error based on the tags some of the units have. We will say that a cluster is colored with red or blue, based on the majority of the tagged group in the clusters. Now the error is the sum of red units inside blue clusters, and blue units inside red clusters. This error is averaged over the 100 iterations. The results are summarized in table

Covariance type	**Numbers of clusters**	**Average Error**
Full	**7**	**6.6 (min 6)**
Full	6	7.65 (min 5)
Diag	6	12.38
Tied	2	17.73
Tied	6	8.61
Spherical	5	10.0
Full	7	6.6 (min 6)

*Table 9: GMM error values for different GMM covariance modes. *the tied appears twice since it isn't clear from the graph which number of cluster we need to choose.*

Following those results, we chose the mode with the lowest error, and performed another 200 iterations using 7 GMM components and a 'full' covariance mode. Based on all the evaluated models, we returned the model with the lowest error, which is 6 This model allows us to classify new data to one of 6 clusters, based on the color of the cluster. We will add that the cell types can be divided to at least 6 groups (original goals for the project).

The mean waveform and ACH curve were averaged across each cluster. The results are visualized in the following figure:

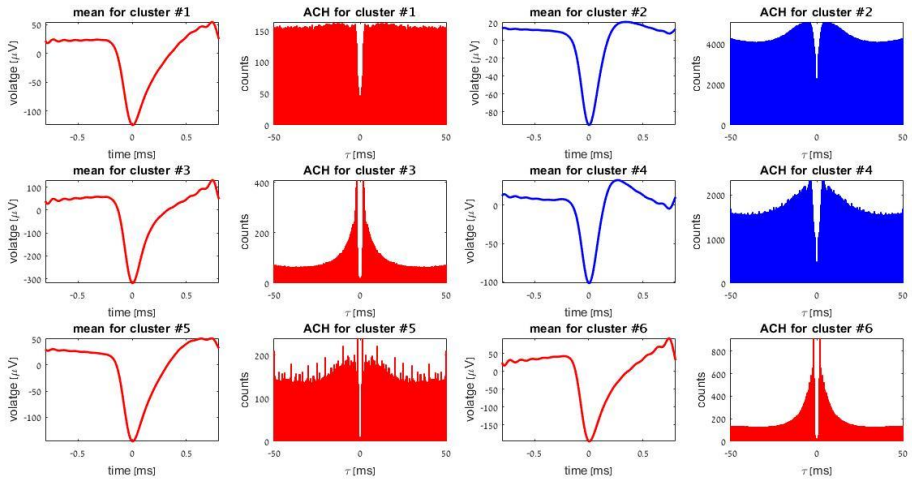

Figure 45: the mean waveform and ACH, averaged for every cluster.

One can see that the GMM clustering was able to distinguish between 2 distinct types of red units. One type has a wide ACH curve as in cluster 1 and 5, and the other has a narrow ACH curve like in cluster 3 and 6.

Using the GMM model we can classify new data. for a new unit we will project it to the feature space, and choose to associate it with the nearest cluster in terms of L2 distance. The predicted tag will be according to the color of the nearest cluster.

The GMM clusters features are summarized in the following tables:

	number of units in cluster	red units in cluster	blue units in cluster
1	67	24	0
2	162	1	92
3	61	35	0
4	26	1	10
5	82	17	1
6	318	206	1

cluster	fwhm	peak2peak	break_measure	rise_coef	smile_or_cry	get_acc
1	0.44 ±0.36	1507.49±637.96	-61.6±116.4	0.7409 ±0.0047	11.98 ±6.48	-5.45±32.82
2	0.22 ±0.05	991.19±438.55	81.2 ±103.9	0.3706 ±0.1137	-5.26±9.71	56.57 ±25.75
3	0.26 ±0.03	3687.11±1179.74	-171±82.4	0.7471 ±0.0033	5.19 ±1.87	-11.15 ± 14.21
4	0.47±0.54	1210.6±879.27	92.6 ±134.9	0.3156 ± 0.0863	-13.6±17.91	64.71 ±23.35
5	0.31±0.05	1660.2±783.08	-94.9 ±98.8	0.6669±0.076	10.26 ±5.36	-12.03±30.34
6	0.31±0.03	2366±883.27	-114.8±73.7	0.7438±0	9.42 ±3.33	-14.71±14.61

cluster	unif_ach_idx	ach_risetime	ach_risetime	pds_feature_1	pds_feature_2	max_speed
1	0.0072 ±0.051	1.63 ±0.84	97.9 ±154.3	157.9 ±33.9	37.6 ±13.2	0.41 ±0.14
2	0.0101 ±0.0198	1.02 ±0.67	293.8 ±184.6	199.6 ±42.5	24.7 ±6.8	0.21 ±0.07
3	0.0658 ±0.0293	1.31 ±0.36	147.1 ±129.8	114.9 ±20.9	46.2 ±9	0.2 ±0.05
4	0.0016 ±0.0481	1.57 ±1.48	55.7 ±297.2	187.2 ±56.2	60.1 ±41.8	0.19 ±0.04
5	-0.0402 ±0.05	3.04 ±2.06	1.7 ±195.9	184.5 ±40.2	60.2 ±45.3	0.38 ±0.12
6	0.0749 ±0.0236	1.22 ±0.26	216.9 ±121.4	111.5 ±9.2	45.9 ±9.8	0.4 ±0.16

Table 10: A summary of the mean value for each cluster, the color code is decided according to the majority of tagged units in the cluster.

To quantify how good this error measure, is we preformed shuffling in the GMM cluster numbers 1000 times, and then calculated the error. We made a histogram based on the

error values that resulted from that iteration, for the 'FULL' mode, and got the following error distribution:

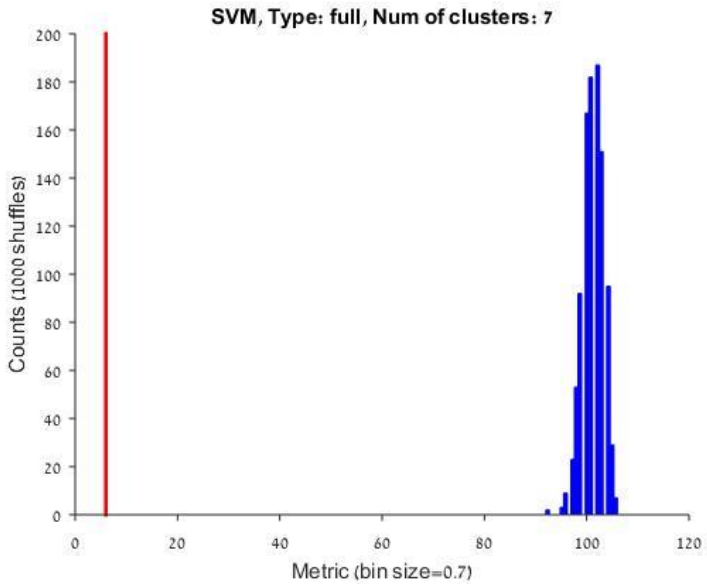

Figure 46: the error distribution for the GMM model using the 'full' covariance mode, and 6 clusters. The red line indicates the error of our model.

We can see that our error is significantly lower than mean of the error distribution.

Random forest

Random forests or random decision forests are an ensemble learning method for classification or clustering based on creating multiple decision trees and pruning them to get a more accurate and stable prediction. The random forest is utilized by separating the data into trees and leafs. From the output of the forest, one can derive the pair-wise distances between different units, and expect similar units to have small distance from each other. [TAU ML course lec 9] [3]

The first step in this process is **data augmentation**. This means generating a matrix X_{synth} of the same shape of X, only with synthetic data drown from the same marginal distributions of the features. The model is evaluated using both X_{synth} and X. In our case X is the feature matrix, which is of dimension 12*949. Each colon correspond to a feature, and each row correspond to a unit the features were calculated for.

The model has tunable parameters that we preformed a grid search on[4].

In each iteration of the grid search the forest is evaluated. Based on the forest, a similarity matrix is calculated for the real observations (not including X_{synth}). The similarity matrix is a symmetric matrix, which dimensions are $n \times n$, where n is the number of units in the feature matrix.

[3] http://ml-intro-2016.wdfiles.com/local--files/course-schedule/scribe-9-Boosting+Ensemble.pdf
[4] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011
random forest implemintation

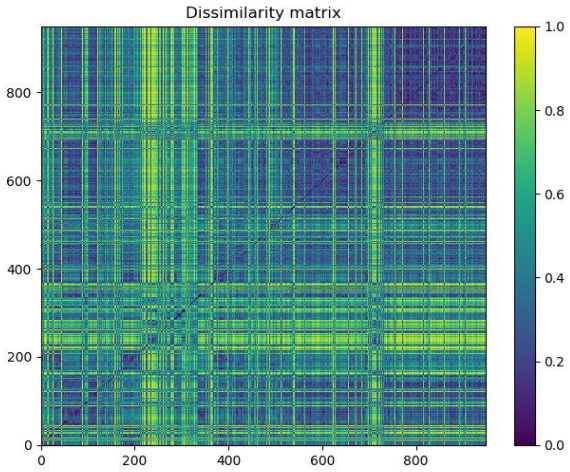

Figure 47: Example of a disimilarity matrix used for the tSNE visualization.

The similarity matrix is visualized using the tSNE dimensionality reduction scheme. tSNE stands for **t-Distributed Stochastic Neighbor Embedding** which in short takes a high dimensional data and projects it to a low dimensional space in a way that conserve structure. The tSNE can also work in a mode where it gets a distance map, and return the representation based on it[REF][5]. This is done by setting its 'precomputed' attribute to 1

The tSNE itself is a tunable algorithm, with its parameter elaborated in the link [REF][3]. The data is to be visualized using the tags, this is done in order to see if the clusters are homogenous with respect to the tags.

A summary of the process used:

1. We generate X_{synth} based on the feature matrix X
2. The optimal Random forest parameters are found using a grid search, using both X and X_{synth}
3. We fix the model using the parameters found in step 2.
4. A distance map between all the units is computed based on how many mutual visits in the tree leafs two units share.
5. This distance map is fed to a t-SNE model, which output a two dimensional feature space.

The tSNE visualization we got is shown in **figure 48**

tSNE implemintation

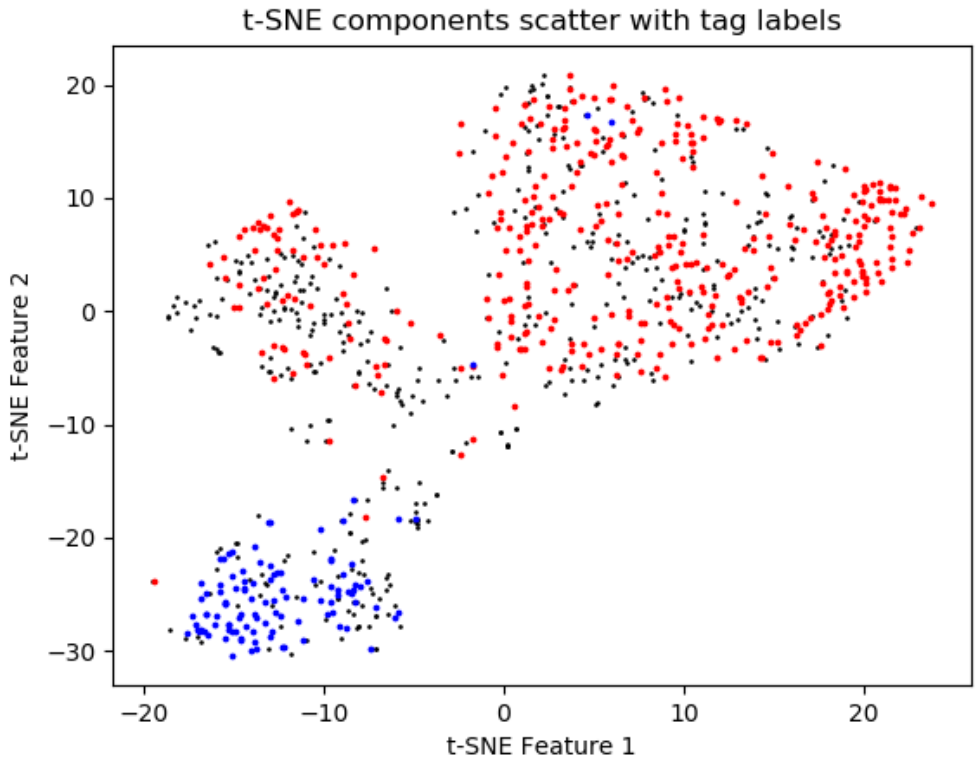

Figure 48: tSNE visualization. The red dots correspond to units that were tagged as red, and blue dots to units that were tagged blue. Black are unlabeled units.

As one can see in the visualization that the blue and red units are separated, based on the high dimensional data.

Now using the 2D space that resulted with this process is fed to the same process as with the GMM on the entire data:

1. A GMM model is evaluated using varying number of clusters. And using the 'full' covariance mode.
2. The number of clusters is chosen using the ELBOW method on the gradient of the BIC curve.
3. The model is fixed using the number of clusters from step 2.
4. We preform error analysis.

In the following figure we summarize the process. First we made the BIC curve.

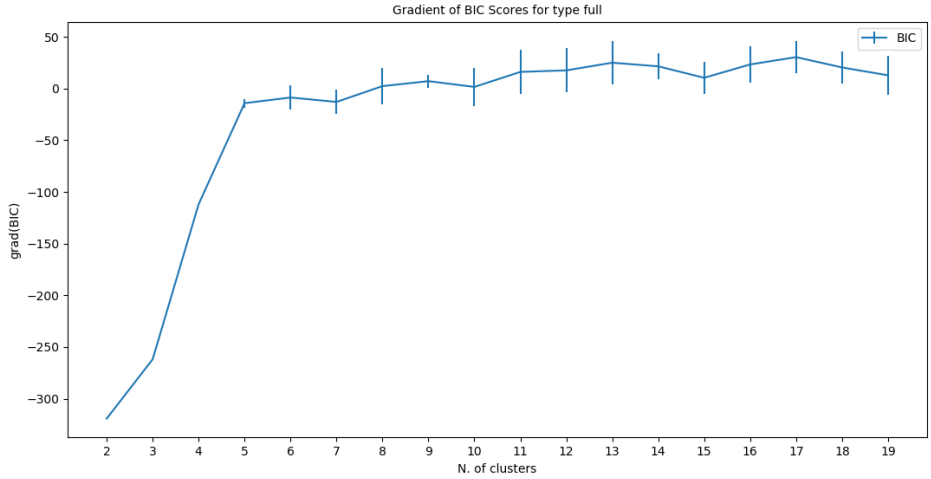

Figure 49: BIC gradient graph used for choosing the number of clusters for the GMM, in the 'full' covariance mode

Based on this graph we chose 4 clusters. In the following, we evaluated the GMM model using this number of clusters.

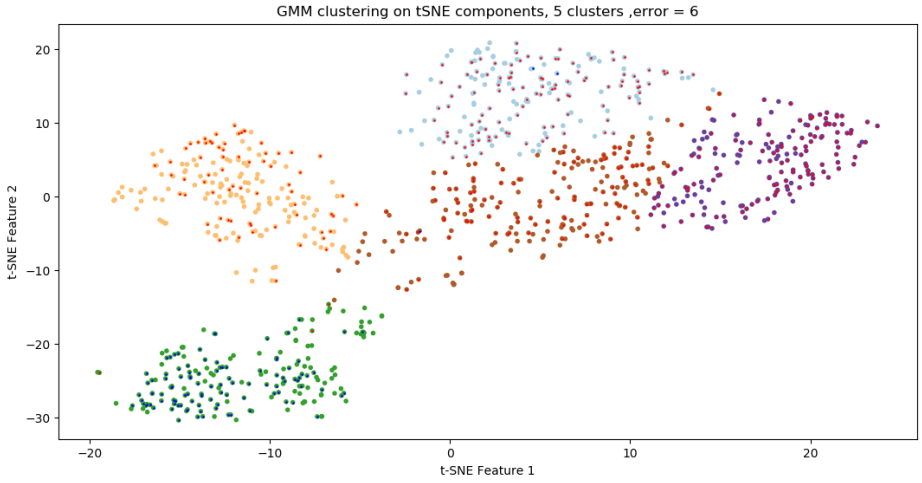

Figure 50: GMM clustering on the tSNE components. The small dots correspond to the true tags. The large dots color stands for the GMM clustering.

Finally, we preform the error analysis as it was made for the GMM on the entire data. The error we got is 6, just as in the GMM analysis on the feature space. From this we can conclude that most of the meaningful information is not loss in the tSNE dimensionality reduction.

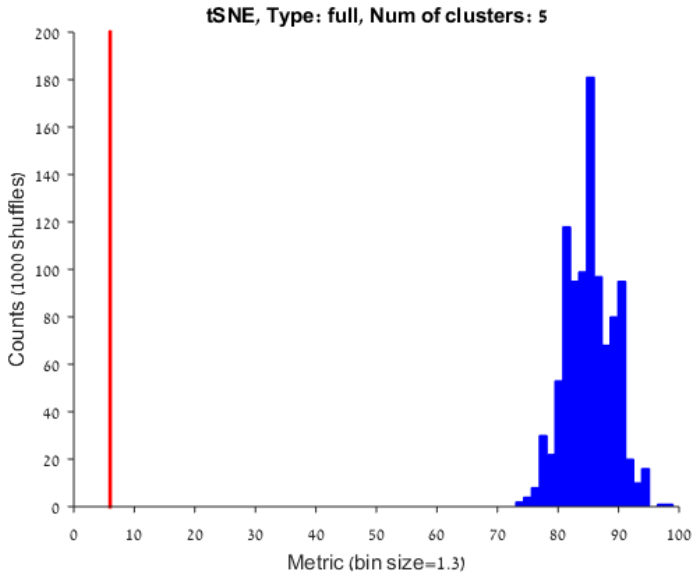

Figure 51: error analysis for the GMM on the tSNE 2D space.

We will state that the tSNE components do not allow projection of new data on them. hence the visualization is possible using the data set, but new data cannot be introduced to this low dimensional space. Visualization on new data can be made using the full scheme each time. Moreover every iteration results with a different visualization due to randomness in the algorithm, hence practically the tSNE algorithm is only useful for visualization.

Discussion

This project follows a similar research that was published in 2013 **[ref].** in that paper only 2 features were extracted, the trough2peak, and the FWHM. On this feature space a GMM was evaluated, and no error analysis was made.

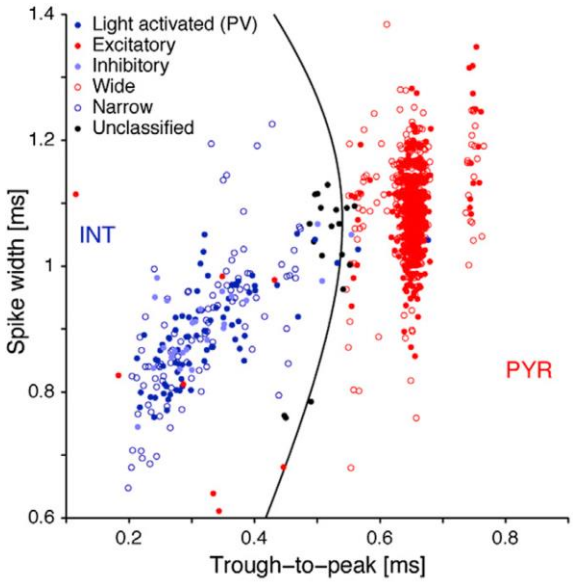

Figure 52: the GMM classification from the original paper.

In our work we added 7 more morphological features, and deemed one of the original feature as redundant. On top of that we added new type of features that addressed the temporal statistics of the spiking units. Those features were designed after inspecting the representative curves we created for each unit, and a redundancy analysis was made.

Based on the features we built a non-linear model in a supervised setting. We chose the model parameters and the features used in the model evaluation via a grid search, and chose the parameters and features that had the best generalization score. Those parameters were fixed and the model intrinsic error was evaluated. The features that we chose include one morphological feature and one temporal feature.

We made a second model in a unsupervised manner, using GMM. This model created homogenous clusters with respect to the tagged data, and based on an error measure we made we found that its error is significantly low with respect to the error NULL distribution. Moreover we were able to split the data to more clusters then there are tags, and made an inner separation inside groups that were previously considered the same.

To conclude the project we will say that we bettered the previous results, created inner separation within the existing classes, and furthered the depth of the morphological and temporal analysis of the spiking units. The separation to inner group can be the used as a basis for future research about new cell types.

Bibliography

- https://www.ncbi.nlm.nih.gov/pubmed/18599766

  Neuronal diversity and temporal dynamics: the unity of hippocampal circuit operations. Klausberger T, Somogyi P.Science. 2008 Jul 4;321(5885):53-7. doi: 10.1126/science.1149381.

- https://www.ncbi.nlm.nih.gov/pubmed/15378039

  Interneurons of the neocortical inhibitory system.Markram ,Toledo-Rodriguez , Wang , Gupta, Silberberg, Wu. Nat Rev Neurosci. 2004 Oct;5(10):793-807.

- https://www.sciencedirect.com/science/article/pii/S0896627313008647

  Inhibition-Induced Theta Resonance in Cortical Circuits
  Stark, Eichler,Roux,,Fujisawa,Rotstein,Buzsáki. Volume 80, Issue 5, 4 December 2013, Pages 1263-1276.

- http://www.jmlr.org/papers/v12/pedregosa11a.html
  Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

- https://www.ncbi.nlm.nih.gov/pubmed/?term=hazan+zugaro
  Hazan L, Zugaro M, Buzsáki G (2006) Klusters, NeuroScope, NDManager: a free software suite for neurophysiological data processing and visualization. J Neurosci Methods, 155:207-16

- https://www.sciencedirect.com/science/article/pii/S0896627315000860.
  Tools for Probing Local Circuits: High-Density Silicon Probes Combined with Optogenetics.Buzsáki,Stark,Berényi,Khodagholy,Kipke,Yoon,Wise. Neuron

- Volume 86, Issue 1, 8 April 2015, Pages 92-105