Spike sorting toolbox for rehabilitation

Author: Guillem Boada[1]
Supervisor: Florian Helmhold[2]

[1] Graduate Training Centre of Neuroscience (GTC), Universität Tübingen
[2] Institute of Medical Psychology and Behavioral Neurobiology, Universität Tübingen

ABSTRACT

Context. Patients that suffered from a stroke or a spinal cord injury often retain a partial or total paralysis, losing their autonomy and their ability to independently perform activities of daily living. This problem can be addressed with neurorehabilitation via Brain-Computer Interface (BCI) systems. While most methods rely on the electroencephalogram (EEG) to record the neural activity due to its accessibility and low cost, multielectrode extracellular recordings seem to increase the performance and widen the possibilities of such treatments thanks to their high spatial resolution. Directly using the threshold crossings of the raw signals is enough to drive a BCI system, however, this ignores the fact that a same electrode may be recording activity from different neurons.
Aims. In this project we aimed to develop a set of methods to select the channels containing neural activity and perform spike sorting on those. The output is aimed at the timestamps of the detected spikes in each useful channel together with a label indicating the inferred source. This information is potentially useful for improving the neurorehabilitation efficiency as well as for investigating the plasticity changes though monitoring of the single neurons activity.
Materials & methods. We proved the efficiency of the following implemented methods by testing them in a dataset consisting of Utah array recordings from the perilesional area in the primary motor cortex of a stroke patient. The expected results are obtained, from which a sample is shown in the Results section. The whole task has been divided in subtasks for a better management and maintenance of the code. Firstly, a convolutional neural network (CNN) approach is used to discard the channels which apparently do not present neural activity but just noise and artifacts. Secondly, two classes are used to perform the spike sorting. One detects and extracts the waveforms, the other labels them through clustering.
Results. The resulting spike sorting for two channels is presented, including the waveforms in the time and principal component space. In the latter, it is possible to observe how the clustering algorithm performed. Moreover, to check the validity of the clusters as single unit sources, we provide the cross- and autocorrelograms of the clusters.
Discussion. On one hand, for one of the channels the algorithm managed to discriminate the two apparent sources and they are likely to consist of neural activity because the autocorrelograms fulfil the characteristic refractory period of neurons. On the other hand, for the other channel the algorithm did not discriminate the multiple origin that can be seen with the naked eye. This is due to the non-Gaussian shape of the clusters and the limited ability of the clustering method to automatically determine the number of sources to be searched. The clusters shapes and their potential origin are hypothesized.

Key words. Spike sorting, brain-computer interface, stroke, multielectrode recording, Utah array, Python toolbox, deep learning, convolutional neural networks, contextual learning, object-oriented programming.

1. Introduction

Patients that suffered from a stroke or a spinal cord injury often retain a partial or total paralysis, losing their autonomy and their ability to independently perform activities of daily living. This problem can be addressed with neurorehabilitation tasks which set a contingent link between the intention and the movement, triggering the plasticity of the perilesional networks (Jackson and Zimmermann, 2012). A reliable way to set this link is via Brain-Computer Interface (BCI) system. While most methods rely on the electroencephalogram (EEG) to record the neural activity due to its accessibility and low cost, multielectrode extracellular recordings seem to increase the performance. They widen the possibilities of such treatments thanks to their high spatial resolution. Directly using the threshold crossings of the raw signals is enough to drive a BCI system, however, this ignores the fact that a same electrode may be recording activity from different neurons, i.e. multiunit activity. Selecting the electrodes which contain neural activity and sorting the detected spikes on those to uncover the firing pattern of individual neurons coding for different aspects of movement intention and performance may be essential in understanding the neural correlates of human motor control and, eventually, use this knowledge to shape the activity and connectivity in order to re-establish specific connections in paralyzed patients. A solution to automatically select channels and sort spikes is needed.

For channel selection, it is today common to select the useful channel by manual inspection of each channel signal. This is nowadays feasible because the array with the most electrodes approved by the FDA and CE marking is the Utah array with 96 electrodes, an amount which can still be dealt manually. However, it is interesting to find methods that could automatize this task to be ready for larger amount of electrodes or for online systems. Regarding spike sorting in a single channel, the trials to automatize their detection and labelling date back to the 80s, when pioneers noticed that each spike has a particular shape which depends on the morphology, physiology and location of the neuron that fires it and managed to sort spikes with purely analogue hardware (Schmidt, 1984). Since the invention of digital computers, a zoo of spike sorting algorithms appeared. Many

of them share a common pipeline. Gibson, Judy, and Marković, 2011 presents an extensive review on spike sorting approaches.

2. Materials & methods

In this section, the analyzed dataset and the set of used tools consisting of algorithms are presented.

2.1. Dataset

The dataset consists of Utah array recordings from the perilesional area in the primary motor cortex (M1) of a stroke patient. Figure 1 shows the Utah array (b) and its location (a). The shown analysis has been performed on signal acquired during a rehabilitation task which mainly involved grasping movement with both arms, the healthy and the affected one.

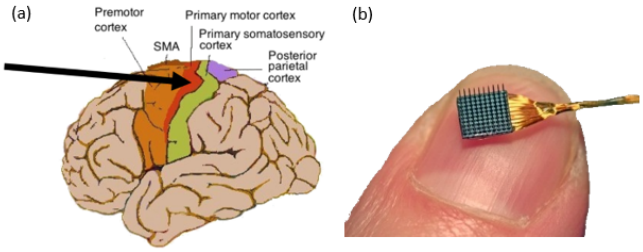


Fig. 1. (a) Brain representation with an arrow indicating the approximate locationof the Utah array in the M1. (b) Utah array. Source: (a) Iamozi. Wikicommons. (b) Maynard, Nordhausen, and Normann, 1997.

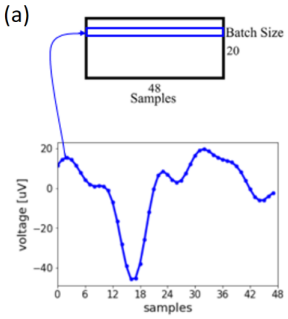
2.2. Algorithms

The first task of automatic channel selection is performed using an implementation of a pretrained convolutional network coined by their authors as SpikeDeeptector (Rehman et al., 2019). Later, we sort spikes using two classes consecutively, the `Waveformer` for waveform extraction from the raw signal, and the `WaveformSorter`, for labelling of the extracted waveforms.

2.2.1. SpikeDeeptector

In multielectrode recordings it is usual that some of the channels do not record any apparent neural activity. This can be due to encapsulation of the electrode by the tissue or that the electrode has been placed in a particularly silent area in the cortex, among many other causes. Nevertheless, these channels also suffer from noise and artifacts which cross the threshold and are detected by the spike sorting methods. Therefore, to avoid unnecessary sorting of non-spike waveforms and their later use in a BCI decoder, the first task we face is selecting the channels containing neural activity. Rehman et al., 2019 proposed and trained a fully connected and a convolutional neural network in various datasets of multielectrode recordings to obtain a method which can universally predict a channel content as either “Neural” or “Artifact”.

Deciding whether a single waveform is a spike or an artifact can be complicated even for expert neuroscientists. For those cases, it is helpful to observe other waveforms that were also captured with the same electrodes. This helps the observer to get a better idea of the nature of the noise affecting the channel and take the decision. In other words, context helps. This idea is the main point of this CNN, which instead of being fed with a single waveform, it uses a stack of waveforms of shape 20x48, as represented in Figure 2 (a). Once the waveforms of a channel are structured in such batches, they are input to the CNN and the overall prediction counts yield the channel label. If there are more batches predicted as “Neural”, the channel is labelled as “Neural” and later fed to the spike sorting pipeline. Oppositely, if there are more batches predicted as "Artifact", the channel is labelled as "Artifact". Moreover, from the proportion of labelled batches, a confidence ratio is provided. An exemplar output can be seen in Figure 2 (b).



(b)

ch	label	confidence [%]
12	Artifacts	100.0
13	Artifacts	100.0
14	Neural	85.71

Fig. 2. (a) Input to the CNN consisting of a batch of 20 vertically stacked waveforms of length 48 samples. (b) Final output of the SpikeDeeptector obtained by feeding the CNN with all the input batches obtained from a specific channel and counting the amount of batches predicted as "Neural" and "Artifact". That proportion is used to compute the confidence ratio. Source: Rehman et al., 2019

.

The details of the SpikeDeeptector functioning can be found in the original paper, it is however worthwhile giving an insight on how the structure of the kernels was essential for the contextual learning and what they learnt. The first layer of the network is a convolutional layer whose filter is of shape 1x3 therefore, this filter does not use elements of different waveforms at the same time and, thus, it strictly learns temporal information from the waveforms. In Figure 3 (a), it is possible to observe that the learnt weights are mainly ascending and descending gradients. Following, the temporal features extracted through the first layer are input to the second layer which is also convolutional, but this time with kernels of shape 20x3, therefore taking 3 temporal features of each waveform at each operation. Here, the waveforms are compared and the context is learnt. In the same way, some kernels are shown in Figure 3 (b). However, as there are so many kernels for this layer, one cannot see with the naked eye what patterns have been learnt.

Regarding the implementation, the original SpikeDeeptector was built and trained using MATLAB (The MathWorks, Inc) with its “Deep learning” and “Neural network” toolboxes. Nevertheless, for the sake of using a single programming language in this project, we decided to do a Python implementation of this work using Keras. The source code is available online[1].

2.2.2. `Waveformer` & `WaveformSorter`

These two classes are an object-oriented programming implementation of the common spike sorting pipeline summarized in the literature (Gibson, Judy, and Marković, 2011) and shown in Figure 4. This has been split in two parts. First part, the `Waveformer`, performs the spike detection and alignment. The arguments used when generating an instance

[1] https://github.com/guillemboada/SpikeDeeptector-in-Python

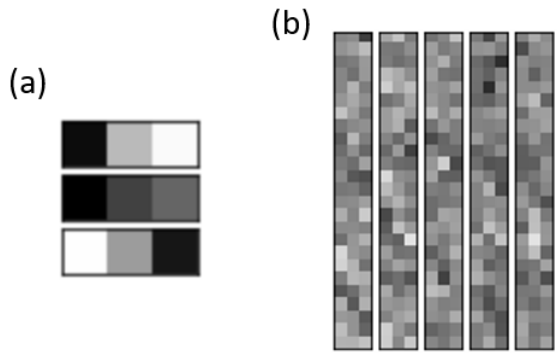


Fig. 3. Representative kernels selected from the first two convolutional layers. Dark and white code for low and high weight values, respectively. The kernels from the first layer (a) have shape 1x3, which makes them extract temporal information from each waveform (rows) independently. The top and middle ones look for ascending gradients and the bottom-most matches descending gradients. The second layer kernels (b) have shape 20x3, taking in this way three temporal features of each waveform in the same convolution operation. This allows for the contextual learning. As there are many of these filters, the learnt features are not clearly seen in the filters weights.

of this class let the user decide over the algorithm parameters: filter type, cutoff frequencies, filter order, emphasizing method, threshold factor, waveform length, and aligning method. Second part, the `WaveformSorter`, clusters the extracted waveforms or waveform features through a mixture of Gaussians with automatic number of clusters selected using the Bayesian Information Criterion (BIC). These classes are compatible and designed to be used with the Scikit-learn architecture `sklearn.pipeline.Pipeline`, building in this way a framework which leaves freedom to select the most appropriate methods for feature selection and dimensionality reduction from the extensive Scikit-learn library and other compatible libraries. Figure 5 exemplifies the use of these classes in the pipeline architecture.

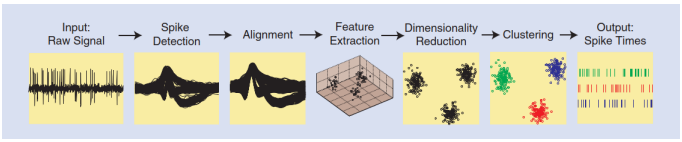


Fig. 4. Common spike sorting pipeline. Starting from the left, a high-pass filtered trace of a multi-unit recording from which spikes are detected. Their waveformes are extracted and aligned. Then, the features from each spike are extracted and they can be represented in the feature space (three-dimensional for easy visualization). Next, the dimensionality of this space is reduced by selecting some of the features and, finally, a clusterer discriminates the different spikes groups. The output are the spike times of each recorded neuron. Source: Gibson, Judy, and Marković, 2011

Again, the source code is available online[2].

3. Results

The resulting spike sorting of several channels through the pipeline shown in Figure 5 is presented, including the waveforms in the time and principal component space. The dimensionality reduction is from 60 samples in the time domain to 3 principal components shown in a pairplot. This already assured that at least an 80% of the variance was explained. In the principal component space is possible to observe how the clustering algorithm performed. Moreover, to check the validity of the clusters as single unit sources, we provide the cross- and autocorrelograms of the clusters. Assuming that the refractory period of a neuron is about 2 milliseconds, we would expect that the two bins on the left and two on the right from the centered zero would be empty. If not, it is because not all waveforms from that cluster come from the same neural source.

In Figure 6 we can observe the algorithm outcome for two channels that recorded signal for about one hour. One in the left column, the other in the right one.

```python
# Build pipeline
pipeline = Pipeline([
    ('waveformer', Waveformer(visualize=False)),
    ('pca', ColumnTransformer([('timestamp', 'passthrough', slice(0,1)),
                               ('pca', PCA(n_components=3), slice(1,None))])),
    ('sorter', WaveformSorter(visualize=False))
])
```

Fig. 5. Use of the `Waveformer` and the `WaveformSorter` in the Scikit-learn architecture `sklearn.pipeline.Pipeline`. This frameworks allows for a free choice of the feature extraction and dimensionality reduction algorithms. In this case, a principal component analysis was chosen.

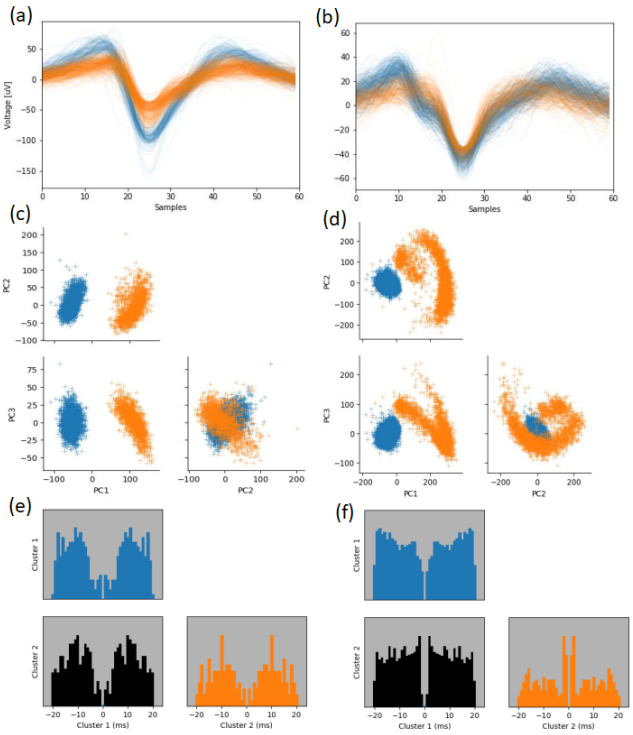


Fig. 6. Results obtained from feeding two channels of about one hour of recording to the spike sorting pipeline. The left column corresponds to a rather simple spike sorting case, while the right column presents a more complex situation. (a-b) Show the filtered waveforms in the time domain, (c-d) in the principal component space, and (e-f) the corresponding auto- and crosscorrelograms of the clusters.

4. Discussion

The left example is a rather simple spike sorting case because the two apparent sources are already easily distinguished in the temporal domain (Figure 6 (a)). However, this is a good example to check that the clustering algorithm performed well labelling

[2] https://github.com/guillemboada/Spikesorting

the waveforms in the principal component space (Figure 6 (c)). Additionally, the autocorrelograms show that particular centered valley that we would expect from single unit cluster, although not perfected because there are some counts in the -2, -1, +1 and +2 milliseconds bins (Figure 6 (d)). But this may be due to some artifacts being classified in the same cluster as the neural source.

The right example shows a more complex case. In the temporal space (Figure 6 (b)), the distinction between the apparent waveform sources is not clear. Then, in the principal component space (Figure 6 (d)), we realize that the waveform clouds do not have a typical Gaussian shape but the rightmost cloud is rather elongated and curved. Notice that the orange source does not only include one cloud but rather two or three, depending on the observer. This indicates some flaws in the algorithm: (1) the clustering in the `WaveformSorter` is only able to model the data as Gaussians, hence not capturing the real data structure in some cases; and (2), the automatic selection of the number of clusters is too limited, labelling for only two clusters in cases in which there are more than two.

We propose to solve (1) by investigating the cause of such elongated shapes. One could track the data over time and discover if and how the spiking features of a specific neuron have changed during the recording. Maybe it is due to a constant change in the neurons neurophysiology which causes the waveforms to change in time; or maybe is due to burst spiking, in which the earlier spikes will stronger than the later ones. Additionally, more complex clustering algorithms could capture them by making use of their generation nature, e.g. valley seeking clustering (Zhang et al., 2007), density-based spatial clustering of applications with noise (DBSACN) (Ester et al., 1996), spectral clustering (Ng, Jordan, and Weiss, 2001). For the number of cluster flaw (2), we propose still using the BIC to select it but in a different way which does not limit that much the amount of searched clusters. Other metrics as the silhouette score may also be helpful.

We propose to solve (1) by either using a more complex clustering algorithm and investigating the cause of such elongated shapes. Maybe it is due to a constant change in the neurons neurophysiology which causes the waveforms to change in time; or maybe is due to burst spiking, in which the earlier spikes will stronger than the later ones. For the number of cluster flaw (2), we propose still using the BIC to select it but in a different way which does not limit that much the amount of searched clusters. Other metrics as the silhouette score may also be helpful.

Acknowledgements. This work has been possible thanks to the RamosLab team at the Institute of Medical Psychology and Behavioral Neurobiology which gave us an necessary support and the opportunity to work with such valuable dataset.

References

Ester, Martin et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In: pp. 226–231. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220.

Gibson, Sarah, Jack W. Judy, and Dejan Marković (2011). "Spike sorting: The first step in decoding the brain". In: *IEEE Signal Processing Magazine* 29.1, pp. 124–143. ISSN: 10535888. DOI: 10.1109/MSP.2011.941880.

Jackson, Andrew and Jonas B. Zimmermann (2012). *Neural interfaces for the brain and spinal cord - Restoring motor function*. DOI: 10.1038/nrneurol.2012.219.

Maynard, Edwin M., Craig T. Nordhausen, and Richard A. Normann (1997). "The Utah Intracortical Electrode Array: A recording structure for potential brain-computer interfaces". In: *Electroencephalography and Clinical Neurophysiology* 102.3, pp. 228–239. ISSN: 00134694. DOI: 10.1016/S0013-4694(96)95176-0.

Ng, Andrew Y., Micahel I. Jordan, and Yair Weiss (2001). "On spectral clustering: analysis and an algorithm". In: *NIPS*.

Rehman, Muhammad Saif-ur et al. (2019). "SpikeDeeptector: a deep-learning based method for detection of neural spiking activity". In: *Journal of Neural Engineering* 16. DOI: 10.1088/1741-2552/ab1e63. URL: https://doi.org/10.1088/1741-2552/ab1e63.

Schmidt, Edward M. (1984). "Computer separation of multi-unit neuroelectric data: a review". In: *Journal of Neuroscience Methods* 12.2, pp. 95–111. ISSN: 01650270. DOI: 10.1016/0165-0270(84)90009-8.

Zhang, Chaolin et al. (2007). "Neighbor number, valley seeking and clustering". In: *Pattern Recognition Letters* 28.2, pp. 173–180. ISSN: 01678655. DOI: 10.1016/j.patrec.2006.07.003.