Wave clus

Automatic spike detection and sorting using wavelets and super-paramagnetic clustering.

Overview:

A large amount of research in neurophysiology is based on the analysis of extracellular potentials recorded with microwires that capture the action potentials (spikes) of neurons in their surroundings. For many applications it is crucial to know which spike correspond to which neuron, namely -spike sorting- and since recent acquisition systems allows the simultaneous recording of hundreds of channels, it is also important to do this automatically (or semi-automatically) and fast. Wave_clus is a fast and unsupervised algorithm for spike detection and sorting. Although it gives a first unsupervised solution, this can be further modified according to the experimenters' preference (semi-automatic sorting).

Distribution and reference:

Wave_clus is free (and therefore without any warranty) for any non commercial applications. For any commercial application please contact the author (rodri@vis.caltech.edu). You can refer to this algorithm just by citing the paper where it is described:

Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering.
R. Quian Quiroga, Z. Nadasdy and Y. Ben-Shaul
Neural Computation 16, 1661-1687; 2004.

Requirements:

Wave_clus runs under windows or lynux and requires **matlab 6.5 (R13)** or higher. It uses the function cluster.exe (or cluster_linux.exe for lynux), provided by Eytan Domani, which is an executable that does the super-paramagnetic clustering of the data. Two matlab toolboxes are also necessary: the signal processing toolbox for band-pass filtering the continuous data, and the wavelet toolbox for extracting the spike features.

Brief description of the code and parameters:

Each DataType has an associated **set_parameters** file in wave_clus/Parameters_files, or at the beginning of the batch files. Parameters that can be set by the user appear in red.

i) Spike detection: it first filters the continuous data between fmin and fmax (as set in the parameters file). Then, it calculates an optimal amplitude threshold, which is set as stdmin times the estimated standard deviation of the noise (see the paper for details and advantages of this estimation). It also calculates a maximum threshold, stdmax, to avoid high amplitude artifacts. The parameter detection sets the threshold to be either positive, negative, or both. For each spike, w_pre datapoints before and w_post datapoints after the spike peak are stored. Spike alignment is done after interpolation of the spike shape with cubic splines. The interpolation is set with the parameters interpolation and int_factor. To avoid double detections, spikes should be separated at least a certain number of datapoints, given by ref. The sampling rate is set with the parameter sr. The parameter segments is for reading the data in pieces in order to avoid memory overloads. The code reads tmax seconds of data starting from tmin.

ii) Feature extraction: it uses a selection of wavelet coefficients chosen with a Kolmogorov Smirnov test of Normality. The parameter inputs sets the total number of coefficients and scales is the number of scales for the wavelet decomposition. Alternatively, the code allows the use of the first 3 principal components as the spike features instead of wavelets with the parameter features (see the paper for comparisons of the different features). In some cases this will be the first step of the analysis, since many systems already give the detected spikes and not the continuous data.

iii) Clustering: uses super-paramagnetic clustering (SPC), an automatic clustering algorithm based on ideas from statistical mechanics developed by the group of Eytan Domani. See the paper for a description of its advantages for spike sorting, and papers by the group of Eytan Domani for other applications. The main variable to change with super-paramagnetic clustering is the 'temperature' (see paper). At low temperatures, all spikes will be assigned to a single cluster and at the high temperature limit; each spike will form a single cluster. The optimal temperature for clustering lies in between this two extremes (in the super-paramagnetic regime). In practice, the optimal temperature is set as the largest temperature for which a cluster with at least min_clus members appears. The code actually calculates results for all temperatures between mintemp and maxtemp in steps of tempstep. Changing the temperature is usually the main supervised correction done after clustering, but since results are already calculated for all temperatures in this range, it only implies loading and saving a different set of results.

iv) Force membership: In many cases, points far from any cluster or in between two clusters are not assigned to any cluster. The function force_membership, (button Force in the GUI) will assign these points to the closest cluster unless they are too far. The algorithms in the Force files subdir were developed in collaboration with Casimir Wierzynski at Caltech.

v) Template matching: Wave_clus also has the option of doing template matching after max_spk number of spikes, if the option match is set to 'y'. That means that wave_clus will do SPC clustering for the first max_spk spikes and the rest of the spikes (if any) will be assigned via template matching. The principle of template matching is the same as the one used for

Force_membership. The spikes within max_spk define the templates and the remaining spikes should be assigned with the Force button. Note that template matching can save a lot of time if there are too many spikes to be clustered. In particular, SPC starts taking too long for more than ~30000 spikes.

vi) Fix buttons: This option is for fixing a template before changing the temperature of the force button. It is useful if a selection of clusters at different temperatures is needed. For example you may want to keep, say, cluster 2 at temperature 5 and clusters 1 and 2 at temperature 10. Then you click on the fix button for cluster 2 at temperature 5 and then change the temperature to 10. The cluster you fixed will appear as a new cluster 3 at temperature 10. You may also don't want to use the force option for all clusters. Then force all clusters first, fix those you want to keep with the forced option and click the force button again.

Data Input and Output:

Data input is according to each of the DataType options. In each case, the parameters used are set in the corresponding set_parameters file in the subdir wave_clus/Parameters_files. The CSC and Sc options are for data collected with Neuralynx systems. An exemplary file is given in wave_clus/Sample_data/UCLA_data/CSC4.Ncs. The output of wave_clus (obtained either using the Save clusters button in the GUI or the one given automatically by the batch files) is times_[filename].mat , which is a matlab file containing the following variables: par (parameters used for clustering), spikes (a matrix with the spike shapes), inspk (a matrix with the features of the spike shapes) and cluster_class (a matrix with the clustering results). The variable cluster_class has 2 columns and *nspk* rows (*nspk* is the number of spikes). The first column is the cluster class, with integers denoting the clusters membership and a value of 0 for those spikes not assigned to any cluster. The second column are the spike times in ms.

Getting started:

i) Copy the wave_clus.zip file into your hard drive and unzip it. This will create a wave_clus directory with corresponding subdirs. If you want also to get the simulated data files (the ones used in the Neural Computation paper), download the file Simulator.zip in the wave_clus/Sample_data subdirectory and unzip it there. This will create the subdir Simulator with all the files inside it. If you want to get the real human recording, download UCLA_data.zip in wave_clus/Sample_data and unzip it there. This will create the subdir UCLA_data with a 30' recording inside it. Delete the .zip files once uncompressed. In matlab go to the menu File/Set Path and add the directory wave_clus with subfolders to the matlab path. Copy the font in the subdir wave_clus/wave_clus_font into your windows/fonts/ directory. Now you are ready to use wave_clus.

ii) If you are using Linux, you should set the parameter system to linux in the corresponding set_parameters file and in the Do_clustering batch file.

iii) For starting the GUI just type wave_clus in your matlab command line. You can start playing with the simulated data used in the Neural Computation paper by choosing the option Simulator in the DataType menu. Now you can load the simulated data (with the Load Data button) from the subdir wave_clus/Sample_Data/Simulator and compare the results with ones of the paper. You won't necessarily get exactly the same results, since the SPC clustering is an statistical (i.e. not deterministic) method. Moreover, if you run the same clustering twice you can get slightly different results (but in general is very robust). Check the file set_parameters_simulation in the subdir wave_clus/Parameters_files for the parameters settings for these data.

iv) The file CSC4 is an example of a 30' multiunit recording from a human epileptic patient. This data was collected at the lab of Itzhak Fried at UCLA, using a Neuralynx system (Tucson, Arizona). If you have a Neuralynx system, you can directly use the DataType options CSC, CSC (pre-clustered), Sc and Sc (pre-clustered) with your data. To cluster this data use the option CSC in DataType and load the data. Alternatively, you can load already clustered data with the option CSC (pre-clustered) in DataType. This is one nice feature of wave_clus, that you can save all possible clustering options and then you just restore different options without recalculating everything (see how to use the batch files below). If you use the option Plot_average instead of Plot_all, results are restored much faster since plotting all the spikes may take too long. It may well happen that there is not a unique temperature for getting all the clusters (e.g. 1 cluster appears at temp=6 and another one at temp=7). If this is the case, you will have to save the results separately and then combine them in matlab by changing the 1st column of the variable cluster_class in the times_[filename] output file.

v) The ASCII options in the DataType menu are for loading matlab files. These are the ones that you will most likely be using for your data. The option ASCII if for continuous data, the option ASCII spikes is for clustering spike shapes that have already been detected (e.g. detected on-line by the acquisition system). In both cases there is also an option (pre-clustered) to load results previously calculated with the batch files (the GUI saves the output but doesn't save the clustering results for all temperatures). The input data should be a matlab file (extension .mat). It should be either a vector named data for ASCII files or a matrix named spikes (nr. of spikes x length(spike shapes)) plus a vector index with the spike times in ms for the ASCII spikes option. Before starting don't forget to set the proper sampling rate sr in the file set_parameters_ascii. In the subdir wave_clus/Sample_data there is a test.mat file with a short segment of simulated data that can be loaded using the ASCII option and test1_spikes.mat that can be loaded with the ASCII spikes option.

Batch files:

Since wave_clus gives an unsupervised clustering of the data, it is possible to run batch processes that will go through several files and save the results. You can later restore the results in the GUI and change them, if needed, loading the data with the (pre-clustered) option. The nice thing is that you don't need to calculate everything again. Wave_clus saves the results for all possible temperatures, so, changing the temperature (which is the main parameter to be changed, if needed) means just loading another set of results. There are mainly 2 batch files: Get_spikes (which does the spike detection) and Do_clustering (which does the spike sorting). Get_spikes reads the continuous data and outputs a file [filename]_spikes with the variables index (the spike times in ms) and spikes with the spike_shapes. If you already have the spike shapes, you don't have to run Get_spikes, but you should have the same structure as the one saved by Get_spikes. Do_clustering inputs the [filename]_spikes files and does the clustering automatically. The results are saved in the file times_[filename].mat, which has the same structure as the one saved with the GUI. Although it will save and print results with the automatically chosen temperature, it stores results for all temperatures, so that it can be easily changed later with the GUI. Both Do_clustering and Get_spikes go through all the files specified in files.txt.

Comments and updates:

If you have any comments please send them to me at rodri@vis.caltech.edu. I really hope this algorithm will be useful for you. If it does, or if for some reason is not adequate for your data please let me know. I can't promise that I can introduce suggestions immediately, but I'll try to do it in a reasonable time. Also let me know by email if you want to keep updated on the release of any new version, related paper, etc.

FAQs:

- What should I change if I don't like the automatic results?
The main parameter to change is the temperature. You can also use the force button to add some of the non-clustered spikes to the already defined clusters.

- How can I run wave_clus faster?
There are few tricks to run faster. First, if you have too many spikes, you can do clustering on a first segment that will define the clusters, and assign the remaining spikes via template matching (the templates being the clusters defined in the first segment). This is set with the parameter max_spk. You can also use the option plot_average instead of plot_all, since plotting all spike shapes may take too long. If you just want to check a short segment of the file you can do this using the parameter tmax.

- How can I avoid high temperatures?
Sometimes wave_clus chooses high temperatures that tend to overcluster the data. To avoid picking up high temperatures, you can set the parameter min_clus to a larger number.

- What are the Accept / Reject / Multiunit buttons?
Accept saves the clusters by assigning a corresponding integer (say 1 for the first cluster, etc.). Reject gives the cluster a value of 0; i.e. it will merge it with the non-clustered spikes. Multiunit gives the cluster a value of -1.

- What are the numbers appearing at the top and bottom of each cluster?
The top number is the number of spikes for the cluster. The bottom one is the number of spikes with an inter-spike-interval (ISI) of less than 3ms. A relatively large number of spikes within 3ms ISI is a sign of a multiunit cluster.

- What happens if I have more than 3 clusters?
In this case as many extra figures as necessary will be generated (starting from Figure 10).

- Why the spike features do not show the clusters well separated?
Wave_clus is plotting only the first 2 wavelet coefficients. It may well happen that other coefficients will separate the clusters even better. You can plot other coefficients manually using the variable inspk saved in the times_[filename] output file.

- Can I edit the plots?
Yes, you can edit, zoom, copy and paste, etc each plot using the Tools menu at the top of the GUI.

- What is the set_parameters button?
So far it has no function… Sorry.

- I get a memory overflow error when loading the data, what can I do?
Just use more segments for loading the data in smaller pieces.