Large Scale Spike Sorting With Electrical Artifact
User's guide.

Gonzalo Mena, 03/2016

Main features:

- Allows you to do spike sorting for arbitrarily many cells given a stimulation pattern and Ei's of targeted cells.
- Also (optional) it can detect the onset of axonal bundle artifact (it is not actually the onset, but a measure that detects changes in the 'variance' regime across electrodes, which turns out to be a sensible proxy for the onset of axonal bundle artifact).
- If used, this 'proxy' bundle onset information can be used to: 1)terminate execution (leading to computational savings) or 2)change the artifact model, leading to improved accuracy (although there will still probably be false positives due to axonal bundle activity).
- Also, (optional) it can use information from the stimulating electrode. The current version for this is still quite unstable, but the goal would be to include it, as it can be helpful for somatic or near somatic stimulation, where the stimulating electrode also captures the strongest Ei signal (for distal stimulation don't using the stimulating electrode won't hurt; actually it is a sensible choice).
- I have tested it in ~700 patterns and accuracy is between 97-98 percent but it increases to over 99 percent (and only ~5 false positives) if we consider information only up to the onset of axonal bundle (or it's proxy). Fortunately, in that case, one is still able to detect activation of some neurons, although to a much limited extent.

How doest it work? (read along with examples)

- Apart from pattern and neurons It is fed with a preparation folder and some data00x indexes to look for movie files. Also, a path to a .ei file must be provided (alternatively, one could use elecResp files but that would require changing the code).
- Given a pattern and many cells the algorithm:
    - First, initializes some global hyperparameters that inform about the 'lengthscales', nonstationarities and concentration of variation in the system. That is done in *InitializeArray*, which estimates those parameters using any available pattern (you can decide one for yourself, and may not correspond to the one you want to do spike sorting).
    - Then, it does spike sorting based on those computed 'hyperparameters'.
- Core of the Algorithm explained in a nutshell.
    - it is based on a spatial model for the artifact over the entire array; in other words, it 'borrows' statistical strength from the shared spatial information, thus enhancing estimation.
    - The main underlying machine learning concept is 'scalable gaussian process' (GP): artifact is conceived as a GP that depends on the aforementioned

hyperparameters. With this framework we can use the underlying GP to do filtering and extrapolation operations.
    - Indeed, the algorithm relies heavily on such extrapolation as it swipes forward through the increasing stimulus amplitudes and for each of such amplitudes: : first, it constructs an estimate (extrapolation) of the artifact based on the artifact estimated at previous stimuli. Second, given this estimate, it seeks to place spikes if they lead to a better explanation of data. Third, with this placement of spikes, it constructs a new artifact estimate. Fourth, it goes to the second point repeating until no further changes in spikes are achieved.
- Data structures: **params** contains all required parameters and 'local' and temporal information. After execution, results are saved in the structure **Output**

- Very important: initialization is relatively slow (up to ~10 minutes) but it has to be done **ONLY ONCE PER PREPARATION**. That means, if you want to do spike sorting for many patterns in the same preparation, just initialize once, hopefully using a pattern as nice as possible. A nice pattern is one that doesn't have too much bundle, and where the stimulating electrode is close to the center of the array. If you have TTX measurements of the same preparation, USE THEM!. If you want to waste time thinking in a good pattern, just leave in blank and the algorithm will chose one 'at random', which in the worst case won't be that bad.

Limitations:

- It assumes movie files are sorted increasingly with stimulus amplitude, and that there are no repetitions (no two movie files with same stimulus amplitude). It should be easy to get rid of this constraint, by tweaking the *loadTracesArt* function (talk to me if you want to do it), although this might imply a slowdown, as it would require to load movies twice, and loading data is usually a computational bottleneck.
- There can be at most one stimulating electrode. If necessary, I can try to extend this framework to account for multiple electrodes, hopefully it won't break the nice structure that makes this scalable.
- It assumes data comes from the usual MEA with 512 arrays. However, it is possible to extend to others, but that would require some work.
- I have mainly used Lauren's stimuli, but have also played a little bit with Sasi's stimuli. I don't know to which extent the artifact model assumed in this method would still apply for such new stimuli, so I cannot assure good results to Sasi's stimuli as well (also, because there isn't ground truth). But it is certainly a question that has to be addressed by the very use of the method with Sasi's data.
- Be careful with weak Ei signals. By default, the spike detection is based only on electrodes with maximum EI amplitude >30 daqs (you can change that default), but if neurons are such that the available electrode with the maximum available signal is close

to that threshold, then it is possible that problems will arise: that will be typically indicated by very noisy spike activation curves, where essentially spike detection becomes coin flipping.

- Be careful also with many neurons! Mostly if their Ei's overlap (equivalently, they are hard to distinguish) due to closeness and/or if the Ei signals are weak (as above). In those cases spike sorting may end up in loops, where spikes are alternatively assigned to one neuron or the other. That will be indicated in the Log, which contains information about how many iterations (estimation of artifact followed by estimation of spikes) were carried about for each stimulus. If that number exceeds the maximum default number of iterations that should be taken as an indication of loops (and warnings will be recorded to Log.messages). Probably, a closer look to this situation would be necessary to improve accuracy in these 'hard' cases, so we recommend to start by looking at more single situations, and then pin down the hard cases and look at the actual causes of this misbehavior.
- Also, a big limitation: we are assuming Ei's contains true information, which is not necessarily true. So here we 'cross' our fingers that our Ei estimates are close enough to the truth, but if not, the algorithm is helpless and accounting for that would require major changes. Notice that this misspecification can add up to the above problems (for example imagine two neurons overlapping whose Ei's are very different but according to our estimated Ei's they are very similar. Then, problems will arise.)
- Finally: It would be possible to use Karthik' axonal bundle detection methods instead, if they lead to better performance in some aspect. That should be straightforward, but still requiring some coding