Illustration for Combining Sorting Results from Multiple Sorters

Shangmin Guo

November 14, 2019

In this document, I would give a brief illustration of the current derivation and implementation of how I combine different sorting results from different sorters. The Github repo of this algorithm is https://github.com/Shawn-Guo-CN/spikecombine.

1 Derivation of the Algorithm

1.1 Preliminary

Assume that we first have the following things:

- K sorters, i.e. $\{s_1, s_2, \ldots, s_K\}$;
- $N^{(k)}$ units from with their metric values $\mathbf{m}_i^{(k)} = \left\{m_{i,1}^{(k)}, m_{i,2}^{(k)}, \ldots, m_{i,D}^{(k)}\right\}$, $\forall i \in \{1, 2, \ldots, N^{(k)}\}$ for every sorting s_k, where D is the number of metrics we have.

Then, we task is: for a specific sorter s_k, decide whether unit $u_i^{(k)}$ it detected is a true positive (TP) unit (i.e. $u_i^{(k)} = 1$) or false positive (FP) one (i.e. $u_i^{(k)} = 0$), based on their corresponding metric values. As we would only curate the sortings from one sorter, from then on, I will ignore the serial number (k) of different sorters.

1.2 Derivation

Based on the Bayesian theorem and the task, I could derive the posterior probability of some unit u_i being a TP as follow:

$$
\begin{aligned}
p(u_i = 1|\mathbf{m}_i) &= \sum_{k=1}^{K} p(u_i = 1, s_k|\mathbf{m}_i) \\
&= \sum_{k=1}^{K} \frac{p(u_i = 1, s_k, \mathbf{m}_i)}{\sum_{k'=1}^{K}\sum_{i'=1}^{N^{(k')}} p(u_{i'}, s_{k'}, \mathbf{m}_{i'})} \\
&\propto p(u_i = 1, s_k, \mathbf{m}_i) \\
&= p(\mathbf{m}_i|u_i = 1, s_k)p(u_i = 1|s_k)p(s_k)
\end{aligned} \tag{1}
$$

Similarly, we could have:

$$
p(u_i = 0|\mathbf{m}_i) \propto p(\mathbf{m}_i|u_i = 0, s_k)p(u_i = 0|s_k)p(s_k) \tag{2}
$$

Note: $p(u_i|s_k)$ is a delta distribution whose value is decided by *spikecomparison.compare_two_sorters()* . So, it doesn't have a probabilistic explanation.

Then, take $p(u_i = 1|\mathbf{m}_i)$ for example, what we need to estimate is only $p(\mathbf{m}_i|u_i = 1, s_k)$, i.e. the posterior distribution of metric values on TP units from sorter s_k. As for the prior distribution $p(s_k)$, we can set it to uniform. And $p(u_i = 1|s_k)$ could be estimated by: i) the agreement score between s_k and ground-truth during learning; ii) the agreement score between s_k and other sorter $s_{k'}$ during inference.

2 Implementation of the Algorithm

2.1 Hierarchy of Classes

In the project, all the classes and their brief descriptions are given as follows:

- **SpikeSortersCombiner**, the class implementing the algorithm with a function *fit* for fitting $p(\mathbf{m}_i|u_i = 1, s_k)$ and a function *predict* for performing the inference in Equation(1) and Equation (2).
- **DataGenerator**, the class is to load a folder containing lots of *MEArecRecordingExtractor* in .h format, and (more importantly) generate a folder that is specifically organised to be taken as the input for *DataLoader*.
- **DataLoader**, the class is to take the folder generated by *DataGenerator* as the input, and provide *SpikeSortersCombiner* APIs to get access the *sortings* from different sorters with the corresponding *ground-truth sorting* and *original recording*.

2.2 Descriptions of Key Steps

Here, I did not provide pseudocodes for the key steps, as the codes themselves are all very straightforward.

2.2.1 For Fitting

The steps of fitting parameters of $p(\mathbf{m}_i|u_i = 1, s_k)$ of **every sorter** s_k are given as follows:

1. by comparing with the corresponding ground-truth sorting, classify the units to TP U_k^{TP} ones and FP ones U_k^{FP} for every sorter s_k, based on the agreement scores between s_k and ground-truth;

2. estimate the mean and covariance matrix on: i) U_k^{TP}, which gives μ_k^{TP} and σ_k^{TP}; ii) U_k^{FP}, which gives μ_k^{FP} and σ_k^{FP}.

2.2.2 For Inference

The steps to do inference on sortings from s_k are exactly following Equation (1) and (2):

1. compare s_k with sortings from all other different sorters, and store the agreement scores between them;

2. calculate the metric $\mathbf{m}_i^{(k)}$ for every detected unit $u_i^{(k)}$;

3. for every $u_i^{(k)}$, calculate $p(u_i = 1|\mathbf{m}_i)$ and $p(u_i = 1|\mathbf{m}_i)$ following Equation (1) and (2) respectively;

4. if $p(u_i = 1|\mathbf{m}_i) < p(u_i = 1|\mathbf{m}_i)$, then exclude $u_i^{(k)}$ from the output of s_k.