A simple algorithm for averaging spike trains.

Conor Houghton

Mathematical Neuroscience Laboratory
School of Mathematics
Trinity College Dublin

Bristol, 9 September 2011
A simple algorithm for averaging spike trains.

Motivation.

The zebra finch.
A simple algorithm for averaging spike trains.

Motivation.

Spike trains.
A simple algorithm for averaging spike trains.

Motivation.

The STRF/aEIF model: summary.
A simple algorithm for averaging spike trains.

Motivation.

Coding.

‘what the . . .?’

‘well now that’s clear!’
A simple algorithm for averaging spike trains.

The space of spike trains.

Averaging points.

\[
\begin{align*}
(0,1) & : \frac{0 + 0 + 2}{3} = \frac{2}{3} \\
(0,0) & : \frac{0 + 1 + 1}{3} = \frac{2}{3} \\
(2,1) & : \frac{0 + 0 + 2}{3} = \frac{2}{3}
\end{align*}
\]
Spike trains don’t have coördinates, they do have a metric.
A simple algorithm for averaging spike trains.

The space of spike trains.

A non-Euclidean metric: Metrics in towns.
A simple algorithm for averaging spike trains.

The van Rossum metric.

- A spike train is a list of spike times.
  \[ u = \{ u_1, u_2, \dots, u_m \} \]

- Map spike trains to functions of \( t \)
  \[ u \mapsto f(t; u) = \sum_{i=1}^{m} h(t - u_i) \]

- \( h(t) \) is a kernel, here, it is a causal exponential function
  \[ h(t) = \begin{cases} 
  \exp(-t/\tau) & t > 0 \\
  0 & t \leq 0 
  \end{cases} \]

- Now
  \[ d(u, v) = \sqrt{\int dt [f(t; u) - f(t; v)]^2} \]
The van Rossum metric.

Two steps

- Maps from spike trains to functions using a filter.
- Use the metric on the space of functions.
A simple algorithm for averaging spike trains.

The van Rossum metric.
A simple algorithm for averaging spike trains.

The van Rossum metric.

Mediod?

If we have a distance we have a mediod!
A simple algorithm for averaging spike trains.

The van Rossum metric.

Mediod?

Doesn’t seem to work so well for spike trains!

Spike trains ‘sort of’ live in a high dimensional space so there is unlikely to be a point near the center.
Why not copy the spirit of the van Rossum metric and filter the spike trains first and then do the averaging in the function space?
A simple algorithm for averaging spike trains.

That is . . .

Given spike trains

\[ u_a = \{u_{a1}, u_{a2}, \ldots, u_{am_a}\} \]

then

\[ \{u_1, u_2, \ldots, u_n\} \]
\[ \downarrow \]
\[ \{f(t, u_1), f(t, u_2), \ldots, f(t, u_n)\} \]
\[ \downarrow \]
\[ \bar{f}(t) = \frac{1}{n} \sum_{a=1}^{n} f(t, u_a) \]
A simple algorithm for averaging spike trains.

Averaging algorithm
A simple algorithm for averaging spike trains.

Construct average spike train.

Finally, find the spike train that best accounts for the average function.

Find $\bar{u}$ that minimizes

$$\mathcal{E}(\bar{u}) = \int [\bar{f}(t) - f(t; \bar{u})]^2 dt$$
A simple algorithm for averaging spike trains.

Averaging algorithm

\[ \bar{f}(t) = \sum_{i=1}^{m} h(t - u_i) = f(t; \bar{u}) \]
A simple algorithm for averaging spike trains.

The greedy algorithm.

The greedy algorithm.

The minimization itself is done with the greedy algorithm.

\[ 14c = \]
A simple algorithm for averaging spike trains.

The greedy algorithm.

The greedy algorithm doesn’t always work . . .

14 pence =

. . . but it is often a good approximate answer.
A simple algorithm for averaging spike trains.

The greedy algorithm.

The greedy algorithm and averaging.

Add successive spikes to $\bar{u}$ so that each new spike reduces $\mathcal{E}$ as much as possible.

Say spikes $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{p-1}$ have already been added to $\bar{u}$ then adding a spike $\bar{u}_p$ changes the error by

$$
\delta \mathcal{E} = \frac{1}{nt} \sum_{a,i} e^{-|u_{ai} - \bar{u}_p|/\tau} - \frac{1}{\tau} \sum_{j=1}^{p-1} e^{-|\bar{u}_j - \bar{u}_p|/\tau}.
$$

This analytic formula can be minimized using for example golden section minimization. On a technical note, continuing until the number of spikes is correct works better than stopping when $\mathcal{E} > 0$. 
A simple algorithm for averaging spike trains.

Results.

Seems to work!
A simple algorithm for averaging spike trains.

Results.

Seems to work!
A simple algorithm for averaging spike trains.

---

Results.

Clustering test.

The averaging algorithm has been tested using the very large Zebra Finch dataset made available to the Collaborative Research in Computational Neuroscience database by the Frederic Theunissen lab at UC Berkeley. This includes 450 sets of spike trains.

|                | average | all $k = -2$ | all $k = 1$ | mediod |
|----------------|---------|--------------|--------------|--------|
| average $h$    | 0.53    | 0.49         | 0.43         | 0.37   |
| better than average | n/a    | 0.16         | 0.01         | 0.03   |
| fraction correct | 0.53    | 0.43         | 0.39         | 0.37   |
Conclusions.

- Seems to work!
- Need to test on other data!
- Need to consider ‘spike-train-ness’ of the average spike train!
- What about other maps to functions?
- Is this process represented in biology?