The Wohlgemuth-Ronacher Spike Distance

Robert Mill – v0.1

Introduction

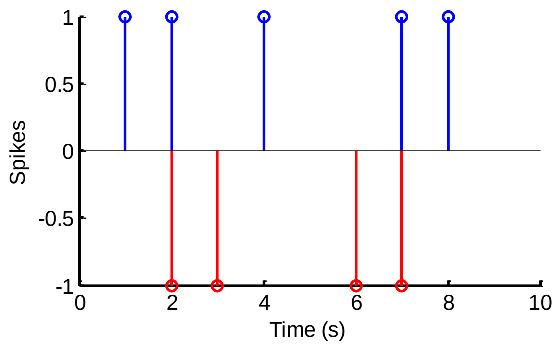
The Wohlgemuth-Ronacher (W-R) spike distance metric[1] provide a means to measure the "distance" d between a pair of spike trains. Because the d is a metric, the following common-sense ideas about distances carry over:

- $d(S_1, S_2) \geq 0$ – all distances are positive, with equality only when $S_1 = S_2$
- $d(S_1, S_2) = d(S_2, S_1)$ – distances are symmetric
- $d(S_1, S_3) \leq d(S_1, S_2) + d(S_2, S_3)$ – a "triangle" inequality

Informally, the W-R metric is calculated as follows. Assume we have two sets of spike times $s_1, s_2, \dots, s_{N_S} \in T$ and $t_1, t_2, \dots, t_{N_T} \in T$. As an example, let's take:

$$S = \{1,2,4,7,8\}$$
$$T = \{2,3,6,7\}$$

Then, we can plot the spikes (pretty literally) in the following way (using `wr_eg1`).



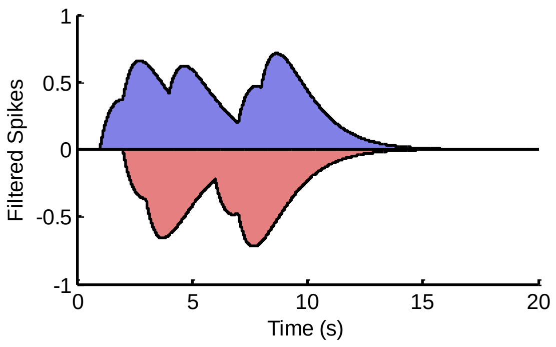
We could, from here, simply count exact spike coincidences (a so-called coincidence metric) but this is obviously too sensitive. If we perturbed one coincident spike by the smallest amount, the metric changes. In practice, spike times take continuous values, so exact matching is out of the question anyway.

Basically the W-R (and other) spike distance metrics get round this problem by filtering the spikes first to introduce a degree of overlap. The filtered sequence (for times in S) looks has the equation

$$y_S = \sum_{i=1}^{N_S} t \exp(-\alpha s_i)$$

Filtering both example spike trains with $\alpha = 1$, leads the modified plot

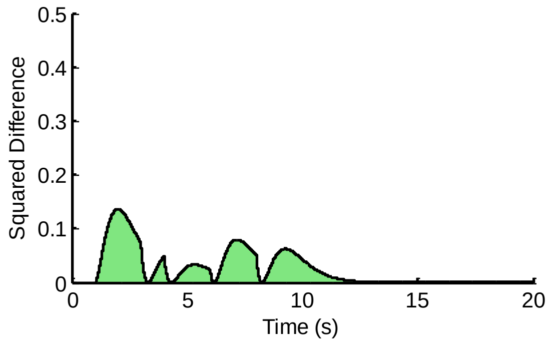
[1] See S. Wohlgemuth & B. Ronacher (2007), Auditory Discrimination of Amplitude Modulations Based On Metric Distances of Spike Trains, *J. Neurophysiol.* **97**:3082-3092.



This plot (and the one below) is generated using `wr_eg2`. Now in principle, we can measure the distance by differencing the waveforms (one of which is plotted upside-down of course), squaring the difference and integrating, i.e.,

$$d = \int (y_S - y_T)^2 dt$$

The difference waveform for the above, incidentally, is



and the green area under the curve is 0.4818 – so this is the distance measured.

Note that lower values of α lead to more smoothing and hence are more "lenient" when it comes to spikes being misaligned. Conversely small values of α lead less smoothing and hence are more "strict" when it comes to spikes being misaligned. (In the limit of small α, we are back to a coincidence detector).

Using the WR code

Some simple code has been written in C to compute the spike train distances. (The code performs all the integrations under the curve analytically, rather than numerically, using small steps.) The code accepts a column cell array of vectors, each vector being a column array of (double-precision) spike times.

To run the example above, type:

```
wr_metric({[1; 2; 4; 7; 8]; [2; 3; 6; 7]}, 1)
```

This returns

```
ans =


                0    0.4813

           0.4813         0
```

The zeros along the diagonal indicate that the spike trains are “zero” distance from themselves. The off-diagonal elements indicate the distance between the spike trains. (The same value we arrived at earlier only calculated numerically.) The routine returns large, square, symmetric distance matrices for larger numbers of spike trains.

Normalisation

The original W-R paper is rather unclear as to the *normalisation* that is applied to the distance. The current code does not normalise the integral. However, it is likely that the integral is supposed to be normalised by multiplying with $4\alpha^3$. This ensures that:

- There is a minimum where the optimum filtering time constant falls
- Dilating the time scale of the spikes and the filter response by the same factor does not affect the distance
- If α is really small, $\sqrt{d}$ gives the difference in total spike count.
- If α is really large, d gives the number of spikes that are *not coincident*.

To use normalisation, type `wr_metric(spks, alpha, 'norm')`. Or, if you want to use another normalisation, divide the distance by whatever factor seems appropriate.

A Note on Implementation

The code which implements this distance metric solves in blocks that go from one spike time to the next (regardless of which train they are in). In other words, it has to stop and perform calculations at each unique spike time. When comparing very many spike trains there are going to be a large number of unique spike times across the entire set. If speed becomes a problem, round all the spikes to an appropriate order of magnitude (e.g., 0.1 ms). For example, if the algorithm previously had to stop at times 0.9997, 1.0000, 1.0003, 1.0004 and 1.00045, it now only has to stop and perform computations at *one* step – with minimum disruption to the spike times themselves. (However, if speed is not an issue, leave things as they are.)