On the efficient calculation of van Rossum distances.

Conor Houghton[1*] and Thomas Kreuz[2†]

[1] *School of Mathematics, Trinity College Dublin, Ireland.*

[2] *Institute for complex systems - CNR, Sesto Fiorentino, Italy*

January 30, 2012

Abstract

The van Rossum metric measures the distance between two spike trains. Measuring a single van Rossum distance between one pair of spike trains is not a computationally expensive task, however, many applications require a matrix of distances between all the spike trains in a set or the calculation of a multi-neuron distance between two populations of spike trains. Moreover, often these calculations need to be repeated for many different parameter values. An algorithm is presented here to render these calculation less computationally expensive, making the complexity linear in the number of spikes rather than quadratic.

Introduction

The van Rossum metric is used to calculate a distance between two spike trains (van Rossum, 2001). It has been shown to be a good metric in the sense that the distance between multiple responses to the same stimulus is typically smaller than the distance between responses to different stimuli (Machens et al., 2003; Narayan et al., 2006). Like some other spike train

*houghton@maths.tcd.ie

†thomas.kreuz@cnr.it

distances (Victor and Purpura, 1996), the van Rossum metric depends on a parameter τ which determines the time scale in the spike trains to which the metric is sensitive. In the $\tau \to \infty$ limit of the parameter range the metric measures the difference in spike numbers while in the other limit, where $\tau = 0$, it counts non-coincident spikes.

If the two spike trains have roughly the same number of spikes, n say, then the complexity of the calculation of the van Rossum metric is of order n^2. Often an entire distance matrix is required containing the pairwise distances between all the spike trains in a set. This is needed, for example, when calculating a preferred value of the time scale τ: the standard method involves metric clustering and this requires the calculation of a distance matrix for multiple values of the time scale (Victor and Purpura, 1996; Machens et al., 2003; Wohlgemuth and Ronacher, 2007; Narayan et al., 2006; Toups et al., 2011). The optimal time scale is sometimes interpreted as indicative time scale for the temporal precision in coding. Although this interpretation should be treated with some caution (Chicharro et al., 2011), the optimal time scale is nonetheless useful as a way of maximizing the performance of the metric in applications. If there are N spike trains in the set, calculating the van Rossum distance matrix is, at face value, an order N^2n^2 calculation. For large data sets this can be a considerable computational burden.

The same holds true for a number of variations of the van Rossum metric such as the synapse-like metric introduced in (Houghton, 2009) and the multi-neuron metric (Houghton and Sen, 2008). The latter is designed to estimate the distance between two sets of neuronal population responses and is particularly troublesome from the point of view of computational expense.

Here, a trick is presented to reduce the computational cost for all three of these cases: For the regular van Rossum distance between two spike trains of length n the cost is reduced from order n^2 to order n and, hence, the cost of calculating a matrix of distances for N such spike trains from order N^2n^2 to order N^2n. The same speed-up is obtained for the van Rossum-like metrics. Similarly, in the multi-neuron case the cost of calculating the distance between two populations of P spike trains each will be reduced from order P^2n^2 to order P^2n.

Methods

The van Rossum metric

To describe the van Rossum distance it is useful to first define a map from spike trains to functions: given a spike train

$$\mathbf{u} = \{u_1, u_2, \cdots, u_n\} \tag{1}$$

it is mapped to $f(t; \mathbf{u})$ by filtering it with a kernel $h(t)$:

$$\mathbf{t} \mapsto f(t; \mathbf{u}) = \sum_{i=1}^{n} h(t - u_i). \tag{2}$$

The kernel function has to be specified, here, as in (van Rossum, 2001), the causal exponential is used

$$h(t) = \begin{cases} 0 & t < 0 \\ e^{-t/\tau} & t \geq 0 \end{cases} . \tag{3}$$

The decay constant τ is the time scale which parameterizes the metric.

The van Rossum metric is induced on the space of spike trains by the L^2 metric on the space of functions. In other words, the distance between two spike trains $\mathbf{u}$ and $\mathbf{v}$ is given by

$$d(\mathbf{u}, \mathbf{v}; \tau) = \sqrt{\frac{2}{\tau} \int dt [f(t; \mathbf{u}) - f(t; \mathbf{v})]^2}. \tag{4}$$

where the normalizing factor of $2/\tau$ is included so that there is a distance of one between a spike train with a single spike and one with no spikes. In fact, for the causal exponential filter this integral can be done explicitly to give a distance (Schrauwen and Van Campenhout, 2007; Paiva et al., 2009)

$$\begin{aligned} [d(\mathbf{u}, \mathbf{v}; \tau)]^2 &= \sum_{i,j} e^{-|u_i - u_j|/\tau} + \sum_{i,j} e^{-|v_i - v_j|/\tau} \\ &\quad -2 \sum_{i,j} e^{-|u_i - v_j|/\tau}. \end{aligned} \tag{5}$$

Now, the trick relies on the calculation of an extra vector for each spike train in the set; this extra vector will be called the *markage vector*. Given a

spike train $\mathbf{u}$, the markage vector $\mathbf{m}(\mathbf{u})$ will have the same number of element as there are spikes, so $\#(\mathbf{m}(\mathbf{u}))=\#(\mathbf{u})$. The entries in the markage vector are defined recursively so that $m_1 = 0$ and

$$m_i = (m_{i-1} + 1)e^{-(u_i - u_{i-1})/\tau}. \tag{6}$$

This means

$$m_i = \sum_{j|u_i > u_j} e^{-(u_i - u_j)/\tau} \tag{7}$$

where $j|u_i > u_j$ indicates that the sum is restricted to values of j where $u_j < u_i$. In fact, it will be assumed here that the spike times are in ascending order, which in most real situations they will be; this means that $j|u_i > u_j$ is actually equivalent to $j < i$. However, note that this notation for restricted values of an index is used below in other contexts where it can not be simplified in this way. Also, when needed for clarity the notation $m_i(\mathbf{u})$ will be used for the ith component of $\mathbf{m}(\mathbf{u})$, but in the above, the argument has been left out for brevity. The function $f(t)$ is discontinuous and m_i is equal to the left limit of $f(t)$ at u_i:

$$m_i = f(u_i-) = \lim_{t \to u_i -} f(t). \tag{8}$$

The idea is to use the markage vector to reduce the double sums in the expression for $d(\mathbf{u}, \mathbf{v})$, Eq. 5, to single sums. For convenience this equation is first rewritten to avoid the use of the absolute value

$$\begin{aligned}[d(\mathbf{u}, \mathbf{v}; \tau)]^2 \;=\; & \frac{n_1 + n_2}{2} + \sum_i \sum_{j|u_i > u_j} e^{-(u_i - u_j)/\tau} + \sum_i \sum_{j|v_i > v_j} e^{-(v_i - v_j)/\tau} \\ & - \sum_i \sum_{j|u_i > v_j} e^{-(u_i - v_j)/\tau} - \sum_i \sum_{j|v_i > u_j} e^{-(v_i - u_j)/\tau}\end{aligned} \tag{9}$$

where $n_1 = \#(\mathbf{u})$ and $n_2 = \#(\mathbf{v})$ are the number of spikes in the two spike trains. This yields

$$\sum_i \sum_{j|u_i > u_j} e^{-(u_i - u_j)/\tau} = \sum_i m_i(\mathbf{u}). \tag{10}$$

The cross-like terms are trickier, let $J(i)$ denote the index of the last spike time in $\mathbf{v}$ that is earlier than u_i, hence

$$J(i) = \max(j|u_i > v_j), \tag{11}$$

which leads to

$$\begin{aligned}\sum_i \sum_{j|u_i>v_j} e^{-(u_i-v_j)/\tau} &= \sum_i e^{-(u_i-v_{J(i)})/\tau} \sum_{j|v_{J(i)}\geq v_j} e^{-(v_{J(i)}-v_j)/\tau} \\ &= \sum_i e^{-(u_i-v_{J(i)})/\tau} \left(1+\sum_{j|v_{J(i)}>v_j} e^{-(v_{J(i)}-v_j)/\tau}\right) \\ &= \sum_i e^{-(u_i-v_{J(i)})/\tau} \left[1+m_{J(i)}(\mathbf{v})\right]. \end{aligned} \tag{12}$$

Since the other two terms in the expression for $d(\mathbf{u},\mathbf{v})$ are identical to the two above with $\mathbf{u}$ and $\mathbf{v}$ switched they can be calculated in the same way, so, for example,

$$\sum_i \sum_{j|v_i>u_j} e^{-(v_i-u_j)/\tau} = \sum_i e^{-(v_i-u_{J(i)})/\tau} \left[1+m_{J(i)}(\mathbf{v})\right]. \tag{13}$$

For time ordered spike trains, this reduces the calculation of all four terms from n^2 to n. The two cross terms, containing times from both $\mathbf{u}$ and $\mathbf{v}$ are the most important since when calculating a distance matrix these are the most expensive, involving N^2n^2 calculations without markage, the two square terms need only be calculated once for each spike train and are therefore linear in N.

Using the markage vector does introduce extra calculations. Along with the calculation of the markage vector itself, there is the need to calculate $J(i)$, the index of the time on one spike train immediately prior to the ith time in the other. If the spike trains were not time ordered this calculation would be order n^2; however, for the more biologically relevant case of time ordered spike trains, $J(i)$ can be calculated iteratively by advancing it from its previous value when necessary. It is seen below that the constant prefactor to N^2n in the algorithm with markage is larger than the prefactor to N^2n^2 in the traditional algorithm, but that it is worthwhile using the new algorithm even for quite short spike trains.

Other van Rossum-like metrics

There are a number of variations of the van Rossum metrics. One generalization is to consider other kernels. In the standard formulation the kernel $h(t)$

is a causal exponential, Eq. 3, but other kernels can be used: a square pulse or normal curve are popular alternatives. It is also possible to regard the exponential factors in the integrated formula, Eq. 5 as kernels and replace these with other functions (Schrauwen and Van Campenhout, 2007). The algorithm proposed here will not, in general, be useful for these other cases. It might be possible to generalize the sum formula for the markage, Eq. 7, but this is an order n^2 calculation and would contribute Nn^2 to the overall calculation of a distance matrix. In the causal exponential case considered here there is also an iterative formula for the markage, Eq. 6. This will not work for all choices of kernel.

There is, however, another generalization of the van Rossum metric in which the map from spike trains to functions is viewed as a differential equation, rather than as a filtering. In this description, $\mathbf{u}$ is mapped to $f(t;\mathbf{u})$ where

$$\tau \frac{df}{dt} = -f \tag{14}$$

with discontinuities

$$f(u_i+) = f(u_i-) + 1 \tag{15}$$

at the spikes. The synapse-like metric introduced in (Houghton, 2009) changes the map from spike trains to functions in order that it mimics aspects of the synaptic conductivity. Specifically, it accounts for the depletion of available binding sites by changing how much f is increased by each spike; $f(t;\mathbf{u})$ is defined as the solution to the differential equation

$$\begin{aligned} \tau \frac{d}{dt} f &= -f \\ f(u_i+) &= (1-\mu) f(u_i-) + 1 \end{aligned} \tag{16}$$

at the spike times u_i. The first parameter τ, as before, is a time scale, associated with the precision of spiking, while the second parameter μ reflects a reduction of the significance of spike timing in spike clusters.

To calculate the integrated form of this distance it is convenient to first define an *increment vector* $\boldsymbol{\delta}(\mathbf{u})$, where $\delta_1 = 1$ and

$$\delta_i = 1 - \mu \sum_{j=1}^{i-1} \delta_i e^{-(u_i - u_j)/\tau}. \tag{17}$$

The quantity δ_i is equal to the increment in $f(t;\mathbf{u})$ at u_i. Now, letting

$\boldsymbol{\alpha} = \boldsymbol{\delta}(\mathbf{u})$ and $\boldsymbol{\beta} = \boldsymbol{\delta}(\mathbf{v})$ the synapse distance is

$$\begin{aligned} d(\mathbf{u}, \mathbf{v}) &= \sum_{i,j} \alpha_i \alpha_j e^{-|u_i - u_j|/\tau} + \sum_{i,j} \beta_i \beta_j e^{-|v_i - v_j|/\tau} \\ &\quad -2 \sum_{i,j} \alpha_i \beta_j e^{-|u_i - v_j|/\tau}. \end{aligned} \tag{18}$$

Again, this calculation can be made more efficient using a markage vector; this time it will have the form

$$m_i = \sum_{j=1}^{i-1} \delta_i e^{-(u_i - u_j)/\tau}. \tag{19}$$

The markage and increment vector can be calculated iteratively in leap-frog fashion

$$\begin{aligned} m_i &= (m_{i-1} + \delta_{i-1}) e^{-(u_i - u_{i-1})/\tau} \\ \delta_i &= 1 - \mu m_i. \end{aligned} \tag{20}$$

For example,

$$\begin{aligned} \sum_{i,j} \alpha_i \alpha_j e^{-|u_i - u_j|/\tau} &= \sum_i \alpha_i^2 + 2 \sum_i \alpha_i \sum_{j|u_i > u_j} \alpha_j e^{-|u_i - u_j|/\tau} \\ &= \sum_i \alpha_i (\alpha_i + 2 m_i) \end{aligned} \tag{21}$$

with similar expressions for the other terms; in short, the formulae for the efficient calculation of the synapse metric only differ from formulae for the original van Rossum metric in the inclusion of the components of the increment vector.

The multi-neuron van Rossum metric

There is also a multi-neuron van Rossum metric (Houghton and Sen, 2008) which was proposed as tool for quantifying if and how populations of neurons cooperate to encode a sensory input. Here, computational expense is an even more acute problem but the same trick can be used.

In this setup there are two population responses $\mathcal{U}$ and $\mathcal{V}$ with $p = 1, ..., P$ spike trains each. Spikes are represented as u_i^p and v_j^q with $i = 1, .., n_u^p$ and

$j = 1, .., n_v^q$ where n_u^p and n_v^q denote the numbers of spikes in the pth and qth spike train of the populations $\mathcal{U}$ and $\mathcal{V}$, respectively. By locating the P spike trains of the two population responses $\mathcal{U}$ and $\mathcal{V}$ in a space of vector fields, with a different unit vector assigned to each neuron, interpolation between the two extremes, the labeled line (LL) coding and the summed population (SP) coding, is achieved by varying the angle θ between unit vectors from zero (SP) to $\pi/2$ (LL) or, equivalently, the parameter $c = \cos\theta$ from one to zero:

$$d(\mathcal{U}, \mathcal{V}; \tau) = \sqrt{\frac{2}{\tau} \sum_p \left(\int_0^\infty |\delta_p|^2 dt + c \sum_{q \neq p} \int_0^\infty \delta_p \delta_q dt \right)} \tag{22}$$

with $\delta_p = f_p(t) - g_p(t)$.

In the original description in (Houghton and Sen, 2008) the metric is described in terms of the unit vectors corresponding to each neuron in the population. In the formula above the dot-products between the unit vectors have already been performed, giving a simpler formula.

Similar to the single-neuron case (Schrauwen and Van Campenhout, 2007; Paiva et al., 2009) the multi-neuron van Rossum distance can also be estimated in a more efficient way by integrating analytically:

$$d(\mathcal{U}, \mathcal{V}; \tau) = \sqrt{\sum_p \left(R_p + \ c \sum_{q \neq p} R_{pq} \right)} \tag{23}$$

with

$$R_p = \sum_{i,j} e^{-|u_i^p - u_j^p|/\tau} + \sum_{i,j} e^{-|v_i^p - v_j^p|/\tau} - 2 \sum_{i,j} e^{-|u_i^p - v_j^p|/\tau} \tag{24}$$

being the bivariate single neuron distance, as in Eq. 5, between the p-th spike trains of u and v and

$$R_{pq} = \sum_{i,j} e^{-|u_i^p - u_j^q|/\tau} + \sum_{i,j} e^{-|v_i^p - v_j^q|/\tau} - \sum_{i,j} e^{-|u_i^p - v_j^q|/\tau} - \sum_{i,j} e^{-|v_i^p - u_j^q|/\tau} \tag{25}$$

representing the cross-neuron terms that are needed to fully capture the activity of the pooled population. As in the estimate in Eq. 22 the variable c interpolates between the labeled line and the summed population distance.

These are the two equations for which the markage trick can be applied since all of these sums of exponentials can be rewritten using markage vectors

in the same way as it was done in Eqs. 9, 10, and 12, so, taking the first term in R_{pq} as an example

$$\begin{aligned}\sum_{i,j} e^{-|u_i^p - u_j^q|/\tau} &= \sum_i e^{-(u_i^p - u_{J(i)}^q)/\tau} \left[1 + m_{J(i)}(\mathbf{u}^q)\right] \\ &\quad + \sum_i e^{-(u_i^q - u_{J(i)}^p)/\tau} \left[1 + m_{J(i)}(\mathbf{u}^p)\right] \end{aligned} \tag{26}$$

In case of $\tau = \infty$ Eq. 23 remains still valid, however, the calculation of the contributing terms simplifies to

$$R_p = n_u^p(n_u^p - n_v^p) + n_v^p(n_v^p - n_u^p) \tag{27}$$

and

$$R_{pq} = n_u^p(n_u^q - n_v^q) + n_u^q(n_u^p - n_v^p) + n_v^p(n_v^q - n_u^q) + n_v^q(n_v^p - n_u^p). \tag{28}$$

If $\tau = \infty$ and $c = 1$, the calculation of the population van Rossum distance D simplifies even further:

$$D(\mathbf{u}, \mathbf{v}; \tau) = \sqrt{R} \tag{29}$$

with

$$R = \sum_{p=1}^{P} n_u^p \sum_{p=1}^{P} (n_u^p - n_v^p) + \sum_{p=1}^{N} n_v^p \sum_{p=1}^{P} (n_v^p - n_u^p). \tag{30}$$

There is a generalization of this metric in which there are many different angles, or equivalently, many different c parameters; this allows each pair of neurons to lie on a different point in the interpolation between summed coding and labeled coding (Houghton and Sen, 2008). The simplifications discussed here can also be applied in this case.

Results

The algorithm of the regular van Rossum metric was tested using simulated spike trains. The spike trains were generated as a homogeneous Poisson process with rate 30Hz. The algorithm (with the timescale set to $\tau = 12$ms) was implemented in `C++` and run times were compared to the same algorithm without the markage method. A further optimization was considered: the

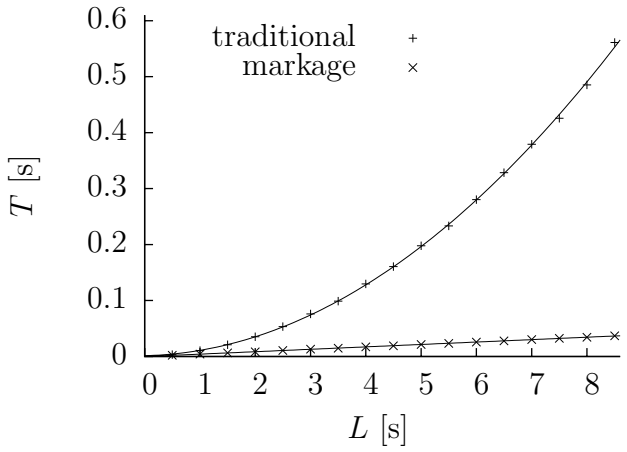


Figure 1: Van Rossum metric: Run times plotted against spike train length. Here the run time T is plotted against spike train length L for the traditional algorithm and the algorithm with markage. T and L are both measured in seconds; the data set contains 100 spike trains and for each value of L the whole matrix of pairwise distances is calculated and the time calculated by averaging over 200 trials. The line and parabola are least square fits calculated using `wolframalpha.com`.

exponential factors of the form $\exp(t_0/\tau)$ and $\exp(-t_0/\tau)$, with t_0 a spike time, were calculated just once at the start. Although this improvement does not alter the dependence on n, the number of spikes, it does speed up both algorithms considerably. The run times discussed here do not include the calculation of the simulated spike trains, but do include all initializations involved in the metric calculation, and the calculation of the markage vector and the pre-calculated positive and negative exponential vectors, where required.

The `C++` code was compiled using `g++ v4.6.1`, part of the GNU compiler collection. It was compiled with the directive `-O3` but with no further optimizations and run on one core of an `Intel Core i5-2520M` CPU. In Fig. 1 the run times T are plotted against spike train length L. The largest L value considered is $L = 8.5$s. Without further modification the pre-calculation of exponentials would be problematic for longer spike trains since $\exp(t_0/\tau)$ will be too large to store as a `double`.

The graph in Fig. 1 shows both the recorded run times and a least squares fit to a parabola, for the traditional algorithm, and to a line for the algorithm with markage. For the algorithms with exponential pre-calculation shown in the graphs the best-fit curves are

$$T_0(L) = 0.00738L^2 + 0.00209L + 0.00145 \tag{31}$$

for the traditional algorithm and

$$T_1(L) = 0.00432L - 0.00011 \tag{32}$$

for the algorithm with markage. Of course, the run times depend on platform, hardware and implementation, but the ratio should give an approximate indication of how well the algorithm performs

$$T_1 = 0.58\frac{T_0}{L} + O(1) \tag{33}$$

or, in terms of spike number, n,

$$T_1 \approx 17.4\frac{T_0}{n} \tag{34}$$

for larger values of n. For small values of n the quadratic is a poor fit and $T_0 = 0.00214$ for $L = 0.5$s, equivalent to $n = 15$, while $T_1(0.5) = 0.00329$. Exponential pre-calculation has a more substantial effect on the traditional algorithm than on the algorithm with markage; the traditional algorithm requires many more exponential factors. Fitting curves to the run times without exponential pre-calculation gives

$$T_0(L) = 0.219L^2 + 0.00496L + 0.00135 \tag{35}$$

for the traditional algorithm and

$$T_1(L) = 0.0185L - 0.000433 \tag{36}$$

for the markage algorithm, so

$$T_1 = 0.084\frac{T_0}{L} + O(1) \approx 2.5\frac{T_0}{n}. \tag{37}$$

The ratio between the markage algorithm with exponential pre-calculation and the traditional algorithm without is $0.6/n$. Varying N, the number of spike trains, verifies that the run times have an N^2 dependence.

If the same analysis is applied to the synapse metric with exponential pre-calculation and with $\mu = 0.5$, we obtain

$$T_1 \approx 15.3\frac{T_0}{n}. \tag{38}$$

The multi-neuron metric was also studied; using $P = 10$, $N = 20$ and $\cos\theta = 0.5$ yielded

$$T_1 \approx 18.0\frac{T_0}{n}. \tag{39}$$

In Fig. 2 the running time for the multi-neuron metric is plotted against N and P, the number of sets of spike trains and the number of spike trains in each set. In each case, again considerably smaller run times are observed for the algorithm with markage.

Discussion

The algorithms described here are more complicated to implement than the more straight-forward approaches; however, for large spike trains the speed up is considerable and may make practical an analysis technique that would not otherwise be quick enough to be useful.

Although the method presented here is a computational trick, it is interesting to note that a biologically inspired quantity, like the van Rossum metric and its multi-neuron extension, might be expected to permit a trick of this sort. The trick works by replacing a sum over history by a running tally; this sort of replacement is typical of biological systems.

Matlab[1] and c++ [2] source codes for both the regular and the multi-neuron van Rossum distances are available.

Acknowledgements

We gratefully acknowledge useful discussions with Daniel Chicharro. CH is grateful to the James S McDonnell Foundation for a Scholar Award in Understanding Human Cognition. We thank one of the anonymous referee's for suggesting the pre-calculation of the exponentials.

[1] `http://www.fi.isc.cnr.it/users/thomas.kreuz/sourcecode.html`

[2] `http://sourceforge.net/p/spikemetrics`

References

Chicharro D, Kreuz T, Andrzejak RG. 2011. What can spike train distances tell us about the neural code? *Journal of Neuroscience Methods* 199:146–165.

Houghton C. 2009. Studying spike trains using a van Rossum metric with a synapses-like filter. *Journal of Computational Neuroscience* 26:149–155.

Houghton C, Sen K. 2008. A new multi-neuron spike-train metric. *Neural Computation* 20:1495–1511.

Machens CK, Schütze H, Franz A, Kolesnikova O, Stemmler MB, Ronacher B, Herz A. 2003. Single auditory neurons rapidly discriminate conspecific communication signals. *Nature Neuroscience* 6:341–2.

Narayan R, Graña G, Sen K. 2006. Distinct time scales in cortical discrimination of natural sounds in songbirds. *Journal of Neurophysiology* 96:252–258.

Paiva ARC, Park I, Príncipe JC. 2009. A reproducing kernel Hilbert space framework for spike train signal processing. *Neural Computation* 21:424–449.

Schrauwen B, Van Campenhout J. 2007 Linking non-binned spike train kernels to several existing spike train metrics. *Neurocomputing* 70:1247–1253.

Toups JV, Fellous J-M, Thomas PJ, Sejnowski TJ, Tiesinga, PH. 2011. Finding the event structure of neuronal spike trains. *Neural Computation* 23:2169–2208.

van Rossum M. 2001. A novel spike distance. *Neural Computation* 13:751–763.

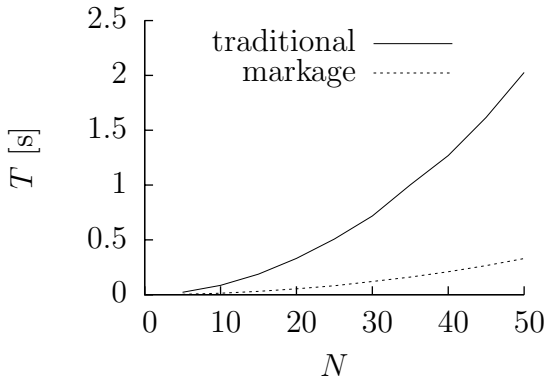
Victor JD, Purpura KP. 1996. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of Neurophysiology* 76:1310–1326.

Wohlgemuth S, Ronacher B. 2007. Auditory discrimination of amplitude modulations based on metric distances of spike trains. *Journal of Neurophysiology* pp. 3082–3092.

Declaration of interest

The authors report no conflicts of interest. The authors alone are responsible for the content and writing of the paper.

A



B

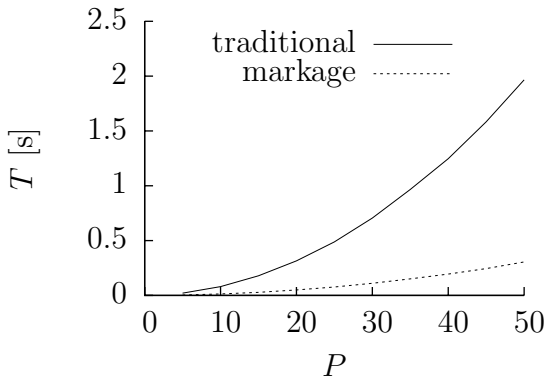


Figure 2: Multi-neuron van Rossum metric: Run times plotted against N and P. **A** shows run times plotted against the number of sets of spike trains, N, for $P = 10$, **B** shows run times against the number of spike trains in each set, P, for $N = 10$. In each case the run times were averaged over 200 trials with the train length $L = 3$s, with angle $\cos\theta = 0.5$ and with exponential pre-calculation. Since the dependence on both N and P is quadratic, the two graphs are similar to the point of being almost indistinguishable.