ELSEVIER

Journal of Neuroscience Methods 161 (2007) 351–360

JOURNAL OF NEUROSCIENCE METHODS

www.elsevier.com/locate/jneumeth

Dynamic programming algorithms for comparing multineuronal spike trains via cost-based metrics and alignments

Jonathan D. Victor [a,*], David H. Goldberg [b], Daniel Gardner [a,b]

[a] *Department of Neurology and Neuroscience, Weill Medical College of Cornell University, 1300 York Avenue, New York City, NY 10021, United States*
[b] *Laboratory of Neuroinformatics, Department of Physiology and Biophysics, Weill Medical College of Cornell University, 1300 York Avenue, New York City, NY 10021, United States*

Received 9 August 2006; received in revised form 31 October 2006; accepted 1 November 2006

Abstract

Cost-based metrics formalize notions of distance, or dissimilarity, between two spike trains, and are applicable to single- and multineuronal responses. As such, these metrics have been used to characterize neural variability and neural coding. By examining the structure of an efficient algorithm [Aronov D, 2003. Fast algorithm for the metric-space analysis of simultaneous responses of multiple single neurons. J Neurosci Methods 124(2), 175–79] implementing a metric for multineuronal responses, we determine criteria for its generalization, and identify additional efficiencies that are applicable when related dissimilarity measures are computed in parallel. The generalized algorithm provides the means to test a wide range of coding hypotheses.

Keywords: Neural coding; Spike trains; Metric space; Sequence comparison; Informatics

1. Introduction

Spike train metrics (Victor and Purpura, 1996, 1997) are used to characterize neural variability and neural coding in a range of neurophysiologic contexts, especially sensory systems; see recent review by Victor (2005). Metric approaches formalize spike train neural activity as a sequence of events (Segundo and Perkel, 1969). This viewpoint contrasts with two ways to represent neural activity in a vector space: as a rate (i.e., a continuous function of time) or as a function of discretely sampled time (Rieke et al., 1997). The choice of viewpoint has implications for the overall approach to data analysis. Vector spaces (including spaces of time series) have a natural means of defining a Euclidean "distance", based on their inner (scalar) product. In contrast, distances that naturally arise from sequence comparisons are typically non-Euclidean (Aronov and Victor, 2004). Non-Euclidean distances are necessary to account for some aspects of neural coding (Hopfield, 1995; Wuerger et al., 1995).

However, non-Euclidean distances determined by spike train metrics are more difficult to compute than distances in a vector space. For sequences of activity of a single neuron, the algorithmic problem for a simple spike train metric, $D^{\text{spike}}[q]$, where q is the cost per unit time for moving a spike, is closely analogous to that of biological sequence comparison (Needleman and Wunsch, 1970; Sellers, 1974), and a very similar dynamic programming algorithm is applicable (Victor and Purpura, 1996, 1997). Computational complexity is M^2, where M is the typical number of spikes in the responses to be compared. Multineuronal activity, recordings of which are becoming more and more widely available (Gray et al., 1995; Kralik et al., 2001; Nicolelis et al., 2003), can be considered as a sequence of labeled events (in which the label indicates the neuron of origin). The metric-space method readily extends to this context (Aronov et al., 2003; Samonds and Bonds, 2004). A straightforward extension of the algorithm for the single-neuron metric $D^{\text{spike}}[q]$ leads to an algorithm for a simple multineuronal metric, $D^{\text{spike}}[q, k]$. This algorithm has a computational complexity of M^{2L}, where L is the number of distinct neurons. Recently, Aronov (2003) dramatically improved this to M^{L+1}, via a dynamic-programming algorithm that treats the compared spike trains asymmetrically.

Generically, spike train metrics have parameters that indicate the extent to which the metric is sensitive to various features of the spike train (e.g., sensitivity to timing in $D^{\text{spike}}[q]$ and $D^{\text{spike}}[q, k]$ is determined by q, and sensitivity to neuron of ori-

* Corresponding author.
E-mail address: jdvicto@med.cornell.edu (J.D. Victor).

0165-0270/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.jneumeth.2006.11.001

gin in $D^{\text{spike}}[q, k]$ is determined by k). In this paper, we extend this algorithm to a wide range of single- and multineuronal spike train metrics. We also show how the algorithm can be modified to calculate spike train metrics for many values of the parameters in an efficient manner. As part of an ongoing effort to provide information-theoretic tools to the neuroscience community, implementations of algorithms for $D^{\text{spike}}[q]$ and $D^{\text{spike}}[q, k]$ described below are available in the Spike Train Analysis Toolkit (Goldberg et al., 2005) and can be obtained from the website http://neuroanalysis.org.

After some preliminary definitions, we review the notion of a cost-based metric, and then consider a partially distinct approach to define dissimilarities of event sequences, based on "alignments." We observe that the intersection of cost-based metrics and alignments includes the basic multineuronal cost-based spike time metric $D^{\text{spike}}[q, k]$. As we then show, the efficient dynamic programming algorithm (Aronov, 2003) for $D^{\text{spike}}[q, k]$, when viewed as an algorithm to identify alignments, is capable of substantial generalization. We then comment on matters of implementation and provide examples.

2. Results

2.1. Preliminaries

A spike train (a neural response) is considered to be a sequence of events, each occurring at a specific time and associated with a discrete label (the neuron of origin). More formally, a *spike train* A is a sequence of $M(A)$ spike times $A_1, A_2, \ldots, A_{M(A)}$, each with an associated label, $a_1, a_2, \ldots, a_{M(A)}$. The spike times A_k are non-decreasing real numbers, and may coincide. The labels are drawn from a set $\{1, \ldots, L\}$ of abstract tags. We use $A[w]$ to denote the subsequence of A that includes just the spikes associated with a label w. Thus, $M(A[w])$ is the number of spikes in A with the label w. We also use $\vec{M}(A)$ to denote a vector whose wth element is $M(A[w])$. A spike train is the disjoint union of its subtrains $A[w]$, i.e.,

$$\bigcup_{w=1}^{L} A[w] = \{A_1, \ldots, A_{M(A)}\}, \quad \text{and} \quad \sum_{w=1}^{L} M(A[w]) = M(A).$$

2.2. Metrics, alignments, and strains

We will consider two ways to measure dissimilarity between two spike trains A and B. The first, "cost-based metrics" (Victor, 2005; Victor and Purpura, 1996, 1997) is based on a set of elementary transformations between spike trains. Each elementary transformation (e.g., deleting a spike, inserting a spike, shifting a spike in time, or changing the label of a spike) is associated with a cost. In a cost-based metric, the distance (dissimilarity) between two spike trains is the minimum total cost of any sequence of elementary steps that transforms A into B.

The second way of measuring dissimilarity is based on the notion of an *alignment* between two spike trains, $X(A, B)$ (or simply X). An alignment (Fig. 1A) indicates a correspondence between the components of spike train A and those of spike train B, via a set of links between pairs of spikes. As shown in Fig. 1A, an alignment may leave some spikes unlinked in one or both of the two trains. We will assign a *strain* to each alignment, and the measure of dissimilarity between A and B will be taken to be the smallest strain of all possible alignments $X(A, B)$.

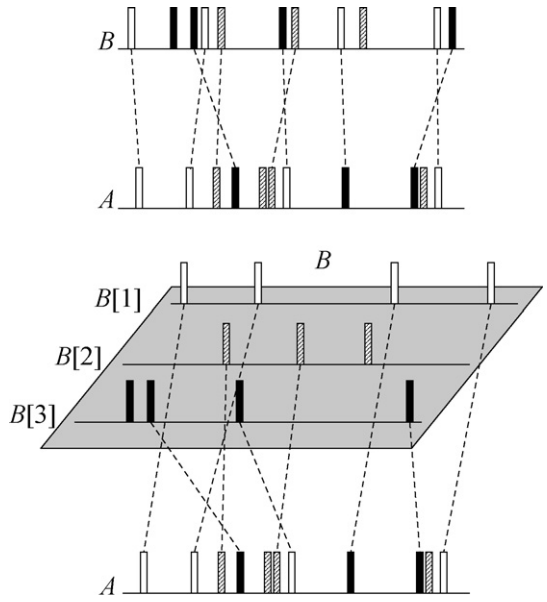

Fig. 1. (A) A candidate alignment of two multineuronal spike trains. The neuron of origin (the "label") of each spike is indicated by shading, and an alignment consists of links between pairs of individual spikes. An alignment may link spikes from different neurons, and all spikes need not be linked. The alignment shown is not necessarily an "efficient" alignment, but is typical of one considered by the dynamic programming algorithm (Aronov, 2003) for $D^{\text{spike}}[k, q]$. (B) A necessary, but not sufficient, condition for an efficient alignment. The subtrains of spike train B are separated into individual parallel lines. This defines a unique plane for each label w, containing the subtrain $B[w]$ and the full train A. In each plane (i.e., the plane of links in which the second member of each pairing comes from the same subtrain $B[w]$) links cannot intersect. For further details, see text. Adapted from Aronov (2003) with permission.

We will restrict consideration to strains that depend only on the number of unlinked spikes and on the time differences between the spikes connected by the links. We will further restrict consideration to strains in which these components combine additively. More formally, we will consider strains of the form

$$S(X(A, B)) = \sum_{w=1}^{L} I_{1,w}(U_{1,w}) + \sum_{w=1}^{L} I_{2,w}(U_{2,w}) + \sum_{w_1=1}^{L}\sum_{w_2=1}^{L}\sum_{j} J_{w_1,w_2}\left(\left|T_{w_1,w_2,j}\right|\right). \quad (1)$$

In this equation, $U_{1,w}$ and $U_{2,w}$ are the number of unlinked spikes of label w in each train, and T is a table of lists $T_{w_1,w_2,j}$ of the time differences between linked spikes of label w_1 in A and label w_2 in B. $I_{1,w}$ and $I_{2,w}$ are non-decreasing functions that determine the contribution to the strain of unpaired spikes of each label (w) in each train (1 for A, 2 for B). Similarly, $J_{w_1,w_2}(t)$ is a non-decreasing function that determines the contribution to

the strain of a link of length t between a spike with label w_1 in train A, and a spike with label w_2 in train B.

Note that the form (1) allows for the dependence of S on unlinked spikes to depend on the label, and to be different for the first and second trains in the alignment. The dependence of S on time differences may also depend on the labels associated with each pair of linked spikes.

An alignment that achieves the minimum strain is called an "efficient" alignment. Thus, calculation of the least strain between two spike trains can be reduced to finding an efficient alignment.

2.3. Relationship between metrics and alignments

The cost-based metric $D^{\text{spike}}[q]$ and its multineuronal extension $D^{\text{spike}}[q, k]$ have been found useful in several neurophysiologic contexts (Aronov et al., 2003; Samonds and Bonds, 2004; Victor, 2005). As we now show, these metrics can also be considered to be the strain of an efficient alignment. $D^{\text{spike}}[q, k]$ is defined by the following allowed elementary transformations and costs: inserting or deleting a spike has a cost of 1; moving a spike by an amount Δt has a cost $q\Delta t$, and changing the label of a spike has a cost k. In a sequence of elementary steps that achieves minimal total cost, no spike need be moved in more than one direction, and spikes that are inserted or deleted need not be moved. Thus, a minimal-cost transformation can always be broken down into the following components: (1) spikes deleted from A or inserted into B; (2) spikes in A that are moved into position of spikes of the same label of B, and (3) spikes in A that are moved into the position of spikes of a different label in B, and then change their label.

Together the deleted and inserted spikes constitute the unlinked spikes of an alignment, and the moved spikes constitute the links of an alignment. Conversely, any alignment can be reinterpreted as a sequence of the elementary steps allowed by $D^{\text{spike}}[q, k]$.

Consequently, the cost-based metric $D^{\text{spike}}[q, k]$ between two spike trains can be recast as the minimum strain of an alignment, where the strain is of the form (1), with

$$I_{1,w}(U_{1,w}) = U_{1,w},\ I_{2,w}(U_{2,w}) = U_{2,w},\ \text{ and }\ J_{w_1,w_2}(T_{w_1,w_2}) = k(1-\delta(w_1,w_2)) + q\sum_j \left|T_{w_1,w_2,j}\right|. \quad (2)$$

However, cost-based metrics and alignments are not equivalent in general. Cost-based metrics such as $D^{\text{interval}}[q]$, in which elementary steps (Victor and Purpura, 1996, 1997) include changing the length of an interspike interval, cannot readily be considered as alignments of individual spikes. Conversely, strains of a form similar to Eq. (1), in which terms are combined following application of a power law, cannot readily be construed as the sum of costs of elementary steps.

Cost-based metrics must satisfy the triangle inequality: the length of the shortest path between two points cannot be more than the length of a path between those two points that is constrained to pass through a third point. Cost-based metrics must also be symmetric in their arguments. Measures of dissimilarity that arise from strains (Eq. (1)) need not have either property, and thus, may not be "metrics". Such notions of dissimilarity are important in neuroscience applications. Asymmetry may arise if the spike trains being compared play distinct roles, for example, the first spike train may represent a standard by which the second spike train is to be classified. Non-metric measures of dissimilarity also arise naturally in studies of perception (Maloney and Yang, 2003; Tversky, 1977; Tversky and Gati, 1982) and information theory, e.g., the Kullbach–Leibler divergence (Cover and Thomas, 1991; Kullbach and Leibler, 1951).

2.4. Generalizing the algorithm for cost-based metrics

We next review the dynamic programming algorithm (Aronov, 2003) for $D^{\text{spike}}[q, k]$. By casting the algorithm as a method to minimize the strain of an alignment, and focusing on the properties of a strain required for the algorithm's validity, we will extend the applicability of the algorithm to a much wider notion of dissimilarities between spike trains, including, but not limited to, other cost-based distances.

As in the algorithm for genetic sequence comparison (Needleman and Wunsch, 1970; Sellers, 1974), each stage of the algorithm determines a distance $D^{\text{spike}}[q, k](A^{(\alpha)}, B^{(\vec{\beta})})$ between subtrains $A^{(\alpha)}$ and $B^{(\vec{\beta})}$ (defined below), from previously calculated distances. At the last stage, $A^{(\alpha)} = A$ and $B^{(\vec{\beta})} = B$, so the calculation is complete. The two spike trains are treated asymmetrically. For the first spike train, $A^{(\alpha)}$ denotes the subtrain consisting of the first α spikes of A, independent of label. For the second spike train, $\vec{\beta}$ is a vector index with one entry for each of the L labels, and $B^{(\vec{\beta})}$ consists of the first β_w spikes of each subtrain $B[w]$. This asymmetry is key to limiting the computational complexity to M^{L+1}.

The algorithm's strategy is based on the fact that only a small number of subtrains need to be considered to find an efficient alignment $X(A^{(\alpha)}, B^{(\vec{\beta})})$. This fact follows from a geometric argument made by Aronov (2003) for the strain corresponding to $D^{\text{spike}}[q, k]$. Any alignment, efficient or not, can be displayed as a set of line segments that link subsets of spikes between the two trains, each considered as identified points on parallel lines (Fig. 1A). The spikes of the subtrains $B[w]$ can be separated into parallel lines, one for each subtrain, while keeping all of the spikes of train A on a single line (Fig. 1B). When this is done, a necessary (but not sufficient) condition that the strain is efficient is that the line segments that correspond to links do not intersect (Fig. 1B). The non-crossing rule allows us to reduce the problem of finding an efficient alignment of the original trains to one of finding an efficient alignment of a restricted number of smaller trains. Below we will identify properties of a strain that are sufficient to guarantee that this non-crossing rule holds.

In any alignment $X(A, B)$, at least one of the following three possibilities must hold: (i) the terminal spike in A is unlinked, (ii) the terminal spike in at least one of the subtrains $B[w]$ is unlinked, or, (iii) the terminal spike in A is linked, as are all of the terminal spikes of the subtrains $B[w]$. We want to set up conditions under which each of these possibilities leads to a

simple way to reduce an efficient alignment $X(A^{(\alpha)}, B^{(\vec{\beta})})$ to an efficient alignment of smaller trains.

In case (i), the terminal spike in $A^{(\alpha)}$ is unlinked. We would like to be able to remove this spike, and be assured that the resulting alignment, $X(A^{(\alpha-1)}, B^{(\vec{\beta})})$, is also efficient. It suffices to assert *Condition 1*: Consider any two alignments $Y(A, B)$ and $Z(A, B)$ of spike trains A and B. Add a spike to A to form a new spike train A', and form new alignments $Y'(A', B)$ and $Z'(A', B)$ from $Y(A, B)$ and $Z(A, B)$ by keeping the new spike unlinked. Then the rank order of the strains of $Y(A, B)$ and $Z(A, B)$ is the same as the rank order of the strains of $Y'(A', B)$ and $Z'(A', B)$ (i.e., if $Y > Z$ then $Y' > Z'$, and conversely).

In case (ii), the terminal spike in one of the subtrains $B[w]$ is unlinked. Here, we would like to remove this spike and be assured that the resulting alignment is efficient. In a manner completely analogous to Condition 1, this can be guaranteed if we assert *Condition 2:* augmenting alignments by adding an unlinked spike to any one of the subtrains $B[w]$ does not change the rank order of their strains.

In case (iii), the terminal spike in $A^{(\alpha)}$ is linked, and the terminal spikes in all of the subtrains $B^{(\vec{\beta})}[w]$ are also linked. Reduction of the alignment X now requires removal of linked spikes. First, we want to be assured that removal of a link results in an efficient alignment. This can be guaranteed by asserting *Condition 3:* augmenting alignments by adding the same linked pair of spikes does not change the rank order of their strains. However, *Condition 3* does not guarantee that the terminal spike in $A^{(\alpha)}$ is linked to a *terminal* spike in one of the subtrains $B^{(\vec{\beta})}[w]$—the formal equivalent of the non-crossing rule.

Without this simplification, there would be a combinatorial explosion of the number of subtrains of $B^{(\vec{\beta})}[w]$ whose alignments with $A^{(\alpha-1)}$ need to be considered, since there may be efficient alignments in which all terminal spikes of A and B are paired, but each are paired with non-terminal spikes in the other train. To exclude this possibility, we need a fourth condition, to ensure that in any efficient alignment, links between $A^{(\alpha)}$ and $B^{(\vec{\beta})}[w]$ do not cross. Thus, we assert *Condition 4:* in any alignment $Y(A, B)$ with an intersection of two links between spikes in A and a subtrain $B[w]$, then the alignment derived by uncrossing the links cannot have a larger strain.

2.5. Statement of the basic algorithm

Given the above *Conditions 1–4*, an efficient alignment of $X(A, B)$ may be found as follows.

Initialize: Efficient alignments with the null train Ø (namely, $X(A^{(\alpha)}, \text{Ø})$ or $X(\&\text{Oslash};, B^{(\vec{\beta})})$), trivially have all spikes unpaired.

Extend: An efficient alignment of the subtrains $X(A^{(\alpha)}, B^{(\vec{\beta})})$ is the alignment that has the least strain among the following possibilities:

(i) Add an unlinked terminal spike at time A_α and with label a_α to the alignment $X(A^{(\alpha-1)}, B^{(\vec{\beta})})$,

(ii) add an unlinked terminal spike at time $B[w]_{\beta_w}$ and with label w to the alignment $X(A^{(\alpha)}, B^{(\vec{\beta}-\vec{1}_w)})$, or

(iii) add a link between a terminal spike at time A_α and with label a_α and a terminal spike at time $B[w]_{\beta_w}$ and with label w to the alignment $X(A^{(\alpha-1)}, B^{(\vec{\beta}-\vec{1}_w)})$.

Terminate: when $A^{(\alpha)} = A$ and $B^{(\vec{\beta})} = B$, i.e., when $\alpha = M(A)$ and $\vec{\beta} = \vec{M}(B)$.

2.6. Analysis of the conditions

We now identify sufficient conditions for *Conditions 1–4* to hold, and hence, for validity of the above algorithm.

Conditions 1–3 state that in the calculation of the strain, the unlinked spikes in each train and the lengths of the pairings do not interact—i.e., that one cannot change the rank order of two strains by adding unlinked spikes or pairings. Strains of the form (1) necessarily satisfy these conditions. These conditions also hold for after generalizing the form (1) by applying any monotonically increasing function to each term.

Condition 4, which implies non-crossing of any efficient alignment, is somewhat subtler. As shown in Fig. 2, an alignment with intersecting links between A and $B[w]$ can always be untangled by a sequence of uncrossings: by uncrossing, at each

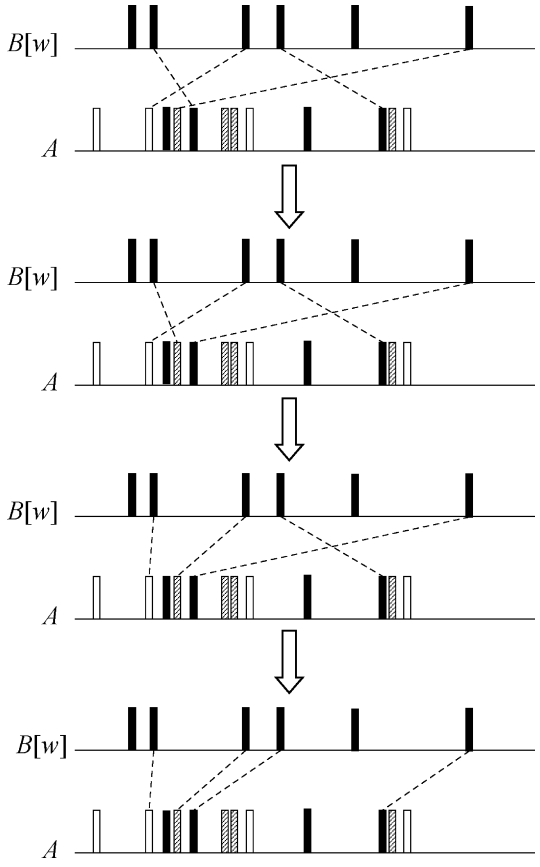

Fig. 2. A configuration of links between a multineuronal spike train A and a subtrain $B[w]$. The intersections can be eliminated one by one, by uncrossing a pair of links whose spikes in train A are the closest. The reduction of a general pattern of intersecting links by a sequence of single uncrossings does not depend on the spike labels, and does not place additional restrictions on the strain. Condition 4 requires that each single uncrossing does not increase the strain; this is further analyzed in Fig. 3.

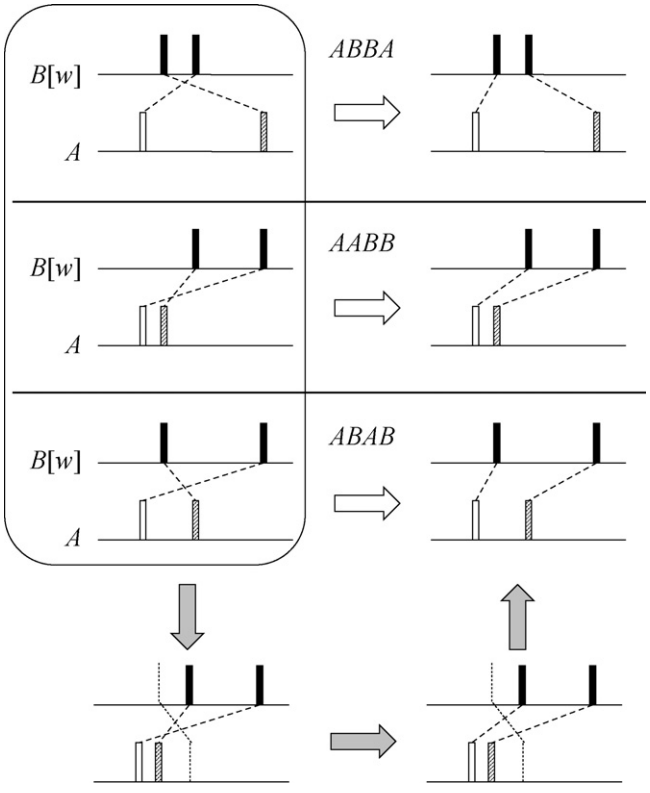

Fig. 3. The three configurations of intersections between pairs of links. For the first two configurations, the requirement that uncrossing does not increase the strain places constraints on Eq. (1). The sequence of transformations indicated by the gray arrows is used to demonstrate that the third configuration does not lead to additional constraints.

stage, a pair of links whose spikes in train A are the closest. This reduction from multiple uncrossings to single uncrossings does not place any restrictions on the strain, and does not depend on the spike labels. Thus, to guarantee *Condition 4*, it suffices to guarantee that single uncrossings never increase the strain. We now analyze the requirements for this.

Since the strain (Eq. (1)) depends only on the absolute value of the time differences associated with each link, we need only consider the sequences in which the spike in train A occurs first. The four spikes must occur in one of the following temporal configurations: *ABBA*, *AABB*, and *ABAB* (see Fig. 3).

For the first configuration (*ABBA*), uncrossing the links reduces the lengths of each link. Thus, *Condition 4* will be satisfied if each of the functions J_{w_1,w_2}, is non-decreasing. We assume that these functions are differentiable, so we may write

$$\frac{dJ_{w_1,w_2}(t)}{dt} \geq 0. \tag{3}$$

For the second configuration (*AABB*), uncrossing the links reduces the length of the longer link and increases the length of the shorter link, leaving the total link length unchanged. For constraints of the form (1), *Condition 4* requires that for any t

$$J_{w_i,w}(t-\tau) + J_{w_l,w}(t+\tau) \tag{4}$$

is a non-decreasing function of τ for $\tau \in [0, t]$, where w_i and w_l are the labels of the spikes in train A. Provided that these functions are differentiable, the condition that Eq. (4) is non-decreasing is equivalent to the condition that its derivative is non-negative:

$$-\frac{dJ_{w_i,w}(t)}{dt} + \frac{dJ_{w_l,w}(t)}{dt} \geq 0 \tag{5}$$

Since the inequality of Eq. (5) must also hold when the roles of w_i and w_l are reversed, we conclude that equality must hold. That is, $J_{w_i,w}(t)$ and $J_{w_l,w}(t)$ differ from each other by at most a constant:

$$J_{w_l,w}(t) = J_w(t) + c(w_l). \tag{6}$$

Since the expression (4) is non-decreasing, the inequality

$$J_{w_i,w}(t-\tau) + J_{w_l,w}(t+\tau) \geq J_{w_i,w}(t) + J_{w_l,w}(t) \tag{7}$$

must hold for all t and $\tau \in [0, t]$. Eq. (7) implies (with $t_- = t - \tau$, $t_+ = t + \tau$, and Eq. (6)) that

$$J_w(t_-) + J_w(t_+) \geq 2J_w\left(\frac{t_- + t_+}{2}\right), \tag{8}$$

i.e., that J_w is concave (Rudin, 1976). Provided that J_w has a second derivative, Eq. (8) is equivalent to

$$\frac{d^2 J_w}{dt^2} \geq 0, \tag{9}$$

Conversely, it is straightforward to see that Eqs. (6) and (9) are sufficient to ensure that the expression (4) is non-decreasing.

For the third configuration (*ABAB*), no further conditions are necessary if the conditions implied by the first two configurations are satisfied. This is because the uncrossing can be accomplished by a sequence of three steps, each of which is guaranteed not to increase the strain (lower half of Fig. 3). In step 1, the positions of the middle spikes are interchanged. That is, the first spike in train B is moved to the time of the second spike in train A, and the second spike in train A is moved to the time of the first spike in train B. This does not change the lengths of the links, and therefore does not change the strain. Since the spikes are now in the *AABB* configuration, the links can be uncrossed (step 2) without increasing the strain. In step 3, the middle spikes are moved back to their original position. This cannot increase the lengths of the links, and therefore (by Eq. (3)), cannot increase the strain.

In sum, we have found that for strains of the form (1), Eqs. (3), (6), and (9) are sufficient for validity of the dynamic programming algorithm stated above.

2.7. All-parameter calculation

In typical applications of spike metrics to characterize neural coding (Aronov et al., 2003; Samonds and Bonds, 2004; Victor, 2005; Victor and Purpura, 1997), it is necessary to compute measures of dissimilarity across a range of parameter values. These measures of dissimilarity are then examined to determine which parameter values provide the most faithful representation of the stimuli or conditions that elicited the neural responses. These parameters are then taken as characterizations of the neural code.

In a typical application, one might want to calculate $D^{\mathrm{spike}}[q, k]$ for several hundred pairs of values of q and k. More generally, one might want to calculate minimum strains of the form (1) for a variety of choices for the terms I_1, I_2, and J_{w_1,w_2}. In these situations, it may be possible to realize a substantial computational saving over the straightforward strategy of running the above algorithm once for each value of the parameters.

The basic strategy is as follows. First, we observe that for typical strains, it is not necessary to store the most efficient alignment explicitly at each stage of the above basic algorithm (i.e., for each pair of indices α and $\vec{\beta}$). Rather, one may be able to store merely a set of values that allow for calculation of the strain of the most efficient alignment, and of strains of larger alignments determined at later stages. Thus, for strains that are sums of contributions from their components (such as those of the form (1) with linear I_1, I_2, and J_{w_1,w_2}), it suffices to keep track of the strain of the most efficient alignment at each stage, rather than the alignment itself. This substantially decreases the storage requirements of the algorithm, since now only a scalar, rather than an alignment, needs to be stored for each pair of indices α and $\vec{\beta}$. As we show below, extending this idea yields a strategy to keep track of the strain of optimal alignments across *all* choices of the metric's parameters. This in turn leads to an algorithm for calculating measures of dissimilarity for all parameter values at once.

2.7.1. Single-neuron spike time metric

As a simple example, consider the single-neuron metric $D^{\mathrm{spike}}[q]$ and its dependence on a single parameter q, the cost per unit time for moving a spike. This metric can be expressed as the strain of an efficient alignment (Eqs. (1) and (2) with $L = 1$ and $k = 0$): the strain is the sum of the total number of unpaired spikes and the product of q and the total length of all links. At each stage of the algorithm, we keep track of all of the alignments that might be efficient for at least one value of q. Even though q can assume a continuum of values, this is always a discrete set: it is parameterized by the number of unpaired spikes, and, for each number of unpaired spikes, it is the alignment with the minimum total link length of all alignments that are constrained to have a particular number of unpaired spikes, U. Equivalently, this set can be parameterized by the total number of links, here denoted r, since every spike is either unpaired or part of a link. Moreover, the alignments themselves need not be stored; it suffices to store (for each pair of indices α and β) the minimum total link length $l(\alpha, \beta, r)$, of all alignments that have a total of r links. (Note that in the single-neuron case considered here, there is only one kind of label, and the vector index $\vec{\beta}$ can be replaced by a scalar index $\beta = \vec{\beta}_1$.)

The above dynamic programming structure works, since at each stage of the algorithm, $l(\alpha, \beta, r)$ depends only on $l(\alpha - 1, \beta, r)$ (add a terminal unpaired spike to A), $l(\alpha, \beta - 1, r)$ (add a terminal unpaired spike to B), and $l(\alpha - 1, \beta - 1, r - 1)$ (add a terminal linked pair). Once the final $l(M(A), M(B), r)$ has been calculated, then

$$D^{\mathrm{spike}}[q] = \min_r (M(A) + M(B) - 2r + q l(M(A), M(B), r)), \tag{10}$$

where $M(A) + M(B) - 2r$ is the number of unpaired spikes in an alignment of A and B with r links. The net result is an algorithm that calculates $D^{\mathrm{spike}}[q]$ for all values of q.

2.7.2. Implementation

The all-parameter algorithm iteratively determines a minimum total link length $l(\alpha, \beta, r)$ of all alignments between the first α spikes of A and the first β spikes of B that have a total of r links. These values constitute a three-dimensional array. However, to apply Eq. (10) to calculate the distance between the spike trains A and B, we only require $l(M(A), M(B), r)$, the values on a single edge of this array. This, coupled with the fact that $l(\alpha, \beta, r)$ depends only on $l(\alpha - 1, \beta, r)$, $l(\alpha, \beta - 1, r)$, and $l(\alpha - 1, \beta - 1, r - 1)$, points the way to a memory-efficient implementation that replaces the three-dimensional array $l(\alpha, \beta, r)$ by two-dimensional arrays $l_{\mathrm{cur}}(\alpha, \beta)$ and $l_{\mathrm{prev}}(\alpha, \beta)$ each of size $(M(A) + 1) \times (M(B) + 1)$. The outer loop of the iteration is on r, and the start of each iteration on r, the previously calculated l_{cur} becomes the current l_{prev}. The full calculation may be written as follows:

```
for r = 1,... min(M(A), M(B))
    l_prev ← l_cur
    for α = 1,... M(A)
        for β = 1,... M(B)
            l_cur(α, β) = min{ l_cur(α − 1, β), l_cur(α, β − 1), l_prev(α − 1, β − 1) + |A_α − B_β| }
        end β
    end α
end r
```

Initializations are accomplished by noting that $l_{\mathrm{cur}}(\alpha, \beta) = 0$ when $r = 0$, and omitting consideration of values of $l_{\mathrm{cur}}(\alpha, \beta)$ where it is undefined ($\alpha = 0$ or $\beta = 0$) during the inner computation.

2.7.3. Multineuronal metric

For the multineuronal metric $D^{\mathrm{spike}}[q, k]$, a somewhat more complicated procedure accomplishes the analogous goals. It suffices (see Eqs. (1) and (2)) to store the total number of unpaired

spikes, the total length of all links, and the total number of links between spikes of different labels. Equivalently, one may store $l(\alpha, \vec{\beta}, r, s)$, the minimum total link length of all alignments that have r links between spikes of the same label and s links between links of different labels. $l(\alpha, \vec{\beta}, r, s)$ depends only on $l(\alpha-1, \vec{\beta}, r, s)$, $l(\alpha, \vec{\beta}-\vec{1}_w, r, s)$, $l(\alpha-1, \vec{\beta}-\vec{1}_w, r-1, s)$, and $l(\alpha-1, \vec{\beta}-\vec{1}_w, r, s-1)$ with the latter three possibilities considered for each $w \in \{1, \ldots, L\}$. Once the final $l(M(A), \vec{M}(B), r, s)$ has been calculated, then

$$D^{\text{spike}}[q,k] = \min_{r,s}(M(A)+M(B)-2r - 2s + ks + ql(M(A), \vec{M}(B), r, s)), \quad (11)$$

where $M(A)+M(B)-2r-2s$ is the number of unpaired spikes in an alignment of A and B with r matched-label links and s unmatched-label links. The net result is an all-parameter algorithm that calculates $D^{\text{spike}}[q, k]$ for all values of q and k.

2.7.4. Implementation

Computation of the minimum total link lengths requires four nested loops, indexed by α, the number of matched links r, the number of unmatched links s, and $\vec{\beta}$. $\vec{\beta}$ is a multi-index, with one component for each label $w \in \{1, \ldots, L\}$, but it is kept as a single dimension with values ranging from 0 to $M(B) = \prod_{w=1}^{L} M(B[w])$.

The fact that $l(\alpha, \vec{\beta}, r, s)$ depends only on $l(\alpha-1, \vec{\beta}, r, s)$, $l(\alpha-1, \vec{\beta}-\vec{1}_w, r-1, s)$, $l(\alpha-1, \vec{\beta}-\vec{1}_w, r, s-1)$, and $l(\alpha, \vec{\beta}-\vec{1}_w, r, s)$ can be used to achieve a memory-efficient implementation. We require two-dimensional arrays, $l_{\alpha\text{prev, rcur}} = l(\alpha-1, \bullet, r, \bullet)$ and $l_{\alpha\text{prev, rprev}} = l(\alpha-1, \bullet, r-1, \bullet)$ each of size $(S+1) \times (M(B)+1)$, and one three-dimensional array l_{temp} of size $(R+1) \times (S+1) \times (M(B)+1)$, where R is the maximum number of matched links, and S is the maximum number of unmatched links.

At the start of each α-iteration, the contents of $l_{\text{temp}}(0, \bullet, \bullet)$ are copied into $l_{\alpha\text{prev, rcur}}$. At the start of each r-iteration, the contents of $l_{\alpha\text{prev, rcur}}$, are copied into $l_{\alpha\text{prev, rprev}}$, and the contents of $l_{\text{temp}}(r, \bullet, \bullet)$ are copied into $l_{\alpha\text{prev, rcur}}$. For a given r', $l_{\text{temp}}(r', \bullet, \bullet)$ corresponds to $l(\alpha-1, \bullet, r, \bullet)$ for $r < r'$ and to $l(\alpha, \bullet, r, \bullet)$ for $r > r'$. The computation may be written as follows:

for $\alpha = 1, \ldots, M(A)$

$l_{\alpha prev,rcur} \leftarrow l_{temp}(0, \bullet, \bullet)$

for $r = 0, \ldots, R$

$l_{\alpha prev,rprev} \leftarrow l_{\alpha prev,rcur}$

$l_{\alpha prev,rcur} \leftarrow l_{temp}(r, \bullet, \bullet)$

for $s = 0, \ldots, S$

for $\vec{\beta} = 1 \ldots \vec{M}(\beta)$

if $a_\alpha = b[w]_{\beta_w}$

% A-spike and B-spike have same label

$$l_{temp}(r,s,\vec{\beta}) = \min\left\{l_{\alpha prev,rcur}(s,\vec{\beta}), l_{temp}(r,s,\vec{\beta}-\vec{1}_w), l_{\alpha prev,rprev}(s,\vec{\beta}-\vec{1}_w) + \left|A_{\alpha-1} - B_{\vec{\beta}-\vec{1}_w}\right|\right\}$$

else

% A-spike and B-spike have different labels

$$l_{temp}(r,s,\vec{\beta}) = \min\left\{l_{\alpha prev,rcur}(s,\vec{\beta}), l_{temp}(r,s,\vec{\beta}-\vec{1}_w), l_{\alpha prev,rcur}(s-1,\vec{\beta}-\vec{1}_w) + \left|A_{\alpha-1} - B_{\vec{\beta}-\vec{1}_w}\right|\right\}$$

endif

end $\vec{\beta}$

end s

end r

end α

Initializations are accomplished by noting that $l(\alpha, \vec{\beta}, 0, 0) = 0$, and by making exceptions for out-of-range values in the inner computation.

2.7.5. Comparison: basic algorithm versus all-parameter algorithm

As described above, the computation of $D^{\mathrm{spike}}[q]$ and $D^{\mathrm{spike}}[q, k]$ can be carried out either by the basic algorithm for each set of values of the metric parameters, or by the all-parameter algorithm, in which the minimal link lengths $l(M(A), M(B), r)$ or $l(M(A), \vec{M}(B), r, s)$ are calculated and then $D^{\mathrm{spike}}[q]$ and $D^{\mathrm{spike}}[q, k]$ are determined via Eqs. (10) or (11). In the first strategy, the calculation time scales as M^{L+1}, where M is the typical number of spikes of each label, and L is the number of labels (here, we are assuming that each label occurs on approximately the same number of spikes). Thus the length of the α-loop is LM, and the length of the multi-index $\vec{\beta}$-loop is M^L. At each stage of the iteration, $1 + 2L$ possibilities need to be considered (unlinked last spike in train A, unlinked last spike in one of the L subtrains $B^{[w]}$, or a link between the last spike from train A and the last spike of one of the L subtrains $B^{[w]}$). The calculation needs to be repeated for each set of values of the metric parameters. Thus, we anticipate a computational time T_{basic} that scales as

$$T_{\mathrm{basic}} \approx K_{\mathrm{basic}} P(1 + 2L) L M^{L+1} \tag{12}$$

where P is the number of sets of values of the metric parameters, and K_{basic} is a proportionality constant that depends on hardware, implementation, and the "shape" of the data (e.g., the typical temporal patterns of spikes in time).

For the all-parameter strategy, we consider $D^{\mathrm{spike}}[q]$ and $D^{\mathrm{spike}}[q, k]$ separately. For $D^{\mathrm{spike}}[q]$, the time required to calculate the link lengths $l(M(A), M(B), r)$ scales as $3M^3$, since the length of each of the α-, β-, and r-loops is typically M (we include the factor of 3 to represent the three alternatives considered in the inner loop, for consistency with the analysis of the multineuronal metric, below). This stage of the calculation is independent of the number of sets of metric parameters. Following calculation of the link lengths, distances are calculated via Eq. (10). The time for this calculation is proportional to PM, since a maximum over each of the M link lengths must be determined for each of the P values of q. Thus, the computation time $T_{\mathrm{allparameter}}$ is anticipated to scale as

$$T_{\mathrm{allparameter}} \approx 3K_{\mathrm{links}} M^3 + K_{\mathrm{dists}} PM, \tag{13}$$

where K_{links} and K_{dists} are proportionality constants analogous to K_{basic}. For $D^{\mathrm{spike}}[q, k]$, the time required to calculate the link lengths $l(M(A), \vec{M}(B), r, s)$ scales as $(1 + 2L)L^2M^{L+3}$, using LM as a conservative estimate of the length of the s-loop, the previous estimates of LM for the length of the α-loop, M^L for the $\vec{\beta}$-loop, M for the r-loop, and a factor of $1 + 2L$ to indicate the dependence of the number of inner-loop comparisons on L. Thus, the computation time is anticipated to scale as

$$T_{\mathrm{allparameter}} \approx K_{\mathrm{links}}(1 + 2L)L^2 M^{L+3} + K_{\mathrm{dists}} PLM^{L+1}. \tag{14}$$

In Eqs. (13) and (14), we anticipate that $K_{\mathrm{links}} \gg K_{\mathrm{dists}}$, since the distance calculation only requires a multiplication, followed by finding the minimum of a list. For the same reason, we anticipate that K_{basic} (Eq. (12)) will be much larger than K_{dists}. That is, in comparison to Eq. (12), both Eqs. (13) and (14) have a term that is large and independent of the number of parameter pairs P, followed by a term that depends on P but has a much shallower growth. Thus, for sufficiently large P, the all-parameter algorithm will be more efficient than the basic algorithm.

We can estimate the asymptotic behavior of the breakeven point $P_{\mathrm{breakeven}}$ as follows. For $D^{\mathrm{spike}}[q]$, set $T_{\mathrm{basic}} = T_{\mathrm{allparameter}}$ in Eqs. (12) (with $L = 1$) and (13), to obtain, for large M,

$$P_{\mathrm{breakeven}} \approx M \frac{K_{\mathrm{links}}}{K_{\mathrm{basic}}}. \tag{15}$$

For $D^{\mathrm{spike}}[q, k]$, set $T_{\mathrm{basic}} = T_{\mathrm{allparameter}}$ in Eqs. (12) and (14) to obtain

$$P_{\mathrm{breakeven}} \approx LM^2 \frac{K_{\mathrm{links}}}{K_{\mathrm{basic}} - K_{\mathrm{dists}} L/(1 + 2L)}. \tag{16}$$

2.7.6. Examples

As a test case, we applied both algorithms to responses from a pair of neurons in primary visual cortex of a macaque monkey (recording 410106.st of (Aronov et al., 2003)). Stimuli were transient presentations of stationary luminance gratings; 64 responses to each of 16 spatial phases were recorded. Thus, there were 1024 responses (each containing $L = 2$ labels) in the dataset, and approximately 500,000 pairs of distances to calculate for each set of metric parameters q and k. On average, there were $M = 12.7$ (mean) spikes per neuron, and no response had more than 20 spikes.

The algorithm implementations were those used in the open source Spike Train Analysis Toolkit (http://neuroanalysis.org). The algorithms are coded in C and provided with a Matlab interface. The analysis was performed on a Dell Precision 450 with 1 GB RAM and a single-core 2.8 GHz Xeon processor.

Runtime measurements for $D^{\mathrm{spike}}[q]$ are depicted in Fig. 4A. The tradeoff is at $P = 2$; for larger values of P, the all-parameter algorithm is more efficient. For $D^{\mathrm{spike}}[q, k]$ (Fig. 4B), the tradeoff is at $P_{\mathrm{breakeven}} \approx 36$, a value that is substantially higher because of the computations required to obtain the link lengths. Nevertheless, this is well within the range of values typically required for data analyses.

Note that the all-parallel algorithm increases storage requirements by a factor of $O(M)$ for the single-neuron metrics and by $O(M^2)$ for multineuronal metrics. For typical neuroscience uses of the algorithm (as in the multineuronal example above), this additional storage is readily handled by a desktop PC.

We also ran calculations to illustrate how run time depends on the number of neurons for $L > 2$ neurons. We generated pairs of multineuron responses with spike times randomly distributed in a uniform fashion over a 0.5 s interval. The calculations were performed using the basic algorithm, and they were parameterized by the number of spikes per train with each label, M, and the number of labels (neurons), L. The run times are depicted

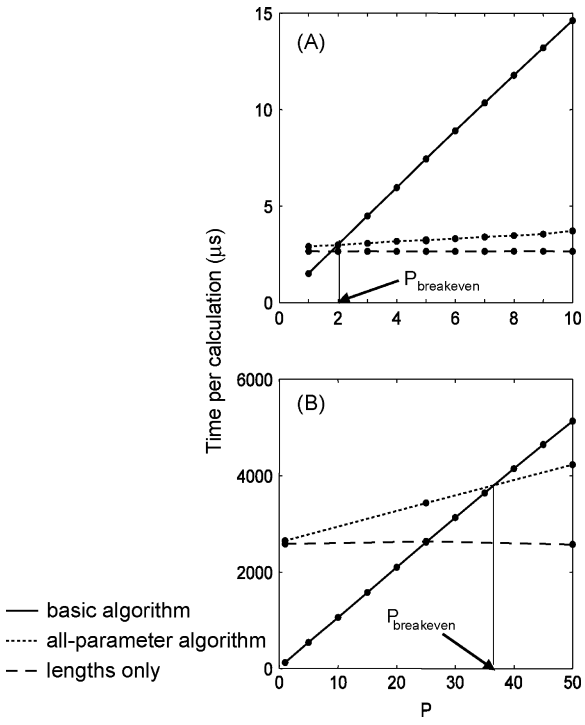

Fig. 4. Comparison of computation times for the basic algorithm, estimated by Eq. (12), and the all-parameter algorithms, estimated by Eq. (13) for $D^{\text{spike}}[q]$ (panel A) and Eq. (14) for $D^{\text{spike}}[q, k]$ (panel B). The dataset consists of neuronal responses in primary visual cortex of a macaque monkey elicited by transient presentations of stationary gratings of varying spatial phase (Aronov et al., 2003). (A) Unit 410106.s of (Aronov et al., 2003). (B) Unit 410106.s and 410106.t of (Aronov et al., 2003). The abscissa, P, is the number of values for the parameter q(panel A) or of (q, k)-pairs (panel B). The "lengths-only" plot is runtime of the portion of the algorithm that determines $l(M(A), \bar{M}(B), r, s)$, corresponding to the term in Eqs. (13) and (14) that depends on K_{links}. $P_{\text{breakeven}}$ is breakeven point at which the all-parameter algorithm becomes more efficient than the basic algorithm.

on a log–log plot in Fig. 5. Once the run times exceed 1 ms, the predicted $O(M^{L+1})$ behavior (Eq. (12)) is observed.

Although the asymptotic regime is not reached, the breakeven point $P_{\text{breakeven}}$ at which the all-parameter algorithm becomes more efficient than the basic algorithm increases with M and

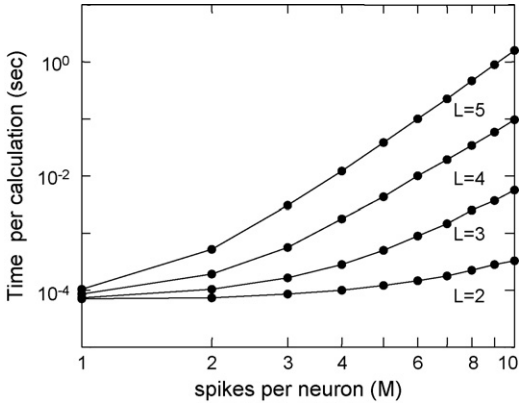

Fig. 5. Dependence of computation times for the basic algorithm on the number of spikes per neuron, M, train and the number of neurons, L. Spikes are distributed uniformly in time.

L, as expected by Eq. (16): $P_{\text{breakeven}} \approx 25$ for $M = 4$, $L = 3$; $P_{\text{breakeven}} \approx 300$ for $M = 10$, $L = 3$; $P_{\text{breakeven}} \approx 60$ for $M = 4$, $L = 4$; and $P_{\text{breakeven}} \approx 450$ for $M = 10$, $L = 4$.

Thus, the basic algorithm may be more efficient than the all-parameter algorithm for modest spike trains and/or a relatively small number of parameter values P. The breakpoint is likely to be data-, implementation-, and hardware-dependent, and it may be useful to carry out pilot analyses prior to large-scale calculations to determine which approach is faster.

3. Discussion

We have considered notions of dissimilarity that can be applied to compare multineuronal responses, viewed as sequences of labeled events. Our main result is that the dynamic programming algorithm (Aronov, 2003) for $D^{\text{spike}}[q, k]$ can be extended to calculate measures of dissimilarity based on minimizing the "strain" of an alignment. The four conditions that the strain must satisfy are simple and relatively intuitive. The first three conditions specify that adding unlinked spikes or a linked pair of spikes to alignments cannot influence their rank order. The fourth condition is that strain is never increased by uncrossing two links, and also expresses the notion that one long link is never better than two short links of the same total length. For strains of the form (1), these conditions are expressed by three requirements (Eqs. (3), (6) and (9)) for the penalty $J_{w_1,w_2}(t)$ assigned to a link of length t between spikes of labels w_1 and w_2. The form (1) not only includes the standard spike time metrics $D^{\text{spike}}[q]$ and $D^{\text{spike}}[q, k]$ (Victor, 2005; Victor and Purpura, 1996, 1997), but also variants in which the component of the strain associated with unpaired spikes ($I_{1,w}$ and $I_{2,w}$ in Eq. (1)) depends on the label w, and strains which increase as an accelerating function of distance. These latter strains violate the triangle inequality. They therefore provide notions of spike train sequence comparisons that are more general than those based on metrics, which may be helpful in understanding neural coding of non-metric perceptual phenomena (Maloney and Yang, 2003; Tversky, 1977; Tversky and Gati, 1982).

3.1. Relationship to algorithms for biologic sequence comparison

The algorithms discussed here are similar to the dynamic programming algorithms for biologic sequence comparison (Needleman and Wunsch, 1970; Sellers, 1974), both in their goal and structure. However, the continuous nature of spike times adds a distinctive flavor. One aspect of this is that for biologic sequences, an increase in efficiency can be obtained (Arlazarof et al., 1970; Myers, 1992) by breaking down each spike train into small subsequences, and using a lookup table to calculate the distances between these subsequences. For spike trains, this approach (known informally as the "Four Russians" speedup), is not readily applicable, unless one is willing to calculate only approximate distances by binning spike times coarsely. The general "divide and conquer" strategy (e.g., Grice et al., 1997; Tonges et al., 1996) often used to improve the efficiency of sequence alignments is moot here, since we are not interested

in identifying the optimal alignment *per se*, nor in aligning more than a single pair of spike trains. However, extensions of these algorithms to identify the "consensus" sequence among several spike trains might benefit from such an approach.

Conversely, certain aspects of the sequence comparison for spike trains may suggest efficiencies that are of use in sequence comparisons more generally. In particular, for spike train analysis, it is often useful to explore a parametric family of metrics, to determine which metrics provide the most faithful or reliable separation of neural responses. In phylogenetic analysis, it can be useful to compare genetic sequences according to a family of parametrically related distances, which differ by the relative costs of the six possible substitutions among the four nucleotides (Bos and Posada, 2005). By exploiting the manner in which strain depends on the parameter values, it is possible to keep track of all efficient alignments for the *entire* parametric family of strains. As we show here, this can achieve a substantial computational saving.

Acknowledgments

The authors thank John Zollweg at the Cornell Theory Center for assistance with optimizing the Matlab implementation of the $D^{\mathrm{spike}}[k, q]$ algorithm, and Lucy A. Skrabanek for helpful comments related to genetic sequence analysis. This work was supported by Human Brain Project/Neuroinformatics MH68012 from NIMH to DG and by EY9314 from NEI to JDV.

References

Arlazarof VL, Dinic EA, Kronrod MA, Faradzev IA. On economic construction of the transitive closure of a directed graph. Dokl Acad NaukSSR 1970;194:487–8.

Aronov D. Fast algorithm for the metric-space analysis of simultaneous responses of multiple single neurons. J Neurosci Methods 2003;124(2):175–9.

Aronov D, Reich DS, Mechler F, Victor JD. Neural coding of spatial phase in V1 of the macaque monkey. J Neurophysiol 2003;89:3304–27.

Aronov D, Victor JD. Non-Euclidean properties of spike train metric spaces. Phys Rev E Stat Nonlin Soft Matter Phys 2004;69(6 Pt 1):061905.

Bos DH, Posada D. Using models of nucleotide evolution to build phylogenetic trees. Dev Comp Immunol 2005;29(3):211–27.

Cover TM, Thomas JA. Elements of information theory. In: Wiley series in telecomunications. New York: Wiley; 1991. p. 542.

Goldberg DH, Victor JD, Gardner, D. The spike train analysis toolkit: a component of a computational neuroinformatic resource. Society for Neuroscience: Washington, DC; 2005 (p. Program No. 984.19. 2005).

Gray CM, Maldonado PE, Wilson M, McNaughton B. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. J Neurosci Methods 1995;63(1–2):43–54.

Grice JA, Hughey R, Speck D. Reduced space sequence alignment. Comput Appl Biosci 1997;13(1):45–53.

Hopfield JJ. Pattern recognition computation using action potential timing for stimulus representation. Nature 1995;376(6535):33–6.

Kralik JD, Dimitrov DF, Krupa DJ, Katz DB, Cohen D, Nicolelis MA. Techniques for long-term multisite neuronal ensemble recordings in behaving animals. Methods 2001;25(2):121–50.

Kullbach S, Leibler R. On information and sufficiency. Ann Math Stat 1951;22:79–86.

Maloney LT, Yang JN. Maximum likelihood difference scaling. J Vis 2003;3(8):5, doi:10.1167/3.8.5.

Myers G. A four Russians algorithm for regular expression pattern matching. J Assoc Comput Mach 1992;39(4):430–48.

Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 1970;48(3):443–53.

Nicolelis MA, Dimitrov D, Carmena JM, Crist R, Lehew G, Kralik JD, et al. Chronic, multisite, multielectrode recordings in macaque monkeys. Proc Natl Acad Sci USA 2003;100(19):11041–6.

Rieke F, Warland D, de Ruyter van Steveninck RR, Bialek W. Spikes: exploring the neural code. Cambridge, MA: MIT Press; 1997.

Rudin W. Principles of mathematical analysis. New York: McGraw-Hill; 1976. p. 325.

Samonds JM, Bonds AB. From another angle: differences in cortical coding between fine and coarse discrimination of orientation. J Neurophysiol 2004;91(3):1193–202.

Segundo JP, Perkel DH. The nerve cell as an analyzer of spike trains. In: Brazier MAB, editor. The interneuron. Berkeley: University of California Press; 1969. p. 349–90.

Sellers P. On the theory and computation of evolutionary distances. SIAM J Appl Math 1974;26:787–93.

Tonges U, Perrey SW, Stoye J, Dress AW. A general method for fast multiple sequence alignment. Gene 1996;172(1):GC33–41.

Tversky A. Features of similarity. Psychol Rev 1977;84(4):327–52.

Tversky A, Gati I. Similarity, separability, and the triangle inequality. Psychol Rev 1982;89(2):123–54.

Victor JD. Spike train metrics. Curr Opin Neurobiol 2005;15(5):585–92.

Victor JD, Purpura KP. Nature and precision of temporal coding in visual cortex: a metric-space analysis. J Neurophysiol 1996;76(2):1310–26.

Victor JD, Purpura KP. Metric-space analysis of spike trains: theory, algorithms and application. Network 1997;8:127–64.

Wuerger SM, Maloney LT, Krauskopf J. Proximity judgments in color space: tests of a Euclidean color geometry. Vis Res 1995;35(6):827–35.