On the computation and application of prototype point patterns

Katherine E. Tranbarger[1] and Frederic Paik Schoenberg[1]

Abstract

This work discusses computational problems related to the implementation of Victor and Purpura's spike time distance metric for point processes. Properties of the metric and extensions of its use are investigated. These extensions include prototype point patterns that can be used for describing a typical point pattern and various clustering algorithms that can be applied to point process data through use of spike-time distance and prototype patterns.

Key words: distance metric, point process, prototype pattern.

[1] Department of Statistics, 8125 Math-Science Building, University of California, Los Angeles, 90095-1554.

1 Introduction

Methods involving distance metrics for point patterns have received increasing use recently. Such methods have had particularly important applications in the analysis of collections of neuron spike trains (Victor and Purpura 1997). In addition, point pattern distance metrics are required for the computation of a prototype point pattern, a construct introduced in Schoenberg and Tranbarger (2004) and shown to be useful in the description of earthquake aftershock sequences. The present paper investigates the computation and implementation of distance metrics, especially in the context of their use in prototype point pattern analysis.

The spike-time distance metric proposed by Victor and Purpura (1997) involves matching points in one point pattern to the points in another point pattern. The metric is somewhat analogous to several distance metrics currently in use in computer imaging and other areas. For instance, the concept of Earth Movier's Distance (EMD) introduced by Rubner, Tomasi, and Guibas (1998) evolved from the Hitchcock (1941) solution to the original transportation problem first discussed by Monge (1781). EMD measures the amount of work required to transform a histogram of the values in one image into that of the values in another image using basic operations (see Rubner et al., 2000), and was shown by Levina and Bickel (2004) to be equivalent to Mallows distance on probability distributions when the two image signatures in question are appropriately weighted by their sizes.

In contrast to EMD, the spike-time distance of Victor and Purpura (1997) focuses on matching the points of the two processes rather than their summary histograms. In addition, unlike EMD, the spike-time distance can be used to compare two point patterns of unequal length, and no modification involving only partial matching is required. Indeed, the difference in the number of points is a key ingredient in spike-time distance.

In Section 2, we review spike-time distance and explore various properties that aid in its computation. Prototype point patterns are described in Section 3 and issues in their computation and approximation are discussed. Algorithms for clustering collections of observed point patterns based on their spike-time distances and prototype point patterns are discussed in Section 4. A discussion and suggestions for further research are presented in Section 5.

2 Calculation of spike-time distance

Let X and Y be temporal point patterns, i.e. each is a collection of points on the real line, and corresponds to a σ-finite non-negative integer-valued measure on $\mathbf{R}$ (see Daley and Vere-Jones 2003). (We refer to such a collection of points as a point *pattern*, as distinguished from a point *process*, which is a random variable whose outcomes are point patterns.)

The spike-time distance between X and Y is defined as the total cost of transforming X into Y using a series of basic operations (Victor and Purpura, 1997). Points from X can be deleted at a cost p_d, added at a cost of p_a, or moved horizontally a distance Δ at a cost of $p_m\Delta$. The sum of costs for the deletion, addition, and moving operations necessary to transform point pattern X into point pattern Y is the spike-time distance between X and Y. Note that in order for this to be a distance metric, it must be symmetric and thus the constraint $p_a = p_d$ must be imposed. While in some applications it might be desirable for p_a and p_d to take on different values, this paper focuses on the case where the spike-time distance is a symmetric distance metric. Hence in what follows we assume $p_a = p_d$.

2.1 Basic Properties

The minimal sequence of operations required to transform point pattern X into point pattern Y can be difficult to determine. In the special case where no points of X or Y are added or deleted, however, the computation is quite trivial in view of the following result.

Theorem 1. Suppose that temporal point patterns X and Y each consist of exactly n points and that $p_m << p_a = p_d$, so that in effect addition and deletion of points are not permitted. Then

$$d(X,Y) = p_m \sum_{i=1}^{n} |x_i - y_i|, \tag{1}$$

where $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ are the sorted points of X and Y, respectively.

Proof.

The statement is trivial for $n = 1$.

Suppose $n = 2$, and without loss of generality assume $x_1 \geq y_1$.

If also $x_2 \geq y_2$, then

$$|x_1 - y_2| + |x_2 - y_1| \geq |x_1 - y_1 + x_2 - y_2| = |x_1 - y_1| + |x_2 - y_2|,$$

since both $(x_1 - y_1)$ and $(x_2 - y_2)$ are non-negative.

Alternatively, if $x_2 < y_2$, then $y_2 - x_1 \geq y_2 - x_2 > 0$, since $x_1 \leq x_2 < y_2$. Similarly, $0 \geq y_1 - x_1 \geq y_1 - x_2$. Therefore, $|y_1 - x_1| \leq |y_1 - x_2|$ and $|y_2 - x_2| \leq |y_2 - x_1|$, so $|y_1 - x_1| + |y_2 - x_2| \leq |y_1 - x_2| + |y_2 - x_1|$.

Hence the basic operations that move the point x_1 to y_2 and move the point x_2 to y_1

have a total penalty that is at least as large as the penalties for moving x_1 to y_1 and moving x_2 to y_2.

Now suppose $n = k + 1$ and that the result in Theorem 1 holds for $n = k$. Consider any sequence of elementary operations that includes moving x_{k+1} to y_i and moving x_j to y_{k+1}, where $i, j < k + 1$. By the $n = 2$ case proven above, $|x_{k+1} - y_i| + |x_j - y_{k+1}| \geq |x_j - y_i| + |x_{k+1} - y_{k+1}|$. That is, the elementary operations considered have cost at least as large as those obtained by moving x_{k+1} to y_{k+1} and moving x_j to y_i. Hence the minimal total cost in aligning X and Y is obtained by the elementary operations that involve moving x_{k+1} to y_{k+1}; the cost of aligning the other k points of X with the other k points of Y is given by (1) with $n = k$. The result folllows by induction. $\square$

In light of Theorem 1, the problem of determining the best sequence of operations to transform point pattern X into point pattern Y reduces to the problem of determining whether each point in the point patterns will be *removed* (i.e. deleted from one string or equivalently added to the other), or whether it will be *kept*, i.e. *paired* to a point in the other string. Once the points to be kept are determined, the best approach for matching the kept points in X to those in Y is simply sequential, based on Theorem 1. Evaluation of each potential set of kept points is thus remarkably straightforward as each potential distance is simply the number of points removed summed with the integrated difference between the cumulative functions associated with the remaining points in the two temporal point patterns (Schoenberg and Tranbarger 2004).

Given two temporal point patterns X and Y whose spike-time distance is sought, the following two results are useful in determining which points will be kept and which will be removed.

Lemma 2. Any point that is more than $(p_a + p_d)/p_m = 2p_d/p_m$ away from its nearest neighbor in the alternate point pattern will be removed.

Proof. The proof is immediate: for such a point x in X, it is more costly to move x to a point y in Y than to delete x and add to X a point at y. □

Theorem 3. Any pair of points x_i and y_j such that

$$|x_i - y_j| < \min_{k \neq i} \{|x_k - y_j|\} < \min_{k \neq j} \{|x_i - y_k|\} < 2p_d/p_m \tag{2}$$

will be kept.

Proof. There are three possible outcomes for points x_i and y_j satisfying the condition (2). Either both points are removed, one point is kept while the other is removed, or both points are kept. We will show that the first two of these outcomes are excluded.

Suppose that both points are removed. Then the distance penalty function includes the penalty $2 * p_d$ associated with those deletions. Since $|x_i - y_j| < 2 * p_d/p_m$, the penalty could be reduced by keeping both points and moving them to each other, at a cost of $|x_i - y_j| * p_m$. Therefore, removing both x_i and y_j cannot yield the minimum penalty.

Suppose that one point is kept while the other is removed. Assume without loss of generality that x_i is kept while y_j is removed. Then x_i is paired with some point y_k such that $y_k \neq y_j$ and the spike-time distance includes both penalty p_d for removal of y_j and $|x_i - y_k| * p_m$ for the move of point x_i to y_k. Since $|x_i - y_k| > |x_i - y_j|$ the total spike-time distance penalty could be reduced by removing point y_k rather than point y_j and moving x_i to y_j. Therefore, removing one point while keeping the other cannot yield the minimum penalty.

The only option that remains is that both x_i and y_j are kept in the sequence of moves yielding minimal total cost. □

Using these three properties of the optimal sequence of operations in the spike-time distance metric, the problem of determining the distance between process X and process Y is simplified to identifying points known to be kept or deleted, then checking the total cost associated with each possible combination of potentially kept points. Calculating the total cost associated with each potential combination of kept points in the two point patterns is not computationally time prohibitive, as the sequential ordering prescribed by Theorem 1 makes this calculation extremely straight-forward.

2.2 Application and Penalty Selection

It is not uncommon in point process applications to observe a collection of independent realizations of point processes observed on a common space. An example discussed in Schoenberg and Tranbarger (2004) is the collection observed times of aftershocks from mainshocks in different regions, over a fixed period of time following the mainshock. Alternatives include the origin dates of wildfires in a region, where each year is viewed as a separate point pattern. Throughout this work, we refer to such a collection of point processes simply as a *point process dataset*.

The number and proximity of points in each point pattern observed in a point process dataset are important to consider when determining the penalties used in the distance metric. When selecting these penalties, it is the ratio of the deletion (addition) penalty p_d to p_m that governs the results. For simplicity, we suggest setting $p_a = p_d = 1$ and determining p_m by examining the spread of the points in the data. If p_m is too large, then little movement of points will take place in the computation of distances between point patterns, as the cost of deletion and addition will be less than most potential moves. Alternatively, if p_m is too small, then moves will be made between points that are not very close at all. With either extreme, the resulting spike-time distance will measure little more than the sum of, or difference between, the number of points in the two observed point patterns. That is, the spike-time distances will approach the sum of the two point pattern lengths for very large values of p_m, and will approach the absolute difference between these lengths for very small p_m values.

With these extremes in mind, one option is to set p_m to a value such that points closer than the the typical inter-point distance are paired, and points further than this are removed and reintroduced in calculating the spike time distance. This leads to selecting penalties such that:

$$(T/M)p_m = 2p_d \tag{3}$$

where M is the median number of points in the observed point patterns.

3 Prototype point patterns

With a distance metric clearly defined, it becomes possible to identify a prototype point patern that can be used for describing a typical observation within the point process dataset. We define this prototype to be the point pattern Y such that the sum

$$\sum_{i=1}^{n} d(X_i, Y), \tag{4}$$

is minimized, where X_i, $i = 1...n$, are the n observed point patterns in the dataset.

3.1 Basic properties of prototype points

Fortunately, one need not search over all possible point patterns in determining the prototype for a given point process dataset. A convenient feature making prototypes easy to identify is described in the next result. Before stating this fact, we first turn to the definition of the median of a sorted list of numbers $\mathbf{z} = \{\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_m}\}$, whose length m is even. Many texts define the median of such a list as the mean of the two entries $z_{m/2}$ and $z_{m/2+1}$. Instead, let us refer to any value M such that $z_{m/2} \leq M \leq z_{m/2+1}$ as *a* median of $\mathbf{z}$.

With this convention in mind, we return to the problem of determining prototypes. Suppose that Y is the prototype of a point process dataset consisting of n point patterns $X_1, \ldots, X_n$. For any point p in the prototype, consider the collection of points in the point process dataset $\mathbf{z_p} = \{z_1, z_2, \ldots, z_m\}$ to which p is paired. That is, each point z_i is the point to which p is moved in determining the spike-time distance between Y and X_j, for some j. Note that $m \leq n$, since p might not be kept in the spike-time distance between the prototype and some of the point patterns in the dataset.

Theorem 4. Any point p in the prototype is a median of $\mathbf{z_p}$.

Proof. Fix any prototype point p and the list $\mathbf{z_p} = \{z_1, \ldots, z_m\}$ of points in the dataset to which p is paired. Note that, in order for p to be a point of the prototype, the sum of distances from z_i to p must be less than or equal to the sum of distances from z_i to any other point q. Let q be a median of $\mathbf{z_p}$, and suppose that p is not a median of $\mathbf{z_p}$. We will show that the sum of distances from z_i to q is less than the sum of distances from from z_i to p, contradicting the assumption that p is a point of the prototype.

First suppose that the length m of $\mathbf{z_p}$ is odd, and without loss of generality assume $p < q$.

The sum of all m distances from z_i to q is:

$$\begin{aligned}\sum_{i=1}^{m}|z_i - q| &= \sum_{i=1}^{(m-1)/2}|z_i - q| + \sum_{i=(m+3)/2}^{m}|z_i - q| + |z_{(m+1)/2} - q| \\ &= \sum_{i=1}^{(m-1)/2}|z_i - q| + \sum_{i=(m+3)/2}^{m}|z_i - q| \qquad (5)\end{aligned}$$

since $|z_{(m+1)/2} - q| = 0$. The sum of the m distances from z_i to p is:

$$\begin{aligned}\sum_{i=1}^{m}|z_i - p| &= \sum_{i=1}^{(m-1)/2}(|z_i - q| - (q-p)) + \sum_{i=(m+3)/2}^{m}(|z_i - q| + (q-p)) + (z_{(m+1)/2} - p) \\ &= \sum_{i=1}^{m}|z_i - q| - (\frac{m-1}{2})(q-p) + (\frac{m-1}{2})(q-p) + (z_{(m+1)/2} - p) \\ &= \sum_{i=1}^{m}|z_i - q| + (z_{(m+1)/2} - p). \qquad (6)\end{aligned}$$

Therefore, the sum of distances from z_i to p in (6) is greater than the sum of distances to q, which is a contradiction.

If m is even, then without loss of generality assume that $p < z_{m/2}$. The sum of all m distances from z_i to q is:

$$\begin{aligned}\sum_{i=1}^{m}|z_i - q| &= \sum_{i=1}^{(m/2)-1}(|z_i - z_{(m/2)}| + (q - z_{(m/2)})) + \sum_{i=(m/2)}^{(m/2)+1}(|z_i - q|) \\ &\quad + \sum_{i=(m/2)+2}^{m}(|z_i - z_{(m/2)+1}| + (z_{(m/2)+1} - q)) \\ &= \sum_{i=1}^{(m/2)-1}(|z_i - z_{(m/2)}|) + (z_{(m/2)+1} - z_{m/2}) + \sum_{i=(m/2)+2}^{m}(|z_i - z_{(m/2)+1}|) \\ &\quad + (\frac{m}{2}) * (z_{(m/2)+1} - z_{m/2}) \qquad (7)\end{aligned}$$

and the sum of distances from z_i to p is:

$$\begin{aligned}
\sum_{i=1}^{m} |z_i - p| = & \sum_{i=1}^{(m/2)-1} (|z_i - z_{(m/2)}| - (z_{m/2} - p)) + \sum_{i=(m/2)}^{(m/2)+1} (|z_i - p|) \\
& + \sum_{i=(m/2)+2}^{m} (|z_i - z_{(m/2)+1}| + (z_{(m/2)+1} - p)) \\
= & \sum_{i=1}^{(m/2)-1} (|z_i - z_{(m/2)}|) + (z_{(m/2)+1} - z_{m/2}) + (z_{m/2} - p) + \sum_{i=(m/2)+2}^{m} (|z_i - z_{(m/2)+1}|) \\
& + (\frac{m}{2} - 1) * (z_{(m/2)+1} - z_{m/2}) \\
= & \sum_{i=1}^{m} |z_i - q| + (z_{m/2} - p), \qquad (8)
\end{aligned}$$

which again contradicts the assumption that p is a point of the prototype. $\square$

A consequence of Theorem 4 is that, for any point process dataset, there exists a prototype made up entirely of points observed in the dataset. That is, in searching for a prototype, one may limit one's search to all possible combinations of entries in the dataset.

3.2 Penalty selection considerations

Penalties selected for p_m, p_a, and p_d play a significant role in prototype determination. As discussed in Schoenberg and Tranbarger (2004), a point at time t can only be part of the prototype if at least $p_a/(p_a + p_d)$ of the point patterns in the dataset contain a point within the interval $[t - 2p_a/p_m, t + 2p_a/p_m]$. Selection of moving penalties that are too large will lead to prototypes that seem unusually short, since the range $t \pm 2p_a/p_m$ will be quite small.

When the primary purpose of the prototype pattern is to serve as an example of a typical point pattern, it is useful to set p_m to be quite small compared to what might be advisable for distance calculations. Setting p_m to a small positive value when $p_a = p_d$ enables the prototype to take on the median number of points in the dataset. Because each prototype point must be matched to points within at least $p_a/(p_a + p_d)$ of the observed point patterns, it is not possible to achieve a prototype of greater than median length while maintaining the $p_a = p_d$ restriction.

3.3 Prototype Determination

Theorem 4 may be useful in determining the prototype of a point process dataset. In addition, note each point of the prototype must be within $(p_a+p_d)/p_m$ of a fraction $p_a/(p_a+p_d)$ of the dataset's point patterns, as discussed in Schoenberg and Tranbarger (2004). Let n be the number of point patterns in the dataset. Then each point z in the dataset for which at least $np_a/[2(p_a+p_d)]$ other points from distinct point patterns in the dataset are in the range $[z - (p_a + p_d)/p_m, z]$ and at least $np_a/[2(p_a + p_d)]$ other points from distinct point patterns in the dataset fall in the range $[z, z + (p_a + p_d)/p_m]$ is a candidate for inclusion in the prototype. Note that for the case of $p_a = p_d$, this is simply $n/4$ points from distinct point patterns in the dataset occurring on either side of z, within a distance $2 * p_d/p_m$. In practice, this observation may significantly decrease the number of candidates to be considered as potential points in the prototype.

3.3.1 Direct algorithms for prototype determination

With a finite number n_{cp} of candidate prototype points to consider, one may compute the sum in equation (4) for each possible prototype, i.e. each possible collection of points that are potentially in the prototype. For sufficiently small p_m, the prototype will have length equal to the median length M of the n point patterns, so one may limit one's search to the $\binom{n_{cp}}{M}$ collections of length M only, as candidates for the prototype. Depending on the size of the dataset of interest, this may be a reasonable task to undertake.

Alternatively, if p_m is not small enough to allow for a prototype of length M, then the best prototype of length k can be iteratively sought for $k = 0, 1, 2, \ldots M$. One may begin by finding the optimal prototype candidate of length $k = 0$ and progressively increase k, ending the search once the minimum sum of distances between the point patterns in the dataset and the potential prototype for length $k + 1$ is greater than the minimum sum of distances for length k.

3.3.2 Forward and reverse stepwise approximation

Finding the prototype of a set of point patterns as prescribed in Section 3.3.1 may prove to be prohibitively tedious for large datasets. In this case, stepwise methods can be implemented to attain an approximate prototype solution. Note that such methods may provide useful approximations of a prototype, but are generally not exact and will not select the true prototype of the point process dataset as defined via (4).

A reverse stepwise approach can be useful when the length of the prototype is unknown. By first finding the optimal prototype of median length M, possible prototypes of shorter length can be found by eliminating points from the length M prototype one at a time. This strategy continues until the optimal prototype of length $k - 1$ has a larger sum of distances (4) than the best prototype candidate of length k. It is important to note that if $p_a \neq p_d$ then it is possible that the prototype may have length greater than M. When $p_a < p_d$, prototypes longer than M might be optimal, while setting $p_a > p_d$ will create shorter prototype lengths.

In general, an upper bound on the length of the prototype is the $p_d/(p_a + p_d)$ percentile length of all observed pattern lengths in the dataset.

While the reverse stepwise approach will entail fewer computations than finding the prototype directly, even the reverse stepwise approximation might prove prohibitively cumbersome due to the long length of the prototype computed in the first stage. Finding the best prototype of length M involves calculating (4) for $\binom{n_{cp}}{M}$ possible prototypes, a task which might be too large an undertaking depending on the median pattern length M, and the number of identified candidate prototype points n_{cp}. An alternative is to use a forward stepwise approach, which can further reduce computation time by beginning with a short prototype candidate and then progressively expanding it. In each iteration, the sum of pattern-prototype distances for the prototype candidate of length k is compared with the sum for the prototype candidate of length $k - 1$. Provided the length k prototype has a smaller sum of distances (4) than the prototype of length $k - 1$, the process continues by finding the optimal prototype of length $k+1$ that contains all points included in the solution of length k. Once the sum of distances () for the length $k+1$ prototype exceeds the sum for the length k prototype, the process ends and the prototype of length k is selected.

4 Classification through clustering

Using the spike-time distance metric and prototype, various clustering algorithms established for standard multivariate data can be modified for use in point process applications. This Section describes three such adapted methods.

4.1 HMEANS and KMEANS

HMEANS and KMEANS clustering are two closely related iterative techniques useful for clustering multivariate data. Both HMEANS and KMEANS clustering begin by randomly assigning each observation to one of c clusters and finding the center of each cluster. While centers are conventionally defined as centroids in standard HMEANS and KMEANS clustering of multivariate data (Spaeth 1980, Anderberg 1973, Bock 1970, Howard 1966), in the case of a point process dataset one may instead use the prototype as the definition of the center of a cluster of point patterns.

In HMEANS clustering, once the cluster centers are defined, the distances from each of the observed point patterns to each of the c cluster prototypes is measured and patterns are reassigned to the cluster with the nearest prototype. Once cluster prototypes are calculated according to the new assignments, the process repeats until no point pattern assignments are changed. This approach is not guaranteed to yield the optimal assignment of the n observed patterns into c clusters. In fact, HMEANS may not result in a total of c clusters, as one or more of the clusters may be eliminated during the iterative process.

A similar approach, KMEANS, will maintain all c clusters and has been shown to produce the optimal partitioning possible, given the starting cluster assignments, for standard multivariate applications (Spaeth 1980). Rather than calculating the distance from each observed point pattern to each of the c cluster prototypes and then reassigning all observed patterns at once, KMEANS considers only one point pattern at a time. The distances between the ith observed pattern in the data and each of the c cluster prototypes are calculated and then weighted according to the number of patterns present in each cluster. This weighting scheme is intended to estimate the effect adding the ith pattern might have to each cluster's prototype. Once the i^{th} pattern is reassigned to the cluster with the minimum weighted distance prototype, the prototypes of the two affected clusters are recalculated. The process cycles through iterations of $i = 1, ..n$ and continues until no reassignments are made through a complete cycle of all n observed patterns. While this approach is much more computationally involved than the related HMEANS, it provides a powerful approach to classification of point patterns into c groups. By eliminating the possibility of creating fewer groups than intended, KMEANS eliminates the risk present in HMEANS of assigning all patterns to one group.

4.2 Agglomerative approach

A third option is an agglomerative approach that assigns each observed point pattern to its own cluster and then systematically combines clusters. This approach is especially useful in cases where HMEANS and KMEANS fail to converge after numerous iterations.

With each observed point pattern assigned to a cluster of its own, each cluster's prototype is simply the member pattern. The pairwise distances between each cluster prototype can be found and the nearest two clusters joined to form one larger two-pattern cluster. From here, the new cluster prototype is determined, and the new pairwise distances between prototypes are calculated. The process repeats the until only c desired clusters remain. It is important to note that when new cluster prototypes are determined, each of the original point patterns contained in the clusters of interest is considered, rather than using only the two prototypes of the two former clusters.

If the amount of data makes computing all pairwise distances unfeasible, a similar approach is to consider pairwise distances for only one point pattern at a time in either a randomly assigned order, or in order of length from the longest pattern to the shortest. Our investigations suggest that this procedure of ordering the point patterns from longest to shortest and then progressively merging the point patterns minimizes problems that can occur if the p_a, p_d, and p_m penalties are set such that pattern length overshadows other features in the distance measures. More specifically, this ordering is helpful because the agglomerative method tends to consider point patterns of longer lengths only at the end, as these patterns often are measured to be far from other patterns due to the higher numbers of point additions, deletions, and point movements involved. Beginning the agglomerative process with longer patterns ensures that patterns with higher numbers of points will be paired to point patterns with similar features early in the process rather than being left to be paired with one of only a few shorter prototype patterns in the final steps of the algorithm.

4.3 Penalty choice considerations

While setting p_m to a small value is useful in determining prototypes, very small p_m values are not desirable in clustering applications. As discussed in Section 2.2, small penalties for movement of points will lead to distances that approach the absolute difference in pattern lengths, while large movement penalties lead to distance calculations that approach the sum of point pattern lengths. In clustering, either of these cases will lead to clusters assigned by grouping patterns of similar lengths rather than by grouping patterns with similarly placed points. For this reason, penalties assigned by (3) are recommended for clustering applications.

5 Multi-dimensional Extensions and Discussion

The spike-time distance metric as examined in this work, and as originally proposed by Vector and Pupura (1997), has thus far been applied primarily to temporal point patterns as defined in Section 2. As explored in Schoenberg and Tranbarger (2004), the spike-time metric, the related prototype pattern technique, and related clustering algorithms can be extended to point process data with points occurring in R^d.

In R^d, the basic definition of the spike-time distance metric as a measurement of the operations required to transform one collection of points into another remains. Points can be added with penalty p_a, deleted with penalty p_d, or moved along the i^{th} axis a distance of Δ at a cost of $p_m^{(i)}\Delta$. Moving penalties $p_m^{(1)}, p_m^{(2)}, ...p_m^{(d)}$ may be set independently for movement along each of the d axes as in (3) for distance calculations and clustering application settings. As with the one-dimensional case, smaller values for movement penalties are again useful for prototype determination to enable longer length prototype point patterns. While the ratio of p_a and p_d to moving penalty p_m was of primary importance for temporal point process work, the relative values of the d movement penalties must also be considered when multiple dimensions are involved. These ratios must be considered so as to avoid (or allow) inter-point distances along one or more dimensions being more heavily weighted in distance calculations.

As might be expected, spike-time distance calculations are far more cumbersome in R^d, as the result of Theorem 1 does not hold, so multiple pairing arrangements must be considered for each possible set of kept points. The process to determine which points will be kept and which will be removed remains unchanged as Lemma 2 and Theorem 3 (and their results) extend immediately to multiple dimensions.

For prototype pattern determination, a modified version of Theorem 4 applies to the multi-dimensional setting. With more than one dimension, rather than each prototype point p being a median of the points $(z_1, z_2, ..., z_m)$ it is paired with, each *coordinate* of prototype point p will be a median of the corresponding coordinate of the points $(z_1, z_2, ..., z_m)$. Therefore, while the prototype may not contain points in the dataset, there exists a prototype made entirely of points such that each coordinate of each prototype point is a coordinate in one of the points in the dataset.

Clustering algorithms discussed in Section 4 can be applied to multi-dimensional point pattern data without modification. For some datasets, the computation time involved in prototype determination impedes the use of the clustering algorithms as described in Section 4 and slight modifications can be made, such as considering only one of the d dimensions at a time in each step of the prototype and/or distance calculation. Such a monothetic approach is used in Schoenberg and Tranbarger (2004) for clustering earthquake aftershock activity considering the time, magnitude, and location of each aftershock.

While the spike-time distance metric is only one of countless possible distance metrics for point pattern data, the concept of a prototype sequence is one that exists independently of distance metric specifics. Therefore, though many of the theorems presented in this work apply solely to the spike-time metric, the approaches to determining prototypes and clusters derived from the ability to measure distances between point patterns should remain useful in conjunction with other point pattern distance metrics.

6 Acknowledgements.

We thank Chuck Woody for introducing us to the work of Victor and Purpura, and we thank Pepe Segundo for introducing us to Chuck Woody. We also thank David Jackson and Yan Kagan for their helpful comments. This material is based upon work supported by the National Science Foundation under Grant No. 0306526. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7 References

Daley, D., and Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes, Volume 1: Elementary Theory and Methods, 2nd ed.* Springer-Verlag, New York.

Hitchcock, F.L. (1941). The distribution of a product from several sources to numerous localities. *J. Math. Phys.*, **20**, 224–230.

Levina, E. and Bickel, P.J. (2001). *The Earth Mover's Distance is the Mallows Distance: Some Insights from Statistics.* Proceedings of ICCV 2001, Vancouver, Canada, 251-256.

Monge, G. (1781). *Mémoire sur la Théorie des Déblais et des Remblais.* Histoire de l'Académie Royale des Sciences, Paris.

Rubner, Y., Tomasi, C., and Guibas, L. (1998). A Metric for Distributions with Applications to Image Databases. *IEEE Int'l Conf. Computer Vision*, 59-66.

Rubner, Y., Tomasi, C., and Guibas, L. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* **40(2)**, 99-121.

Späth, Helmuth (1980). *Cluster Analysis Algorithms for Data Reduction and Classification of Objects.* E. Horwood, Chichester and Halsted Press, New York.

Victor, J. and Purpura, K. (1997). Metric-space analysis of spike trains: theory, algorithms and application. *Comput. Neural. Syst.***8**, 127–164.