Prototype Construction for Clustering of Point Processes based on Imprecise Synchrony

Christian Braune[1] **Christian Borgelt**[2]

[1]Otto von Guericke University (Magdeburg)
Universitätsplatz 2, 39106 Magdeburg, Germany
cbraune@ovgu.de
[2]European Centre for Soft Computing
c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Asturias, Spain
christian@borgelt.net

Abstract

We consider the task to cluster realizations of point processes, that is, lists of points in time. Our guiding principle is that two such lists are the more similar, the more (approximately) synchronous points they contain. This task occurs in the analysis of parallel spike trains in neurobiology, where it arises from the desire to detect assemblies of neurons, which are characterized by the synchronous spiking activity they exhibit. While earlier approaches along similar lines employed mainly hierarchical agglomerative clustering, based on distance measures for spike trains, we try to make prototype-based clustering approaches (like (fuzzy-)c-means clustering) applicable by proposing a method to construct cluster prototypes. For this we draw on an idea that is inspired by mountain clustering. In addition, we adapt a method that was originally developed for outlier detection in order to actually single out relevant groups of related realizations of point processes in front of a background of noise, and thus to identify neuron assemblies in parallel spike train data.

Keywords: clustering, prototype, distance, point process, spike train analysis, temporal imprecision

1. Introduction

Point processes are (random) processes that produce isolated points, most commonly in time, but sometimes also in (geographical) space [11]. They are often used to describe event sequences or event locations, like incoming phone calls, the arrival of customers, or the places of lightning strokes. The application domain that motivated our investigation is the analysis of (parallel) spike trains in neurobiology [19]. In this domain the spikes (impulses, action potentials) emitted by a neuron are modeled as a point process (in time).

While it is fairly well understood how individual neurons work and are triggered to emit impulses, comparatively little is known about how networks of neurons process information. Among several competing theories, *cell assemblies* have been suggested as the building blocks of higher-level neural processing [22]. Such assemblies are expected to exhibit synchronous spiking activity, that is, now and then they emit spikes jointly. Since it is possible nowadays, due to recent technological advances, to record the impulses of hundred(s) of neurons in parallel [9], the desire arises to test the fairly popular cell assembly hypothesis by analyzing recorded parallel spike trains with the objective of detecting the predicted synchronous spiking activity.

The core challenges of this detection task are *temporal imprecision* and *selective participation*. The former means that the joint spiking cannot be expected to be perfectly synchronous, which is partially due to limitations of the recording equipment and the necessary preprocessing steps, but likely also caused by imprecisions in the underlying neural process. The latter is due to similar reasons and requires us not to expect to see a spike from every neuron in an assembly in every joint spiking event. Rather a varying subset of the neurons comprising the assembly may participate.

Furthermore, detecting synchronous spiking activity can be addressed (at least) on the following different levels: (a) test whether it is present, possibly with a lower bound on the number of involved neurons, but without identifying the participating neurons (e.g. [34, 35, 28]); (b) test for individual neurons whether they participate in synchronous spiking activity, but without identifying the groups of correlated neurons [3]; (c) actually identify the members of cell assemblies that exhibit synchronous spiking activity (e.g. [18, 17]).

Our approach falls into the third category and is related to the clustering approach of [17], which can be seen as a restricted version of the older, enumeration-based Accretion algorithm [18]. The core idea of both is to measure the distance between two neurons (or sets of neurons) by the p-value of an independence test (e.g., Pearson's χ^2). The test statistic is computed from the counters of a 2×2 contingency table, which is derived from time-binned data, where using time bins of a certain length is meant to cope with the temporal imprecision problem. In [17] the p-values are then used as

the distances in a hierarchical agglomerative clustering algorithm [33, 25] in which cluster merging is terminated as soon as the (remaining) p-values exceed a user-specified significance level. Unfortunately, this renders the detection performance very sensitive to accidental characteristics of the data, which may let it choose bad initial (pair) mergers. The enumeration approach of [18], which is related to frequent item set mining [2], suffers less from this problem (although it is not free of it either), but is computationally much more expensive.

Some of the drawbacks of the mentioned methods can be overcome by a prototype-based clustering algorithm [1, 21, 6, 26, 7, 8, 23], which does not choose early and irrevocably that two neurons (or their spike trains) have to be in the same cluster. However, to apply such an algorithm it is not enough to have a measure for the distance of two spike trains; we also need a way of constructing a prototype. This is the first contribution of this paper: we propose a way of constructing a prototype for clustering the realizations of point processes, which takes temporal imprecision into account.

An obvious alternative would be a medoid-based approach: choose the most central spike train of each cluster as its prototype. However, this taints prototypes with the accidental, non-synchronous spikes of the chosen spike trains. Therefore we refrain from such an approach and opt for an actual construction procedure that generates a spike train like object that need not occur in the data itself.

The idea underlying our prototype construction approach is inspired by mountain clustering [37] and motivated by the fact that (approximately) synchronous spikes signify the similarity of spike trains. That is, we obtain a prototype—as a list of (time) intervals—by finding the (highest) mountains in a step function resulting from summing, over all neurons, the influence functions induced by their spikes and cutting these mountains at an appropriate level.

The second contribution of this paper concerns the fact that it is likely that most of the recorded neurons are merely background noise, while only few recorded neurons may form one or more cell assemblies. Therefore we need a method that singles out groups of relevant clusters (i.e. sets of spike trains, not single spikes) while ignoring unrelated background noise. For this we adapt a procedure that was suggested originally as an outlier detection method [31]. This method is inspired by noise clustering [12, 13] and iteratively adapts the noise cluster distance to contract the actual clusters to the relevant group or groups. In our approach, however, we dispense with a noise cluster and merely remove in each step the data point farthest from the cluster center(s). Although similar in spirit, this method leads to a more robust behavior than using an actual noise cluster (as we found in experiments).

The remainder of this paper is structured as follows: in Section 2 we review the time-binning approach, reveal its several drawbacks and explain the influence region approach [29, 5], which we employ as an alternative. In Section 3 we briefly review common distance measures for sets and binary vectors [10] and how they can be generalized to a continuous domain [5]. In Section 4 we introduce our prototype construction method by deriving it from mountain clustering [37]. Section 5 is devoted to the actual assembly detection algorithm, which combines our cluster prototype construction with a stepwise removal of the farthest data points. The actual assemblies are finally found by analyzing the resulting sequence of farthest distances from the prototype. In Section 6 we report about experiments on artificially generated data to demonstrate the performance of our method. Finally, in Section 7 we draw conclusions from our discussion.

2. Temporal imprecision

In the most common approach to handle the temporal imprecision of synchronous spiking, the spike trains are discretized with a user-specified time bin length, thus turning them into binary vectors, one for each recorded neuron: either a time bin contains a spike of a neuron or it does not.

Unfortunately, such time binning, especially the exact placement of the boundaries of the time bins, has unintuitive effects: two spikes that are almost as far apart as the time bin length are seen as synchronous if they fall into the same time bin, while two spikes that are close(r) together, but on different sides of a time bin boundary, are regarded as not synchronous (boundary problem). As a consequence, shifting the start time of the first bin may change which spikes are seen as synchronous and thus may lead to considerably different results. Furthermore, the notion of synchrony underlying this approach is bivalent: two spikes are either seen as synchronous or not (bivalence problem). There is no concept of a "degree of synchrony" that could reflect how precise the spike synchrony is, even though it is not implausible to assume that spikes that are close together in time create a higher net input for downstream neurons than spikes that are farther apart and thus may be more effective in generating output spikes in downstream neurons. Finally, two spikes may fall into the same bin. Since it is only considered whether a neuron produces a spike in a time bin and thus multiple spikes (of the same neuron) in the same bin are reduced to one (so-called *clipping*), the binning procedure reduces the total number of spikes. Although the impact of this is usually not very severe (at least if sufficiently short time bins are used), it may still lose relevant information.

As an alternative to time binning we introduced so-called *influence maps* [5, 29]: for each spike s of a spike train S an interval $[s - \frac{1}{2}\Delta t, s + \frac{1}{2}\Delta t]$ is created, which describes the region in which the spike may be seen as synchronous to spikes of other

trains (i.e., other neurons). Formally, the function

$$f_S(t) = 1, \text{ if } \exists s \in S\colon\ t \in [s - \tfrac{1}{2}\Delta t, s + \tfrac{1}{2}\Delta t] \quad (1)$$

describes the influence function of a spike train $S = \{s_1, \ldots, s_{n_S}\}$, which is 1 for every time t that can be covered by an influence map of S. Alternatively, one may simply collect all intervals that are generated by the spikes as an interval list and then merge overlapping intervals into one larger interval.

Note that influence maps essentially allow us to place time bins dynamically, centered around each spike instead of having fixed bin boundaries. In such a way the degree of synchrony between two spikes can be measured by calculating the overlap between their influence maps. The following section describes how we can exploit this to calculate the distance or similarity of two spike trains.

3. Distance measures

If the spike trains are discretized using conventional binning, the result can be interpreted as a high-dimensional binary vector. Each component of such a vector corresponds to a bin and its entry to whether the bin contains a spike or not. There is an abundance of different distance or similarity measures that can be used to compare such collections of dichotomous variables, like, for example, the Jaccard Index [24], the Hamming distance [20] or Dice's measure of association [14] (cf. Table 1). These measures can easily be generalized to the continuous domain of spike trains (or rather the interval lists representing them based on influence maps around spikes) by replacing the dichotomous variables with a continuous domain. For instance, the Hamming distance is the number of disagreements between two binary vectors, that is, the number of components in which either the first vector is 0 and the second is 1 or *vice versa*. These cases can be denoted as n_{01} and n_{10}, respectively, were each of these quantities contains the number of components of two vectors $\vec{a}$ and $\vec{b}$ that fulfill the above conditions. The quantities n_{00} and n_{11} can be defined analogously as the number of components in which both vectors are zero or both are one, respectively. All four quantities represent the entries in a contingency table derived from the binary vectors, in which n_{11} alone represents the "degree of synchrony" mentioned in Section 2.

With these four quantities the (normalized) Hamming distance of two binary vectors is defined as

$$d_{\text{Hamming}}(\vec{a}, \vec{b}) = \frac{n_{01} + n_{10}}{n_{00} + n_{01} + n_{10} + n_{11}}. \quad (2)$$

To generalize these measures to the continuous domain we simply replace the quantities n_{00}, n_{01}, n_{10} and n_{11} by the amount of overlap between the corresponding interval lists of two spike trains. As n_{11} usually describes the number of components that are 1 in both vectors, it now describes the total length of all intersections of all pairs of intervals drawn from either interval list. Technically, we have

$$n_{11} = \frac{1}{\Delta t} \int_0^T f^{(11)}_{S_1 S_2}(t) \mathrm{d}t,$$

where T is the considered duration of time and

$$f^{(11)}_{S_1 S_2}(t) = 1, \text{ if } \exists s \in S_1\colon\ t \in [s - \tfrac{1}{2}\Delta t, s + \tfrac{1}{2}\Delta t] \\ \wedge\ \exists s \in S_2\colon\ t \in [s - \tfrac{1}{2}\Delta t, s + \tfrac{1}{2}\Delta t].$$

The other three quantities are defined in an analogous fashion using functions $f^{(00)}_{S_1 S_2}$, $f^{(01)}_{S_1 S_2}$ and $f^{(10)}_{S_1 S_2}$.

4. Prototype construction

Generating a prototypical point for a set of points in a metric space does not pose much of a problem as one can always use the (weighted) mean of the points. However, for spike trains—or the interval lists representing them—such a construction is not possible without further effort. We already investigated generating a metric representation of spike trains in [5], but this is a costly procedure. In addition, although it yields a suitable representation, the results yields no intuitive representation of a spike train and the mapping onto a metric space is not reversible. Therefore we use here a different approach that can cope with temporal imprecision and yields a representation of the prototype within the context of spike trains as well.

To construct the prototype we use the function from Equation 1 for every spike train considered for the prototype and sum them up into a step function. To allow for fuzzy clustering to be performed with these prototypes we incorporate weights into the function such that spike trains can contribute differently to the joint influence function:

$$f_{\mathcal{S}}(t) = \sum_{S_i \in \mathcal{S}} w_i f_{S_i}(t), \quad (3)$$

where $\mathcal{S}$ is the set of spike trains considered and w_i are the weights with which they enter the prototype construction. Since the sum of these spike trains can easily cover the whole time frame $[0, T]$, we need some way to choose a proper set of intervals that represents the prototype appropriately and can still be interpreted as a spike train. For this several different possibilities exist, such as choosing only those intervals in which the step function is maximal. However, this has the obvious disadvantage that there may exist intervals in which more than only the assembly neurons are active and thus we may introduce several false positive classifications into our algorithm. The resulting interval list may also contain considerably fewer intervals as the spike trains that contribute to the prototype (possibly even just one), and therefore one could argue that it is not a "typical" spike train itself. Choosing an (arbitrary) threshold lower than the maximum results in a broader coverage of the recorded period

Jaccard [24]	d_{Jaccard}	$= \frac{n_{10}+n_{01}}{n_{11}+n_{10}+n_{01}}$
Tanimoto [32]	d_{Tanimoto}	$= \frac{2(n_{10}+n_{01})}{n_{11}+n_{00}+2(n_{10}+n_{01})}$
Dice [14]	d_{Dice}	$= \frac{n_{10}+n_{01}}{2n_{11}+n_{10}+n_{01}}$
Correlation [16]	$d_{\text{Correlation}}$	$= \frac{1}{2} - \frac{n_{11}n_{00}-n_{01}n_{10}}{2\sqrt{(n_{10}+n_{11})(n_{01}+n_{00})(n_{11}+n_{01})(n_{00}+n_{10})}}$
Yule [38]	d_{Yule}	$= \frac{n_{01}n_{10}}{n_{11}n_{00}-n_{01}n_{10}}$
Hamming [20]	d_{Hamming}	$= \frac{n_{01}+n_{10}}{n_{**}}$

Table 1: Different distance measures used for binary vector comparison.

and may lead to a better representation of the contained spike trains. One possibility is to preserve all the intervals in which more than the average number of spike trains (plus a user-specified factor times the standard deviation) are active. These intervals would be the "mountain tops" we are trying to single out and which the mountain clustering algorithm [37] inspired us to look for. Mountain clustering looks for local maxima and associates them with cluster centers. We draw from this insight that we can try to associate coincident spikes from the step function with local maxima.

Another alternative, which we choose for our purposes here, is to select the threshold in such a way that the number of resulting intervals roughly matches the average number of intervals present in the original spike trains. To find such a threshold, we search from the top for the lowest cut level that leads to a prototype containing no more intervals than the average number of intervals in the participating interval lists. This renders the prototype a representative of a fairly "typical" spike train, not only in terms of the number of intervals, but also in the locations of these intervals. In order to ensure even better correspondence to a "normal" spike train, we expand the intervals of the prototype that are shorter than Δt to a width of at least Δt each, because this is the minimum width of an interval in an influence map representation of a spike train.

5. Assembly detection

With the ability to construct a prototype from a set of spike trains that again is an interval list (like the spike trains itself after placing influence regions around spikes), and a way to calculate distances between interval lists, we can begin to detect neural assemblies in spike train data. For this we propose the following algorithm: first we generate a prototype from all spike trains available and calculate the distance from each spike train to this prototype. The spike train that is farthest from the prototype is then removed from the spike train set. Its distance is stored and we repeat the process of computing a new prototype and removing the spike train farthest from it for the remaining spike trains until no spike train is left or the remaining number of spike trains is smaller than a user-specified threshold (for example, the minimum size of a group of neurons we are willing to accept as an assembly).

Note that this procedure is similar to the outlier detection method proposed in [31], only that we always remove the one farthest spike train, while in [31] the distance to a noise cluster is slowly reduced. The difference consists mainly in the fact that in our setting reducing a distance to a noise cluster tends to remove multiple spike trains in one step (also due to the changes of the prototype brought about by removing a spike train), but we need a higher resolution (smaller step width) to properly identify assemblies. Thus we cannot and need not construct a *real* noise cluster and do not need to calculate weights or degrees of membership and can consider all weights as 1.0 for future calculations. However, it is clearly possible to extend our method so that it uses a noise cluster or multiple clusters for actual assemblies. We leave the investigation of this obvious possibility for future work, though.

Note also that with this procedure the neurons that hardly contribute to the highest peaks in the step function (cf. Equation 3) are removed first and only those neurons remain that share a lot of intervals with each other, that is, which have a lot of (approximately) synchronous spikes. At the point when only neurons belonging to an assembly are present, they all become fairly similar to the prototype. Therefore the stored farthest distance (recorded when the spike train farthest from the prototype is removed) is usually smaller than the distances of spike trains removed earlier.

Unfortunately just looking for the largest drop in the stored distances does not lead to satisfactory results as this drop usually occurs after several assembly neurons have already been removed. This happens because the assembly neurons do not only share their coincident spikes from joint activity but some of them may also show similar behavior within their background firing. These spike trains appear even more similar to each other than those within the assembly already do. Just looking at the largest difference between the distances at which two consecutive spike trains are removed thus favors smaller assemblies and leads to many false negative results.

Should the probability with which the assembly neurons participate in a particular synchronous

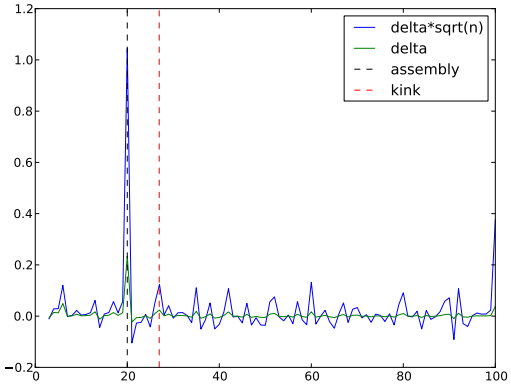

Figure 1: Differences of distances of removed spike trains to their prototypes weighted (blue) and unweighted (green). Copy probability is 1.0.

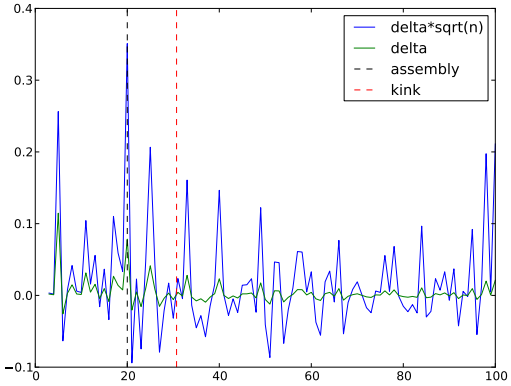

Figure 2: Differences of distances of removed spike trains to their prototypes weighted (blue) and unweighted (green). The highest peak in the green line obviously does not refer to the assembly. Copy probability is 0.6.

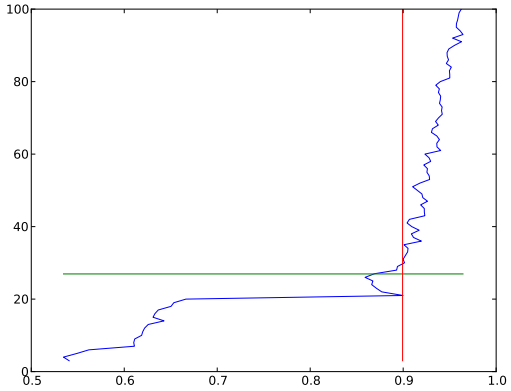

Figure 3: Weighted distances of removed spike trains together with the identification of the kink point. Copy probability is 1.0.

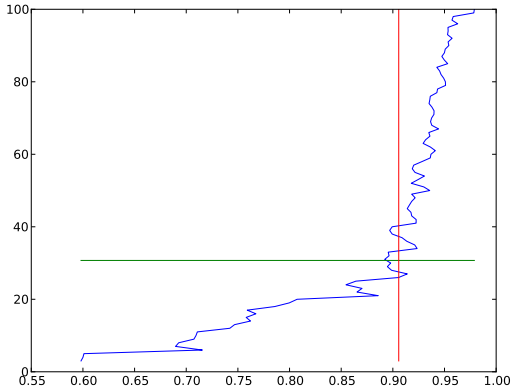

Figure 4: Weighted distances of removed spike trains together with the identification of the kink point. Copy probability is 0.6.

spiking event be less than 1.0, we also face the problem that subsets of the real assembly may show such behavior even at those places where there is actual synchronous activity. Reducing the set of spike trains to such a subset also leads to a significant drop in the observed distances, which might be mistaken for an assembly indicator.

Weighting the difference by the number of neurons participating in the prototype (or the square root of this) improves the results drastically and leads to a method favoring larger assemblies. However, the drop in distances from the maximum distance to the first removed spike train may be so large that weighting it even with only the square root of the total number of spike trains may still be larger than the drop we are actually looking for.

However, when looking at plots of the distance curves (see next section), it becomes obvious that there is a kink present, which indicates where the differences between subsequent distances become larger on average. This kink can be detected by a simple algorithm: first we find the point in the data that is farthest from an imaginary line drawn from the first to the last point. This point is used as a seed point. For a certain number of split points on both sides of the seed point a linear regression is performed (including the split point in both point sets) and the angle between the regression lines is calculated. The split point that yields the largest angle between the two regression lines is selected as the best split point. The intersection between the regression lines resulting from this best split point is the kink point, which we use as the starting point for our assembly detection: only drops below the number of neurons indicated by the kink point are taken into account and the largest drop is used to identify the cluster.

6. Experiments

Acquiring spike trains from living specimen poses two major problems. First, beside ethical reasons, collecting parallel spike trains is a costly procedure that makes it nearly impossible to gather enough data to get reliable results. Second, we do not know which parameters truly occur in such data and therefore cannot validate the results we might see. Due to this we use artificially generated data sets to test and validate our algorithm. For this each spike train is modeled as a stationary Poisson process.[1] Spike trains that belong to neurons that form an assembly have their background firing rates reduced such that the overall firing rate matches the background firing rate of non-assembly neurons. Coincident spikes are generated by a mother process with either a fixed firing rate or a fixed number of spikes generated. The spikes of this process are then copied into the spike trains that form an assembly. As both processes are Poisson processes the resulting process is still a Poisson process. Spike times are sampled by retrieving the inter-spike intervals, i.e. the time between two consequent spikes, from an exponential distribution until the first spike would have to be placed outside the required length of the spike train. To model the temporal imprecision (jitter) of the spike times all spikes are moved by up to $\pm$ 3ms independent of each other.

For our experiments every trial contained 100 parallel spike trains with an assembly of 20 neurons injected into background firing with a rate of 20Hz. The number of coincidences within the assembly was fixed at 50 which corresponds with a coincidence rate of 5Hz. Throughout the different experiments we varied the copy probability from 1.0 (like the so-called *single interaction process* (SIP), see [27]) down to 0.6 (any copy probability less than one is usually referred to as a *multiple interaction process* (MIP), see [27]), that is, 40% of all synchronous spiking information is lost.

As we know from the generating procedure which neurons belong to an assembly we can use external cluster evaluation measures such as the Rand Index [30] to assess the quality of our method. The results are shown in Figure 5.

As we can see, we get nearly perfect classification results for copy probabilities of 1.0 and 0.8 and even if 40% of the coincidence information is lost, the majority of results is still favorable. The classification we use here uses the labels assigned to each single neuron. Commonly used are also measures that consider it a (true) positive result if either all assembly neurons have been found or no additional neuron has been found. Using any of these measures would even increase the quality of our method, especially in the first two cases, where mostly either one or two additional neurons were present or missing. Each of the alternative measures here would rank these results higher than the Rand Index does.

[1] While Poisson processes are not too far from reality, the stationarity of such spike trains may be debatable, but should be acceptable as a setting for a fundamental evaluation.

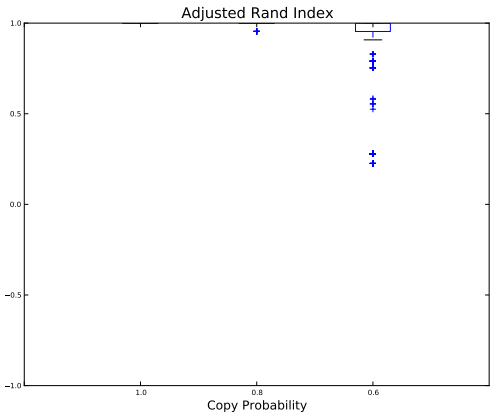

Figure 5: Adjusted Rand Index for each 1000 experiments with varying copy probabilities.

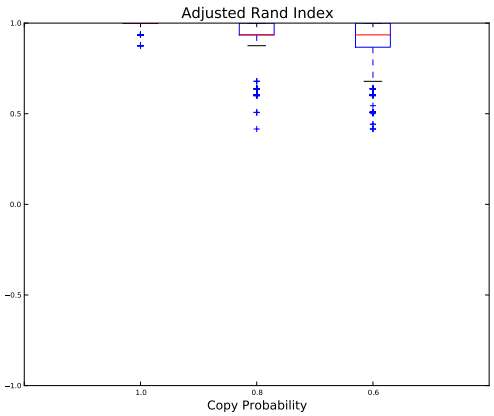

Figure 6: Adjusted Rand Index for each 1000 experiments with varying copy probabilities at only 6s length.

On the other hand we can see in Figure 6 how the length of the recorded process influences the results. In these examples only 6 seconds of spike train data were generated and though the results are still very good they are significantly worse than in the previous case. This is due to the reduced number of coincidences that we are able to observe.

7. Conclusion

In this paper we proposed a method to construct cluster prototypes from a set of realizations of point processes. The core idea underlying our approach is to handle temporal imprecision by *influence regions* around each point, so that each realization is described by a list of *intervals* or, in a more general setting, by *fuzzy sets*. Summing the representing interval functions over the realizations that are assigned to a cluster prototype yields a step function.

By choosing an appropriate cut, which singles out the "mountain tops" (and thus areas of many approximately synchronous points), we then construct a prototype that is also a list of intervals. As a consequence, it can be compared directly to the interval lists representing the original data, using distance measures that are natural continuous generalizations of well-known binary distance measures. With these ingredients we can execute prototype-based clustering algorithms like (fuzzy-)c-means.

Inspired by the idea of a *noise cluster* to capture noise and outliers, so that they do not affect the actual grouping, and the idea of adapting the noise cluster distance to contract the actual clusters to the relevant groups, we finally derived a method for detecting neuron assemblies in parallel spike train data: in a similar spirit we iteratively remove the farthest spike train and recompute the cluster prototype. The actual assemblies are then found by analyzing the resulting sequence of largest distances.

Our experiments show that this method is able to handle both core problems, namely *temporal imprecision* and *selective participation*, very well and is able to detect neural assemblies even under fairly unfavorable conditions.

Acknowledgment

This work was partially supported by the Spanish Ministry for Economy and Competitiveness (MINECO Grant TIN2012-31372).

References

[1] G.H. Ball and D.J. Hall. A Clustering Technique for Summarizing Multivariate Data. *Behavioral Science* 12(2):153–155. J. Wiley & Sons, Chichester, United Kingdom 1967

[2] D. Berger, C. Borgelt, M. Diesmann, G. Gerstein, and S. Grün. An Accretion Based Data Mining Algorithm for Identification of Sets of Correlated Neurons. *18th Annual Computational Neuroscience Meeting: CNS*2009* 10(Suppl 1), 18–23. 2009

[3] D. Berger, C. Borgelt, S. Louis, A. Morrison, and S. Grün. Efficient Identification of Assembly Neurons within Massively Parallel Spike Trains. *Computational Intelligence and Neuroscience* 2010, Article ID 439648 (doi:10.1155/2010/439648). Hindawi Publishing Corp., New York, NY, USA 2010

[4] C. Braune, C. Borgelt, and S. Grün. Finding Ensembles of Neurons in Spike Trains by Non-linear Mapping and Statistical Testing. *Advances in Intelligent Data Analysis X* LNCS 7014:55–66. Springer, Berlin / Heidelberg, Germany 2011

[5] C. Braune, C. Borgelt, and S. Grün. Assembly Detection in Continuous Neural Spike Train Data. *Advances in Intelligent Data Analysis XI* LNCS 7619:78–89. Springer-Verlag, Berlin/Heidelberg, Germany 2012

[6] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, NY, USA 1981

[7] J.C. Bezdek and N. Pal. *Fuzzy Models for Pattern Recognition*. IEEE Press, New York, NY, USA 1992

[8] J.C. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Dordrecht, Netherlands 1999

[9] G. Buzśaki. Large-Scale Recording of Neuronal Ensembles. *Nature Neuroscience* 7(5):446–451. Nature Publishing, New York, NY, USA 2004

[10] S.-S. Choi, S.-H. Cha, and C.C. Tappert. A Survey of Binary Similarity and Distance Measures. *Journal of Systemics, Cybernetics and Informatics* 8(1):43–48. Int. Inst. of Informatics and Systemics, Caracas, Venezuela 2010

[11] D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Springer, New York, USA 1988

[12] R.N. Davé. Characterization and Detection of Noise in Clustering. *Pattern Recognition Letters* 12:657–664. Elsevier Science, Amsterdam, Netherlands 1991

[13] R.N. Davé and R. Krishnapuram. Robust Clustering Methods: A Unified View. *IEEE Trans. on Fuzzy Systems* 5:270–293. IEEE Press, Piscataway, NJ, USA 1997

[14] L.R. Dice. Measures of the Amount of Ecologic Association between Species. *Ecology* 26:297–302. Ecological Society of America, Ithaca, NY, USA 1945

[15] J.C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3(3):32–57. American Society for Cybernetics, Washington, DC, USA 1973 Reprinted in [7], 82–101

[16] Edwards, A.: *An introduction to linear regression and correlation*. WH Freeman, New York, NY, USA (1984)

[17] S. Feldt, J. Waddell, V.L. Hetrick, J.D. Berke, and M. Ochowski. Functional Clustering Algorithm for the Analysis of Dynamic Network Data. *Physical Review E* 79:056104. American Physical Society, College Park, MD, USA 2009

[18] G.L. Gerstein, D.H. Perkel and K.N. Subramanian. Identification of Functionally Related Neural Assemblies. *Brain Research* 140(1):43–62. Elsevier, Amsterdam, Netherlands 1978

[19] S. Grün and S. Rotter (eds.) *Analysis of Parallel Spike Trains*. Springer-Verlag, Berlin, Germany 2010

[20] R.V. Hamming. Error Detecting and Error Correcting Codes. *Bell Systems Tech. Journal* 29:147–160. Bell Laboratories, Murray Hill, NJ, USA 1950

[21] J.A. Hartigan and M.A. Wong. A *k*-Means Clustering Algorithm. *Applied Statistics* 28:100–108. Blackwell, Oxford, United Kingdom 1979

[22] D.O. Hebb. *The Organization of Behavior*. J. Wiley & Sons, New York, NY, USA 1949

[23] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. J. Wiley & Sons, Chichester, United Kingdom 1999

[24] P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles* 37, 547–579. France 1901

[25] S.C. Johnson. Hierarchical Clustering Schemes. *Psychometrika* 32:241–254. Psychometric Society, USA 1967

[26] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. J. Wiley & Sons, New York, NY, USA 1990

[27] A. Kuhn, A. Aertsen, and S. Rotter. Higher-order Statistics of Input Ensembles and the Response of Simple Model Neurons. *Neural Computation* 15:67–101. MIT Press, Cambridge, MA, USA 2003

[28] S. Louis, C. Borgelt, and S. Grün. Complexity Distribution as a Measure for Assembly Size and Temporal Precision. *Neural Networks* 23(6):705–712. Elsevier, Amsterdam, Netherlands 2010

[29] D. Picado-Muiño, I. Castro-León, and C. Borgelt. Fuzzy Frequent Pattern Mining to Identify Frequent Neuronal Patterns in Parallel Spike Trains. *Advances in Intelligent Data Analysis XI* LNCS 7619:289–300. Springer-Verlag, Berlin/Heidelberg, Germany 2012

[30] W. Rand. Objective criteria for the evaluation of clustering methods *Journal of the American Statistical association* 336(66):846–850 Taylor & Francis, 1971

[31] F. Rehm, F. Klawonn, and R. Kruse. A Novel Approach to Noise Clustering for Outlier Detection. *Soft Computing* 11(5):489–494. Springer-Verlag, Heidelberg, Germany 2007

[32] D.J. Rogers and T.T. Tanimoto. A Computer Program for Classifying Plants. *Science* 132:1115–1118. American Association for the Advancement of Science, Washington, DC, USA 1960

[33] R.R. Sokal and P.H.A. Sneath. *Principles of Numerical Taxonomy*. Freeman Books, San Francisco, CA, USA 1963

[34] B. Staude, S. Grün, and S. Rotter. Higher-order Correlations in Non-stationary Parallel Spike Trains: Statistical Modeling and Inference. *Frontiers in Computational Neuroscience* 4:16 (doi:10.3389/fncom.2010.00016). Frontiers Media, Lausanne, Switzerland 2010

[35] B. Staude, S. Rotter, and S. Grün. CuBIC: Cumulant Based Inference of Higher-order Correlations in Massively Parallel Spike Trains. *Journal of Computational Neuroscience* 29(1–2):327–350 (doi:10.1007/s10827-009-0195-x). Springer-Verlag, New York, NY, USA 2010

[36] T. Tanimoto. IBM Internal Report (November 17, 1957)

[37] R.R. Yager and D.P. Filev. Generation of Fuzzy Rules by Mountain Clustering. *Journal of Intelligent & Fuzzy Systems* 2(3):209–219. IEEE Press, Piscataway, NJ, USA 1994

[38] G. Yule. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London*, Series A, 194:257–319. Blackwell, Oxford, United Kingdom 1900