Behavioral Clustering for Point Processes

Christian Braune[1], Christian Borgelt[2], and Rudolf Kruse[1]

[1] Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
[2] European Centre for Soft Computing
Calle Gonzalo Gutiérrez Quirós s/n, E-33600 Mieres (Asturias), Spain
christian.braune@ovgu.de, christian@borgelt.net,
kruse@iws.cs.uni-magdeburg.de

Abstract. Groups of (parallel) point processes may be analyzed with a variety of different goals. Here we consider the case in which one has a special interest in finding subgroups of processes showing a behavior that differs significantly from the other processes. In particular, we are interested in finding subgroups that exhibit an increased synchrony. Finding such groups of processes poses a difficult problem as its naïve solution requires enumerating the power set of all processes involved, which is a costly procedure. In this paper we propose a method that allows us to efficiently filter the process set for candidate subgroups. We pay special attention to the possibilities of *temporal imprecision*, meaning that the synchrony is not exact, and *selective participation*, meaning that only a subset of the related processes participates in each synchronous event.

Keywords: point processes, clustering, spike train analysis

1 Introduction

Point processes occur in many different situations, such as arrivals of customers or phone calls, accidents on highways or firing of neurons in artificial or natural neural networks [8]. They generate a series of points in time or space and can be used to describe different kinds of event sequences. The work we report about in this paper is motivated by the analysis of (parallel) spike trains in neurobiology [14]: each train refers to a neuron, the associated point process records the times at which the neuron emitted an electrical impulse (*action potential* or *spike*).

The mechanisms by which a single neuron is activated by the release of neurotransmitters and emits electrical impulses are fairly well understood. However, how groups of neurons interact with each other and collectively encode and process information (like stimuli) is still the subject of ongoing research and intense debate in the neuroscience community. Several competing theories have been proposed to describe this processing. In particular, neuron assemblies have been suggested by Hebb [15] as the key elements of information processing in the cortex. Hebb suggested that such assemblies should reveal themselves by increased synchronous activity, i.e. they tend to produce (roughly) coincident spikes.

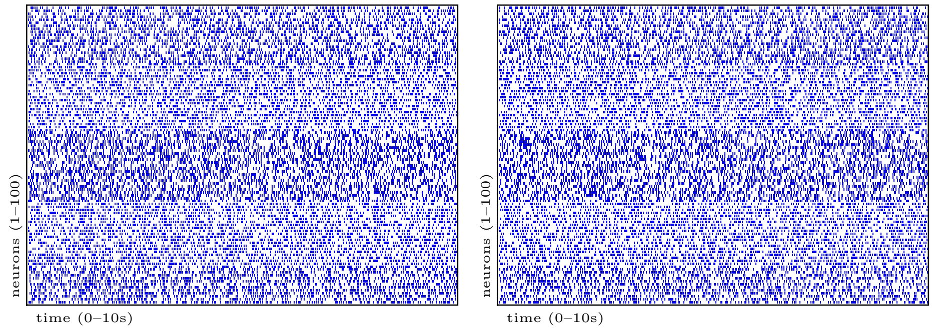

Fig. 1. Two sets of (artificial) parallel spike trains. Right: independent trains, generated as Poisson processes; left: coincident spiking events of 20 neurons (randomly selected) injected into an independent background (see also: [13, 12, 3]).

Since today the recording of several hundred(s) of spike trains in parallel is possible, there is an increased need for efficiently analyzing the data and to test the assembly hypothesis accordingly. The objective is to detect those groups of neurons (or spike trains) that show more synchronous activity than we would expect to see under the assumption that they are all independent.

The main obstacles we need to deal with in this task are threefold: in the first place, the possible combinations of neurons that may form an assembly increases exponentially with the number of neurons recorded. Furthermore we have to cope with *temporal imprecision* and *selective participation*. Temporal imprecision means that we cannot expect two events that originate from the same underlying coincident event to actually appear at (exactly) the same point in time in the spike trains. This may be due to the underlying biological process that generates the spikes but also due to the measurement procedure in which one probe records the electrical potential of (possibly) several neurons in parallel which then have to be separated in a process called *spike sorting*.

Selective participation means that not each and every neuron that belongs to an assembly actually takes part in every coincidence; rather some neurons may miss some of the coincidences. This is quite likely to occur in real neural networks as neurons need some time (so-called *refractory period*) after they emitted a spike to regenerate and be able to emit the next spike. Selective participation may even lead to situations where we do not even see a single coincidence in which all neurons forming the assembly took part.

In this paper we analyze the task of distinguishing between groups of spike trains that contain just random noise (independent spike trains) and groups that exhibit increased synchronous firing, allowing for temporal imprecision as well as selective participation, but without actually identifying the assemblies themselves. Figure 1 shows two samples of parallel spike trains where in the left case 20 neurons are firing with higher synchrony while the right picture shows independent trains. This is to emphasize the difficulty of the problem posed.

The remainder of this paper is organized as follows: in Section 2 we briefly review related work on methods for the analysis of parallel spike trains. In Section 3 we describe how our method works and evaluate it on artificially generated train sets in Section 4. Section 5 concludes the paper with a discussion of the results and an outlook on future research on this topic.

2 Related Work

The problem of finding cell assemblies in parallel spike train data has been the subject of research for quite some time. Early algorithmic attempts to detect assemblies date back at least to [11] where assemblies are detected by performing several pairwise χ^2-tests for independence on the time-discretized spike trains. The pair of spike trains yielding the lowest p-value is then merged into a single spike train, containing only the coincidences. The result of this merger is then added to the pool of spike trains (keeping the originals for further tests) and the tests are repeated until no further significant pairs can be found.

Generally, attempts to identify assemblies in parallel spike train data can be (roughly) categorized in three classes: (1) finding out whether there is (at least) one assembly present in the data (e.g. [18, 21, 22]), (2) answering for each neuron whether it belongs to such an assembly (e.g. [3]) and (3) actually identifying the assemblies (e.g. [10] or for selective participation [2]). The approach we present in this paper belongs to the second category, which is particularly useful for preprocessing, and results from previous work we did on the generation of prototypes for the analysis of spike trains on a continuous time domain [6].

Methods that test whether a neuron belongs to an assembly or not (like [3]) sometimes rely on the generation of surrogate data. Such surrogates are spike trains that retain some (desirably: most) of the statistical properties of the original spike trains while other properties (for instance, synchronous spiking) are destroyed on purpose in order to be able to test for this property. Simple examples are the generation of a spike train that contains the same number of spikes but at different points in time or a spike train that has the same distribution of inter-spike intervals. One may then calculate some statistics for each of the surrogates and if the behavior seen in the original spike train does not occur (or occurs only very rarely) in the surrogates one can assume that it is not the product of a random process. However, though fairly simple and statistically sound, generating surrogates for a large set of spike trains is a very time-intensive procedure that can quickly become infeasible if a large number of surrogate data sets need to be generated to meet a chosen significance level. Methods that allow for faster decision on whether a neuron belongs into an assembly or not are desireable and are very useful as a preprocessing step, and then later (computationally more expensive) analysis can be focused on promising subsets.

In this paper we introduce a method that allows for such quick preprocessing of a data set of parallel spike trains by analyzing the overall behavior of the spike trains, especially w.r.t. synchronous events, which groups them by their

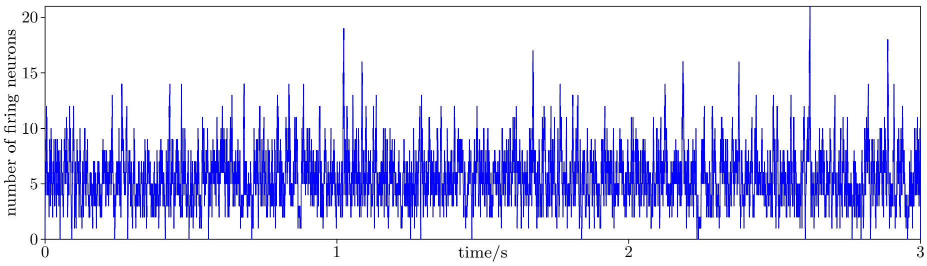

Fig. 2. Spike profile for a set of 100 spike trains with an injected assembly of size 10.

behavior rather than by their actual coincident events. In this sense it may be seen as a classification algorithm [16].

3 Behavioral Clustering

In this section we describe how to compute a clustering of neurons into potential assembly neurons and background neurons from parallel spike train data. The method we propose here circumvents an interpretability problem that occurs when calculating metric representatives for the spike trains (see [5]) in the sense that we do not cluster the spike trains directly but rather their behavior: spike trains that belong to no assembly should behave essentially randomly, while the other spike trains should show a different, more organized behavior.

A spike train is essentially a point process, that is, a set of events that are identified by a point in time. We denote such a set by T, every event (or rather the point in time at which it occurs) by $t_i \in T$. Spike trains are often discretized to form an n-dimensional binary vector, each component of which describes one time bin and records whether the neuron emitted a spike in the corresponding time interval or not. As this approach suffers from various problems (especially the boundary problem, which results from how the bin boundaries fall relative to possible synchronous events), we choose a dynamic window placement. That is, by placing windows of a user-specified length $2\delta t$ around each event (spike), we model the events not as single points in time but as intervals during which they may be considered as coincident with other events. Formally we define:

$$f_T(t) = \begin{cases} 1 & \text{if } \exists t_i \in T : t_i - \delta t \leq t \leq t_i + \delta t, \\ 0 & \text{otherwise.} \end{cases}$$

Spike trains are thus effectively encoded as interval lists that describe the combined influence of all contained spikes. Note that the above definition merges overlapping intervals into a single, longer interval and thus the interval list may contain fewer intervals than the original point process contains points.

A set of parallel recorded spike trains $\mathcal{S} = \{T_1, \ldots, T_n\}$ can now be represented by its spike profile which is simply the sum of all individual influence functions (cf. Figure 2):

$$f_{\mathcal{S}}(t) = \sum_{T \in S} f_T(t).$$

To distinguish spike trains that form an assembly from those that do not we need a representation of the spike trains that allows us to study their behavior. As we pointed out in the motivation, we are looking for a subgroup of processes that exhibits higher synchronous activity than we would expect under independence. Synchronous activity should show itself by several spike trains containing events that occur (roughly, in the presence of temporal imprecision) at the same time. As a pairwise comparison of the interval data would be too costly we can use the profile to identify the behavior of each spike train individually with respect to the other spike trains.

To extract what may be called a "behavior profile", we create different interval lists from the spike profile by using a flooding-like approach. That is, we extract from the spike profile all intervals where $f_{\mathcal{S}}(t) > 0$ holds. This, as we may say, "prototype" interval list is then compared to each individual spike train (or rather its representation as an interval list). To this end we calculate the overlap between the interval lists. Formally, we compute $P_x = \{t \mid f_{\mathcal{S}}(t) \geq x\}$ as a "prototype" interval list and then calculate $s_T(x) = d(T, P_x)\ \forall T \in \mathcal{S}$, $\forall x \in \{0, 1, \ldots, \max f_{\mathcal{S}}(t)\}$, where d is the overlap of the two interval lists T and P_x. More technically, we define the cut level function (for level x)

$$f_{\mathcal{S},x}(t) = \begin{cases} 1 & \text{if } f_{\mathcal{S}}(t) \geq x, \\ 0 & \text{otherwise} \end{cases} \qquad \text{and then} \qquad s_T(x) = \int f_{\mathcal{S},x}(t) \cdot f_T(t)\,\mathrm{d}t.$$

That is, the function $s_T(x)$ describes the total length of the time intervals in which both the cut level function (for level x) and the spike train function are 1. For a sample set of 100 spike trains with an injected assembly of size 10 the resulting curves can be seen in Figure 3 (left diagram, $\delta = 3ms$).

The profile curves of the assembly can already be distinguished by visual inspection on this leftmost graph as they descend slightly later. To enhance the distinction, we exploit the plausible argument that higher levels are more important to detect synchronous activity. Therefore we weight each point of the behavior profile with the square of the level, i.e., the number of participating spike trains. Formally, we have $\forall T \in \mathcal{S} \colon \forall x \in \{0, 1, \ldots, \max f_{\mathcal{S}}(t)\}$:

$$s'_T(x) = x^2 \cdot s_T(x).$$

The resulting curves are shown in the middle diagram of Figure 3.

Finally, in order to make the assembly stand out even more, we normalize the curves by subtracting for each point the minimum value over the spike trains. Formally, we have $\forall T \in \mathcal{S} \colon \forall x \in \{0, 1, \ldots, \max f_{\mathcal{S}}(t)\}$:

$$s''_T(x) = x^2 \cdot s_T(x) - m_x \qquad \text{where} \qquad m_x = \min_{T \in \mathcal{S}} \big(x^2 \cdot s_T(x)\big).$$

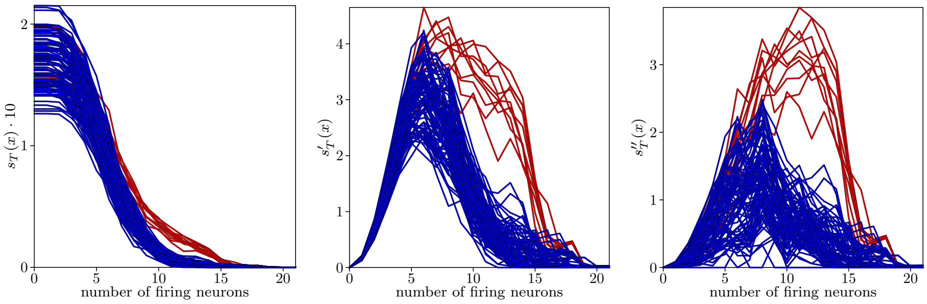

Fig. 3. Profile curves for a set of 100 spike trains with an assembly of size 10 injected. Left: similarity with the prototype/cut level; middle: similarity with the prototype/cut level, weighted with the square of the number of participating neurons (i.e., the level height); right: similarity with the prototype/cut level, weighted with the square of the number of participating neurons (i.e., the level height) and normalized by subtracting the minimum value over all spike trains.

The resulting curves are shown in Figure 3 on the right. Here the assembly clearly stands out from the rest of the spike trains so that we may use the profile curves obtained from the function s'' to describe the behavior of a spike train and perform a clustering on this data, using the vector of values $s''_T(x)$ for $x \in \{0, 1, \ldots, \max f_{\mathcal{S}}(t)\}$ as points in a metric space.

To automatically separate the two groups we decided to test both a density-based clustering algorithm (DBSCAN, [9]) and a simple hierarchical clustering with complete linkage. While the former should be able to detect the number of assemblies as well, the latter was set to report only two clusters. Directly reporting the assemblies contained in the set of spike trains would be a nice feature, as it would lift the method from a pure classification of neurons (into assembly and non-assembly neurons) to an assembly detection algorithm. But unfortunately the assembly neurons behave very similar when compared against the rest of the spike trains as can be seen in Figure 4 where two assemblies are shown as red and green lines respectively. Only if the two assemblies differed significantly in size and/or activity the curves would be different enough to become distinguishable. For the time being we consider them indistinguishable and leave better approaches for future work. As we want a procedure that decides without taking too much time if a spike train should be considered noise, we chose to still evaluate DBSCAN as it showed promising results in separating at least noise from assembly spike trains.

As input both algorithms received a similarity matrix, computed from the squared point-wise difference, i.e. the squared Euclidean distance, of the "behavior profiles". The calculation of this matrix is much faster than the calculation of a similarity matrix as we employed it in [5].

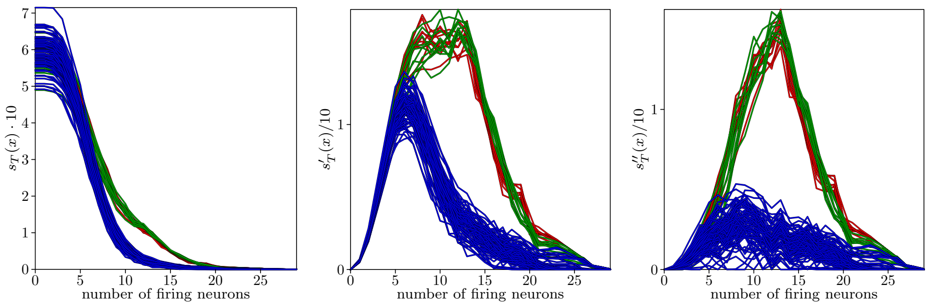

Fig. 4. Profile curves for a set of 100 spike trains with two assemblies injected, each of size 10. Noise is colored in blue, assemblies in red or green. The two disjoint assemblies are nearly indistinguishable.

Both algorithms may report the clusters found in arbitrary order so that we still need a criterion to distinguish them. For that we first calculate the mean profile curve as

$$m_{\mathcal{C}}(x) = \frac{1}{\|\mathcal{C}\|} \sum_{T \in \mathcal{C}} s''_T(x)$$

for each cluster $\mathcal{C}$ found and then calculate the area under the curve (AUC) for each $m_{\mathcal{C}}$. The one that yields the smallest AUC has the smallest overlap with time frames that many spike trains have contributed to. So it is fair to assume that this is the prototype for the behavior of the noise. The remaining spike trains will be labeled as potential assembly candidates and can be further processed with other methods.

As we only need to decide which spike trains should be labeled as noise and which as assembly candidates, we can also justify the choice of restricting the hierarchical clustering to report exactly two clusters. One will be the noise while the other one will contain the assembly spike trains.

4 Evaluation

To evaluate the method we proposed in Section 3 we generated several artificial sets of spike trains and ran our algorithm to report assembly and non-assembly spike trains. As this is a kind of classification, we can use classification quality measures such as the Adjusted Rand Index (ARI, [20]), Adjusted Mutual Information (AMI, e.g. [23]) and others for the evaluation of our method. Both aforementioned measures calculate the agreement between two different clustering results based on the predicted cluster labels but independent of the order of the cluster labels. The first is based on the absolute number of agreements while the latter is based on the mutual information shared by both clusterings.

To generate an artificial spike train, we sample the inter-spike intervals (time between two subsequent events) from an exponential distribution until a specified length of the spike train is reached (i.e., we generate Poisson point processes).

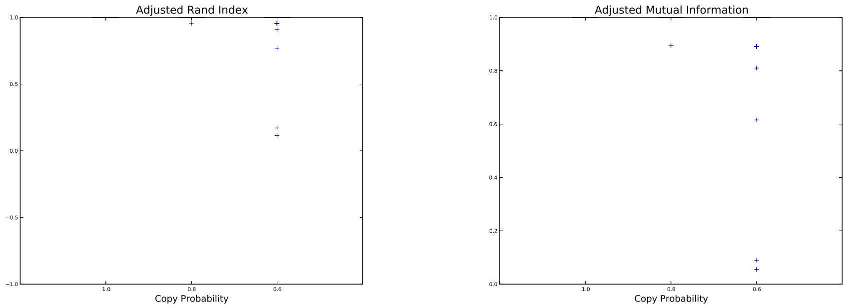

Fig. 5. Results for 1000 sets of parallel spike trains of 10 seconds length with an assembly of 20 spike trains injected, analyzed with DBSCAN.

Firing rates were set to 20Hz for the non-assembly spike trains (which is a typical reference in the field). For the assembly spike trains a mother process was generated from which the coincident events were copied into the assembly spike trains with a certain probability ($c = 1.0$ if full participation was to be present, $c < 1.0$ if selective participation was to be modeled). The background firing was adjusted such that the overall firing rate (background and coincidences) was the same as for the noise spike trains (20Hz). Thus the spike trains cannot be distinguished by merely looking at the number of spikes. The temporal imprecision was modeled by shifting each spike after its generation by a certain, specified amount (here: $\pm$5ms, i.e. $\forall t_i : t_i := t_i + U(-5, 5)$; or $\delta t = 5$ms).

For our experiments we generated 1000 sets of parallel spike trains, consisting of 100 spike trains each. 20 of the spike trains form a single assembly with a coincidence rate of 5Hz embedded. The length of the simulated recording was 10 seconds in the first trials with copy probabilities of 1.0, 0.8 and 0.6. Each of these sets has been analyzed in the same way with either DBSCAN or hierarchical clustering grouping the spike trains together. To show the effects of the assembly size on the detection quality we ran the same number of tests on sets of parallel spike trains with an assembly size of only 10 and only 6 seconds length. The results of these tests can be seen in Figures 5 and 6 respectively for DBSCAN. Please note that the boxplots used seem to disappear in some cases. This is due to the fact that (almost) all values for the quality measures are actually 1.0 which means that the algorithm returned a clustering that perfectly matched the ground truth. Even if we reduce the copy probabilities to 0.6 all non-perfect results have to be considered outliers (i.e. they lie at least 2.698σ / outside the 99% interval from the median).

As the result for shorter spike trains with an average participation probability of 0.8 was significantly worse than we expected, we also used hierarchical clustering to analyze the last set of spike trains (see Figure 7). For copy probabilities of $c = 0.8$ the results are clearly better than when using density-based clustering, but for smaller copy probabilities the results are still bad albeit better. With

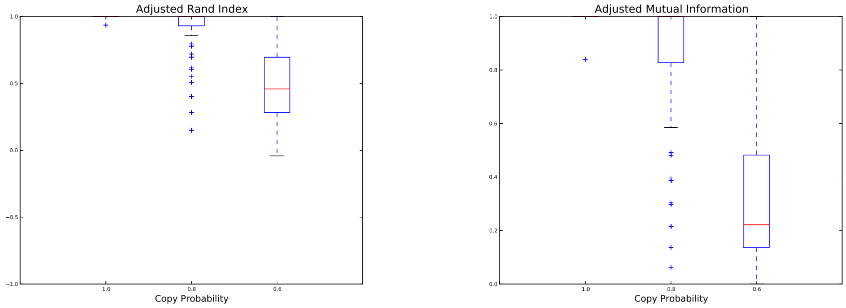

Fig. 6. Results for 1000 sets of parallel spike trains of 6 seconds length with an assembly of 10 spike trains injected, analyzed with DBSCAN.

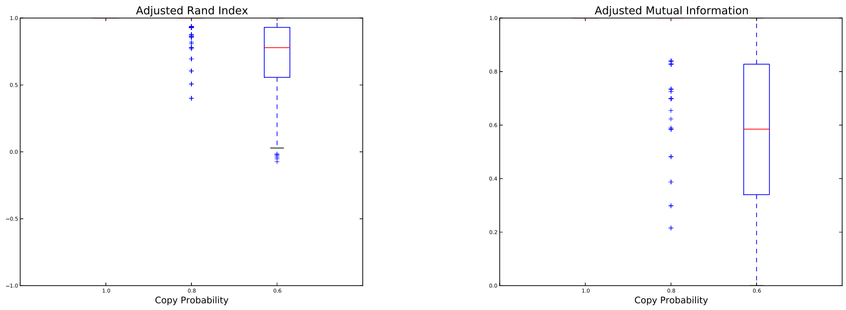

Fig. 7. Results for 1000 sets of parallel spike trains of 6 seconds length with an assembly of 10 spike trains injected, analyzed with hierarchical clustering.

only 10 spike trains forming an assembly and only six of them taking part in a coincidence on average the number of events we can use for the detection of the assembly neurons is already close to the number of coincidences we would expect to see in totally independent processes.

5 Conclusion and Future Work

In this paper we presented a method to group sets of point processes by the similarity of their behavior with respect to the behavior of the other processes by means of clustering algorithms. Synchronization between processes can be detected quite well for processes of different length and under additional obstacles such as selective participation and temporal imprecision. We evaluated our method in several different settings of artificially generated spike trains, i.e. point processes as they commonly appear in neurobiology. The artificial nature of our data allows us to control the experiments and perform a much more restrictive analysis as we can clearly calculate the number of false-positive or false-negative results. We use different measures for classification evaluation to aggregate these rates and the results show that our method is capable of recognizing and distinguishing groups of synchronized processes quite well from those that show no synchronization. We have to admit, though, that our method is not (yet) capable of reporting different groups present in the data. However, it is valuable as a preprocessing method that can focus more expensive methods for actual assembly detection on a set of promising candidates.

Acknowledgements

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

References

1. 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).* (doi:10.1186/1471-2202-10-S1-P254) Berlin, Germany 2009
2. C. Borgelt and T. Kötter. Mining Fault-tolerant Item Sets using Subset Size Occurrence Distributions. *Advances in Intelligent Data Analysis X*, LNCS 7014:43–54. (doi:10.1007/978-3-642-24800-9) Springer-Verlag, Berlin/Heidelberg, Germany 2011
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*, Article ID 439648. (doi:10.1155/2010/439648) Hindawi Publishing Corporation, 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. (doi:10.1007/978-3-642-24800-9) Springer-Verlag, 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. (doi:10.1007/978-3-642-34156-4) Springer-Verlag, Berlin/Heidelberg, Germany 2012
6. C. Braune and C. Borgelt.Prototype Construction for Clustering of Point Processes based on Imprecise Synchrony. *8th Conf. of the European Society for Fuzzy Logic and Technology (EUSFLAT 2013)* (submitted, under review)
7. E.N. Brown, R.E. Kass, and P.P. Mitra. Multiple Neural Spike Train Data Analysis: State-of-the-art and Future Challenges. *Nature Neuroscience* 7(5):456–461. (doi:10.1038/nn1228) Nature Publishing, New York, NY, USA 2004
8. D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes.* (doi:10.1007/978-0-387-49835-5) Springer, New York, USA 1988
9. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu: A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD 96, Portland, Oregon)*, 226–231. AAAI Press, Menlo Park, CA, USA 1996

10. 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. (doi:10.1103/PhysRevE.79.056104) American Physical Society, College Park, MD, USA 2009
11. G.L. Gerstein, D.H. Perkel and K.N. Subramanian. Identification of Functionally Related Neural Assemblies. *Brain Research* 140(1):43–62. (doi:10.1016/0006-8993(78)90237-8) Elsevier, Amsterdam, Netherlands 1978
12. S. Grün, M. Abeles, and M. Diesmann. Impact of Higher-order Correlations on Coincidence Distributions of Massively Parallel Data. *Dynamic Brain — From Neural Spikes to Behaviors*, LNCS 5286:96–114. (doi:10.1007/978-3-540-88853-6_8) Springer-Verlag, Berlin/Heidelberg, Germany 2008
13. S. Grün, M. Diesmann, and A.M. Aertsen. 'Unitary Events' in Multiple Single-neuron Spiking Activity. I. Detection and Significance. *Neural Computation* 14(1):43–80. (doi:10.1162/089976602753284455) MIT Press, Cambridge, MA, USA 2002
14. S. Grün and S. Rotter (eds.) *Analysis of Parallel Spike Trains.* (doi:10.1007/978-1-4419-5675-0_10) Springer-Verlag, Berlin, Germany 2010
15. D.O. Hebb. *The Organization of Behavior.* J. Wiley & Sons, New York, NY, USA 1949
16. R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, P. Held. *Computational Intelligence.* (doi:10.1007/978-1-4471-5013-8) Springer, London, United Kingdom (2013)
17. M. Lewicki. A Review of Methods for Spike Sorting: The Detection and Classification of Neural Action Potentials. *Network* 9(4):R53–R78. (doi:10.1088/0954-898X_9_4_001) Informa, St. Helier, Jersey, France (1998)
18. S. Louis, C. Borgelt, and S. Grün. Complexity Distribution as a Measure for Assembly Size and Temporal Precision. *Neural Networks* 23(6):705–712. (doi:10.1016/j.neunet.2010.05.004) Elsevier, Amsterdam, Netherlands 2010
19. D. Picado-Muino and C. Borgelt. Characterization of Spike Synchrony without Discretization of Time. *Neuroinformatics* (submitted)
20. W. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association* 336(66):846–850. (doi:10.2307/2284239). Taylor & Francis, Oxford, United Kingdom 1971
21. 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
22. 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
23. W.M. Wells III, P. Viola, H. Atsumi, S. Nakajima, and R. Kikinis. Multi-modal Volume Registration by Maximization of Mutual Information. *Medical Image Analysis* 1(1):35–51. (doi:10.1016/S1361-8415(01)80004-9). Elsevier, Amsterdam, Netherlands 1996