# Finding Ensembles of Neurons in Spike Trains by Non-linear Mapping and Statistical Testing

Christian Braune[1,2], Christian Borgelt[1], and Sonja Grün[3,4,5]

[1] European Centre for Soft Computing
Calle Gonzalo Gutiérrez Quirós s/n, E-33600 Mieres (Asturias), Spain
[2] Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
[3] RIKEN Brain Science Institute, Wako-Shi, Japan
[4] Institute of Neuroscience and Medicine (INM-6), Research Center Jülich, Germany
[5] Theoretical Systems Neurobiology, RWTH Aachen University, Aachen, Germany
`christian.braune@st.ovgu.de`, `christian.borgelt@softcomputing.es`,
`s.gruen@fz-juelich.de`

**Abstract.** Finding ensembles in neural spike trains has been a vital task in neurobiology ever since D.O. Hebb's work on synaptic plasticity [15]. However, with recent advancements in multi-electrode technology, which provides means to record 100 and more spike trains simultaneously, classical ensemble detection methods became infeasible due to a combinatorial explosion and a lack of reliable statistics. To overcome this problem we developed an approach that reorders the spike trains (neurons) based on pairwise distances and Sammon's mapping to one dimension. Thus, potential ensemble neurons are placed close to each other. As a consequence we can reduce the number of statistical tests considerably over enumeration-based approaches (like e.g. [1]), since linear traversals of the neurons suffice, and thus can achieve much lower rates of false-positives. This approach is superior to classical frequent item set mining algorithms, especially if the data itself is imperfect, e.g. if only a fraction of the items in a considered set is part of a transaction.

## 1 Introduction and Motivation

With the help of electrodes it is possible to observe the electrical potential of a single neuron directly, while multi-electrode arrays (MEAs) allow for several neurons to be observed simultaneously and their electrical potential to be recorded in parallel. By analyzing the wave form of the electrical potential it is possible to detect increases in the potential (so-called spikes) as well as to separate signals from multiple neurons recorded by a single electrode [17]. After this step, which is called *spike sorting*, a single spike train consists of a list of the exact times at which spikes have been recorded. Due to the relatively high time resolution it is advisable to perform some time binning in order to cope with inherent jitter. That is, the continuous time indices are discretized by assigning a time index to each spike. A 10s record binned with 1ms time bins therefore leads to a list of up to 10000 entries with the time indices of the spiking events.

In this form, a spike train can also be interpreted as a binary vector with 10000 dimensions, where each dimension represents one time index and the vector elements are either 0 (no spike) or 1 (spike). Figure 1 shows two sets of spike trains, each generated artificially for 100 neurons and 10,000 time bins (10 seconds) with a simple Poisson model. The top diagram shows random noise (independent neurons, no correlations), while the data in the middle diagram contains an ensemble of 20 neurons. Without any computer aid it is hardly possible to distinguish these two data sets, let alone decide with a high degree of certainty which plot belongs to which data set. Only if the correlated neurons are sorted together (bottom diagram), the assembly becomes discernible.

This paper proposes an algorithm to find an ordering of neurons that reflects the fact that ensembles can be found as lines within the background noise if they are properly sorted. Section 5 shows that such an ordering can then be exploited to perform linear statistical testing to find neuronal ensembles. Thus it abolishes the need to perform all pairwise tests, while still being able to detect the inherent dependency structure to a very high degree, often even perfectly.

## 2 Related Work

Gerstein *et al.* (1978) proposed the accretion algorithm which tries to find such ensembles by iterative combination/merging of neural spike trains [9]. Though not executable at the time it was published for massively parallel spike train data, today even standard desktop computers allow to perform the necessary tests in acceptable time. However, there still remains the problem that after merging relevant spike trains, some subsets of neurons are tested more than once and thus the likelihood of receiving false positive results becomes fairly large because of the huge number of tests that are performed.

Attempts have been made to identify the structure of neuronal ensembles with the help of frequent item set mining (FIM) algorithms [1]. In this approach, neurons are mapped to items, time bins to transactions. Although FIM algorithms are able to detect groups of neurons in the background noise, they generally also report much smaller groups as significant ensembles. In most of these smaller groups all but one neuron belong to a real ensemble and the remaining one randomly produces one or two coincidences, which, however, are considered significant by standard statistical measures.

In addition, in the domain of spike train analysis we face the problem that it cannot be reasonably expected that all neurons of an ensemble always participate in a synchronous spiking event. The synfire model [7] suggests that 50 to 80% of the ensemble neurons may participate, with a different selection on each instance. In order to cope with this problem, fault-tolerant (or approximate or fuzzy) frequent item set mining algorithms may be applied (for example, [23, 2, 3]). However, these approaches have the drawback that they enumerate all potential groups/ensembles and thus suffer severely from the false positives problem.

Considering the neural spike trains as binary vectors and the whole set of parallel spike trains as a matrix (see Section 3) an interesting approach was

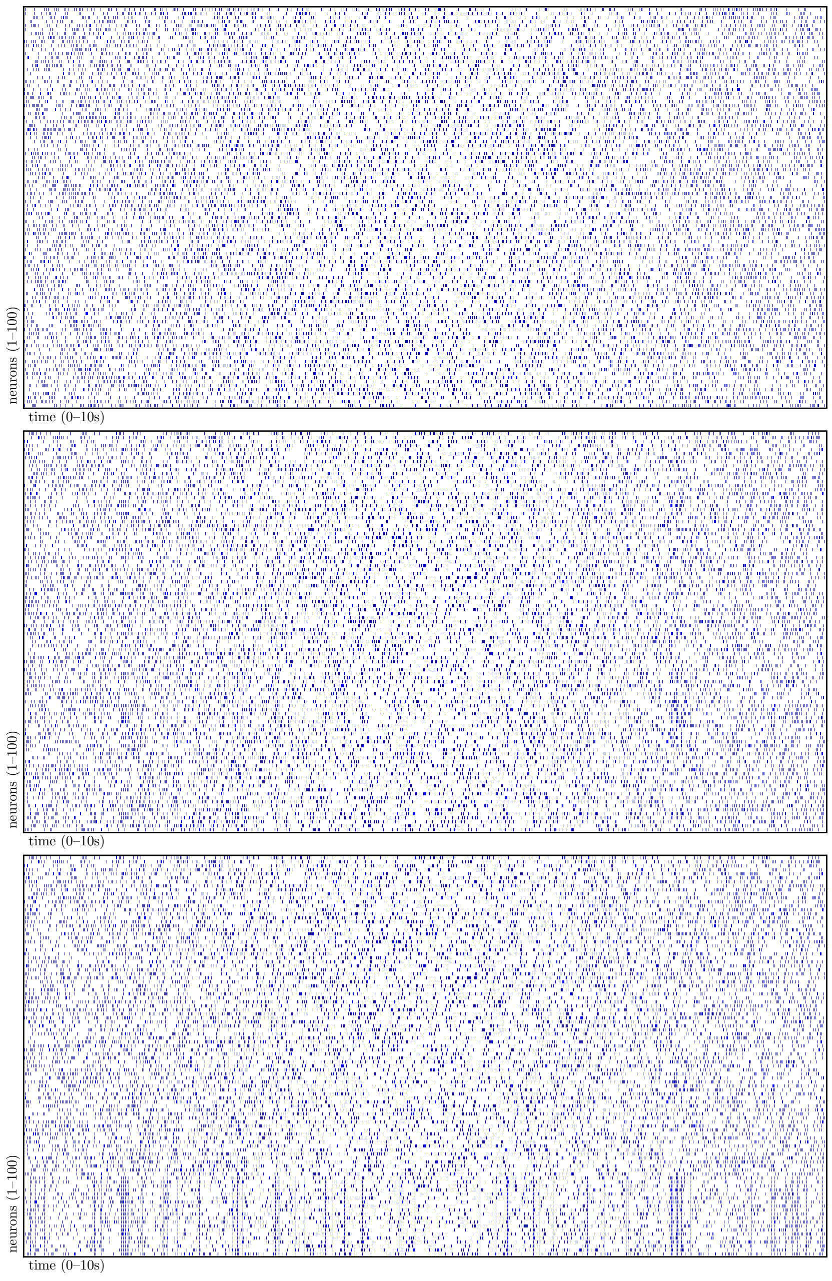

**Fig. 1.** Dot-displays of two sets of parallel spike trains generated with a neurobiological spike train simulator. The top diagram shows independent neurons, while the bottom diagrams contain a group of 20 correlated neurons. In the middle diagram, these neurons are randomly interspersed with independent neurons, while in the bottom diagram they are sorted into the bottom rows (see also: [11–13]).

suggested in [10], which uses the Fiedler vector (i.e. the eigenvector corresponding to the smallest non-zero eigenvalue of the symmetric matrix $L_S = D_S - S$, where $S = (s_{ij})$ is a similarity matrix and $D_S = (d_{ii})$ is a diagonal matrix with $d_{ii} = \sum_j s_{ij}$). The entries/coordinates of the Fiedler vector are used to sort the rows of a given binary matrix such that large tiles (sub-matrices) become visible that contain a lot of 1s. The core argument underlying the approach is that the Fiedler vector minimizes the stress function $x^T L_s x = \sum_{i,j} s_{ij} \cdot (x_i - x_j)^2$. With the constraints $x^T e = 0$ and $x^T x = 1$ this can be easily shown.

In [10] it is claimed that sorting the rows by the corresponding components of the Fiedler vector results in a good ordering by minimizing the inter-row stress, mapping similar objects next to each other. Within this optimization one has to consider $s_{ij}$ to be constant as the similarity of the objects cannot change. Only the components of the vector are subject to change. A problem with this calculation, however, is that even for small similarities a small difference between $x_i$ and $x_j$ would lower the overall stress. Equally an arbitrarily large similarity can still be adjusted by choosing fitting values for $x_i$ and $x_j$. Although it can be considered correct that the Fiedler vector minimizes the overall stress over all suitable vectors, this stress function itself is not necessarily the one that really needs to be minimized to obtain more than a heuristic ordering of the rows.

Experiments we conducted showed that for neural spike trains the Fiedler vector based sorting often yields highly unsatisfactory results. Since we actually want to place similar spike trains close to each other, where high similarity may also be represented as a small distance and vice versa, *Sammon's Non-Linear Mapping* [20] suggests itself as an alternative.

## 3 Presuppositions

With the interpretation of spike trains as binary vectors and parallel spike trains as a binary matrix, we can say that the synchronous spiking activity of neuronal ensembles leads to sub-matrices (or *tiles*) in which—under suitable row and column permutations—every entry is 1. This view presumes that every neuron that is part of an ensemble participates in every coincident spiking event (that is, the probability of copying from the coincidence process into the individual spike trains is 1.0 for all neurons). We start from this simplification, because it helps us to design the algorithm, even though it holds neither for natural processes nor will it be it a necessary precondition for the final algorithm. (Although, of course, best results are achieved if the data is perfect.)

Our task consists in finding a suitable row permutation (re-ordering of the spike trains) and maybe also a column permutation (re-ordering of the time bins) without having to test every pair of vectors (spike trains), and subsequently also every triple, quadruple or even larger groups of vectors by merging all vectors except one and testing the resulting vector against the remaining one (as the Accretion algorithm does [9]). Such an approach is disadvantageous, because group testing and incrementing the size of the groups to be tested leads to an (over-)exponential increase in the number of tests that have to be performed.

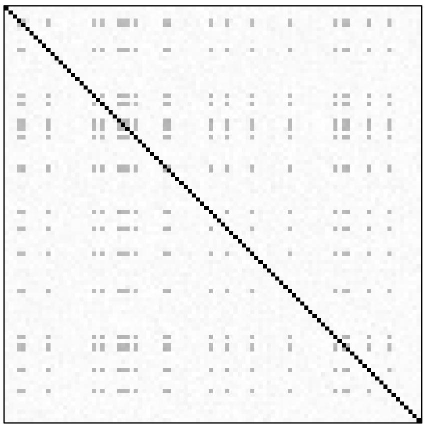

**Fig. 2.** Distance/similarity matrix of 100 neurons (spike trains) computed with the Dice measure with an injected ensemble of size 20 (the darker the gray, the lower the pairwise distance). The data set underlying this distance matrix is depicted as a dot display in the middle diagram of Figure 1: each row (as well as each column) of this diagram corresponds to a row of the diagram in Figure 1 and thus to a neuron.

Hence even a test as simple as the $\chi^2$ test will consume a lot of time—probably without producing significant results. As the low firing probabilities of neurons lead to a relatively low density of the vectors and the even lower rate of coincidences lead to circumstances where the $\chi^2$ test returns excessively many false-positive results, the algorithmic need for larger groups to be tested increases even further. To prevent this, performing an exact test would be desirable but the execution time of tests like Fisher's exact test renders this infeasible.

## 4 Sorting with Non-Linear Mappings

To avoid having to test all subsets of neurons/spike trains and thus to allow the use of an exact test (like Fisher's), some pre-ordering of the neurons/spike trains is necessary, which allows us to carry out the tests on a simple linear traversal through the list of vectors. In this way only neighboring neurons/spike trains have to be tested against each other, reducing the total number of tests required.

Figure 2 shows an example of a distance matrix for a set of spike trains (namely the spike trains shown as a dot display in the middle diagram of Figure 1). Even the untrained human eye can easily distinguish those elements that indicate a smaller distance between two spike trains than the majority of all matrix entries. To make an algorithm able to identify these elements without the aid of image recognition algorithms, it is desirable to sort all neurons/spike trains that are close to each other (according to the used distance measure) to either side (beginning or end) of the sorted set of neurons/spike trains.

Several algorithms for such a (non-)linear mapping are already known, with the classical algorithm known as "Sammon's Non-Linear Mapping" suggesting itself immediately. Usually it is applied to project high-dimensional data onto a two- or three-dimensional space, but we will apply it here to project onto a single dimension. Sammon's algorithm aims at keeping the interpoint distances between two vectors in the target space close to the distances in the original space

**Algorithm 1** Sorting with Sammon's non-linear mapping (SSNLM)

```
Require: List of Spike Trains $L = \{n_0, \ldots\}$, distance measure d, statistic t,
    significance level $\alpha$
Ensure: Set of ensembles A
 1: i := 0
 2: while ($|L| > 1$) do
 3:     $n := |L|$
 4:     $A_i := \emptyset$
 5:     Let $dm$ be a $n \times n$ matrix
 6:     Let $L$ be a list of all IDs
 7:     for all $(a, b) \in L^2, a \geq b$ do
 8:         $dm\,[a, b] = dm\,[b, a] = d(a, b)$
 9:     end for
10:     Perform Sammon mapping with $dm$, $k = 1$, init=PCA, store result in SV
11:     Sort $L$ according to its corresponding entries in SV
12:     if $(t(L[0], L[1]) > t(L[n-2], L[n-1]))$ then
13:         reverse $L$
14:     end if
15:     while $(t(L[0], L[1]) < \alpha)$ do
16:         $A_i := A_i \cup \{n_{L[0]}, n_{L[1]}\}$
17:         $L := L - \{n_{L[0]}\}$
18:     end while
19:     $i := i + 1$
20: end while
21: return  $A = \bigcup_{j=0}^{i-1} \{A_j\}$
```

by minimizing the total error between these distances with an iterative update scheme. The resulting values for each spike train (one-dimensional mapping) are then used as a sorting criterion for the set of spike trains.

## 5 Algorithm

The algorithm assumes that all data points are given as an unordered list and returns a list of lists, each of which contains a possible substructure.

The algorithm works as follows (see Algorithm 1): While there are still neurons/spike trains that have not been assigned to an ensemble or have been rejected, the algorithm continues to look for ensembles (line 2). The first step is to calculate the distance matrix with the distance measure given as an argument (lines 5–9). After the non-linear mapping of this matrix has been performed, the list of neurons/spike trains is sorted according to the result vector (here: a list of real values, lines 10–11). As Sammon's Non-Linear Mapping may map a substructure to any side of the real-valued spectrum, we test for the end of the list that shows stronger evidence for rejecting the null hypothesis of independence and—if necessary—simply reverse the list (lines 12–14). As long as any of the consecutive tests produces $p$-values that are below the chosen level of significance, the corresponding neurons/spike trains are added to the list of the current ensemble (lines 15–18). Each time a test has led to a rejection of the null hypothesis, the first neuron/spike train is also permanently removed from the list of neurons/spike trains, so that it cannot be assigned to any other ensemble.

## 6 Distance Measures

Sammon's algorithm works on a distance matrix, which we have to produce by computing a distance between pairs of binary vectors (namely the discretized spike trains). For this we can select from large variety of distance measures for binary vectors, which emphasize different aspects of dissimilarity (see, for instance, [4] for an overview). As it is to be expected that some of these measures are better suited to find a suitable reordering than others, we compared several of them, in particular those shown in Table 2. All of these measures are defined in terms of the entries of contingency tables as shown in Table 1, which state how often two neurons $A$ and $B$ spike both ($n_{11}$), spike individually without the other ($n_{01}$ and $n_{10}$) and spike neither ($n_{00}$), as well as the row and column sums of these numbers ($n_{0.}$, $n_{1.}$, $n_{.0}$ and $n_{.1}$) and the total number of time bins ($n_{..}$). Although necessarily incomplete, Table 2 lists the most common measures.

In order to assess how well the distance measures shown in Table 2 are able to distinguish between vectors that contain independent, random noise and those that contain possibly correlated data, we evaluated the different distance measures by an outlier detection method, assuming that all neurons/spike trains not belonging to an ensemble are outliers. Such a method has been proposed in [18]. This algorithm, which is inspired by so-called noise clustering [5], introduces a noise cloud, which has the same distance to each point in the data set. A data point is assigned to the noise cloud if and only if there is no other point in the data set that is closer to it than the noise cloud. The algorithm starts with a noise cloud distance of 0, and therefore at the beginning of the algorithm all points belong to the noise cloud. Then the distance to the noise cloud is slowly increased, causing more and more data points to fall out of the noise cloud and to be assigned to a non-noise cluster.

Plotting the distance of the noise cloud against the number of points belonging to the non-noise cluster leads to a curve like those shown in Figure 3. Analyzing the steps in this curve allows us to make some assumptions about possible clusters within the data. Spike trains that have a lot of spikes in common have smaller distances for a properly chosen metric and therefore fall out of the noise cluster earlier than trains that only have a few spikes in common. We assume that neurons/spike trains of an ensemble have significantly smaller distances than neurons/spike trains that do not belong to any ensemble.

The plateaus in each curve of Figure 3 indicate that this method is quite able to identify neurons belonging to an ensemble simply by comparing the inter-vector distances. Even if the copy probability (probability for a coincident spike to be copied to a time bin) drops to values as small as 0.5 the difference between neurons belonging to an ensemble and those not belonging to one remains visible.

**Table 1.** A contingency table for two neurons $A$ and $B$.

| neuron | value | $B$: 0 (no spike) | $B$: 1 (spike) | sum |
|---|---|---|---|---|
| $A$ | 0 (no spike) | $n_{00}$ | $n_{01}$ | $n_{0.}$ |
| $A$ | 1 (spike) | $n_{10}$ | $n_{11}$ | $n_{1.}$ |
| | sum | $n_{.0}$ | $n_{.1}$ | $n_{..}$ |

**Table 2.** Some distance measures used for comparing two item covers.

| | | |
|---|---|---|
| Hamming [14] | $d_{\text{Hamming}}$ | $= \dfrac{n_{01} + n_{10}}{n_{..}}$ |
| Jaccard [16]<br>Tanimoto [22] | $d_{\text{Jaccard}}$ | $= \dfrac{n_{01} + n_{10}}{n_{01} + n_{10} + n_{11}}$ |
| Dice [6]<br>Sørensen [21] | $d_{\text{Dice}}$ | $= \dfrac{n_{01} + n_{10}}{n_{01} + n_{10} + 2n_{11}}$ |
| Rogers &<br>Tanimoto [19] | $d_{\text{R\&T}}$ | $= \dfrac{2(n_{01} + n_{10})}{n_{00} + 2(n_{01} + n_{10}) + n_{11}}$ |
| Yule [24] | $d_{\text{Yule}}$ | $= \dfrac{2n_{01}n_{10}}{n_{11}n_{00} + n_{01}n_{10}}$ |
| $\chi^2$ [4] | $d_{\chi^2}$ | $= 1 - \dfrac{n_{11}n_{00} - n_{01}n_{10}}{\sqrt{n_{0.}n_{1.}n_{.0}n_{.1}}}$ |
| Correlation [8] | $d_{\text{Correlation}}$ | $= \dfrac{1}{2} - \dfrac{n_{11}n_{..} - n_{.1}n_{1.}}{2\sqrt{n_{0.}n_{1.}n_{.0}n_{.1}}}$ |

However, a problem arising from this approach is that even if all neurons are detected that most likely belong to a relevant ensemble, the structure itself remains unknown as neurons are only assigned to the noise or non-noise clusters. Figure 3 shows this quite well: the last diagram on the top row depicts a case where two ensembles of 20 neurons each (both with the same parameters), which were injected into a total set of 100 neurons, cannot be separated.

## 7 Experiments

We conducted several experiments to test the quality of our algorithm w.r.t. different parameter sets. These tests include a different number of ensembles as well as variations of the copy probability from the coincidence process. To generate the test data, a set of non-overlapping ensembles of neurons is chosen and parallel spike trains are simulated with the given parameters.

We considered an ensemble to be (completely) found if and only if every neuron belonging to the ensemble has been successfully identified. We considered an ensemble to be partially found if only a subset of the ensemble's neurons was detected by the algorithm, regardless of how small this subset may be. However,

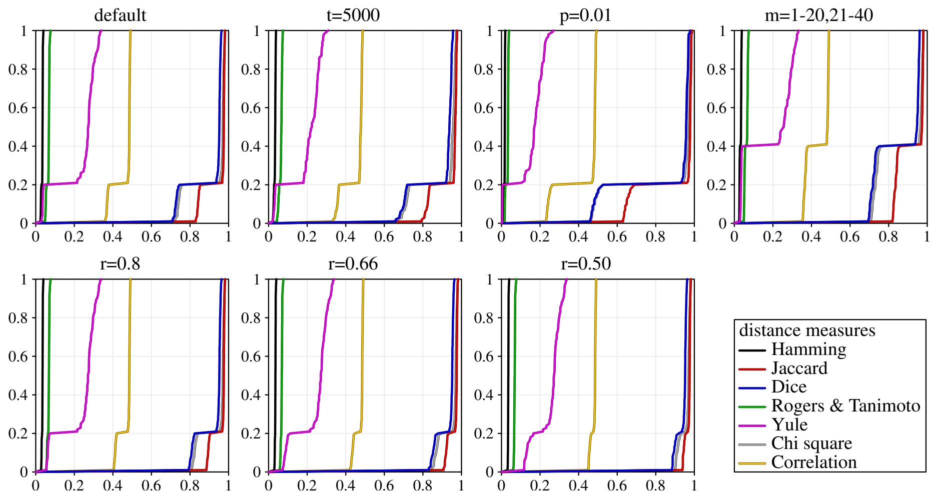

**Fig. 3.** Fraction of neurons *not* assigned to the noise cloud plotted over the distance from the noise cloud for different distance measures. The default parameters of the underlying data sets are: $n = 100$ neurons, $t = 10000$ time bins, $p = 0.02$ probability of an item occurring in a transaction, $m = 1$–$20$ group of neurons potentially spiking together, $c = 0.005$ probability of a coincident spiking event for the group(s) of neurons, $r = 1$ probability with which a spike is actually copied from the coincident spiking process. Deviations from these parameters are stated above the diagrams.

in order to prevent very small (and thus inconclusive) subsets to appear in the result, the minimum ensemble size to be reported was set to three.

The results, for which each configuration was tested several times in order to ensure reliability, are shown in Table 3. The algorithm never reported any neurons belonging to an ensemble not present in the list of actual ensembles. The suppression of false positives in this case most likely originates from the limitation of an ensemble to have at least three neurons or more to be reported.

In addition to the first test, where different parameters were compared, we performed a series of tests with (nearly) fixed parameters where only the copy probability for the coincidences was altered. While the first tests were performed with a copy probability of 1.0 the following tests were performed with a steadily decreasing copy probability. Table 4 shows the results of these tests. It is clearly visible that the detection rate for complete ensembles decreases with decreasing copy probability. This is not surprising, because with copy probabilities as low as 0.5, only about 50% of the neurons participate in a coincident event and every neuron belonging to an ensemble participates in only about 50% of all coincidences. This leaves plenty of space for coincidences simply disappearing in the background noise. On the other hand the rate of partially detected ensembles stays stable for a larger range of copy probabilities, indicating that only a few neurons are not recognized by this method as they are not participating in enough coincidences. Even if the copy probability drops to 0.4 the partial detec-

**Table 3.** Experimental results for the assembly detection method based on reordering the items/neurons with the Sammon projection.

| | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
|---|---|---|---|---|---|
| # of runs/experiments | 50 | 50 | 50 | 50 | 500 |
| # of neurons per runs | 100 | 100 | 100 | 100 | 100 |
| # of time bins per spike train | 10000 | 10000 | 10000 | 5000 | 5000 |
| firing probability | 0.02 | 0.02 | 0.02 | 0.03 | 0.02 |
| coincidence probability | 0.0075 | 0.0075 | 0.0075 | 0.0075 | 0.005 |
| copy probability | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| assemblies per run | 0-5 | 0-6 | 0-5 | 0-5 | 0-6 |
| assembly size | 20 | 5-20 | 10 | 10 | 10 |
| significance level for statistic | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| distance measure used | $d_{\text{Dice}}$ | $d_{\text{Dice}}$ | $d_{\text{Jaccard}}$ | $d_{\text{Dice}}$ | $d_{\text{Yule}}$ |
| total # of assemblies | 108 | 132 | 128 | 118 | 1227 |
| total # of assemblies found | 105 | 97 | 86 | 114 | 1170 |
| total # of partial ens. found | 0 | 5 | 4 | 0 | 57 |
| success rate | 97.2% | 73.5% | 67.2% | 96.6% | 95.4% |
| success rate (incl. partial finds) | 97.2% | 77.3% | 70.3% | 96.6% | 100% |

tion rate remains at almost 80%. This means that even with (on average) 60% of the data missing, still 80% of the ensembles were detected at least partially.

## 8 Conclusions and Future Work

Based on the results shown in the preceding section we can conclude that a testing procedure based on sorting the data and performing only certain combinations of pair-wise tests based on the result of a non-linear mapping produces very good result for the detection of neural ensembles. This holds even if the quality of the data is not ideal—be it lossy data accumulation or missing spikes due to the underlying (biological) process. The tests performed show that even with 60% of the relevant data (i.e. the coincident spikes) lost nearly as much as 80% of the ensembles could be detected at least partially.

Yet, there is still room to improve on the algorithm. At the moment the decision whether a neuron belongs to a relevant ensemble or not is based on rule-of-thumb values. No sophisticated methods are used to analyze the noise/non-noise line of the outlier detection method (see Figure 3). The difference between the two groups is currently only found by a constant value depending on the distance function used rather than using the inflection or the saddle point of the resulting curve. Here a far more analytical approach would be desirable to automate the detection process. Also overlapping groups cannot be distinguished at the moment as data points are assigned to at most one substructure.

**Acknowledgments**

This work has been partially supported by the Helmholtz Alliance on Systems Biology.

**Table 4.** Experimental results for the assembly detection method based on reordering the items/neurons with the Sammon projection, different copy probabilities.

| | Test 6 | Test 7 | Test 8 | Test 9 | Test 10 |
|---|---|---|---|---|---|
| # of runs/experiments | 250 | 250 | 250 | 50 | 100 |
| # of neurons per runs | 100 | 100 | 100 | 100 | 100 |
| # of time bins per spike train | 5000 | 5000 | 5000 | 10000 | 10000 |
| firing probability | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| coincidence probability | 0.005 | 0.005 | 0.005 | 0.005 | 0.005 |
| copy probability | 0.85 | 0.75 | 0.6 | 0.6 | 0.4 |
| assemblies per run | 0-6 | 0-6 | 0-6 | 0-6 | 0-6 |
| assembly size | 10 | 10 | 10 | 10 | 10 |
| significance level for statistic | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| distance measure used | $d_{\text{Yule}}$ | $d_{\text{Yule}}$ | $d_{\text{Yule}}$ | $d_{\text{Yule}}$ | $d_{\text{Yule}}$ |
| total # of assemblies | 648 | 684 | 633 | 129 | 236 |
| total # of assemblies found | 612 | 633 | 417 | 119 | 30 |
| total # of partial ens. found | 35 | 39 | 188 | 4 | 158 |
| success rate | 94.4% | 92.5% | 65.9% | 92.3% | 12.7% |
| success rate (incl. partial finds) | 99.9% | 98.3% | 95.6% | 95.4% | 79.7% |

## 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 Ann. Computational Neuroscience Meeting (CNS*2009)*. BMC Neuroscience, 10(Suppl 1), Berlin, Germany 2009
2. J. Besson, C. Robardet, and J.-F. Boulicaut. Mining a New Fault-Tolerant Pattern Type as an Alternative to Formal Concept Discovery. *Proc. Int. Conference on Computational Science (ICCS 2006, Reading, United Kingdom)*, 144–157. Springer-Verlag, Berlin, Germany 2006
3. C. Borgelt and X. Wang. SaM: A Split and Merge Algorithm for Fuzzy Frequent Item Set Mining. *Proc. 13th Int. Fuzzy Systems Association World Congress and 6th Conf. of the European Society for Fuzzy Logic and Technology (IFSA/EUSFLAT'09, Lisbon, Portugal)*, 968–973. IFSA/EUSFLAT Organization Committee, Lisbon, Portugal 2009
4. 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
5. R.N. Davé. Characterization and Detection of Noise in Clustering. *Pattern Recognition Letters* 12:657–664. Elsevier Science, Amsterdam, Netherlands 1991
6. L.R. Dice. Measures of the Amount of Ecologic Association between Species. *Ecology* 26:297–302. Ecological Society of America, Ithaca, NY, USA 1945
7. M. Diesmann, M.-O. Gewaltig, and A. Aertsen. Conditions for Stable Propagation of Synchronous Spiking in Cortical Neural Networks. *Nature* 402:529–533. Nature Publishing/Macmillan, New York, NY, USA 1999
8. A.L. Edwards. The Correlation Coefficient. *An Introduction to Linear Regression and Correlation*, 33–46. W.H. Freeman, San Francisco, CA, USA 1976

9. 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
10. A. Gionis, H. Mannila, and J.K. Seppänen. Geometric and Combinatorial Tiles in 0-1 Data. *Proc. 8th Europ. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'04, Pisa, Italy)*, LNAI 3202:173–184. Springer-Verlag, Berlin, Germany 2004
11. S. Grün, M. Diesmann, and A. Aertsen. 'Unitary Events' in Multiple Single-neuron Activity. I. Detection and Significance. *Neural Computation* 14(1):43–80. MIT Press, Cambridge, MA, USA 2002
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. Springer, Heidelberg, Germany 2008
13. 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. Hindawi Publishing Corp., New York, NY, USA 2009/2010
14. R.V. Hamming. Error Detecting and Error Correcting Codes. *Bell Systems Tech. Journal* 29:147–160. Bell Laboratories, Murray Hill, NJ, USA 1950
15. D.O. Hebb. *The Organization of Behavior.* J. Wiley & Sons, New York, NY, USA 1949
16. 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
17. M.S. Lewicki. A Review of Methods for Spike Sorting: The Detection and Classification of Neural Action Potentials. *Network: Computation in Neural Systems* 9:R53–R78. Informa, St. Helier, Jersey, France 1998
18. F. Rehm, F. Klawonn and R. Kruse. A Novel Approach to Noise Clustering for Outlier Detection. *Soft Computing — A Fusion of Foundations, Methodologies and Applications* 11(5):489–494. Springer, Berlin, Germany 2007
19. 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
20. J.W. Sammon. A Nonlinear Mapping for Data Structure Analysis. *IEEE Trans. Comput.* 18(5):401–409. IEEE Computer Society, Washington, DC, USA 1969
21. T. Sørensen. A Method of Establishing Groups of Equal Amplitude in Plant Sociology based on Similarity of Species and its Application to Analyses of the Vegetation on Danish Commons. *Biologiske Skrifter / Kongelige Danske Videnskabernes Selskab* 5(4):1-34. Denmark 1948
22. T.T. Tanimoto. IBM Internal Report, November 17, 1957
23. Xiaomeng Wang, Christian Borgelt and Rudolf Kruse. Mining Fuzzy Frequent Item Sets. *Proc. 11th Int. Fuzzy Systems Association World Congress (IFSA'05, Beijing, China)*, 528–533. Tsinghua University Press and Springer-Verlag, Beijing, China, and Heidelberg, Germany 2005
24. G.U. Yule. On the Association of Attributes in Statistics. *Philosophical Transactions of the Royal Society of London*, Series A, 194:257–319, 1900