Simple Pattern Spectrum Estimation for Fast Pattern Filtering with CoCoNAD

Christian Borgelt and David Picado-Muiño

European Centre for Soft Computing
Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Spain
{christian.borgelt|david.picado}@softcomputing.es

Abstract. CoCoNAD (for Co*ntinuous-time* C*losed* N*euron* A*ssembly* D*etection*) is an algorithm for finding frequent parallel episodes in event sequences, which was developed particularly for neural spike train analysis. It has been enhanced by so-called Pattern Spectrum Filtering (PSF), which generates and analyzes surrogate data sets to identify statistically significant patterns, and Pattern Set Reduction (PSR), which eliminates spurious induced patterns. A certain drawback of the former is that a sizable number of surrogates (usually several thousand) have to be generated and analyzed in order to achieve reliable results, which can render the analysis process slow (depending on the analysis parameters). However, since the structure of a pattern spectrum is actually fairly simple, we propose a simple estimation method, with which (an approximation of) a pattern spectrum can be derived from the original data, bypassing the time-consuming generation and analysis of surrogate data sets.

1 Introduction

About a year ago we presented CoCoNAD (for Co*ntinuous-time* C*losed* N*euron* A*ssembly* D*etection*) [4], an algorithm for finding frequent parallel episodes in event sequences, which are defined over a continuous (time) domain. The name of this algorithm already indicates that the application domain motivating our investigation is the analysis of *parallel spike trains* in neurobiology: sequences of points in time, one per neuron, that represent the times at which an electrical impulse (*action potential* or *spike*) is emitted. Our objective is to identify *neuronal assemblies*, intuitively understood as groups of neurons that tend to exhibit synchronous spiking. Such cell assemblies were proposed as a model for encoding and processing information in biological neural networks [8]. As a (possibly) first step in the identification of neuronal assemblies, we look for (significant) *frequent neuronal patterns*, that is, groups of neurons that exhibit *frequent synchronous spiking* that cannot be explained as a chance occurrence [13, 16]. In this paper we draw on this application domain for the parameters of the (artificially generated) data sets with which we tested the proposed pattern spectrum estimation, but remark that our method is much more widely applicable.

The CoCoNAD algorithm differs from other approaches to find frequent parallel episodes in event sequences, like those, for example, in [12, 6, 10] or [15] (some of which are designed for discrete item sequences, although a transfer to a continuous (time) domain is fairly straightforward), by the support definition it employs. While the mentioned approaches define the support of a parallel episode as the (maximal) number of non-overlapping minimal windows covering instances of the episode, CoCoNAD relies on a maximum independent set (MIS) approach. This allows to count different instances of a parallel episode even though the windows covering them overlap, thus leading to a potentially higher support count. Nevertheless the resulting support measure remains anti-monotone, because no spike is contained in more than one counted instance [4].

Furthermore, in order to single out significant frequent patterns from the output, while avoiding the severe multiple testing problem that results from the usually very large number of frequent patterns, we proposed *pattern spectrum filtering* (PSF) in [13].[1] This method relies on generating and analyzing surrogate data sets as an implicit representation of the null hypothesis of items occurring independently. It eliminates all patterns found in the original data, for which a analogous pattern was found in a surrogate data set (since then the pattern can be explained as a chance event, cf. Section 3). This method was further detailed in [16], where it was also extended with *pattern set reduction* (PSR), which strives to eliminate spurious patterns that are merely induced by an actual pattern (that is, subset, superset and overlapping patterns) with a preference relation.[2]

These methods (PSF and PSR) proved to be very effective in singling out patterns from artificially generated data. However, the need to generate and analyze a sizable number of surrogate data sets (usually several thousand) can render the mining process slow, especially if the data exhibits high event frequencies and the analysis window width (maximum time allowed to cover an occurrence of a parallel episode) is chosen to be large. To overcome this drawback, we strive in this paper to exploit the fact that a pattern spectrum actually has a fairly simple structure and thus allows for an (at least approximate) estimation from the original data, bypassing surrogate data generation. The core idea is to count, based on the user-specified analysis window width, the possible "slots" for patterns of different sizes and to estimate from these counts the (expected) pattern support distribution with a Poisson approximation.

The remainder of this paper is structured as follows: Section 2 briefly reviews how (frequent) parallel episodes are mined with the CoCoNAD algorithm and Section 3 how the output is reduced with pattern spectrum filtering (PSF) and pattern set reduction (PSR) to significant, non-induced patterns. Section 4 describes the simple, yet effective method with which we estimate a pattern spectrum from the original data. In Section 5 we report experiments on artificially generated data sets and thus demonstrate the quality of pattern spectrum estimation. Finally, in Section 6 we draw conclusions from our discussion.

[1] Even though pattern spectrum filtering was presented for time-binned data in [13] (which reduces the problem to classical frequent item set mining: each time bin gives rise to one transaction), it can easily be transferred to the continuous domain.

[2] Although time-binned data was considered in [16], the idea of pattern set reduction can easily be transferred to continuous time, requiring only a small adaptation.

2 Mining Parallel Episodes with CoCoNAD

We (partially) adopt notation and terminology from [12]. Our data are (finite) *sequences of events* of the form $\mathcal{S} = \{\langle i_1, t_1\rangle, \ldots, \langle i_m, t_m\rangle\}$, $m \in \mathbb{N}$, where i_k in the *event* $\langle i_k, t_k\rangle$ is the *event type* or *item* (taken from an item base B) and $t_k \in \mathbb{R}$ is the time of occurrence of i_k, $k \in \{1, \ldots, m\}$. Note that the fact that $\mathcal{S}$ is a set implies that there cannot be two events with the same item occurring at the same time: events with the same item must differ in their occurrence time and events occurring at the same time must have different types/items. Note also that in our motivating application (i.e., spike train analysis), the items are the neurons and the events capture the times at which spikes are emitted.

Episodes (in $\mathcal{S}$) are sets of items $I \subseteq B$ that are endowed with a partial order and usually required to occur in $\mathcal{S}$ within a certain time span. *Parallel episodes*, on which we focus in this paper, have no constraints on the relative order of their elements. An *instance (or occurrence) of a parallel episode* $I \subseteq B$, $I \neq \emptyset$, (or a *(set of) synchronous event(s)* for I) in an event sequence $\mathcal{S}$ with respect to a (user-specified) time span $w \in \mathbb{R}^+$ can be defined as a subsequence $\mathcal{R} \subseteq \mathcal{S}$, which contains exactly one event per item $i \in I$ and which can be covered by a (time) window of width at most w. Hence the set of all instances of a parallel episode $I \subseteq B$, $I \neq \emptyset$, in $\mathcal{S}$ is

$$\mathcal{E}_{\mathcal{S}}(I, w) = \big\{\mathcal{R} \subseteq \mathcal{S} \mid \{i \mid \langle i, t\rangle \in \mathcal{R}\} = I \ \wedge\ |\mathcal{R}| = |I| \ \wedge\ \sigma(\mathcal{R}, w) = 1\big\},$$

where the operator σ captures the (approximate) synchrony of the events in $\mathcal{R}$:

$$\sigma(\mathcal{R}, w) = \begin{cases} 1 & \text{if } \max\{t \mid \langle i, t\rangle \in \mathcal{R}\} - \min\{t \mid \langle i, t\rangle \in \mathcal{R}\} \leq w, \\ 0 & \text{otherwise.} \end{cases}$$

That is, $\sigma(\mathcal{R}, w) = 1$ iff all events in $\mathcal{R}$ can be covered by a (time) window of width at most w. We then define the support of a parallel episode $I \subseteq B$ in $\mathcal{S}$ as

$$s_{\mathcal{S}}(I, w) = \max\big\{|\mathcal{U}| \ \mid\ \mathcal{U} \subseteq \mathcal{E}_{\mathcal{S}}(I, w) \ \wedge\ \forall \mathcal{R}_1, \mathcal{R}_2 \in \mathcal{U}; \mathcal{R}_1 \neq \mathcal{R}_2 \colon \mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset\big\},$$

that is, as the size of a maximum independent set of the instances of I. Although in the general case the maximum independent set problem is NP-complete [9] and even hard to approximate [7], the problem instances we are facing here are constrained by the underlying one-dimensional time domain, which makes it possible to devise an efficient greedy algorithm that solves it exactly [14]. Pseudo-code of the support counting procedure can be found in [4].

Frequent parallel episodes are then mined, based on this support definition, with a standard recursive divide-and-conquer scheme that enumerates candidate item sets, which may also be seen as a depth-first search. The search is pruned, as in all such algorithms, with the so-called *apriori property: no superset of an infrequent parallel episode can be frequent*, since the support measure defined above can be shown to be *anti-monotone* (see, for example, [17, 5]). Pseudo-code of the mining procedure including efficient event filtering can be found in [4].

3 Pattern Spectrum Filtering and Pattern Set Reduction

Trying to single out significant patterns proves to be less simple than it may appear at first sight, since one has to cope with the following two problems: in the first place, one has to find a proper statistic that captures how (un)likely it is to observe a certain pattern under the null hypothesis that items occur independently. Secondly, the huge number of potential patterns causes a severe multiple testing problem, which is not easy to overcome with standard methods. In [13] we provided a fairly extensive discussion and concluded that a different approach than evaluating individual patterns with statistics is needed.

As a solution, *pattern spectrum filtering* (PSF) was proposed in [13] based on the following insight: even if it is highly unlikely that a *specific group* of z items co-occurs c times, it may still be likely that *some group* of z items co-occurs c times, even if items occur independently. The reason is simply that there are so many possible groups of z items (unless the item base B as well as z are tiny) that even though each group has only a tiny probability of co-occurring c times, it may be almost certain that *one of them* co-occurs c times.[3] As a consequence, since there is no *a-priori* reason to prefer certain sets of z items over others (even though a refined analysis, on which we are working, may take individual item frequencies into account), we should not declare a pattern significant if the occurrence of a counterpart (same size z and same or higher support c) can be explained as a chance event under the null hypothesis of independent items.

As a consequence, we pool patterns with the same *pattern signature* $\langle z, c\rangle$, and collect for each signature the (average) number of patterns that we observe in surrogate data. This yields what we call a *pattern spectrum* (see Figures 2 and 3). Pattern spectrum filtering consists in keeping only such patterns found in the original data for which no counterpart with the same signature (or a signature with the same z, but larger c) was observed in surrogate data, as such a counterpart would show that the pattern can be explained as a chance event.

The essential part of this procedure is, of course, the generation of surrogate data, for which we rely on a simple permutation procedure: the occurrence times of the events are kept and the items (the event types) are randomly permuted. This destroys any co-occurrence of items that may be present in the data and thus produces data that implicitly represent the null hypothesis of independently occurring items. A discussion of other surrogate data generation approaches that are common in the area of neural spike train analysis can be found in [11].

Note that pattern spectrum filtering still suffers from a certain amount of *multiple testing*: every pair $\langle z, c\rangle$ that is found in the original data gives rise to one test. However, the pairs $\langle z, c\rangle$ are *much fewer* than the number of specific item sets. As a consequence, simple approaches like *Bonferroni correction* [2, 1] become feasible, with which the number of needed surrogate data sets can be computed [13]: given a desired overall significance level α and the number k of

[3] This is actually the case for, say, $z = 5$ and $c = 4$ in our data, for which patterns are essentially certain to occur, see Figures 2 and 3, although the probability of observing a specific set of 5 items co-occurring 4 times is extremely small ($< 10^{-8}$).

pattern signatures to test, at least k/α surrogate data sets have to be analyzed. With the common choice $\alpha = 1\%$ and usually several dozen pattern signatures being observed, this rule recommends to generate several thousand data sets. In our experiments we always chose 10,000, regardless of the actual number of pattern signatures, in order to ensure a uniform procedure for all data sets.

As a further filtering step, *pattern set reduction* was proposed in [16], which is intended to take care of the fact that an actual pattern induces other, spurious patterns that are subsets or supersets or overlap the actual patterns. These spurious patterns are reduced with the help of a preference relation between patterns and the principle that only patterns are kept to which no other pattern is preferred. A simple heuristic, but very effective preference relation is the following: let $X, Y \subseteq B$ be two patterns with $Y \subseteq X$ and let $z_X = |X|$ and $z_Y = |Y|$ be their sizes and c_X and c_Y their support values. The pattern X is preferred to Y if $z_X \cdot c_X \geq z_Y \cdot c_Y$. Otherwise Y is preferred to X. The core idea underlying this method is that under certain simplifying assumptions the occurrence probability of a pattern is inversely proportional to the number of individual events underlying it, that is, to the product $z \cdot c$. Intuitively, the above preference relation therefore prefers the less probable pattern. Alternatives to this preference relation and a more detailed discussion can be found in [16].

4 Pattern Spectrum Estimation

As already mentioned in the introduction, pattern spectrum filtering suffers from the problem that a sizeable number of surrogate data sets (usually several thousand) need to be generated and analyzed, which can render the analysis process slow, especially if due to high event frequencies and a large window width w an individual run already takes some time. Even though pattern spectrum generation lends itself very well to parallelization (since each surrogate data set can be generated and analyzed on a different processor core), it is desirable to find a faster way of obtaining (at least an approximation of) a pattern spectrum.

As a solution, we propose *pattern spectrum estimation* in this paper. This method draws on the idea that by counting the "slots" for patterns of different sizes, we can estimate the support distribution of the patterns via a standard Poisson approximation of the actual binomial distribution. By a "slot" for a pattern size z we mean any collection of z events in the event sequence $\mathcal{S}$ to analyze that can be covered by the chosen analysis window width w. Each such slot can hold an instance of a specific parallel episode $I \subseteq B$, $|I| = z$. With the probability of a pattern instance occurring in such a slot, that is, the probability that the z items constituting the parallel episode are chosen in a random selection (since we want to mimic independent items, as this is the implicitly represented null hypothesis), we obtain a probability distribution over the different numbers of occurrences of the parallel episode in the counted number of slots. This distribution is actually binomial, but it can be approximated well by a Poisson distribution, because the number of slots is usually very large while the occurrence probability of a specific parallel episode is very small.

By scaling the resulting probability distribution over the possible support values to the total number of patterns that can occur, we obtain expected counts for the different pattern signatures with size z. Executing the process for all sizes $z \in \{1, \ldots, |B|\}$ then yields the desired pattern spectrum.

Formally, the number of slots for each pattern size z is defined as

$$\forall z \in \{1, \ldots, |B|\}: \quad N_{\mathcal{S}}(z, w) = \big|\big\{\mathcal{R} \subseteq \mathcal{S} \mid |\mathcal{R}| = z \ \wedge\ \sigma(\mathcal{R}, w) = 1\big\}\big|.$$

However, this formula does not lend itself well to implementation. Therefore, to count the slots for each pattern size, we first pass a sliding window over the event sequence $\mathcal{S}$, stopping at each event $\langle i, t\rangle \in \mathcal{S}$, and collecting the events in the (time) window $[t, t + w]$. That is, we consider the set of event sequences

$$\mathcal{W}_{\mathcal{S}}(w) = \big\{\mathcal{R}_e \mid e = \langle i, t\rangle \in \mathcal{S} \ \wedge\ \mathcal{R}_e = \{\langle i', t'\rangle \in \mathcal{S} \mid t' \in [t, t + w]\}\big\}.$$

Using the mentioned sliding window method, this set is easy to enumerate.

From this set we then obtain the slot counts per pattern size z as

$$\forall z \in \{1, \ldots, |B|\}: \quad N_{\mathcal{S}}(z, w) = \textstyle\sum_{\mathcal{R} \in \mathcal{W}_{\mathcal{S}}, |\mathcal{R}| \geq z} \binom{|R|-1}{z-1}.$$

This formula can be understood as follows: only subsequences in $\mathcal{W}_{\mathcal{S}}$ that contain at least z events can contain slots for a pattern of size z and therefore the sum considers only $\mathcal{R}$ with $|\mathcal{R}| \geq z$. In principle, all subsets of z events in a given $\mathcal{R}$ have to be considered. However, the event sequences in $\mathcal{W}_{\mathcal{S}}$ overlap, and thus summing $\binom{|R|}{z}$ could count the same slot multiple times. We avoid this by counting for each $\mathcal{R}$ only those subsets of size z that contain the first event in $\mathcal{R}$ (that is, the event at which the window defining $\mathcal{R}$ is anchored). Of the remaining $|\mathcal{R}| - 1$ events in $\mathcal{R}$ we then choose $z - 1$ to obtain a slot of size z.

For the support distribution estimation let us first assume that all items (and thus all parallel episodes) are equally likely (we abandon this assumption later, but it simplifies the explanation here) and occur independently (as required by the null hypothesis). Then the probability that a specific parallel episode $I \subseteq B$, $|I| = z$, occurs in a slot of size z is $P_{\mathcal{S}}(I) = 1/\binom{|B|}{z}$. The probability distribution over the support values c can thus be approximated by a Poisson distribution as

$$P_{\mathcal{S}}(\langle z, c\rangle) = \frac{\lambda^c}{c!}\, e^{-\lambda} \qquad \text{with} \qquad \lambda = N_{\mathcal{S}}(z, w) \,/\, \binom{|B|}{z},$$

because $N_{\mathcal{S}}(z, w)$ is (very) large and $1/\binom{|B|}{z}$ is (very) small and thus the standard conditions for a Poisson approximation are met. Multiplying this probability distribution by the number of parallel episodes of size z yields the expected number of patterns with signature $\langle z, c\rangle$, namely

$$E(\langle z, c\rangle) = \binom{|B|}{z} \frac{\lambda^c}{c!} e^{-\lambda},$$

and thus the desired pattern spectrum. To account for the finite number M of surrogate data sets that would have been generated otherwise, one may threshold it with $1/M$ and thus obtain an equivalent to a surrogate data pattern spectrum.

It should be noted, though, that this derivation is only an approximation in several respects. Apart from the Poisson approximation (which, however, is the least harmful, since the conditions for its application are met), it suffers from neglecting the following: in the first place, the support distributions for parallel episodes of the same size are negatively correlated, since more occurrences of one pattern must be compensated by fewer occurrences of other patterns. Hence simply multiplying the individual distributions by the number of possible parallel episodes is not quite correct. Secondly, the "slots" for a given size z overlap, that is, the same event can contribute to multiple slots for a given size. However, in the above derivation the slots are treated as if they are independent. Both of these issues can be expected to lead to an overestimate of the average number of patterns for a signature $\langle z, c\rangle$. The overestimate can be expected to be small, though, because the correlation is small due to the large number of parallel episodes and the amount of overlap is small relative to the total number of slots. Finally, the overlap actually increases the occurrence probability of a pattern, since a slot overlapping one that contains an instance of a pattern has a higher probability of containing the same pattern than an independent slot. This is less relevant, though, because CoCoNAD does not count both of two overlapping instances (see the support definition in Section 2).

However, the most serious drawback of the method as we described it up to now is the assumption that all items (and thus all parallel episodes) are equally likely. This assumption is rarely satisfied in practice, as the firing rates of recorded neurons tend to differ (considerably). Therefore we remove this assumption as follows: since for any practically relevant size of the item base B it is impossible to enumerate all parallel episodes of size z, we draw a sample of K subsets of the item base B having size z (we chose $K = 1,000$ to cover sufficiently many configurations), using equal probabilities for all items. For each drawn parallel episode $I \subseteq B$, $|I| = z$, we compute the Poisson distribution over the support values c as described above and sum these distributions over the elements of the sample. The result, a sum of Poisson distributions with different parameters λ (which take take of the different occurrence probabilities of the items), is then scaled to the total number of possible parallel episodes of size z. That is, the distribution is multiplied with $\binom{|B|}{z}/K$ (and thresholded with $1/M$ where M is the number of surrogate data sets that would have been generated otherwise) to obtain the pattern spectrum.

In this computation one has to take care that the probability of a parallel episode $I \subseteq B$, $|I| = z$, cannot simply be computed as $P_{\mathcal{S}}(I) = \prod_{i \in I} p_i$, where

$$\forall i \in B\colon \quad p_i = |\{t \mid \langle i, t\rangle \in \mathcal{S}\}| \;/\; |\mathcal{S}|$$

is the probability that a randomly chosen event has item i. The reason is that a chosen item cannot be chosen again and therefore the probability should rather be computed, using an order $i_1, \ldots, i_z$ of the items in I, like

$$P_{\mathcal{S}}(I) = \prod_{k=1}^{z} \frac{p_{i_k}}{1 - \sum_{j=1}^{k-1} p_{i_j}}.$$

However, there is no reason to prefer any specific order of the items over any other. To handle this problem, we draw a small sample of orders (permutations) for each chosen parallel episode and average over these orders as well as their reversed forms (unless $z \leq 4$, for which we simply enumerate all orders, since their number is manageable). We consider both a generated item order as well as its reverse, because the computed probabilities are certain to lie on opposite sides of the mean probability. In this way the average over the considered item orders can be expected to yield a better estimate of the mean probability.

In our experiments we found that if the item probabilities actually differed, this approach produced a better pattern spectrum estimate than assuming equal item probabilities. However, it tended to overestimate the occurrence frequencies of the pattern signatures. On the other hand, using equal item probabilities for the estimation (even though the probabilities actually differed) tended to produce underestimates. As a straightforward heuristic to correct these effects, we introduced a factor that contracts the probability dispersion of the items, thus reducing the overestimate. That is, before the support distribution estimation we transform the item probabilities computed above according to

$$p'_i = \bar{p} + \varrho(p_i - \bar{p}) \qquad \text{where} \qquad \bar{p} = 1/|B| \qquad \text{and} \qquad \varrho \in [0, 1].$$

By evaluating the quality of an estimated pattern spectrum relative to one derived from surrogate data, focusing on the expected signature counts close to the decision border for rejecting a found pattern (technically: expected counts $E(\langle z, c\rangle) \in [0.0001, 0.1]$) and using a logarithmic error measure (that is, computing differences of logarithms of pattern counts rather than differences of the counts directly), we found that $\varrho \in [0.4, 0.5]$ is a good choice for basically all parameter combinations that we tested. We only observed a slight dependence on the window width w: for larger values of w, smaller values of ϱ appear to produce better results. For the experiments reported in the next section we used the fixed value $\varrho = 0.5$, but results for other values did not differ much.

5 Experiments

We implemented our pattern spectrum estimation in both Python and C (see below for the source code) and applied it to a variety of data sets that were generated to resemble the data sets we meet in neural spike train analysis (our motivating application area). In total we generated 108 data sets, each of which represented 3 seconds of recording time. We varied the number of neurons (or items, $n \in \{40, 60, 80, 100\}$, which are typical numbers that can be recorded with state of the art equipment), the averaging firing rate ($r \in \{10, 20, 30\}$Hz), the firing rate variation over the neurons (either the same for all neurons or linearly increasing from the lowest to the highest, which was chosen to be 2 or 3 times the lowest rate) and the firing rate variation over time (using either a flat rate profile or a burst profile that mimics presenting and removing a stimulus three times, where the highest rate was chosen to be 2 or 3 times the lowest rate). As an illustration, dot displays of some of these data sets are shown in Figure 1.

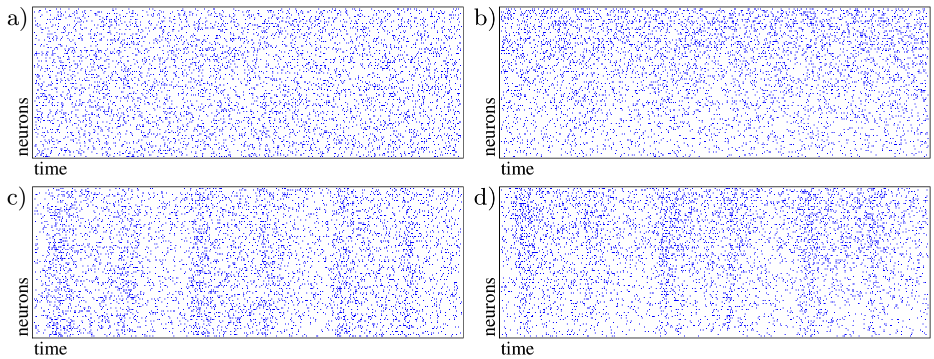

Fig. 1. Some examples of test data sets: a) stationary Poisson processes, same firing rate for all neurons; b) stationary Poisson processes, different firing rates (3:1 highest to lowest); c) burst profile (3:1 highest to lowest rate), same for all neurons; d) burst profile (3:1 highest to lowest rate), different average firing rates (3:1 highest to lowest).

Each data set was then analyzed with four different window widths ($w \in \{2, 3, 4, 5\}$ms), yielding a total of 432 configurations. In each configuration a pattern spectrum was obtained by generating and analyzing 10,000 surrogate data sets and by estimating it with the described method. With this number of surrogate data sets we can be sure to meet an overall significance level of $\alpha = 1\%$ or even lower, since the number of pattern signatures was always clearly less than 100. (See the estimation of the number of needed surrogate data sets via Bonferroni correction in Section 3.) We observed that the estimated pattern spectra match the ones derived from surrogate data sets very well. However, they can be obtained in a small fraction of the time: while estimating a pattern spectrum takes only a fraction of a second, generating and analyzing 10,000 surrogate data sets can take hours and even days (as we experienced for some of the data sets we experimented with). Examples of obtained pattern spectra are shown in Figure 2 ($w = 3$ms) and Figure 3 ($w = 5$ms).[4]

6 Conclusions

Although in several respects a (coarse) approximation, the pattern spectrum estimation we presented in this paper proved to produce very usable pattern spectra for the (artificially generated) data sets on which we tested it. The speed-up that can be achieved by estimation is substantial (often orders of magnitude). This speed-up can be exploited, for example, to automatically determine a proper window width w by trying different values and evaluating the result. Doing the same with surrogate data sets can turn out to be tedious and time-consuming, since each window width requires a new set of surrogates to be generated and

[4] Diagrams of the full set of pattern spectra can be found here: `http://www.borgelt.net/docs/spectra.pdf`

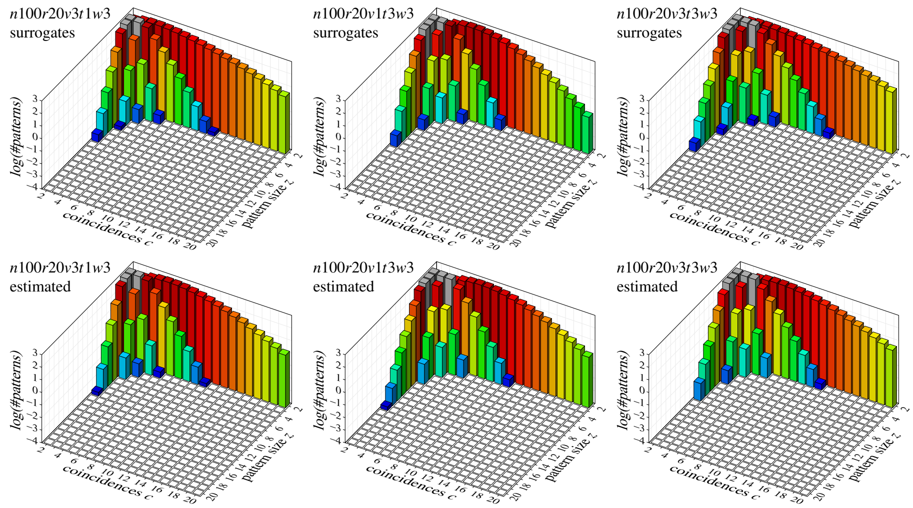

Fig. 2. Pattern spectra for window width 3ms, generated from surrogate data sets (top) or estimated with the described method (bottom). The top word in a diagram title encodes the data set parameters: n—number of neurons, r—firing rate, v—firing rate variation over neurons as $x : 1$ (highest to lowest), t—firing rate variation over time as $x : 1$ (highest to lowest), w—analysis window width. Grey bars extend beyond the top of the diagram, white squares represent zero occurrences. Note the logarithmic scale.

analyzed. We are currently in the process of applying our pattern mining method (CoCoNAD + PSF + PSR, with pattern spectrum estimation as well as deriving a pattern spectrum by generating and analyzing surrogate data sets) to real-world data sets. Preliminary results look very promising.

Software and Source Code

Python and C implementations of the described estimation procedure as well as a Java based graphical user interface can be found at these URLs:

`www.borgelt.net/pycoco.html`   `www.borgelt.net/cocogui.html`

Acknowledgments. The work presented in this paper was partially supported by the Spanish Ministry for Economy and Competitiveness (MINECO Grant TIN2012-31372).

References

1. H. Abdi. Bonferroni and Šidák Corrections for Multiple Comparisons. In: N.J. Salkind, ed. *Encyclopedia of Measurement and Statistics*, 103–107. Sage Publications, Thousand Oaks, CA, USA 2007
2. C.E. Bonferroni. Il calcolo delle assicurazioni su gruppi di teste. *Studi in Onore del Professore Salvatore Ortu Carboni*, 13–60. Bardi, Rome, Italy 1935

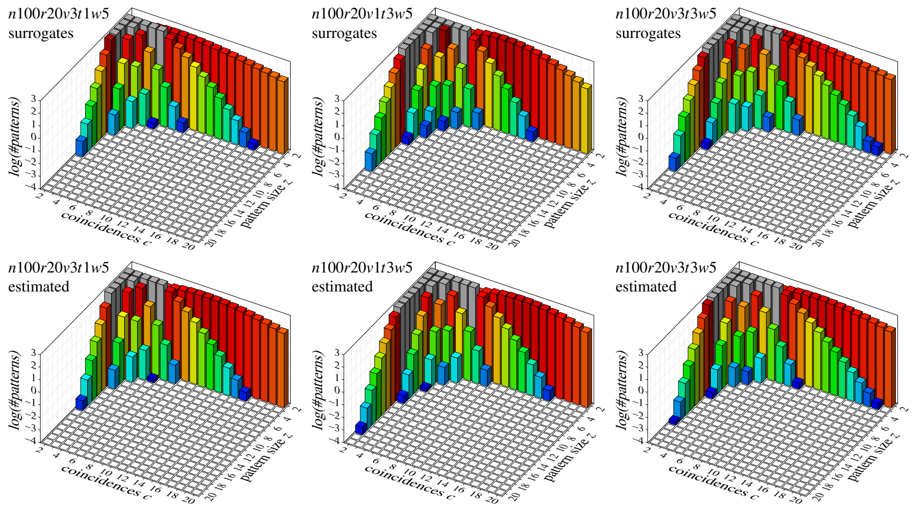

Fig. 3. Pattern spectra for window width 5ms, generated from surrogate data sets (top) or estimated with the described method (bottom). The top word in a diagram title encodes the data set parameters (cf. caption of Figure 2 for details).

3. C. Borgelt. Frequent Item Set Mining. *Wiley Interdisciplinary Reviews (WIREs): Data Mining and Knowledge Discovery* 2:437–456 (doi:10.1002/widm.1074). J. Wiley & Sons, Chichester, United Kingdom 2012
4. C. Borgelt and D. Picado-Muiño. Finding Frequent Synchronous Events in Parallel Point Processes. *Proc. 12th Int. Symposium on Intelligent Data Analysis (IDA 2013, London, UK)*, 116–126. Springer-Verlag, Berlin/Heidelberg, Germany 2013
5. M. Fiedler and C. Borgelt. Subgraph Support in a Single Graph. *Proc. IEEE Int. Workshop on Mining Graphs and Complex Data*, 399–404. IEEE Press, Piscataway, NJ, USA 2007
6. R. Gwadera, M. Atallah, and W. Szpankowski. Markov Models for Identification of Significant Episodes. *Proc. 2005 SIAM Int. Conf. on Data Mining*, 404–414. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA 2005
7. J. Høastad. Clique is Hard to Approximate within n^{1e}. *Acta Mathematica* 182:105–142. Mittag–Leffler Institute, Stockholm, Sweden 1999
8. D. Hebb. *The Organization of Behavior*. J. Wiley & Sons, New York, NY, USA 1949
9. R.M. Karp. Reducibility among Combinatorial Problems. In: R.E. Miller and J.W. Thatcher (eds.) *Complexity of Computer Computations*, 85–103. Plenum Press, New York, NY, USA 1972
10. S. Laxman, P.S. Sastry, and K. Unnikrishnan. Discovering Frequent Episodes and Learning Hidden Markov Models: A Formal Connection. *IEEE Trans. on Knowledge and Data Engineering* 17(11):1505–1517. IEEE Press, Piscataway, NJ, USA 2005
11. S. Louis, C. Borgelt, and S. Grün. Generation and Selection of Surrogate Methods for Correlation Analysis. In: S. Grün and S. Rotter (eds.) *Analysis of Parallel Spike Trains*, 359–382. Springer-Verlag, Berlin, Germany 2010

12. H. Mannila, H. Toivonen, and A. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery* 1(3):259–289. Springer, New York, NY, USA 1997
13. D. Picado-Muiño, C. Borgelt, D. Berger, G.L. Gerstein, and S. Grün. Finding Neural Assemblies with Frequent Item Set Mining. *Frontiers in Neuroinformatics* 7:article 9 (doi:10.3389/fninf.2013.00009). Frontiers Media, Lausanne, Switzerland 2013
14. D. Picado-Muiño and C. Borgelt. Frequent Itemset Mining for Sequential Data: Synchrony in Neuronal Spike Trains. *Intelligent Data Analysis*, to appear. IOS Press, Amsterdam, Netherlands 2014
15. N. Tatti. Significance of Episodes Based on Minimal Windows. *Proc. 9th IEEE Int. Conf. on Data Mining (ICDM'09, Miami, FL, USA)*, 513–522. IEEE Press, Piscataway, NJ, USA 2009
16. E. Torre, D. Picado-Muiño, M. Denker, C. Borgelt, and S. Grün. Statistical Evaluation of Synchronous Spike Patterns Extracted by Frequent Item Set Mining. *Frontiers in Computational Neuroscience*, 7:article 132 (doi:10.3389/fninf.2013.00132). Frontiers Media, Lausanne, Switzerland 2013
17. N. Vanetik, E. Gudes, and S.E. Shimony. Computing Frequent Graph Patterns from Semistructured Data. *Proc. IEEE Int. Conf. on Data Mining*, 458–465. IEEE Press, Piscataway, NJ, USA 2002