

University of Tokyo
Graduate School of Frontier Sciences
Department of Complexity Science and Engineering

博士論文

**Uncovering the functional implications of
spatiotemporal neural activity patterns**
(神経集団の時空間発火パターンの機能的役割の解明)

氏名 渡辺啓太

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Science of the University of Tokyo February 2019

Abstract

The principle of information representation is a fundamental question in neuroscience. Researchers have uncovered that the spatiotemporal activity patterns of neurons play important roles in sensory processing, motor control, and representing internal state of the neuronal population. Moreover, researchers have developed multi-unit extracellular recording schemes, and imaging techniques allow for the recording of hundreds of thousands of neurons to elucidate these codes. However, the current methods are limited in terms of spike pattern detection. A key contribution of this thesis is the development of a framework that facilitates detection of cell-assembly sequences and multi-neuronal temporal activation. The framework utilizes the concept of edit-similarity that defines the similarity between two strings in terms of the number of operations required to transform one text into the other as a distance measure for two different multi neuronal activities recorded at different timings. The effectiveness of the developed method is demonstrated by applying it to synthetic and real recording data obtained from rat hippocampus, prefrontal cortex, and amygdala. The results indicate the practical applicability of the developed method. It could robustly detect spatiotemporal activity patterns, even when the data were noisy. Moreover, we describe possible extensions of the current framework with artificial neural networks and Gaussian process for prediction of animal behavior from cell-assembly sequences through additional optimization of the parameters.

Acknowledgements

First, I would like to express my deep gratitude to my supervisor Prof. Fukai for his useful comments on my thesis and for obliging with the arrangements. Owing to his arrangements, I was afforded ample environments that made it possible to test multiple ideas. Moreover, he created precious opportunities to engage in discussions with many researchers, and these discussions were very enlightening for me. Especially, the advice of Prof. Okamoto was quite helpful for solving the difficulties pertaining to the clustering algorithm. Second, I would like to thank Dr. Haga, who has always supported my research. He willingly shared his precious time to engage in stimulating discussions with me and provided insightful comments and encouragement. Moreover, he suggested that I introduce the edit similarity score for comparison of neuronal population activities. Without this, it would not have been possible to conduct this research. Third, I would like to thank my collaborators Dr. Tatsuno and Dr. David R Euston for their continuous support and advice on the experimental neuroscience side.

I also thank the Buzsáki laboratory for making their data available to the public through the Collaborative Research in Computational Neuroscience website. With their datasets, I could test the proposed framework on real neuronal population activity data.

This research was partially supported by the Brain/MINDS project of AMED (Japan Agency for Medical Research and Development) and RIKEN Junior Research Associate program.

Contents

| | |
|--|------------|
| Abstract | i |
| Acknowledgements | iii |
| 1 Introduction | 1 |
| 1.1 Requirement definition | 3 |
| 1.1.1 Problem of neural coding | 3 |
| 1.1.2 Population coding | 4 |
| 1.1.3 Temporal coding | 4 |
| 1.2 Technical Requirements for temporal coding pattern detection | 5 |
| 1.2.1 Blind detection without any external variables | 5 |
| 1.2.2 Realistic computational complexity | 5 |
| 1.2.3 Robustness against background noise | 6 |
| 1.2.4 Robustness against pattern noise | 6 |
| 1.2.5 Robustness against extension/shrinkage of patterns | 6 |
| 1.3 Policy for the algorithm development | 8 |

| | | |
|----------|--|-----------|
| 1.4 | Outline of this thesis | 9 |
| 1.5 | Related publications and software | 14 |
| 2 | Background Theory | 15 |
| 2.1 | Template matching | 16 |
| 2.1.1 | Limitations | 17 |
| 2.2 | PCA/ICA-based cell assembly detection | 18 |
| 2.2.1 | PCA | 18 |
| 2.2.2 | ICA | 19 |
| 2.2.3 | limitotions | 20 |
| 2.3 | Statistical-significance-based method | 20 |
| 2.3.1 | limitotions | 22 |
| 3 | Edit similarity-based cell assembly sequence detection | 23 |
| 3.1 | Overview of our method | 23 |
| 3.2 | Edit similarity-based cell-assembly sequence detection | 24 |
| 3.2.1 | Edit similarity calculation by N-W algorithm | 25 |
| 3.2.2 | Extended N-W algorithm for neural activity | 28 |
| 3.2.3 | Affine gap penalty | 33 |
| 3.2.4 | Local alignment | 35 |
| 3.2.5 | The proposed scoring schme | 37 |
| 3.2.6 | Metric space and two algorithms for clustering of neural data | 41 |

| | | |
|----------|--|-----------|
| 3.2.7 | Pattern detection by OPTICS | 42 |
| 3.2.8 | Pattern separation by using COPRA | 47 |
| 3.2.9 | Performance evaluation on Density- and community-based algorithms for sequence clustering | 50 |
| 3.2.10 | Dimensionality reduction by t-SNE | 52 |
| 3.2.11 | Tricks for reduction of the computational cost | 53 |
| 3.2.12 | A policy for division of signature matrix | 57 |
| 3.2.13 | Construction of profiles for clustered sequences | 58 |
| 3.2.14 | F_S -score for supervised clustering | 60 |
| 3.2.15 | F_{US} -score for unsupervised clustering | 60 |
| 3.2.16 | Cluster labels for PCA/ICA-based analyses | 61 |
| 3.2.17 | Behavioral labels for clusters | 62 |
| 3.2.18 | Parameter choices | 63 |
| 3.2.19 | Bayesian modeling for edit similarity difference | 63 |
| 3.2.20 | Experimental data | 65 |
| 4 | Application of the edit similarity-based cell assembly detection frame- work | 66 |
| 4.1 | Performance evaluation with artificial data | 66 |
| 4.2 | Place-cell firing sequences in the hippocampus | 70 |
| 4.3 | Cell assemblies in the prefrontal cortex | 76 |
| 4.4 | Anxiety coding cell-assembly sequence in basolateral amygdala . . . | 78 |

| | | |
|----------|--|------------|
| 5 | DISCUSSION | 87 |
| 5.1 | Profile convergence | 87 |
| 5.2 | Comparison with a recent method for sequence detection | 88 |
| 5.3 | Requirements checklist | 91 |
| 5.3.1 | Template matching | 91 |
| 5.3.2 | PCA-/ICA-based method | 91 |
| 5.3.3 | Statistical significance-based method | 91 |
| 5.3.4 | The proposed method | 92 |
| 5.4 | Expansion of the framework | 94 |
| 5.4.1 | Neural-network-based approach for edit-similarity-based feature learning | 94 |
| 5.4.2 | Gaussian Process Regression to find optimal hyper parameters | 97 |
| 6 | Conclusion | 98 |
| 6.1 | Summary of Thesis Achievements | 98 |
| | Bibliography | 100 |
| | Bibliography | 101 |
| A | Statistical Table | 119 |

List of Tables

| | | |
|-----|---|----|
| 1.1 | Technical Requirements Checklist. List of conditions required to detect repetitive spatiotemporal firing patterns from actual multi-neuronal activities. | 7 |
| 3.1 | Example probabilities. The values of the S-shaped function when $b = 5$ and $l = 2$.The probability $1 - (1 - s^l)^b$ that the signatures of two columns are identical in all rows at least one band if the Jaccard similarity for these is s | 56 |
| 5.1 | Requirements checklist for the cell assembly sequence detection. In this thesis, four methods: template matching, PCA-/ICA-based method, statistical significance based method, and the proposed edit similarity based method are considered. | 93 |

List of Figures

| | | |
|-----|---|----|
| 1.1 | Multi-neuronal activity data to be analyzed | 9 |
| 1.2 | Similarity matrix | 11 |
| 1.3 | Illustration of feature space for repeated cell assembly sequence de- tection | 12 |
| 2.1 | Normalized binned spike matrix with bin size 100ms | 18 |
| 3.1 | Edit similarity calculation between strings. We can calculate edit similarity score between “ATCGTAC” and “ATGTTAT” by using a DP table (top). Denoting the characters appearing after diagonal-up moves in the table gives a maximally coincident string (bottom). . . | 26 |
| 3.2 | Edit graph for the equation . Score is changed only when the algorithm takes match. | 27 |

- 3.3 **Detection of repetitive cell-assembly sequences.** (A) Sliding time windows $W(t_k)$ are divided into L bins with an identical size, where t_k refers to the start time of the k -th time window. Population rate vector consists of the spike counts of individual neurons in each bin. (B) A temporal pattern in a sliding time window is schematically illustrated. Such a pattern may contain neurons belonging to a cell assembly as well as non-member neurons. Member neurons may fire at different rates with different temporal precision. Similarity between cell assembly sequences will increase when they share more member neurons and when they fire in a more similar temporal order at similar firing rates with higher temporal precision. Note that each member neuron may appear multiple times at different temporal positions in a time window. (C) Difficulties in detecting repetitive cell-assembly sequences are schematically illustrated. Blue and red bars show spikes of member neurons, while gray bars represent noisy spikes of non-member neurons: (i) temporal structure of target sequence; (ii) contamination by spikes of non-member neurons; (iii) missing spikes of member neurons; (iv) jitters in spike timing of member neurons; (v) arbitrary scaling of sequence length; (vi) member overlaps between different sequences. 29
- 3.4 **Example of extended N-W algorithm. (Left):** DP table of two examples of neuronal activities. **(Right):** Example neuronal activities. The row denotes neuron number, and the column denotes time bin. Coincident activations are represented in red. 30
- 3.5 Needleman and Wunsch algorithm 31
- 3.6 algorithm for making alignments 32

| | | |
|------|--|----|
| 3.7 | First shortcoming of previous scoring scheme. The previous scoring system cannot discriminate patterns with different magnitudes of jitters. | 33 |
| 3.8 | Edit graph for the equation . The left and right tables record the optimal numbers of gap insertions in each cell. | 34 |
| 3.9 | Limitation of the scoring scheme with gap penalty. The rightmost segment has a smaller edit similarity score relative to others in the previous scoring scheme. | 35 |
| 3.10 | Edit graph for the equation . The left and the right tables record the optimal number of gap insertions in each cell. In addition, no-cost jump to an arbitrary location is introduced. | 36 |
| 3.11 | algorithm for making alignments | 39 |
| 3.12 | An algorithm for edit similarity calculation | 40 |
| 3.13 | Illustration of OPTICS algorithm. Each circle represents different data points, and cardinality is shown in black. | 44 |
| 3.14 | Algorithm OPTICS | 45 |
| 3.15 | Example of automatic cluster detection by OPTICS. Three clusters in the dataset detected. | 45 |
| 3.16 | OPTICS fails to distinguish clusters OPTICS fails to differentiate clusters when they share a few data points. | 46 |
| 3.17 | COPRA algorithm example Three steps of the algorithm. | 49 |
| 3.18 | OPTICS can extract densely connected clusters | 49 |
| 3.19 | Combination of OPTICS and COPRA can extract densely connected clusters containing noise | 49 |

- 3.20 **Comparison between different clustering algorithms.** A density-based clustering algorithm (OPTICS) and a community detection algorithm (CORPA) were separately or sequentially applied to a dataset. Data points were generated by a mixture of two Gaussian distributions with different centers and the same variance of 1.3. These points were further mixed with uniformly distributed background data points. (A) OPTICS could remove background noise but failed to discriminate the two clusters. (B) COPRA could separate these clusters but failed to remove background noise. (C) The combined application of OPTICS and COPRA successfully separated the two clusters and removed background noise. (D) Performance of identifying two clusters was compared between the different algorithms, that is, OPTICS only (magenta), COPRA only (yellow) and their combination (cyan). The abscissa represents the distance between the centers of the two clusters. Shaded areas show standard errors. 51
- 3.21 **Min-hashing algorithm by example.** (Left) Three different permutations of rows used to calculate Minhash values (colored in blue, yellow, and red). (Right) Resultant values are stored as the Signature Matrix. 54
- 3.22 **Minhashing algorithm application towards neuronal data.** Similar activity vector tends to be put in the same bucket. The probability is determined by the Jaccard similarity. 55
- 3.23 **The SC curve with $b = 5$ and $l = 2$.** X-axis shows Jaccard similarity values from 0 to 1. Y-axis represents the corresponding probability $1 - (1 - s^l)^b$ that the signatures of two columns are identical in all rows at least one band if the Jaccard similarity for these is s 57

- 4.1 **Comparison between different methods.** (A) An example of the embedded artificial cell assemblies used for the comparison. In the raster plot, each dot is a spike. Each color indicates a cell assembly. For clarity, noisy spikes are not shown. (B) Timing jitters were within ± 10 ms and cell assemblies represented synchronously firing neuron ensembles. F_{US} -score was significantly higher for the proposed method than for PCA- and ICA-based algorithms (p values were less than $2.2e-16$ for both cases: Wilcoxon rank sum test). The time window used was 200ms. (C) Timing jitters were within ± 50 ms and cell assemblies may not be regarded as synchronously firing neuron ensembles. Our method is sensitive to the serial order of firing and hence the score is lowered (p values were less than $2.9e-11$ for both cases: Wilcoxon rank sum test). (D) Nine artificial spike sequences (middle and bottom) were embedded into noisy spike trains. Noise spikes are not shown here. (E) Sequences detected by our method from the artificial data shown in D are presented in two intervals together with noisy spikes (gray). The nine embedded sequences were successfully detected. The results are shown for different values of parameters: $\alpha = 2.0$, MinPts = 20, $v = 2$ in left panels; $\alpha = 0.5$, MinPts = 10, $v = 12$ in right panels. (F) Robustness of cell-assembly detection against background noise. Two scores, Specificity (left) and F_S (right), in the detection of a single cell-assembly are shown against the number of background Poisson spike trains: our method (orange), PCA (green), and ICA (blue). Shaded areas indicate standard errors. (G) Effect on different α for cluster formation. Different shrinkage rates tested: 10 (orange), 5 (green), and 3 (blue). Shaded areas indicate standard errors. 68

- 4.2 Cell assemblies extracted from hippocampal CA1.** (A) The spatial locations in the linear maze are shown for the detected cell-assembly sequences. The x-axis shows time and y-axis shows the position of the rat. Twenty Go clusters and 20 Back clusters of time windows are shown, and the spatial positions at which they were detected were indicated by the rightmost color diagram. Each colored region shows where each segment was observed. Cluster four and ten were additionally colored in blue and pink. Cluster indices are shown on the right, which were sorted and colored according to the order of appearance during the going and returning along the maze. We sorted cluster indices according to their mean kernel density estimate (which is shown in Figure 4.4). The window size was 100ms. (B) t-SNE visualization of the feature space. Note that most of the neighboring colors in A are also adjacent in B, suggesting that two neuronal activities observed at contiguous spatial positions have similar temporal patterns, but are still separate enough in the feature space. (C) Profiles of cell-assembly sequences are shown for four cluster (left four panels). The first 10 neurons for four profiles are shown with number indicating cluster identity.(D) Between-cluster and within-cluster edit similarity scores. Edit similarity was compared between time windows belonging to the same cluster and those belonging to different clusters. Box plot with whiskers from 5 percentile to 95 percentile are shown with outliers (filled circles). . . 73

- 4.3 Cell assemblies detected from CA1.** Four examples are shown from cluster 4 (top) and cluster 10 (bottom). The top, middle and bottom panels display the velocity and spatial position of the rat, spike raster, and local field potentials band-passed at 4-10Hz (theta band) and 150-200Hz (sharp-wave ripples). We calculated criteria (0.095) of the ripple detection with the method described in [28] to confirm our detected patterns during immobility accompanied with sharp wave ripple. In the middle panel, gray vertical bars show noisy spikes and red bars represent the core spikes of the corresponding profile. Neurons are sorted according to their firing position within the average profile. 74
- 4.4 The relationships between hippocampal cell-assembly sequences.** (A) The spatial receptive fields are shown for 36 hippocampal neurons (top) and cell assemblies belonging to the 20 clusters (bottom). The pseudo-color code indicates the probability of firing. (B) Estimated edit similarity difference between the original and shuffled data. Each similarity scores were calculated by using the profiles of cell assemblies and spike trains of hippocampal neural population using a sliding time window. Solid lines correspond to the mean of the estimated differences and blue shaded regions to 95% credible interval. Orange bars designate the spatial locations at which the lower bound of the credible interval is positive. 75

- 4.5 Cell assemblies detected from the prefrontal cortex.** The width of time windows was 250ms. (A) The onset times of detected time windows are shown for all the clusters. (B) The spatial positions and movement directions of a rat are shown at the onset times of detected time windows belonging to three clusters by arrows. Only ten percent of randomly sampled elements are drawn for visualization. (C) Profiles are shown for three prefrontal cell assemblies in terms of the sorted neuron id and relative temporal order. The approximate length of the x-axis coincides the width of temporal windows (250 ms). (D) Cell assembly sequences detected in awake (left three panels) and sleep (rightmost panel) are shown for the three clusters. From top to bottom, each row corresponds to the profile 2, 8 and 9, respectively. Some sleep replay events showed evidence of multiple replays within the 250 ms window. This is most apparent in the first row, where the upward ramp is seen twice. . . . 77
- 4.6 Profile and template of an illustrative session** **4.7 Top:** The template and the profile are shown with three examples of 500-ms neuronal activity after the onset of air puff. 80
- 4.7 Relationship between correlation of neuronal activity versus patterns (profile/template) and acceleration of rat 1.** Top half shows a comparison between profile and neuronal activity/movement of the rats, and bottom half shows the same comparison with the template. 81
- 4.8 Profile and the template of an illustrative session** **4.9 Top:** The template and the profile are shown with three examples of 500-ms neuronal activity after the onset of air puff. 82

- 4.9 Relationship between correlation of neuronal activity versus patterns (profile/template) and acceleration of the rat 2.** The top half shows a comparison between profile and neuronal activity/movement of the rats, and the bottom half shows the same comparison with the template. 83
- 4.10 Profile and template of an illustrative session 4.11 Top:** The template and the profile are shown with three examples of 500-ms neuronal activity after the onset of air puff. 84
- 4.11 Relationship between correlation of neuronal activity versus patterns (profile/template) and acceleration of the rat 3.** The top half shows a comparison between profile and neuronal activity/movement of the rats, and the bottom half shows the same comparison with the template. 85
- 4.12 Pearson's r for all sessions** Left: Pearson's correlation coefficient between the animal's acceleration and the similarity score calculated using the profile. Right: Comparison with the template did not show any strong correlation. 86
- 5.1 Convergence of profile evaluation in CA1 and HPC data analysis.** Profiles were calculated for all clusters in the activity data from CA1 and PFC. The abscissa shows the number of iterations, and the ordinate shows edit similarity between the current profile and the preceding one. The thick lines represent the means over all clusters, and the shaded areas represent the standard errors. 90

| | | |
|-----|--|----|
| 5.2 | Comparison with Russo 2017 (A) F_S -score (top) was significantly higher in case of the proposed method than in case of [112] for the artificial data of length 500 ms (right, p-value is 1.831e-12); however, difference between the two methods was not significant for the artificial data of length 100 ms (left, p-value is 0.6386). (B) The output of Russo 2017 is shown. The distributed computer code of Russo 2017 analyzes spike train data with various temporal precisions (i.e., correlation time scales) ranging from milliseconds to several seconds. Only a part of the results is shown to clarify the characteristic property of Russo 2017: the eight cell assemblies classified as number 10 to number 17 look very similar to each other. | 90 |
| 5.3 | Network architecture. | 95 |
| 5.4 | Training loss during learning. Example training iterations are shown. | 96 |
| 5.5 | Performance of learned network prediction compared with PCA-based approach. | 96 |

Chapter 1

Introduction

The uncovering of neural codes is a fundamental task in neuroscience. The results of several experimental studies suggest that synchronous or sequential firing of cortical neurons plays active roles in primates [1, 46, 124]. In the somatosensory and auditory cortices of rats, spontaneous and stimulus-evoked activities exhibit repetitive neuronal firing sequences [70, 71]. In the rodent hippocampus, place cells exhibit precisely timed, repetitive firing sequences, representing the rodent's trajectory, and subsections of these sequences are repeated during each theta cycle [83, 95, 137]. These sequences are replayed over compressed temporal scales during the awake immobile and the sleep states [16, 20, 35, 65], presumably for memory consolidation [38, 54]. Similar replay events have been observed in the rodent prefrontal cortex as well [33].

The rapid development of techniques for large-scale recording of neuronal activity provides a fertile ground for spike sequence analysis. Calcium imaging facilitates simultaneous measurement of the spike rates of hundreds to thousands of neurons [29, 58, 106, 113, 138], and imaging based on voltage indicators may help overcome the poor temporal resolution of imaging [32, 42, 59, 60]. In addition, extracellular recording of neural activity with multiple electrodes has evolved, thus

allowing access to spike trains from large numbers of neurons [15, 17, 31, 56, 116].

Despite this progress in experimental techniques, methods for analyzing the spatiotemporal structures of cell assemblies remain limited [21]. Template matching is a standard technique for detecting repeated activity patterns [1, 33, 58, 70, 71, 110, 114, 130, 138]. However, the method requires reference events, such as sensory cues and motor responses and is easily disrupted by biological noise, such as jitters in spike timing and variations in sequence length. Very few researchers have attempted blind detection of cell-assembly sequences without relying on reference events [74, 101, 104, 107, 112, 119, 132], and such data analysis remains challenging.

Herein, we develop a method to detect self-similar firing patterns within cell assemblies by using the edit similarity score. The edit similarity metric was originally introduced in computer science to measure the distance between arbitrary strings, and it has been utilized to analyze various types of sequences in computer science and biology [91]. Edit similarity matches two sequences with flexible temporal alignment, an essential feature for detecting noisy spatiotemporal patterns embedded in neural activity. We extend the edit similarity score to a form applicable to neural activity data and develop a clustering method for blind cell-assembly detection. The performance of this method is evaluated using artificial data. The evaluation results show that the proposed method is more robust against background noise than conventional clustering methods. Furthermore, upon application of the proposed method to experimental data recorded from the rat hippocampus [86] and prefrontal cortex [33], it could detect several multi-cell sequences linked with behavior in an unsupervised manner. Based on these findings, the proposed method, which is robust against noise and computationally efficient, will help with exhaustive search of repetitive spatiotemporal patterns in large-scale neural data, which may lead to the elucidation of hidden neural codes.

1.1 Requirement definition

The nervous system adopts several different representation schemes to encode various types of sensory stimuli in an appropriate manner. This section starts with a brief explanation of the major coding schemes. In addition, this section provides a detailed description of the problem, including the types of patterns that must be detected under a given set of conditions. The algorithm development policy is discussed in the last part of this section.

1.1.1 Problem of neural coding

Neurons have the unique ability of propagating signals over long distances by means of characteristic short voltage pulses called spikes or action potentials. Despite the differences in their duration, amplitude, and shape, they are usually treated as identical events, which means that the associated activities can be treated as binary point processes in time. Researchers have invested considerable effort toward uncovering the communication protocol of all-or-none events among the neural population, in other words, neural coding. Studies have revealed several different forms of information representation in different modalities. In this section, we discuss three hypothesized coding schemes: rate coding, population coding, and temporal coding.

Rate coding

In the rate coding model, information pertaining to a certain stimulus is represented solely by the frequency of action potentials, that is, the firing rate. This scheme generally treats trial-by-trial variability of spike trains as noise. As a result, it is considered inefficient, but robust against such variability. In most sensory systems, the firing rate and the stimulus intensity are correlated. For instance,

sensory neurons connected to a muscle exhibit an action potential whose rate is mediated by the degree of stretching of the muscle [57].

1.1.2 Population coding

When a neuronal population collectively represents information as a combination of the firing rates of each of the neurons, the resulting neural code is a population code. In the monkey's medial temporal (MT) cortex, neurons are tuned to the direction of moving objects [82]. A similar representation is found in monkeys trained to move a joystick toward a target [36]. Moreover, in the inferior monkey's temporal (IT) cortex[81], neurons respond selectively to complex visual stimuli, such as faces and objects.

1.1.3 Temporal coding

When precise latencies of spikes or high-frequency firing rate fluctuations in different neurons play an active role, the resulting neuronal code is called a temporal code. Several experimental results suggest that synchronous or sequential firing of cortical neurons plays active roles in primates [1, 46, 124]. In the rat somatosensory and auditory cortices, spontaneous and stimulus-evoked activities exhibit repetitive sequences of neuronal firing [70, 71]. In the rodent hippocampus, place cells exhibit precisely timed, repetitive firing sequences that represent the rodent's trajectory, and subsections of these sequences are repeated during each theta cycle [83, 95, 137]. These sequences are replayed over compressed temporal scales during awake immobile and sleep states [16, 20, 35, 65], presumably for memory consolidation [38, 54]. Similar replay events have been observed in the rodent prefrontal cortex as well [33]. Moreover, the hippocampus encodes information pertaining to distance [137], speed [146], and choice[131].

This thesis focuses on the third coding scheme, namely, temporal coding. The

following section discusses potential difficulties associated with the detection of spatiotemporal spike patterns.

1.2 Technical Requirements for temporal coding pattern detection

The focus of this thesis is a novel methodology for unsupervised temporal pattern detection that facilitates investigation of temporal coding. Several such techniques have been proposed already. This elicits the question: why do we need a new one? To answer this question, in this section, we introduce a technical requirements checklist that an algorithm must satisfy. Throughout this thesis, three major methodologies (template matching, PCA-/ICA-based method, and method that relies on statistical significance) will be introduced. In the discussion, we present a comprehensive summary, including the various requirements fulfilled by the individual methods.

1.2.1 Blind detection without any external variables

Spontaneous firing sequences are ubiquitous in cortical networks [53, 72]. Therefore, the method must be applicable without any external variables, such as sensory cues and behavioral data.

1.2.2 Realistic computational complexity

Recent improvements in experimental equipment have allowed for the recording of thousands of neurons simultaneously [56], and this trend is expected to continue [125]. The methodology must be able to handle such huge datasets within a reasonable computational duration.

1.2.3 Robustness against background noise

Because the algorithm must detect spatiotemporal patterns from such high-dimensional data, the method must be robust against background noise, which comprises action potentials that are not part of any pattern .

1.2.4 Robustness against pattern noise

Moreover, the noisy nature of the nervous system, such as synaptic transmission failure[2, 121] and stochastic ion channel gating [3, 18], makes it difficult to identify individual patterns. Therefore, the method must be able to identify incomplete patterns under such circumstances.

1.2.5 Robustness against extension/shrinkage of patterns

Memory patterns activated when an animal is in an active state are replayed when it is in an immobile or sleeping state. These patterns often reverberate over compressed temporal scales during awake immobile and sleep states [16, 20, 33, 35, 65]. The method must be flexible toward such scale differences.

Table 1.1 lists the above criteria. Throughout this thesis, we will check whether the proposed method satisfies these criteria. Simultaneously, we will test several widely used methodologies, for instance, template matching, the PCA-/ICA-based method, and the statistical-significance-based method.

| |
|--|
| Criteria |
| Without behavioral data |
| Realistic computational complexity |
| Robust against background noise |
| Robust against pattern noise |
| Robust against extension/shrinkage of patterns |
| Shared neurons among different patterns |

Table 1.1: Technical Requirements Checklist. List of conditions required to detect repetitive spatiotemporal firing patterns from actual multi-neuronal activities.

1.3 Policy for the algorithm development

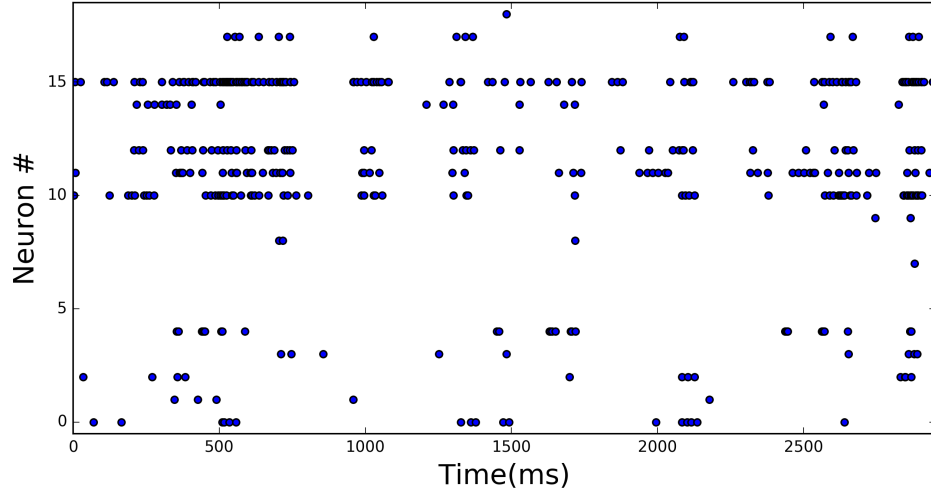
These methods usually require that exact matches of the same pattern for the detection, which is unlikely to be realized as the number of neurons increases and as the time structure considered gets finer [14, 45, 74]. In addition to the difficulty these methods do not scale in terms of the number of neurons as the number of statistical tests to be employed increases dramatically.

The PCA-/ICA-based method, and the template matching method, falls into the later category. The PCA-/ICA-based method in its original form, cannot account for the temporal structure in each segments. Hence, it is not robust against the background noise. The template matching method could capture temporal structure of the pattern, but it is not applicable when no external cues are given.

From these matters under consideration, we decided to derive a novel similarity/distance measurement that could capture the temporal variabilities. We noticed that there is a rich accumulation of information in the field of bioinformatics. As a result of the investigation in the field, we picked up the edit-distance [39, 92, 122]-based similarity measurement and the suffix tree/array [37, 49, 75, 77, 120]-based method as candidates. Although the suffix tree/array-based method is relatively faster in computation, it is less straight forward to make it applicable in neuronal data as it is not straightforward to consider pattern jitters. Hence, we decided to employ the edit similarity based method for the cell assembly sequence detection.

1.4 Outline of this thesis

A



B

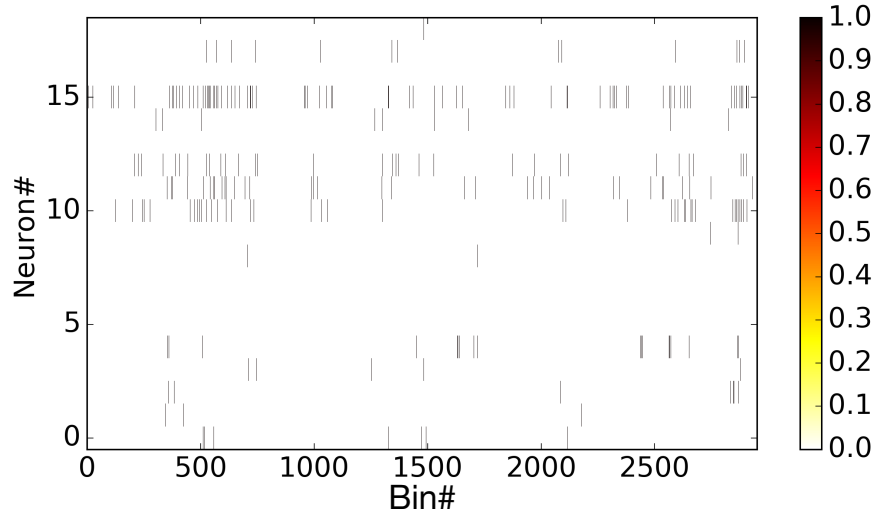


Figure 1.1: Multi-neuronal activity data to be analyzed. (A) Original multi-neuronal spiking activity shown as a raster plot. The X-axis corresponds to the timing of spikes, and the Y-axis denotes ID of the activation. (B) Binned spike activity. A small bin width (1 ms or 10 ms) is used to discretize the data (A). Each column vector represents activation of the neuronal population in each time bin.

From a technical perspective, the main contribution of this thesis is a novel methodology that enables us to quantify and detect spatiotemporal activations that appear repeatedly in multi-neuronal activity data, as in Figure1.1(B). A brief description of the methodology follows. The framework first divides 1.1(B) into

fixed length time segments as a preprocessing step. Suppose we have an appropriate metric that correctly distinguishes segments containing different types of sequences. In the metric, the distance between two segments with similar ordering of neuronal population activities should be small and otherwise; the distance should be significant. With the metric, one can calculate a similarity matrix that represents distances of each possible pair of segments, as in Figure 1.2. Although the corresponding feature space is high-dimensional and difficult to illustrate, one may obtain a two-dimensional projection, as in Figure 1.3, by using the dimensionality reduction technique. In the feature space, repetitions of spatiotemporal activities create a densely localized cluster, such as the cyan, green, and red clusters in Figure 1.3.

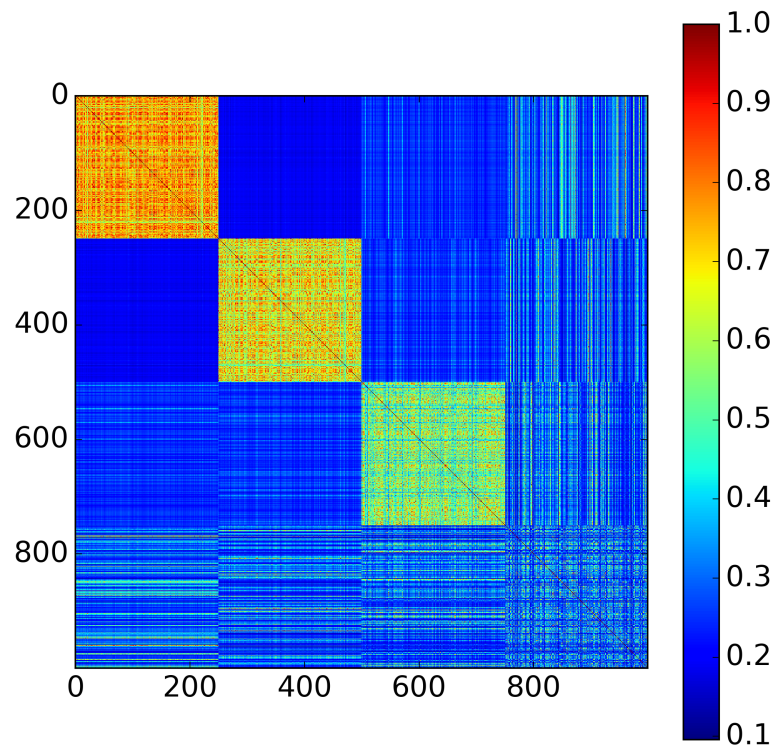


Figure 1.2: Similarity matrix. The matrix is a graphical representation of similar segments in multi-neuronal activity data divided by a fixed length of sliding time window (called a segment).

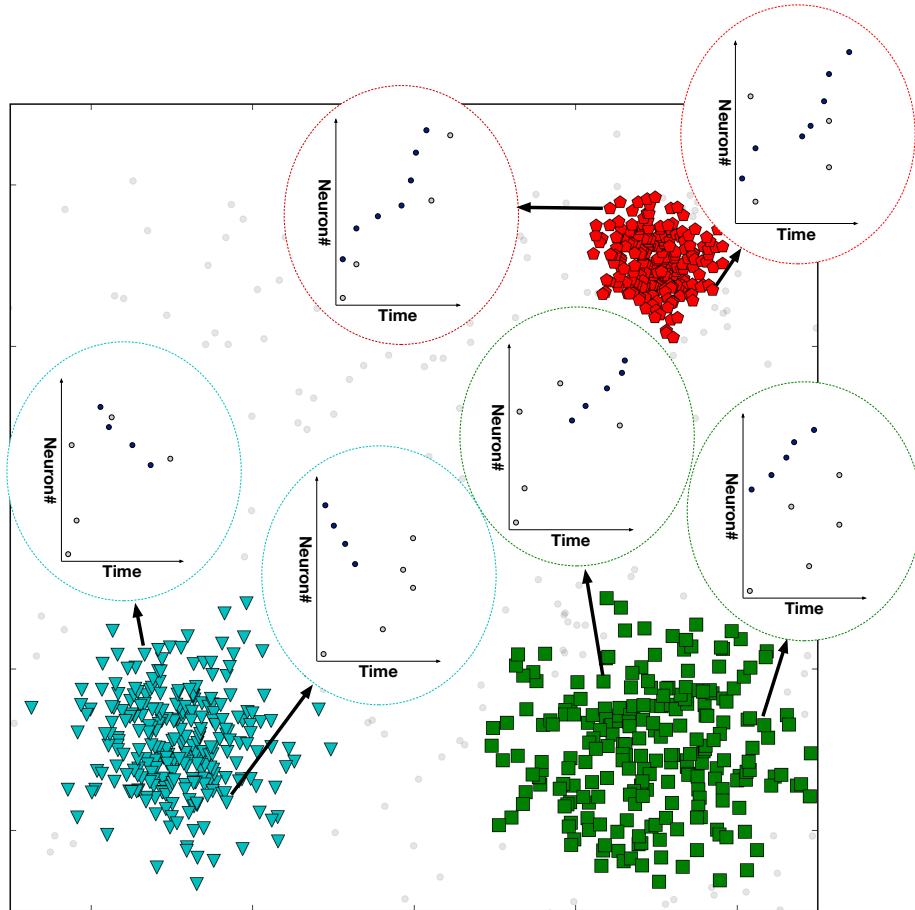


Figure 1.3: Illustration of feature space for repeated cell assembly sequence detection. Clustering result of the synthetic neuronal activity data is shown in cyan, green, and red. Three different types of cell assembly sequences are embedded in the data.

Moreover, in this thesis, we demonstrate spatiotemporal activity patterns detected in the prefrontal cortex, hippocampus, and amygdala. The analysis reveals novel types of cell assembly sequences that have not been observed in previous studies and could not be detected using existing methods.

The remainder of the thesis is structured as follows:

- Chapter 2 reviews the major methodologies employed for cell assembly sequence detection. In this chapter, we explain the underlying algorithms and introduce studies that utilized the aforementioned methodologies and showed the relationship between spatiotemporal activity patterns and mental processes of animals; the focus is on three major approaches: PCA-/ICA-based method ([63, 102, 104]), template matching ([33, 58]), and methods relying on statistical significance tests ([104, 112, 132].
- Chapter 3 introduces the proposed framework. This chapter is concerned with four factors: novel similarity metric between two given multi-neuronal time series, two clustering algorithms that are applied sequentially to detect repeated activation of the same types of cell-assembly sequences, and framework for extracting common timing structure among cluster members.
- Chapter 4 discusses performance evaluation under various conditions with comparison among existing frameworks and three examples in which the proposed framework is applied to datasets of real multi-neuronal activity in rat prefrontal cortex[33], hippocampus CA1 region[86], and amygdala [38] during the behavioral period. The first case employs the data used to show memory traces in PFC with template matching and shows the scalability of the proposed method. In addition, the chapter reports unobserved patterns in the amygdala data that cannot be detected with the template matching.
- Chapter 5 discusses a few characteristics of the proposed framework and discusses possible extensions. Moreover, it presents a comparison of the

proposed approach with that of [112].

- Chapter 6 discusses the key contributions of the thesis.

1.5 Related publications and software

Part of Chapter 3 and Chapter 4 borrows text from a manuscript authored by Watanabe et al[142]. The amygdala data analysis part is based on a publication that is in preparation at the time of writing this thesis. Finally, part of the text in chapter 5 is also retrieved from a manuscript in preparation.

Whole data analysis was done by Python 3.6, Julia 0.6, and Bash shell script. The original implementation of the framework is available at (<https://github.com/KeitaW/Chaldea>). The program is coded with Julia v0.6, Python 3.6, Bash 3.2.57, and javac 1.8.0_65. It offers a set of command line tools implemented in the click library (<https://click.palletsprojects.com/en/7>). Minhashing algorithm is also available as a separate library at (<https://github.com/KeitaW/MinHashing>).

Implementation of the core algorithm in Python 3.6 is available at (<https://github.com/KeitaW/spikesim>, RRID: SCR_016351), which is provided as a library. We used the original implementation of the COPRA [41] publicized by the author [150].

<https://github.com/espg/OPTICS/blob/master/OPTICS.py> was used for the OPTICS. We also used gnu parallel [128] for data processing.

Chapter 2

Background Theory

This chapter reviews the major methodologies employed for cell assembly sequence detection. In this chapter, we explain the underlying algorithms and introduce studies that utilized the aforementioned methodologies and showed the relationship between spatiotemporal activity patterns and mental processes of animals; the focus is on three major approaches: PCA-/ICA-based method ([63, 102, 104]), template matching ([33, 58]), and methods relying on statistical significance tests ([104, 112, 132]).

The capability to detect temporal sequences embedded in a complex sensory stream is required by the neural circuit of the brain [7, 51, 55] and artificial intelligence systems [23, 26, 126] to generate appropriate responses. Hence, detection and characterization of such temporal structures is critical for both biology and machine learning to derive better models that explain and predict information representation in the systems.

In machine learning, researchers have developed various models for analyzing temporal structures, for instance, driving data [129, 133, 143], natural language processing [115, 139, 140], and human motion modeling [43, 141, 145]. Neuroscience researchers are facing the same problem with advanced record-

ing techniques that facilitate investigation of large neuronal population activity data [15, 110]. They have employed various techniques such as the PCA/ICA-based method, template matching [33, 58], statistical-significance-based method [104, 105, 112, 132], state space modeling [119], dynamical system modeling [127], and convolutional non-negative matrix factorization [74, 101].

This chapter reviews a few of these techniques and points out their limitations.

2.1 Template matching

Let x be a $N \times M$ template that is generally produced from certain duration of neuronal population activities after the onset of a specific event, and y be a $N \times M$ partial spike matrix extracted from the original neuronal spike matrix s as follows:

$$y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1M} \\ y_{21} & y_{22} & \cdots & y_{2M} \\ \vdots & \vdots & \cdots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NM} \end{bmatrix} = \begin{bmatrix} s_{1t} & s_{1(t+1)} & \cdots & s_{1(M+t)} \\ s_{2t} & s_{2(t+1)} & \cdots & s_{2(M+t)} \\ \vdots & \vdots & \cdots & \vdots \\ s_{Nt} & s_{N(t+1)} & \cdots & s_{N(M+t)} \end{bmatrix} \quad (2.1)$$

where t is a specific timing to be compared with the template x . In general, the method calculates the similarity of these two matrices. If the similarity is high, y contains similar population activities that are present in x . The method seeks such t . One major similarity score is the standardized Pearson correlation coefficient,

also called Pearson's r. Let $\bar{x}_c, \bar{y}_c, \sigma_{x,c}, \sigma_{y,c}, w_{cm}, z_{cm}$ be

$$\bar{x}_c = \frac{1}{m} \sum_{m=1}^m x_{cm}, \quad (2.2)$$

$$\bar{y}_c = \frac{1}{m} \sum_{m=1}^m y_{cm}, \quad (2.3)$$

$$\sigma_{x,c} = \sqrt{\frac{1}{m} \sum_{m=1}^m (x_{cm} - \bar{x}_c)^2}, \quad (2.4)$$

$$\sigma_{y,c} = \sqrt{\frac{1}{m} \sum_{m=1}^m (y_{cm} - \bar{y}_c)^2}, \quad (2.5)$$

$$w_{cm} = \frac{x_{cm} - \bar{x}_c}{\sigma_{x,c}}, \quad (2.6)$$

$$z_{cm} = \frac{y_{cm} - \bar{y}_c}{\sigma_{y,c}}. \quad (2.7)$$

With these variables, the standardized Pearson correlation coefficient (COR) can be expressed as follows:

$$\text{COR} = \frac{\sum_{n=1}^n \sum_{m=1}^m (w_{nm} - \bar{w})(z_{nm} - \bar{z})}{\sqrt{\sum_{c=1}^n \sum_{m=1}^m (w_{cm} - \bar{w})^2} \sqrt{\sum_{c=1}^n \sum_{m=1}^m (z_{cm} - \bar{z})^2}}, \quad (2.8)$$

where \bar{w}, \bar{z} are

$$\bar{w} = \frac{1}{nm} \sum_{n=1}^n \sum_{m=1}^m w_{nm}, \quad (2.9)$$

$$\bar{z} = \frac{1}{nm} \sum_{n=1}^n \sum_{m=1}^m z_{nm}. \quad (2.10)$$

Significantly high values of COR imply the existence of similar spatiotemporal activity patterns at the timings in question.

2.1.1 Limitations

By definition, it requires pre-determined time points for template evaluation. The limitation can be problematic for the detection of spatiotemporal activity pattern from spontaneous activities. Also, as discussed in the amygdala data analysis

section 4.4, this method is not robust against the existence of noise.

2.2 PCA/ICA-based cell assembly detection

2.2.1 PCA

PCA and ICA employ normalized binned spike matrices $Z = z_{i,j}$ with large seconds bin sizes, such as the one shown in Figure 2.1.

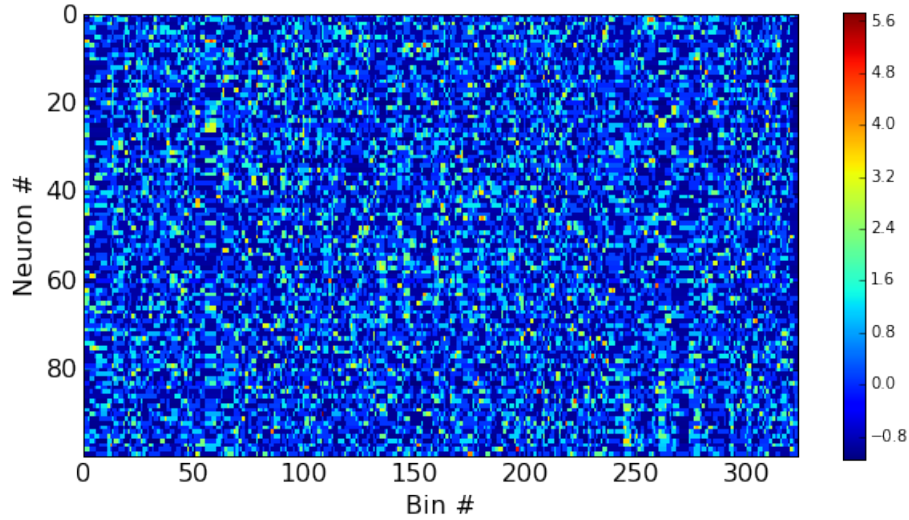


Figure 2.1: Normalized binned spike matrix with bin size 100ms

Each element of the matrix $z_{i,j}$ is defined as follows:

$$z_{ij} = \frac{s_{ij} - \bar{s}_i}{\sigma_{s_i}}. \quad (2.11)$$

From Z , the covariance matrix of C is defined:

$$C = \frac{ZZ^T}{T}. \quad (2.12)$$

Note that T is the transpose operator of a matrix. The principal components \mathbf{c}_i of

matrix Z can be acquired from the decomposition of C :

$$C = \sum \lambda_i \mathbf{c}_i \mathbf{c}_i^T, \quad (2.13)$$

where λ_i is the eigenvalue corresponding to the eigenvector \mathbf{c}_i . To extract cell assemblies from the PCA results, one must determine the number of cell assemblies present in the processed data. Peyrache et al. [102] utilized the Marcenko–Pastur distribution [78] as the null hypothesis for the existence of cell assemblies. In the null hypothesis, the eigenvalues of the correlation matrix of a normal random matrix M follow the following probability distribution:

$$p(\lambda) = \frac{q}{2\pi} \frac{\sqrt{(\lambda_{\max} - \lambda)(\lambda - \lambda_{\min})}}{\lambda}, \quad (2.14)$$

where

$$\lambda_{\max} = \sigma^2(1 + \sqrt{1/q})^2, \quad (2.15)$$

$$\lambda_{\min} = \sigma^2(1 - \sqrt{1/q})^2, \quad (2.16)$$

$$q = \frac{T}{N}, \quad (2.17)$$

and T and N denote the total number of time windows and the total number of neurons, respectively. With statistically independent rows and $q \geq 1$, if all rows are statistically independent, the probability that any eigenvalue λ of m is in the range of the values $\lambda_{\min} \leq \lambda \leq \lambda_{\max}$ is 1. Thus, the number of eigenvalues that exceeds λ_{\max} can be considered the number of cell assemblies in S .

2.2.2 ICA

Use of ICA to quantify interactions among neurons, as proposed by Lanubach et al. [63]. The method statistically extracts independent non-Gaussian components from neuronal population activity data by estimating the values of r and w , where

$r = w^t z$. In this thesis, the fast ICA algorithm is used to approximate the following negentropy:

$$J_{appr}(y) = \sum_i [E(g_i(y)) - E[g_i(v)]]^2, \quad (2.18)$$

where

$$g_1(u) = \frac{1}{a} \log \cosh(au), \quad (2.19)$$

$$g_2(u) = -\exp\left(-\frac{u^2}{2}\right). \quad (2.20)$$

To execute ICA, we must estimate the number of cell assemblies before estimating r and w . Again, the Marcenko–Pastur distribution for the covariance matrix can be used for this purpose.

2.2.3 limitotions

First, since both PCA- and ICA-based method relies on the activity vector that averages activations in the time window for each neuron, these methods do not include temporal order of activations in consideration. Second, the PCA-based method cannot[69] discriminate two cell assemblies if they share neurons. Third, the ICA-based method detects single-neuron activation as a pattern if the distribution of the neuronal activation follows a non-Gaussian distribution.

2.3 Statistical-significance-based method

Multi-variate significance testing techniques are also employed to detect cell-assembly sequences [104, 105, 112, 132]. In this section, we introduce one such technique [112] for illustration.

Assume that we have multi-neuronal spike activity data consisting of N neurons divided into T bins of equal bin width Δ . Let $\{c_{K,t}, \quad t = 1 \dots T\}$ be spike counts

of recorded neuron K . For two neurons A and B , the total joint count $\#_{AB,l}$ at the selected lag \bar{l} is given by the sum of the joint counts $\#_{AB,\bar{l}}$. If the stochastic processes are stationary, the mean and variance under the null hypothesis H_0 can be determined as follows:

$$\mu_{AB,\bar{l}} \equiv E[\#_{AB,\bar{l}}] = \sum_{\alpha=1}^M \frac{\#_A^\alpha \#_B^\alpha}{T - \bar{l}}, \quad (2.21)$$

$$\sigma_{AB,\bar{l}}^2 \equiv E\left[\left(\#_{AB,\bar{l}} - E[\#_{AB,\bar{l}}]\right)^2\right] = \sum_{\alpha=1}^M \text{var}\left(\#_{AB,\bar{l}}^\alpha\right) + 2 \sum_{\alpha=1}^{M-1} \sum_{\gamma>\alpha}^M \text{cov}\left(\#_{AB,\bar{l}}^\alpha, \#_{AB,\bar{l}}^\gamma\right) \quad (2.22)$$

$$= \sum_{\alpha=1}^M \frac{\#_A^\alpha \#_B^\alpha}{\tilde{T}} \frac{(\tilde{T} - \#_A^\alpha)}{\tilde{T}(\tilde{T} - 1)} + 2 \sum_{\alpha=1}^{M-1} \sum_{\gamma>\alpha}^M \frac{\#_A^\gamma \#_B^\gamma}{\tilde{T}} \frac{(\tilde{T} - \#_A^\alpha)}{\tilde{T}(\tilde{T} - 1)} \text{ with } \tilde{T} = T - \bar{l}. \quad (2.23)$$

In the non-stationary case, the following values can be used as H_0 :

$$\mu_{ABBA,\bar{l}} \equiv E[\#_{AB,\bar{l}}] - E[\#_{AB,-\bar{l}}] = 0, \quad (2.24)$$

$$\sigma_{ABBA,\bar{l}}^2 \equiv E\left[\left(\#_{ABBA,\bar{l}} - E[\#_{ABBA,\bar{l}}]\right)^2\right] = 2\sigma_{AB,\bar{l}}^2 - 2\text{cov}\left(\#_{AB,\bar{l}}, \#_{AB,-\bar{l}}\right). \quad (2.25)$$

which approximately follows F -distribution

$$Q_{\bar{l}} \equiv \frac{\left(\#_{ABBA,\bar{l}} - \mu_{ABBA,\bar{l}}\right)^2}{\hat{\sigma}_{ABBA,\bar{l}}^2} \sim F_{1,\nu}, \quad (2.26)$$

with 1 numerator and ν denominator degree of freedom. The framework agglomerates tested pairs and new neurons to find multi-neuronal spatiotemporal activity patterns with the same statistical test with Bonferroni correction[11].

2.3.1 limitotions

Since the method detect set of neurons that fires in specific time lags, this method fails to detect impomplete patterns. Also, the computational complexity can be problematic if the number of neurons increase. Repeated application of Bonferroni correction can be problematic in terms of statistical power[90].

Chapter 3

Edit similarity-based cell assembly sequence detection

This chapter introduces the proposed framework. This chapter is concerned with four factors: novel similarity metric between two given multi-neuronal time series, two clustering algorithms that are applied sequentially to detect repeated activation of the same types of cell-assembly sequences, and framework for extracting common timing structure among cluster members.

3.1 Overview of our method

Our goal is to develop a method to detect similar temporal patterns that occur repeatedly in neural population activity data, without relying on external sensory or behavioral events. Each of these patterns may represent a sequence of neural ensembles firing coincidentally in irregularly repeated temporal windows of certain length (Figure 3.3B). It is difficult to detect sequentially activated cell assemblies because repeated sequences are not usually identical (Figure 3.3C): (i) temporal structure of the target sequence is shown; (ii) sequences may be contaminated

with noisy spikes belonging to none of the cell assemblies; (iii) sequences may not always repeat a complete set of cells; (iv) repeated sequences of the member neurons may exhibit large jitters; (v) sequences can be expanded or compressed from one repetition to another; and (vi) furthermore, a few neurons may belong to multiple sequences. Such overlaps further complicate sequence detection.

To overcome these difficulties, we developed a robust sequence-detection method based on the edit similarity score, an index used in the field of computer science [66]. The basic concept of edit similarity is simple. Suppose we evaluate the similarity between two strings of genes, namely, “ATCGTAC” and “ATGTTAT”. We may naively count the number of coincident bases at the corresponding positions in the two strings. In the above example, the first two bases “AT” coincide, so the similarity is two. However, if we count the maximal number of coincidences preserving the serial orders of bases but allowing the insertion of blanks “-”, we may compare “ATCGT-A-C” and “AT-GTTAT-” to obtain the maximal similarity of five (i.e., A, T, G, T, and A coincide in this order). The Needleman–Wunsch (N–W) algorithm [92] provides a rigorous method for scoring edit similarity between arbitrary strings. In the present study, we extend this algorithm to make it applicable to neural data.

3.2 Edit similarity-based cell-assembly sequence detection

We explain the three major components of the proposed method, namely, edit similarity score with an exponentially growing gap penalty, clustering algorithms, and profile generation algorithm.

3.2.1 Edit similarity calculation by N-W algorithm

We explain the fundamentals of edit similarity score without gap penalty since this metric is not commonly used in neuroscience. Edit similarity score, or edit distance, quantifies the similarity between two strings with the minimum number of operations required to transform one string into the other. We can define arbitrary scoring schemes for each manipulation on strings, that is, insertion of a gap, deletion of a character, and comparison of two characters for coincidence. Needleman and Wunsch [92] proposed one of the most widely used evaluation algorithms of this metric (N-W algorithm).

The original N-W algorithm uses Dynamic Programming (DP), which essentially partitions given problem into subsequent small subproblems to compose a solution of the original problem from those of the subproblems. As an example, we evaluate the score between two strings, $W(1) = \text{ATCGTAC}$ and $W(2) = \text{ATGTTAT}$. As shown in Figure 3.1, we prepare a grid (DP table) and arrange the two strings along the abscissa and ordinate of the DP table. We add a null character “#” to the heads of the two strings and fill the leftmost column and the bottom row with zeros to initialize the following iterative operation.

We assign an appropriate score to each operation (insertion, deletion and coincidence). For the sake of simplicity, in this example without gap penalty we assign +1 to a coincidence, 0 to an insertion and a deletion. Let ϵ_j^i be the number of partial coincidences obtained up to the i -th element of $W(1)$ and the j -th element of $W(2)$. Then, we determine the value ϵ_j^i of the cell (i, j) of DP table by the following recursive equation:

$$\epsilon_j^i = \max \begin{cases} \epsilon_j^{i-1} \\ \epsilon_{j-1}^i \\ \epsilon_{j-1}^{i-1} + \delta(W(1)[i], W(2)[j]) \end{cases}, \quad (3.1)$$

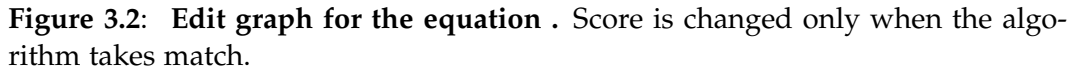
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| T | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| A | 0 | 1 | 2 | 2 | 3 | 4 | 5 | 5 |
| T | 0 | 1 | 2 | 2 | 3 | 4 | 4 | 4 |
| T | 0 | 1 | 2 | 2 | 3 | 4 | 4 | 4 |
| G | 0 | 1 | 2 | 2 | 3 | 3 | 3 | 3 |
| T | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| # | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | # | A | T | C | G | T | A | C |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | T | - | G | T | T | A | T | - |
| A | T | C | G | T | - | A | - | C |

Figure 3.1: Edit similarity calculation between strings. We can calculate edit similarity score between “ATCGTAC” and “ATGTTAT” by using a DP table (top). Denoting the characters appearing after diagonal-up moves in the table gives a maximally coincident string (bottom).

where $\delta(x, y)$ is the Kronecker’s delta: $\delta(x, y) = 1$ if $x = y$ and 0 if $x \neq y$. The edit graph for the recurrence is Figure 3.2. We can fill the grid from the lower left cell to the top right cell with the scores calculated by the above equation (Figure 3.1). We note that $\delta(x, \#) = 0$ for any character x including a null character itself. Then, we obtain the similarity score of two strings $\mathbf{W}(1)$ and $\mathbf{W}(2)$, which is five in this case, at the top right cell e_8^8 . Note that the operation $\epsilon_j^i = \epsilon_j^{i-1}$ corresponds to a deletion of $W(2)[i]$, or equivalently, a gap insertion after $W(1)[j]$. Likewise, $\epsilon_j^i = \epsilon_{j-1}^i$ corresponds to a deletion of $W(1)[j]$ or a gap insertion after $W(2)[i]$, and $\epsilon_j^i = \epsilon_{j-1}^{i-1} + 1$ corresponds to taking a coincidence.

The DP table enables us to obtain the substring that gives the maximum number of coincidence. For this purpose, we usually use the “ β table” that stores the procedural dependency among the cells in the DP table [92]: from the top to the


$$\beta_j^i = \begin{cases} \uparrow & (\epsilon_j^i = \epsilon_j^{i-1}) \\ \rightarrow & (\epsilon_j^i = \epsilon_{j-1}^i) \\ \nearrow & (\epsilon_j^i = \epsilon_{j-1}^{i-1} + \delta(W(1)[j], W(2)[i])) \end{cases}, \quad (3.2)$$

and illustrated with gray arrows in Figure 3.1, and the resultant alignments of $W(1)$ and $W(2)$ are shown at the bottom.

3.2.2 Extended N-W algorithm for neural activity

In this section, we explain the extension of the original scoring method for neural activity. We segmented spike data with a sliding time window of width T_w , and divided each time window into L bins with size b (thus, $L = T_w/b$). If we consider the activity pattern of the neural ensemble in each bin (i.e., the rate vector \mathbf{r} of coincidently firing neurons in Figure 3.3A) as a “letter”, we obtain a string of letters in each time window. Note that each neuron may fire multiple spikes in a bin, so each component of the activity vector represents the number of spikes generated by the corresponding neuron in the bin. The window size and bin size are manually determined from the temporal features of the neural data. Unless otherwise stated, we used the values of T_w ranging from 100 ms to 500 ms and those of b from 1 ms or 10 ms. Our task is to find all segments that contain similar activity vectors in the same temporal order (Figure 3.3C(i) (iv)).

To make the N-W algorithm applicable to spike data, we developed three extensions: scoring with the inner product, exponential gap penalty, and local alignment of starting points. At the outset, we defined the degree of similarity between activity vectors observed in different time windows. In a comparison of two gene sequences, we defined naturally how to count the number of coincident letters (i.e., nucleotide bases) between the two sequences. However, the same scoring scheme cannot be applied to neural activity data because the neural population in vivo will hardly repeat the same patterns owing to various noise sources. In the extended N-W algorithm, we replaced the delta function $\delta(W(1)[j], W(2)[i])$ in the N-W algorithm with the inner product of activity vectors. Let matrices $W(t_k)$ and $W(t_{k'})$ be the spiking activities of N neurons in the windows starting at time points t_k and $t_{k'}$, and $\mathbf{r}_i(t_k)$ and $\mathbf{r}_i(t_{k'})$ be the column vectors in the i -th bin of $W(t_k)$ and $W(t_{k'})$, respectively (see Figure 3.3A):

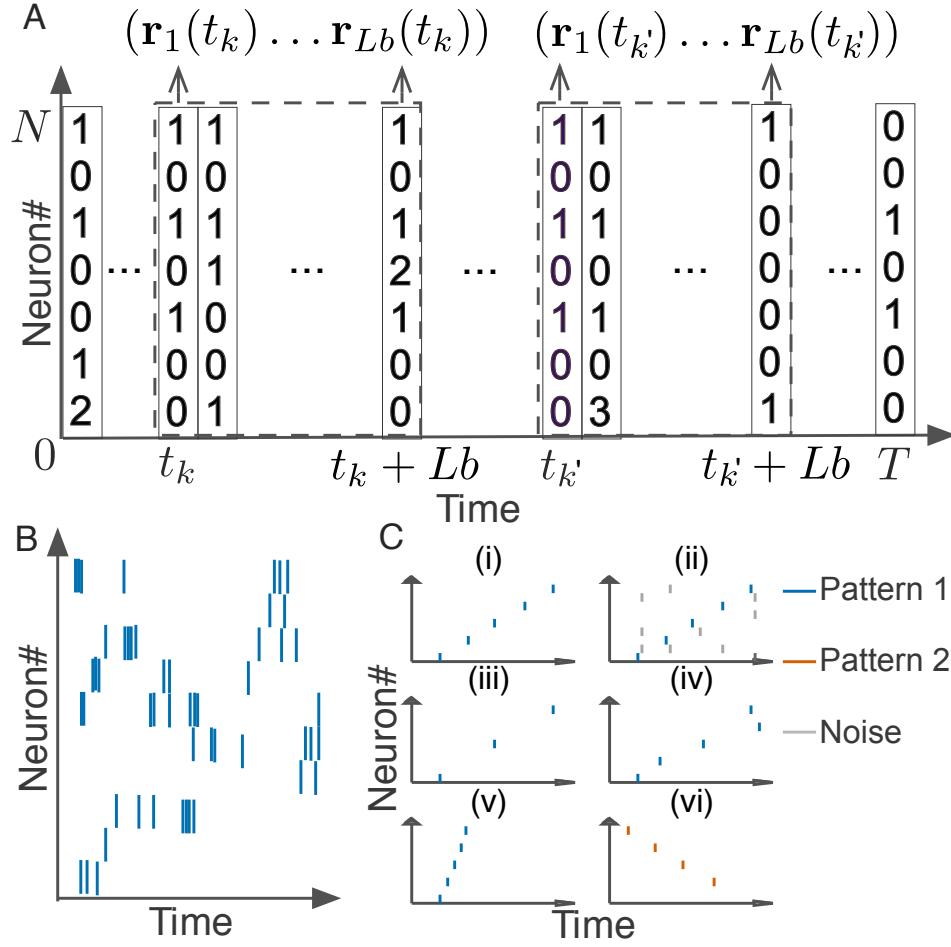


Figure 3.3: Detection of repetitive cell-assembly sequences. (A) Sliding time windows $W(t_k)$ are divided into L bins with an identical size, where t_k refers to the start time of the k -th time window. Population rate vector consists of the spike counts of individual neurons in each bin. (B) A temporal pattern in a sliding time window is schematically illustrated. Such a pattern may contain neurons belonging to a cell assembly as well as non-member neurons. Member neurons may fire at different rates with different temporal precision. Similarity between cell assembly sequences will increase when they share more member neurons and when they fire in a more similar temporal order at similar firing rates with higher temporal precision. Note that each member neuron may appear multiple times at different temporal positions in a time window. (C) Difficulties in detecting repetitive cell-assembly sequences are schematically illustrated. Blue and red bars show spikes of member neurons, while gray bars represent noisy spikes of non-member neurons: (i) temporal structure of target sequence; (ii) contamination by spikes of non-member neurons; (iii) missing spikes of member neurons; (iv) jitters in spike timing of member neurons; (v) arbitrary scaling of sequence length; (vi) member overlaps between different sequences.

$$W(t_k) = (\mathbf{r}_1(t_k), \mathbf{r}_2(t_k), \dots, \mathbf{r}_L(t_k)), \quad (3.3)$$

$$W(t_{k'}) = (\mathbf{r}_1(t_{k'}), \mathbf{r}_2(t_{k'}), \dots, \mathbf{r}_L(t_{k'})). \quad (3.4)$$

By considering $W(t_k)$ and $\mathbf{r}_i(t_k)$ (similarly for $W(t_{k'})$ and $\mathbf{r}_i(t_{k'})$) as a string and a character in the N–W algorithm, respectively, we measured the similarity between the activity patterns at t_k and $t_{k'}$ by using the inner product $\mathbf{r}_i(t_k) \cdot \mathbf{r}_j(t_{k'})$. An example of this evaluation is shown in Figure 3.4.

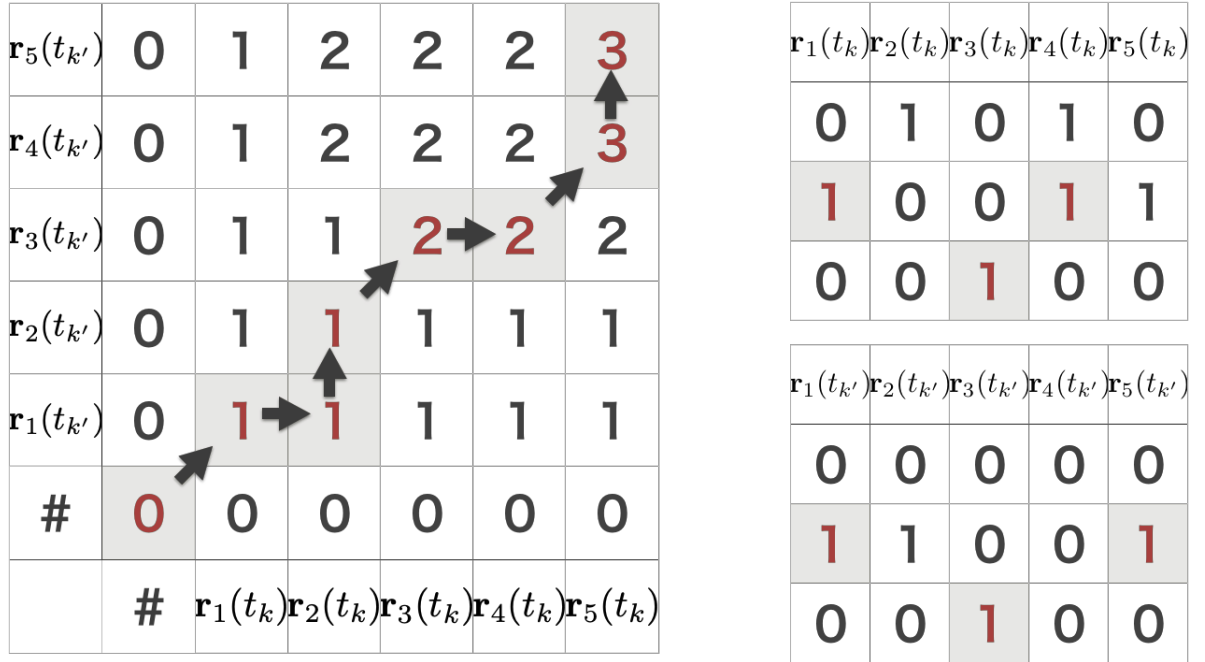


Figure 3.4: Example of extended N–W algorithm. (Left): DP table of two examples of neuronal activities. **(Right):** Example neuronal activities. The row denotes neuron number, and the column denotes time bin. Coincident activations are represented in red.

Data: two partial neuronal activities $(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n), (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_m)$

Result: Two tables that shows edit similarity score and its trace

```

for  $i \leftarrow 1$  to  $n$  do
  |  $s_{i,0} \leftarrow 0,$ 
end
for  $j \leftarrow 1$  to  $m$  do
  |  $s_{0,j} \leftarrow 0,$ 
end
for  $i \leftarrow 1$  to  $n$  do
  | for  $j \leftarrow 1$  to  $m$  do
    |
    | 
$$s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + \mathbf{v}_i \cdot \mathbf{w}_j \end{cases}$$

    | 
$$b_{i,j} \leftarrow \begin{cases} \uparrow & (s_{i,j} = s_{i-1,j}) \\ \rightarrow & (s_{i,j} = s_{i,j-1}) \\ \nearrow & (s_{i,j} = s_{i-1,j-1} + \mathbf{v}_i \cdot \mathbf{w}_j) \end{cases}$$

    |
  | end
  | end
end
return  $s_{n,m}, \mathbf{b}$ 

```

Figure 3.5: Needleman and Wunsch algorithm

Data: Two partial neuronal activities \mathbf{v}, \mathbf{w} to be aligned and backtracking pointer \mathbf{b}

Result: Alignments of two partial neuronal activities $\text{align1}, \text{align2}$

```

while True do
  if  $i == 0$  or  $j == 0$  then
    return
  end
  if  $b_{i,j} == \uparrow$  then
     $\text{align1} \leftarrow v_i + \text{align1}$ 
     $\text{align2} \leftarrow \text{gap} + \text{align2}$ 
  end
  if  $b_{i,j} == \rightarrow$  then
     $\text{align1} \leftarrow \text{gap} + \text{align1}$ 
     $\text{align2} \leftarrow w_j + \text{align2}$ 
  end
  if  $b_{i,j} == \nearrow$  then
     $\text{align1} \leftarrow v_i + \text{align1}$ 
     $\text{align2} \leftarrow w_j + \text{align2}$ 
  end
end
return  $\text{align1}, \text{align2}$ 

```

Figure 3.6: algorithm for making alignments

3.2.3 Affine gap penalty

Although the extension described in the previous section enables us to evaluate the edit similarity scores of neuronal spiking activity data, the scoring system has several shortcomings. The system cannot distinguish the activities, such as the one shown in Figure 3.7. Gap penalty can be introduced to overcome this problem.

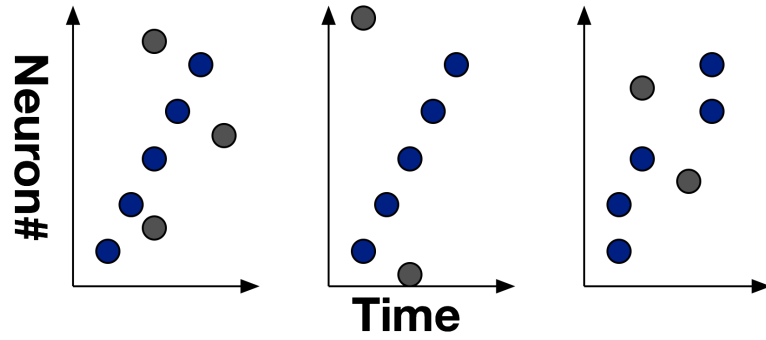


Figure 3.7: First shortcoming of previous scoring scheme. The previous scoring system cannot discriminate patterns with different magnitudes of jitters.

The affine gap penalty linearly penalizes the insertion of continuous gaps. Here, we use the implementation proposed by Gotoh [39]. In the scoring system, the score associated with the introduction of a gap is σ , and an extension is ρ . The system equation is as follows:

$$\begin{aligned}
 s_{i,j}^{\uparrow} &= \max \begin{cases} s_{i-1,j} - \sigma \\ s_{i-1,j}^{\uparrow} - (\rho + \sigma) \end{cases}, \\
 s_{i,j}^{\rightarrow} &= \max \begin{cases} s_{i,j-1} - \sigma \\ s_{i,j-1}^{\rightarrow} - (\rho + \sigma) \end{cases}, \\
 s_{i,j} &= \max \begin{cases} s_{i,j}^{\uparrow} \\ s_{i,j}^{\rightarrow} \\ s_{i-1,j-1} + \mathbf{r}_i(t_k) \cdot \mathbf{r}_j(t_{k'}) \end{cases}.
 \end{aligned} \tag{3.5}$$

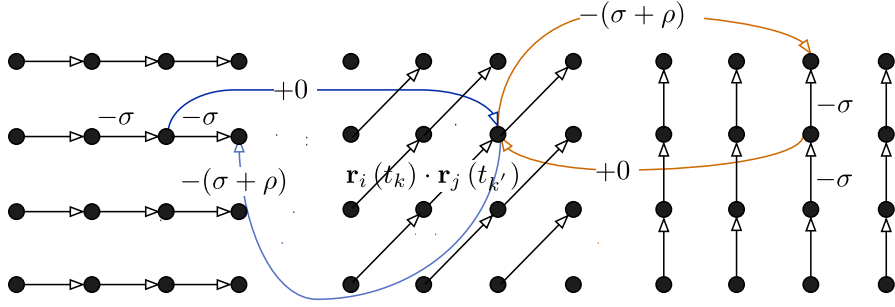


Figure 3.8: Edit graph for the equation . The left and right tables record the optimal numbers of gap insertions in each cell.

Here, variable $s_{i,j}^{\uparrow}$ represents the score that corresponds to the insertion of gaps into the alignment of $\mathbf{r}_i(t_k)$, and $s_{i,j}^{\rightarrow}$ represents the score that corresponds to the insertion of gaps into the alignment of $\mathbf{r}_j(t_{k'})$. A schematic diagram of the algorithm is shown in Figure 3.8.

3.2.4 Local alignment

The extended N–W algorithm aligns two neuronal activity matrices globally. This is useful when the aim is to identify common elements globally, but in our approach, the data to be compared are segments of the original neuronal spiking activities. To detect sequences from such data, a scoring system that can determine pattern onset is required (Figure 3.9).

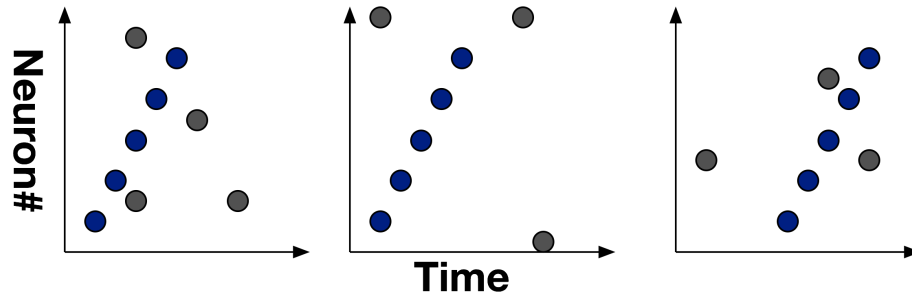


Figure 3.9: Limitation of the scoring scheme with gap penalty. The rightmost segment has a smaller edit similarity score relative to others in the previous scoring scheme.

Smith and Waterman proposed an algorithm to resolve this problem [122] that finds local alignments of two inputs. This is achieved by jumping from the starting points to other points in the edit graph:

$$\begin{aligned}
s_{i,j}^{\uparrow} &= \max \begin{cases} s_{i-1,j} - \sigma \\ s_{i-1,j}^{\uparrow} - (\rho + \sigma) \end{cases}, \\
s_{i,j}^{\rightarrow} &= \max \begin{cases} s_{i,j-1} - \sigma \\ s_{i,j-1}^{\rightarrow} - (\rho + \sigma) \end{cases}, \\
s_{i,j} &= \max \begin{cases} 0 \\ s_{i,j}^{\uparrow} \\ s_{i,j}^{\rightarrow} \\ s_{i-1,j-1} + \mathbf{v}_i \cdot \mathbf{w}_j \end{cases}.
\end{aligned} \tag{3.6}$$

The edit graph of the algorithm is shown in Figure 3.10.

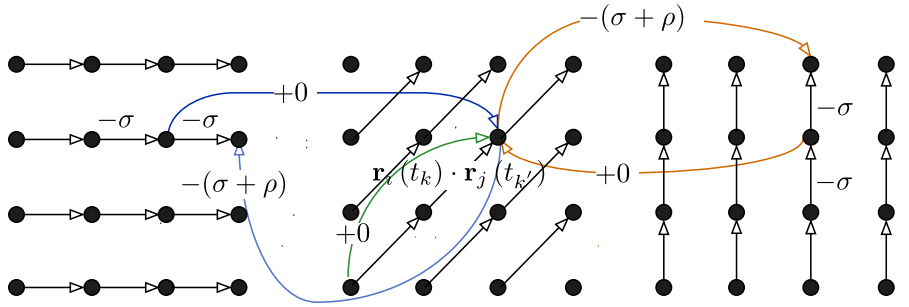


Figure 3.10: Edit graph for the equation . The left and the right tables record the optimal number of gap insertions in each cell. In addition, no-cost jump to an arbitrary location is introduced.

3.2.5 The proposed scoring schme

We developed a scoring scheme with exponential gap penalty, which penalizes the edit similarity score with an exponential discount factor when the corresponding elements appear after different time lags in two sequences, which is similar in spirit to the well-known linear gap penalty scheme [39]. In the following text, the two symbols v_j^i and ρ_j^i represent the optimal numbers of vertical and horizontal gap insertions required for partial comparison up to the i -th element of $W(t_k)$ and the j -th element of $W(t_k)$, respectively. By inserting an additional gap, we may obtain another coincidence at the cost of an additional discount factor in the similarity. Whether one should stop or continue with gap insertion is determined by comparing the cost and benefit of the two operations. To achieve cost-benefit balance, we should insert the maximal number of gaps such that the cost of inserting the gaps is not higher than the accrued benefits. We set the initial conditions $v_1^{1:L+1} = v_{1:L+1}^1 = \rho_1^{1:L+1} = \rho_{1:L+1}^1 = 0$ at the bottom row and the leftmost column of the table, where the notation $v_1^{1:L+1}=0$ means $v_1^1, v_1^2, \dots, v_1^{L+1} = 0$, and similarly, $v_{1:L+1}^1=0$ means $v_1^1, v_2^1, \dots, v_{L+1}^1 = 0$, and so on. Then, we calculate the values of v_j^i and ρ_j^i by using the following recursive formula:

$$v_j^i = \begin{cases} 1 & \epsilon_j^{i-1} - \exp(\alpha) \geq \epsilon_j^{i-1-v_j^{i-1}} - \exp(\alpha v_j^{i-1}) \\ v_j^{i-1} + 1 & \text{otherwise} \end{cases}, \quad (3.7)$$

$$\rho_j^i = \begin{cases} 1 & \epsilon_{j-1}^i - \exp(\alpha) \geq \epsilon_{j-1-\rho_{j-1}^i}^i - \exp(\alpha \rho_{j-1}^i) \\ \rho_{j-1}^i + 1 & \text{otherwise} \end{cases}, \quad (3.8)$$

where α is the weight of gap penalty. The conditions of $v_j^i = 1$ and $\rho_j^i = 1$ in the above equations are satisfied if the cost exceeds the benefit, and then we stop gap insertion. The values of v_j^i and ρ_j^i are calculated before ϵ_j^i in each cell.

We solved the local alignment problem by applying the previously proposed algorithm[122]. In the case of strings (Figure 3.1), the heads of strings from which we start the comparison are obvious. However, the heads of the cell-assembly sequences are not given a priori in neural data. In our scheme, when no significant coincidences are found up to cell (i, j) and the score in that cell is below 0, we restart recursive evaluation by setting ϵ_j^i to 0. In other words, we can jump from the bottom left cell to an arbitrary cell. This scheme results in an automatic search for the optimal starting points of the comparison.

In sum, the recursive equation of the N-W algorithm is changed into the following rule:

$$\epsilon_j^i = \max \begin{cases} 0 \\ \epsilon_j^{i-v_j^{i-1}} - \exp(av_j^{i-1}) \\ \epsilon_{j-\rho_{j-1}^i}^i - \exp(a\rho_{j-1}^i) \\ \epsilon_{j-1}^{i-1} + \mathbf{r}_i(t_k) \cdot \mathbf{r}_j(t_{k'}) \end{cases}, \quad (3.9)$$

which is evaluated along with v_j^i and ρ_j^i . The initial conditions are given as follows:

$$\epsilon_1^1, \epsilon_2^1, \dots, \epsilon_L^1 = 0, \quad \epsilon_1^1, \epsilon_2^2, \dots, \epsilon_1^L = 0, \quad (3.10)$$

and ϵ_{L+1}^{L+1} gives the maximum coincidence between the activity sequences, that is, the edit similarity score as in the standard N-W algorithm. The pseudo code of the algorithm is as follows.

Data: Two partial neuronal activities \mathbf{v}, \mathbf{w} to be aligned, backtracking pointer \mathbf{b} and maxrow, maxcol

Result: Alignments of two partial neuronal activities align1, align2

```

while True do
  if  $i == 0$  or  $j == 0$  then
    | return
  end
  if  $b_{i,j} == \text{start}$  then
    | return align1, align2
  end
  if  $b_{i,j} == \uparrow$  then
    | align1  $\leftarrow v_i + \text{align1}$ 
    | align2  $\leftarrow \text{gap} + \text{align2}$ 
  end
  if  $b_{i,j} == \rightarrow$  then
    | align1  $\leftarrow \text{gap} + \text{align1}$ 
    | align2  $\leftarrow w_i + \text{align2}$ 
  end
  if  $b_{i,j} == \nearrow$  then
    | align1  $\leftarrow v_i + \text{align1}$ 
    | align2  $\leftarrow w_i + \text{align2}$ 
  end
end
return align1, align2

```

Figure 3.11: algorithm for making alignments

Data: two partial neuronal activities $(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n), (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m)$

Result: Two tables that shows edit similarity score and its trace, and indices that corresponding to the maxima

```

for  $i \leftarrow 1$  to  $n$  do
  |  $s_{i,0} \leftarrow 0, s_{i,0}^{\rightarrow} \leftarrow 0, s_{i,0}^{\downarrow} \leftarrow 0$ 
end
for  $j \leftarrow 1$  to  $m$  do
  |  $s_{0,j} \leftarrow 0, s_{0,j}^{\rightarrow} \leftarrow 0, s_{0,j}^{\downarrow} \leftarrow 0$ 
end
for  $i \leftarrow 1$  to  $n$  do
  for  $j \leftarrow 1$  to  $m$  do

     $g_d \leftarrow s_{i-1,j}^{\downarrow}, g_r \rightarrow s_{i,j-1}^{\rightarrow}$ 
     $s_{i,j}^{\downarrow} \leftarrow \begin{cases} 1 & s_{i-1,j} - \exp(a) \geq s_{i-g_d,j} - \exp(ag_d) \\ s_{i-1,j}^{\downarrow} + 1 & s_{i-1,j} - \exp(a) < s_{i-g_d,j} - \exp(ag_d) \end{cases}$ 
     $s_{i,j}^{\rightarrow} \leftarrow \begin{cases} 1 & s_{i,j-1} - \exp(a) \geq s_{i,j-g_r} - \exp(ag_r) \\ s_{i-1,j}^{\rightarrow} + 1 & s_{i,j-1} - \exp(a) < s_{i,j-g_r} - \exp(ag_r) \end{cases}$ 
     $g_d \leftarrow s_{i,j}^{\downarrow}, g_r \rightarrow s_{i,j}^{\rightarrow}$ 
     $\text{down} \leftarrow s_{i-g_d,j} - \exp(ag_d)$ 
     $\text{right} \leftarrow s_{i,j-g_r} - \exp(ag_r)$ 
     $\text{match} \leftarrow \mathbf{a}_i \cdot \mathbf{b}_j$ 
     $s_{i,j} \leftarrow \max \begin{cases} 0 \\ \text{down} \\ \text{right} \\ \text{match} - \text{base} \end{cases}$ 
     $b_{i,j} \leftarrow \begin{cases} \text{start} & s_{i,j} = 0 \\ \uparrow & s_{i,j} = s_{i-1,j} \\ \leftarrow & s_{i,j} = s_{i,j-1} \\ \nearrow & s_{i,j} = s_{i-1,j-1} + \text{match} - \text{base} \end{cases}$ 

  end
end
 $\text{maxrow, maxcol} \leftarrow \text{indices\_that\_marks\_maximum\_score}(s)$ 
return  $s, b, \text{maxrow}, \text{maxcol}$ 

```

Figure 3.12: An algorithm for edit similarity calculation

3.2.6 Metric space and two algorithms for clustering of neural data

The segments form a high-dimensional feature space in which the edit similarity score defines a metric among the data points. That is, from the edit similarity scores $E(k, k')$ between pairs of time windows $W(t_k)$ and $W(t_{k'})$, we can calculate a distance matrix as $D(k, k') = \max(E(k, k')) - E(k, k')$, where the maximum is taken over all possible pairs of segments. In the high-dimensional feature space, time windows containing similar activity patterns are distributed at neighboring locations. Therefore, we can extract similar cell assemblies by clustering the data points. While similar activity patterns yield a dense cluster, time windows containing no repeated patterns are scattered over the feature space as outliers. To remove these “noisy” components, we combined two qualitatively different types of clustering algorithms.

Hence, we searched clustering algorithms under the following criteria:

- The algorithm has to allow clustering of similarity/distance matrix.
- The algorithm has to run in efficient computational time.
- The algorithm has to assign multiple labels to a data point because of the possibility that a segment contains multiple sequences.

Under such conditions, we chose the first algorithm called “OPTICS,”. It finds dense clusters of data points [5] in feature space and it allows us to input the distance matrix. However, this algorithm cannot discriminate between two clusters if they share a non-negligible number of data points. Therefore, we combined a second algorithm “COPRA,” which performs clustering based on a community detection scheme [41], with OPTICS. In short, the data points connected by relatively short distances (large edit similarities) are distinguished from the other data points as a community in the feature space. In this study, we applied OPTICS and COPRA sequentially to exploit the advantages of each method.

3.2.7 Pattern detection by OPTICS

Repetitive sequential activities are packed densely in the feature space. Density-based clustering algorithms are appropriate for detecting such clusters. These algorithms consider that a data point belongs to a cluster if there are a certain number (MinPts) of points within a neighborhood radius ε with the representative point as the center. A more formal definition of the clusters in density-based clustering and an algorithm for detecting densely localized clusters are as follows.

OPTICS algorithm and automatic cluster extraction

We adopt the OPTICS clustering algorithm[5]. Note that the algorithm is used as a pre-processing step that removes "noise" from datasets and extracts cluster-member data points. Here, "noise" means partial neuronal activities that contain no temporal patterns.

- Object p is directly density-reachable from object q with respect to ε and MinPts if $p \in n_\varepsilon(q)$ and $\text{Card}(n_\varepsilon(q)) \geq \text{MinPts}$, where $n_\varepsilon(q) = \{i \in d \mid \text{dist}(i, q) < \varepsilon\}$ and $\text{Card}(n)$ represents the cardinality of set n .
- Object p is density-reachable from an object q with respect to ε and MinPts if there are arbitrary number of objects p_1, p_2, \dots, p_n ($p_1 = q, p_n = p$), where p_i and p_{i+1} ($i = 1, 2, \dots, n-1$) are directly density-reachable.
- Object p is density-connected to object q with respect to ε and MinPts if there are MinPts of objects o for $\forall o \in d$, both p, o and q, o are density-reachable.

The set of objects c is a density-based cluster with respect to ε and MinPts if $\forall p, q \in c$ p and q is density-reachable and $\forall p, q \in c, p$ is density-connected to q . From these variables, one may define "core distance" and "reachability distance" as follows:

$$\text{core-distance}_{\varepsilon, \text{MinPts}} = \begin{cases} \text{undefined} & \text{Card}(n_{\varepsilon}(p)) < \text{MinPts} \\ \text{dist}(p, q) & \text{otherwise } q \text{ is the nearest object} \end{cases}, \quad (3.11)$$

$$\text{reachability-distance} = \begin{cases} \text{undefined} & |n|_{\varepsilon}(o) < \text{MinPts} \\ \max(\text{core-distance}(o), \text{dist}(o, p)) & \text{otherwise} \end{cases}. \quad (3.12)$$

Starting from a random data point, one may sequentially calculate the reachability distance and obtain a reachability plot. The dents in the plot correspond to the clusters. An illustration of the clustering algorithm is shown in Figure 3.13, and an example detection is shown in Figure 3.15.

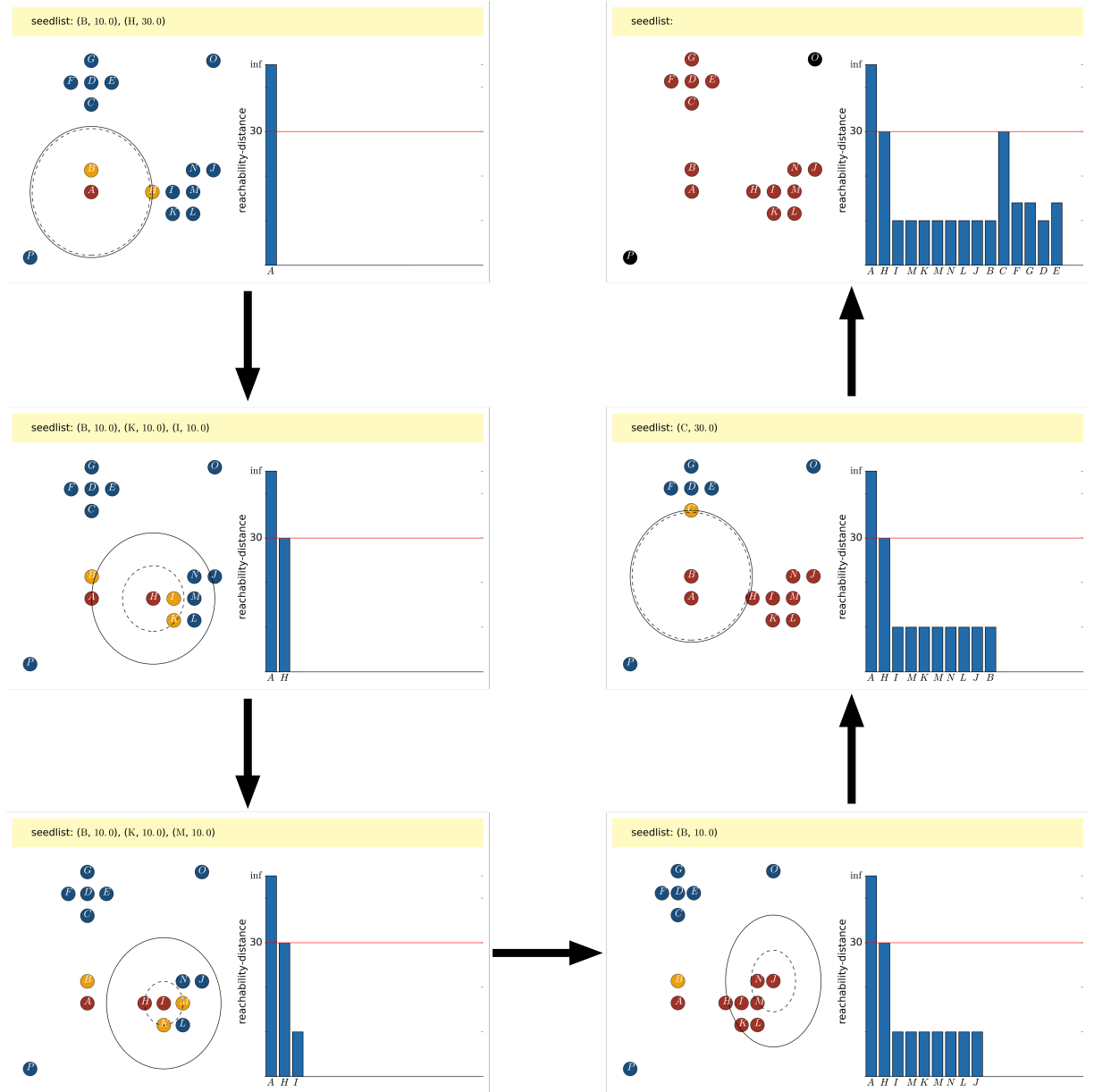


Figure 3.13: Illustration of OPTICS algorithm. Each circle represents different data points, and cardinality is shown in black.

```

Data: setOfObjects,  $\varepsilon$ , MinPts
Result: reachability plot
for object in setOfObjects do
    if object.processed is false then
        | expandClusterOrder(setOfObjects, object,  $\varepsilon$ , MinPts)
    end
end
for object in setOfObjects.orderedList do
    | ReachabilityPlot.append(object.reachabilityDistance)
end
return setOfObjects.orderedList, ReachabilityPlot
Function expandClusterOrder(setOfObjects, object, "", MinPts)
    if core-distance $_{\varepsilon, \text{MinPts}}$ (object)  $\leq \varepsilon$  then
        while object.processed is false do
            object.processed  $\leftarrow$  true
            setOfObjects.orderedList.append(object)
            nextObject  $\leftarrow$  the nearest unprocessed object from the current object
            object.reachabilityDistance  $\leftarrow$  reachability-distance $_{\varepsilon, \text{MinPts}}$ (object, nextObject)
        end
    end
else
    | object.processed  $\leftarrow$  true
end

```

Figure 3.14: Algorithm OPTICS

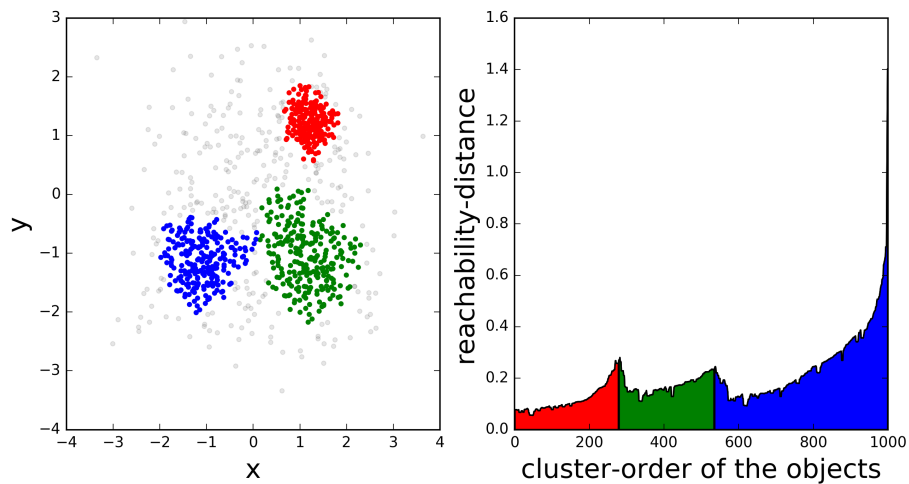


Figure 3.15: Example of automatic cluster detection by OPTICS. Three clusters in the dataset detected.

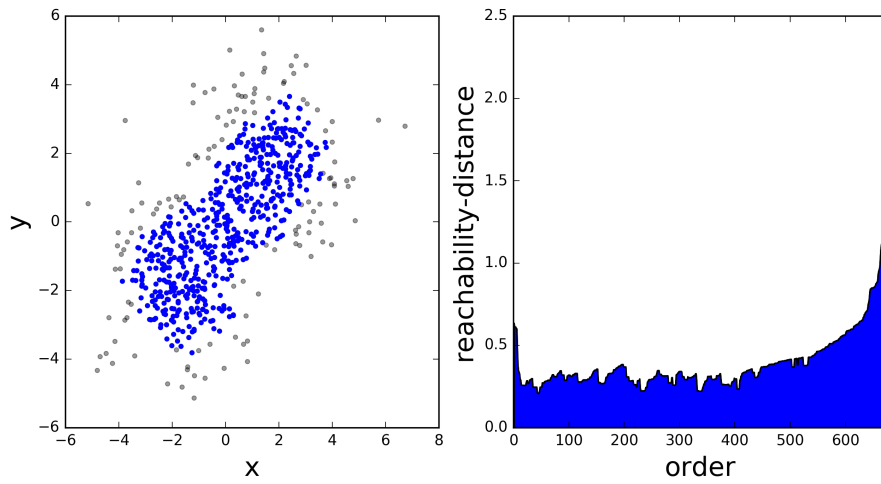


Figure 3.16: OPTICS fails to distinguish clusters OPTICS fails to differentiate clusters when they share a few data points.

3.2.8 Pattern separation by using COPRA

OPTICS cannot distinguish two clusters that share a certain number of data points (Figure 3.16). To overcome this problem, the community detection algorithm COPRA is introduced [41]. The word "community" refers to a sub-graph whose vertices are connected more densely than those of other sub-graphs. Notably, the similarity matrix is visible. The algorithm detects communities inside a graph by propagating labels. The detailed procedure of the algorithm is as follows:

1. First, all vertices are assigned unique labels and unit weights for the labels.
2. In each iteration t , a vertex x is updated by using the following equation:

$$b_t(c, x) = \frac{\sum_{y \in n(x)} b_{t-1}(c, y)}{|n(x)|}, \quad (3.13)$$

where $b_t(c, x)$ is the weight of label c at vertex x .

3. All labels whose coefficients that are smaller than v are dropped. If all labels of the vertices are dropped, remain a label that randomly selected vertex r with $b_t(r, x) = 1$.
4. Calculate values i_t , c_t , and m . i_t is the set of community identifiers in use in the t -th iteration:

$$i_t = \{c \in v \mid \exists x \in v, b_t(c, x) > 0\}, \quad (3.14)$$

c_t is the number of vertices labeled with each community identifier:

$$c_t = \left\{ (c, i) \mid c \in v \wedge i = \sum_{x \in v, b_t(c, x) > 0} 1 \right\}, \quad (3.15)$$

and m is the minimum number of vertices labeled with each community

identifier:

$$m_t = \begin{cases} \{(c, i) \mid \exists p, \exists q((c, p) \in c_{t-1} \wedge (c, q) \in c_t \wedge i = \min(p, q))\} & i_t = i_{t-1} \\ c_t & \text{otherwise} \end{cases} \quad (3.16)$$

5. The algorithm stops label propagation as soon as $m_t = m_{t-1}$; else, it executes procedure 2.

The goodness of the detected community can be measured in terms of overlapping modularity, as proposed by Nicosia et al. [93]:

$$q_{ov} = \frac{1}{m} \sum_{c \in c} \sum_{i, j \in v} \left[\beta_{l(i, j), c} a_{i, j} - \frac{\beta_{l(i, j), c}^{out} k_j^{in} \beta_{l(i, j), c}^{out} k_i^{out}}{m} \right], \quad (3.17)$$

where

$$\mathcal{F}(\alpha_{i, c}, \alpha_{j, c}) = \frac{1}{(1 + \exp(-f(\alpha_{i, c}))(1 + \exp(-f(\alpha_{j, c})))}, \quad (3.18)$$

$$\beta_{l(i, j), c}^{out} = \frac{\sum_{j \in v} \mathcal{F}(\alpha_{i, c}, \alpha_{j, c})}{|v|}, \quad (3.19)$$

$$\beta_{l(i, j), c}^{in} = \frac{\sum_{i \in v} \mathcal{F}(\alpha_{i, c}, \alpha_{j, c})}{|v|}, \quad (3.20)$$

and

$$f(x) = 2px - p \quad p \in \mathcal{R}. \quad (3.21)$$

In the present thesis, the value $p = 30$ recommended by [41] is employed.

Figure 3.17 shows an example of clustering by using COPRA, and Figures 3.18 and 3.19 show that COPRA and the combination of COPRA and OPTICS can successfully detect clusters from noisy datasets.

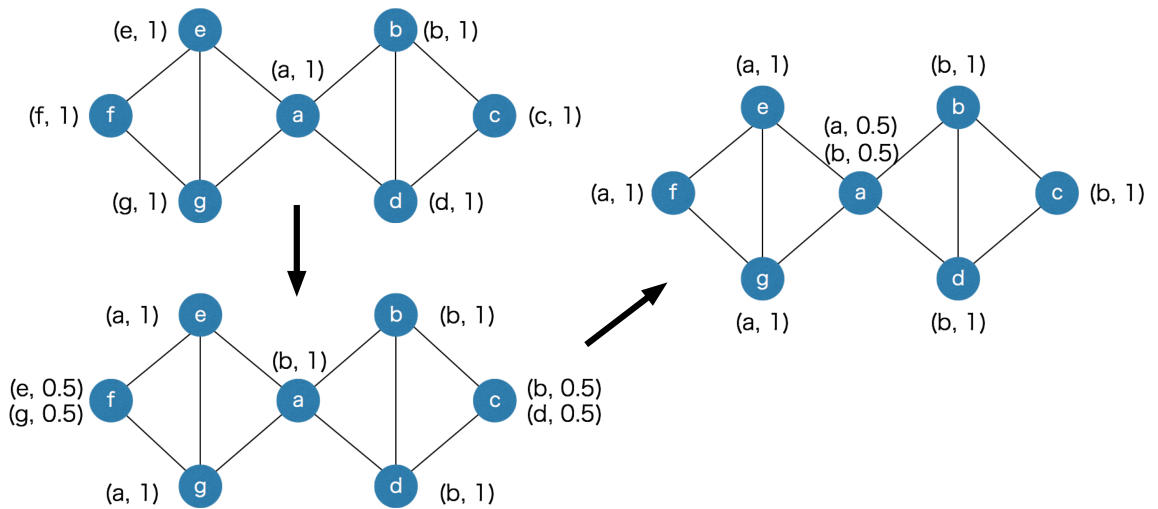


Figure 3.17: COPRA algorithm example Three steps of the algorithm.

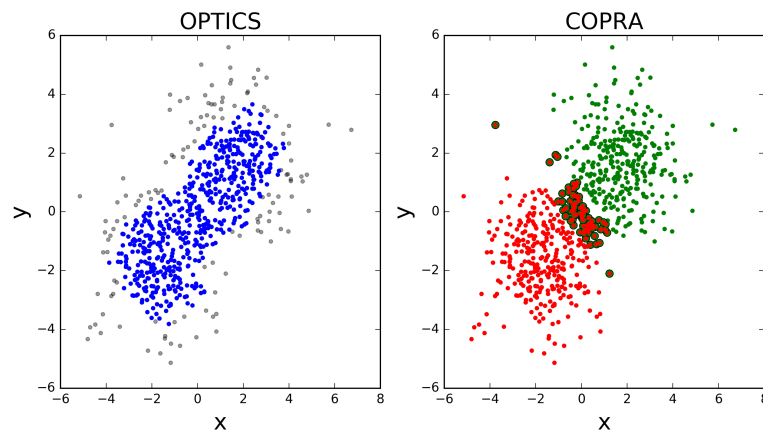


Figure 3.18: OPTICS can extract densely connected clusters

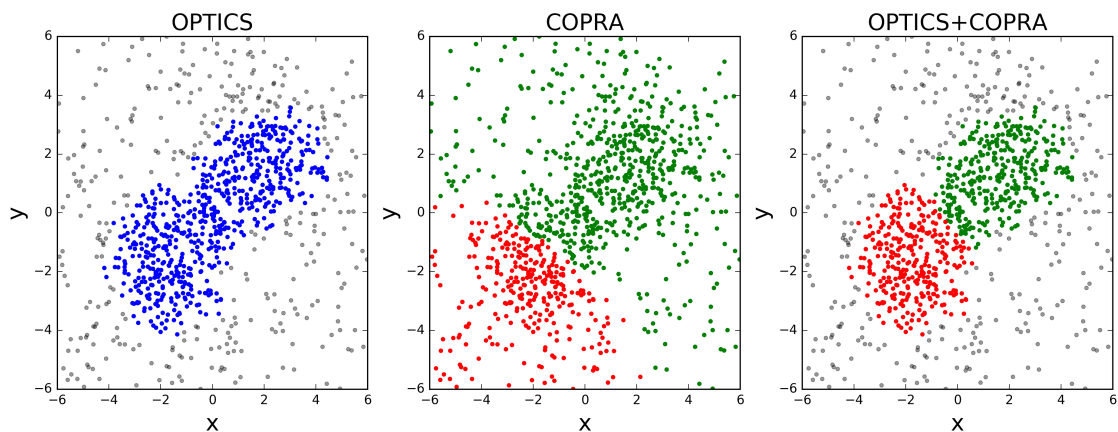


Figure 3.19: Combination of OPTICS and COPRA can extract densely connected clusters containing noise

3.2.9 Performance evaluation on Density- and community-based algorithms for sequence clustering

In our method, neural data are segmented into time windows and those containing similar sequential activity patterns are searched (see Figure 3.3A). These time windows form a high dimensional space with a metric defined by edit similarity score, and similar activity patterns form a cluster of neighboring data points (i.e., a candidate of cell assembly) in this feature space. To identify these clusters, we used a density-based clustering algorithm “OPTICS” [5] and a community-based clustering algorithm “COPRA” [41].

We tested these methods by using an artificial dataset. As shown in Figure 3.20A, OPTICS identified a single dense cluster from noisy data points, but it did not separate the cluster into two parts. On the other hand, “COPRA” identified two separate clusters but each cluster contained a considerable number of noisy data points: an outlier may be invited to a community if its distance from any member of the community is short enough (Figure 3.20B). In this study, we sequentially applied OPTICS and COPRA to take advantage of each method (Figure 3.20C). The combined use of the two algorithms compensated for the weakness of others to improve clustering performance. As studied in Figure 3.20D, the combined use was especially advantageous when the mean distance between data points was small. To our knowledge, such advantage has not previously been pointed out.

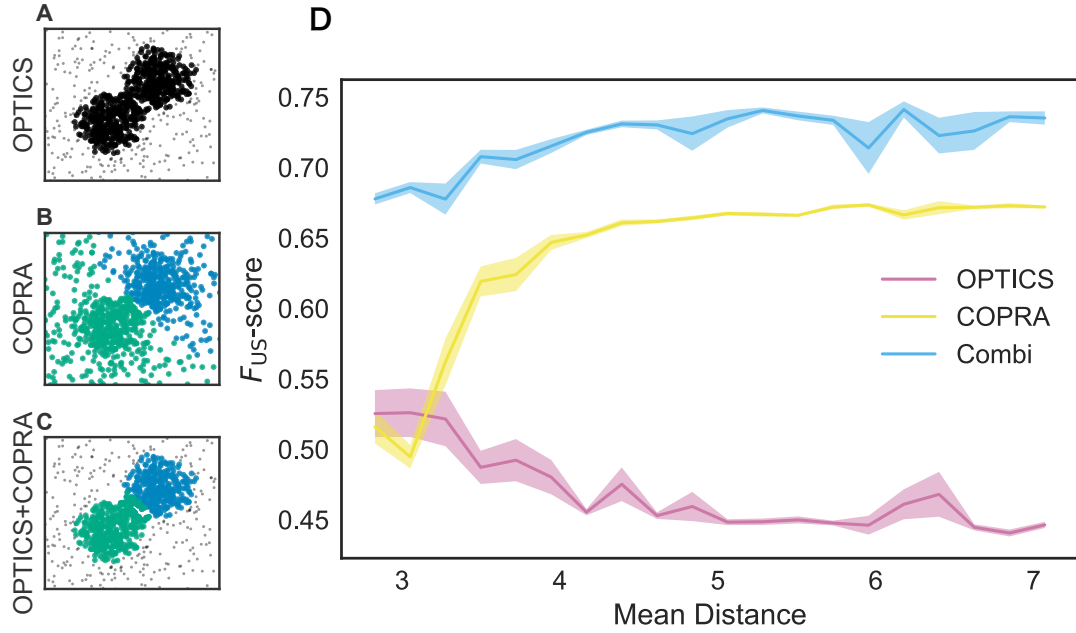


Figure 3.20: Comparison between different clustering algorithms. A density-based clustering algorithm (OPTICS) and a community detection algorithm (CORPA) were separately or sequentially applied to a dataset. Data points were generated by a mixture of two Gaussian distributions with different centers and the same variance of 1.3. These points were further mixed with uniformly distributed background data points. (A) OPTICS could remove background noise but failed to discriminate the two clusters. (B) COPRA could separate these clusters but failed to remove background noise. (C) The combined application of OPTICS and COPRA successfully separated the two clusters and removed background noise. (D) Performance of identifying two clusters was compared between the different algorithms, that is, OPTICS only (magenta), COPRA only (yellow) and their combination (cyan). The abscissa represents the distance between the centers of the two clusters. Shaded areas show standard errors.

3.2.10 Dimensionality reduction by t-SNE

In Figure 3.1A, we visualized the results of clustering in a two-dimensional space by using t-Distributed Stochastic Neighbor Embedding (t-SNE) [73]. t-SNE is an algorithm that maps high-dimensional data into a low dimensional space, typically two or three-dimensional space while maintaining the original data structure in the high-dimensional manifold. In t-SNE, the similarity values are converted to the following conditional probability

$$p_{i|j} = \frac{\exp(-D(i, j)/\sigma_i)}{\sum_{k \neq i} \exp(-D(i, k)/\sigma_i)}, \quad (3.22)$$

and the joint probability is calculated by:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}, \quad (3.23)$$

where n is the number of data points. In Eq. 3.22, the parameter σ_i is modified to tune the visualization effect. We also modeled the joint probability of points y_i and y_j in a low-dimensional embedding space by using a Student t-distribution with one degree of freedom (also known as Cauchy distribution [34]):

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}. \quad (3.24)$$

Note that we set $p_{i|i}$ and $q_{i|i}$ to zero as we only interested in pair wise similarity. The algorithm accomplishes a mapping by reducing the Kullback-Leibler divergence[61, 62] between p_{ij} and q_{ij} :

$$\text{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right), \quad (3.25)$$

by using the gradient descent.

3.2.11 Tricks for reduction of the computational cost

The proposed method, in its original form, requires extensive computational resources when used on neural data. The major difficulty comes from the calculation of a similarity matrix that has a computational complexity of $O(M^2)$, where M is the number of time windows and grows with the data length T . We accelerated the calculation of edit similarity drastically by employing an approximation algorithm based on Jaccard similarity [12], which is originally proposed clustering and eliminating near-duplicates among texts [12, 13, 76] and applied various problems in computer science field [22, 27, 148] and bioinformatics [9, 96]. For instance, this algorithm dramatically reduced the computation time by approximately 97% on the hippocampal data analyzed later.

In this procedure, we reduce the calculation of edit similarity for pairs of time windows that do not share active neurons. Let $(w_i(t_k))$ be a Boolean matrix in which the element (i, k) is 1 if neuron i fires at least once in the time window at t_k or otherwise 0:

$$w_i(t_k) = \begin{cases} 1 & \sum_{b=0}^L (W_b^i(t_k)) \geq 1 \\ 0 & \text{otherwise} \end{cases}, \quad (3.26)$$

where $W_b^i(t_k)$ is the (i, b) element of $W(t_k)$. We measure the similarity between $\mathbf{w}(t_k)$ and $\mathbf{w}(t_{k'})$ by Jaccard similarity defined as

$$\text{Jaccard}(\mathbf{w}(t_k), \mathbf{w}(t_{k'})) = \frac{|\mathbf{w}(t_k) \cap \mathbf{w}(t_{k'})|}{|\mathbf{w}(t_k) \cup \mathbf{w}(t_{k'})|}, \quad (k, k' = 1, \dots, N_w) \quad (3.27)$$

where $|\mathbf{x} \cap \mathbf{y}|$ denotes the inner product of given two vectors, $|\mathbf{x} \cup \mathbf{y}|$ counts number of non-zero elements of the sum of them. The value of Jaccard similarity is between 0 and 1, and is close to unity if the column vectors at time t_k and $t_{k'}$ are similar.

Because calculation of Jaccard similarity for every possible pair of vectors is also $O(M^2)$, we wish to find out pairs that are likely to give highly similar $\mathbf{w}(t_k)$ without direct calculation. For this purpose, we can make use of the statistical properties of Jaccard similarity. Now a trick is to use hash function $\tilde{h}(\mathbf{x})$ which randomly assign an integer to the given integer x . Throughout this study, we used a built-in hash function of programming language Julia (<https://julialang.org/>). We define function $h(\mathbf{x}) = \min \tilde{h}(x'_i), x'_i \in \mathbf{x}, x'_i \neq 0$, which returns the minimum hashed number made by non-zero elements of \mathbf{x} . The value is called the minimum hash (min-hash) value. Importantly, we can prove the following relationship [24]:

$$\text{Prob}[h(\mathbf{w}(t_k)) = h(\mathbf{w}(t_{k'}))] = \frac{|\mathbf{w}(t_k) \cap \mathbf{w}(t_{k'})|}{|\mathbf{w}(t_k) \cup \mathbf{w}(t_{k'})|} = \text{Jaccard}(\mathbf{w}(t_k), \mathbf{w}(t_{k'})) \quad (3.28)$$

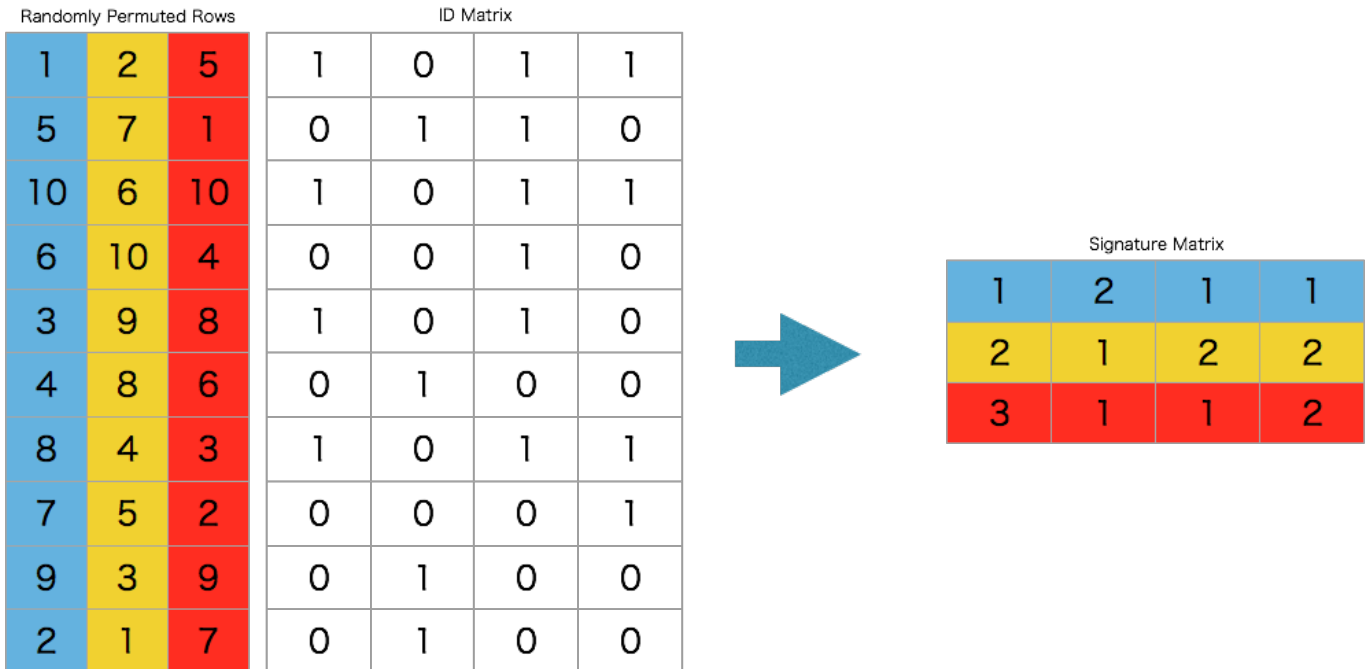


Figure 3.21: Min-hashing algorithm by example. (Left) Three different permutations of rows used to calculate Minhash values (colored in blue, yellow, and red). (Right) Resultant values are stored as the Signature Matrix.

With this relationship, we can obtain the Jaccard similarity without pair-wise comparisons of column vectors:

$$\text{Jaccard}(\tilde{\mathbf{w}}(t_k), \tilde{\mathbf{w}}(t_{k'})) \approx \frac{|\{q | 1 \leq q \leq n \text{ and } \tilde{S}_k^q = \tilde{S}_{k'}^q\}|}{n} \quad (3.29)$$

where \tilde{S} is called a signature matrix that contains the min-hash values over different random permutations of $\tilde{\mathbf{w}}(t_k)$, i.e., $\tilde{S}_k^q = h_q(\tilde{\mathbf{w}}(t_k))$ with h_q being the q -th min-hash function. The total number n of random permutations is dynamically adjusted as explained in the next section. In this matrix, elements in a column are min-hash values of a time window generated with different hash functions, and elements in a row are min-hash values of all time windows generated with a hash function. Figure 3.21 shows an example.

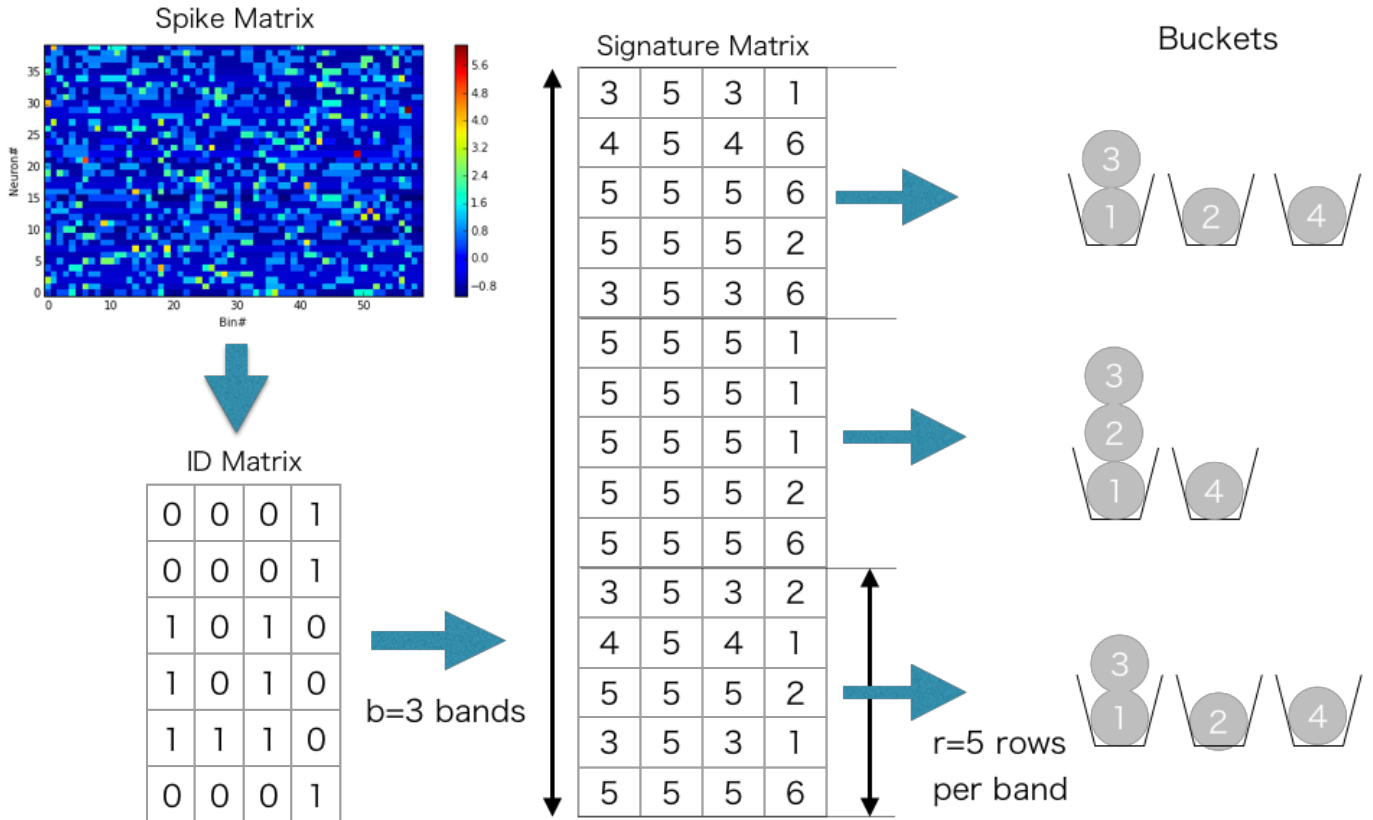


Figure 3.22: Minhashing algorithm application towards neuronal data. Similar activity vector tends to be put in the same bucket. The probability is determined by the Jaccard similarity.

| s | $1 - (1 - s^l)^b$ |
|-----|-------------------|
| 0.0 | 0.000 |
| 0.1 | 0.000 |
| 0.2 | 0.000 |
| 0.3 | 0.007 |
| 0.4 | 0.030 |
| 0.5 | 0.091 |
| 0.6 | 0.216 |
| 0.7 | 0.424 |
| 0.8 | 0.696 |
| 0.9 | 0.931 |
| 1.0 | 1.000 |

Table 3.1: Example probabilities. The values of the S-shaped function when $b = 5$ and $l = 2$. The probability $1 - (1 - s^l)^b$ that the signatures of two columns are identical in all rows at least one band if the Jaccard similarity for these is s .

To further reduce computation, we used the banding technique in the evaluation of Jaccard similarity [24]. We divided \tilde{S} into b bands of l rows each, thus $n = bl$. Suppose that two vectors $\tilde{\mathbf{w}}(t_k)$ and $\tilde{\mathbf{w}}(t_{k'})$ have Jaccard similarity s , then the probability that the min-hash signatures of two columns coincide at least in one row of the matrix is s . Then, the probability that the signatures of two columns are identical in all rows of at least one band is $p(s) = 1 - (1 - s^l)^b$, which is an S-shaped function of s and hence can be used for determining a threshold value of the similarity. We hash all the bands, and search bands in which two columns have the same hash value. The only pairs of time windows that have the same hash value in more than one band are used for similarity matrix calculation. For instance, $p(0) = 0.000$, $p(0.3) = 0.007$, $p(0.5) = 0.091$, $p(0.7) = 0.424$, $p(0.8) = 0.696$, $p(0.9) = 0.931$, and $p(1.0) = 1.000$ when $b = 5$ and $l = 2$. In the present analysis, the values of b and l were dynamically adjusted by data itself. We explain the method for the adjustment in the next section.

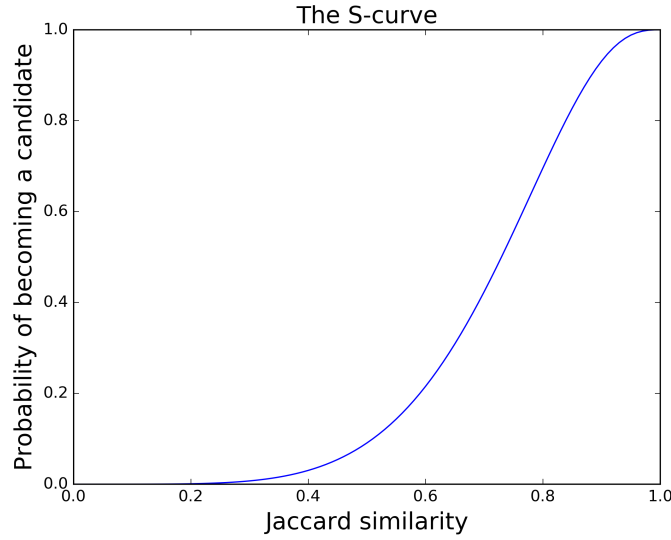


Figure 3.23: The SC curve with $b = 5$ and $l = 2$. X-axis shows Jaccard similarity values from 0 to 1. Y-axis represents the corresponding probability $1 - (1 - s^l)^b$ that the signatures of two columns are identical in all rows at least one band if the Jaccard similarity for these is s .

3.2.12 A policy for division of signature matrix

To apply the above algorithm to neural data, we employed a heuristic method to determine the two parameters b and l for Jaccard similarity from neural data. The aim of this method is to reduce the load of heavy computation for large neural data without losing candidate sequences. We calculated the average firing rates of individual neurons over the entire length of data, and we determined b and l assuming independent Poisson spiking neurons having the same firing rates. The parameters (b and l) for a smaller threshold (Jaccard_1) gives the similarity expected under the assumption of independent Poisson spiking, whereas a parameter for a larger threshold (Jaccard_2) represents the similarity expected when the two time windows contain sequences with a certain length. Let $\#_i$ be the total number of spikes of neuron i during the interval $[0, T]$. From $\#_i$, we can calculate the probability that neuron i has at least one spike in the segment $W(t_k)$ as $p_i = 1 - (1 - (\#_i/T)\Delta)^{(L/\Delta)}$, where Δ is the size of a bin. Then, the index $N_1 = \sum_{i=1}^N p_i$ is the

expected number of active neurons within the time window. Then, the expected number of coincidently active neurons in an arbitrary pair of time windows is $N_2 = \sum_{i=1}^N (p_i)^2$, and Jaccard_1 is calculated as $N_2 / (2N_1 - N_2)$.

Now, suppose that two time windows contain additional N_3 coincidently active neurons. In this case, the expected Jaccard similarity, or Jaccard_2 , is given as $(N_2 + N_3) / (2N_1 - N_2 - N_3)$. In this study, we searched such values of b and l that keep the probability $1 - (1 - s^l)^b$ sufficiently high (e.g., 0.8) for Jaccard_1 and sufficiently low (e.g., 0.1) for Jaccard_2 .

3.2.13 Construction of profiles for clustered sequences

Because activity patterns belonging to a cluster in general exhibit a large variation in the temporal structure, a method was necessary to identify the core temporal structure of the cell assembly sequences belonging to each cluster. Here, we explain our iterative multiple alignment algorithm for constructing profiles of clusters. It is based on the algorithm by [8]. In the original algorithm, we initialize the algorithm with a tentative profile, which is obtained by taking the longest common subsequence between the two time windows in a cluster that show the highest match in edit similarity. After the initialization, we search the next time window that gives the most similar profile to the tentative one, and update the tentative profile using edit similarity. We repeat this procedure until the tentative profile converges.

In our method, we made two major modifications to the original algorithm. First, we chose two arbitrary time windows in the initiation step to reduce the computational cost. The final results did not significantly differ between our approach and the original one. Second, in generating a profile, we used the instantaneous value of z-score of spike count in each time window. Namely, for each neuron we calculated the average and variance of spike count per bin over the

entire data, and then subtracted the average from spike count in each bin and normalized the difference by the variance. The use of z-score suppresses the influences of highly active neurons on the detection of ensemble firing sequences. Finally, in each step, a Gaussian filter with mean 0 and variance σ was applied to the tentative profile. Variance σ was initially as large as the window size and gradually reduced to the bin size as iterations proceeded. This filtering prevented a profile from containing more than two similar sequences, thus enabled a robust detection of minimal sequences.

3.2.14 F_S -score for supervised clustering

Performance of supervised cell-assembly detection from artificial data was scored in terms of F_S -score, which is given as the harmonic mean of Precision and Recall:

$$F_S = 2 \left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}} \right)^{-1}. \quad (3.30)$$

where Precision and Recall are defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (3.31)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (3.32)$$

in terms of true positives (TP), false positives (FP) and false negatives (FN). We also used Specificity, which is defined as

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \quad (3.33)$$

to evaluate the portion of negatives that are correctly classified as such.

3.2.15 F_{US} -score for unsupervised clustering

Performance of unsupervised cell-assembly detection of artificial data was scored in terms of F_{US} -score, which is widely used for unsupervised clustering in the field of machine learning (c.f. [6]). The score is given as the harmonic mean of Purity and Inverse Purity as

$$F_{US} = 2 \left(\frac{1}{\text{Purity}} + \frac{1}{\text{Inverse Purity}} \right)^{-1}. \quad (3.34)$$

We note that this score is different from F_S -score for supervised clustering. Purity is a weighted average of the fractions of true members in detected clusters,

$$\text{Purity} = \sum_{i=1}^m \frac{|C_i|}{T} \frac{|C_i \cap L_j|}{|C_i|}, \quad (3.35)$$

and Inverse Purity is a weighted average of correctly classified portions of true clusters,

$$\text{Inverse Purity} = \sum_{j=1}^n \frac{|L_j|}{T} \frac{|C_i \cap L_j|}{|L_j|}, \quad (3.36)$$

where m is the number of detected clusters $C = \{C_1, C_2, \dots, C_m\}$, n is the number of true clusters plus a noise cluster in the artificial data $L = \{L_1, L_2, \dots, L_n\}$, and T is the total number of data points (i.e., time windows). The noise cluster consists of spurious cell assemblies. $(C_i \cap L_j / C_i)$ represents the fraction of members of the j -th true cluster in the i -th detected cluster. j -th true cluster is selected by $\text{argmax}_k |C_i \cap L_k|$. In the above expressions, weights are determined such that a larger cluster contributes more strongly to the weighted sums. We note that Purity and Inverse Purity take their values within the interval $[0, 1]$. If a classification is perfect, both Purity and Inverse Purity take the maximal value of unity. The harmonic mean of Purity and Inverse Purity penalizes two trivial solutions. In one such solution each data point constitutes an independent cluster (i.e., $m = T$), and in the other solution all data points are classified into a large cluster. In these trivial cases, Purity, but not Inverse Purity, takes the maximum value of unity.

3.2.16 Cluster labels for PCA/ICA-based analyses

To compare our method with the PCA/ICA-based methods for detecting synchronously firing ensembles [69], we have to assign a cluster label to each component detected by these methods. To this end, we calculated overlaps between the population activity vector and the principal (or independent) components for all time windows. Then, in each time window the component yielding the highest

overlap with the instantaneous population activity was assigned to the time window as the corresponding cluster label. If the highest overlap in a time window did not reach a certain threshold, no PCA/ICA components were assigned to the window and this time window was treated as noise in the calculation of F_{US} -score. The above procedure was repeated until all time windows were labeled with some PCA/ICA components. For a fair comparison with our method, an optimal value that maximizes the F_{US} -score was searched for the threshold.

3.2.17 Behavioral labels for clusters

We introduced three behavioral labels (Go, Back, Stop) in the analysis of clusters of hippocampal neurons. First, individual members (time windows) of each cluster were labeled with "Go", "Back" or "Stop" according to the directions and average speed of animal's movement in the corresponding time window. A member was labeled as "Stop" if the average speed was less than 3 cm/s, and Go refers to the movement direction from start to goal and Back refers to the opposite direction. Then, in each cluster we constructed Gaussian kernel density functions (KDF) [99, 111] to fit the spatial distributions of cluster members labeled with the different behavioral labels, and measured the peak heights of these KDEs. Now, for each behavioral label we selected the top 20 clusters yielding the highest peaks of the corresponding KDF. We called thus-obtained three sets of 20 clusters as "Go cluster", "Back cluster" and "Stop cluster" in Figure 4.4A. We note that this categorization scheme allows a cluster of time windows to obtain multiple behavioral labels.

3.2.18 Parameter choices

Here we list the values of parameters in our algorithm. We searched the set of parameter values that maximizes the detection performance in Figure 4.1A. Each parameter was tested within the following range: the number of points for a minimal cluster in OPTICS MinPts from 2 to 20 [5]; parameter for COPRA v from 2 to 20 [41]; the parameter $\alpha = 1.0$; MinPts = 5 and $v = 5$ in Figure 4.1F-G.

For the hippocampal data, the following parameter values were used: the parameter $\alpha = 0.1$; the length of sliding time window $T_w = 100$ ms, which is close to the period of one cycle of theta oscillation; parameter for Jaccard₂ $N_3 = 10$; the number of points for a minimal cluster in OPTICS MinPts = 20; parameter for COPRA $v = 4$.

The prefrontal data were analyzed by using sliding time windows with a wider variety of lengths ranging from 250 ms to 2.5 sec because the characteristic time scale of sequences were not known. However, all the results shown in this study were obtained for the length of 250 ms. The temporal discount factor was set as $\alpha = 0.03$. Other parameters were as follows: parameter for Jaccard₂ $N_3 = 10$; the number of points for a minimal cluster in OPTICS MinPts = 400; parameter for COPRA $v = 30$.

3.2.19 Bayesian modeling for edit similarity difference

In the analysis of hippocampal activity, we searched locations in a linear maze at which the recorded neural activity coincides with a detected cell assembly in a statistically meaningful manner. We divided the linear maze (its total length ranged from 60 cm to 300 cm) into 30 different locations which we may term position bins. In each position bin (denoted as x), we computed edit similarity score ED_x^{raw} between the profile of given cell assembly and neural activity. Similarly,

we calculated a edit similarity score ED_x^{sge} for a surrogate data in which spikes of each neuron were randomly shuffled across time bins (this corresponds to the null model of homogeneous Poisson processes). Then, we used Bayesian modeling for estimating the significance of similarity score at each position bin compared with the null model. Let μ_x be the baseline score at the position bin x . We assumed that the value of μ_x smoothly changes across positions until the score exhibits a sudden jump at some position bins. To be specific, we assumed that ED_x^{raw} and ED_x^{sge} obey the following Gaussian distributions:

$$ED_x^{\text{sge}} \sim \text{Normal}(\mu_x, \sigma^{\text{sge}}) \quad (3.37)$$

$$ED_x^{\text{raw}} \sim \text{Normal}(\mu_x + \delta_x, \sigma^{\text{raw}}) \quad (3.38)$$

Then, we assumed that the change $\mu_x - \mu_{x-1}$ obeys the Gaussian distribution with the mean $\mu_{x-1} - \mu_{x-2}$ and the variance σ_μ and that a jump δ_x obeys the Cauchy distribution with the mean δ_{x-1} and the variance σ_δ :

$$\mu_x \sim \text{Normal}(2\mu_{x-1} - \mu_{x-2}, \sigma_\mu) \quad (3.39)$$

$$\delta_x \sim \text{Cauchy}(\delta_{x-1}, \sigma_\delta) \quad (3.40)$$

For the statistical modeling, we used STAN library[19] with default uniform prior distributions for the variances σ_μ , σ_δ , σ^{raw} and σ^{sge} to sample the parameters from the model using the NUTS(No-U-Turn Sampler[48]).

3.2.20 Experimental data

The data of hippocampal neurons is available at the data sharing website of Collaborative Research in Computational Neuroscience (CRCNS.org., <http://dx.doi.org/10.6080/K09G5JRZ>)[86]. The data used in this study contains the activity of 108 neurons recorded from the hippocampal CA1 region of a male Long-Evans rat during voluntary exploration of a linear maze. The total duration of recordings is 1928 s.

The multi-neuron spike trains of prefrontal neurons used in this study were recorded previously from the medial prefrontal cortex of a male Brown Norway/Fisher hybrid rat with a chronically implanted hyper drive consisting of 12 tetrodes [33]. The data contains the activity of 76 neurons and the total duration of recordings is 11,010 s.

The data of basolateral amygdala neurons is taken from the data sharing website of Collaborative Research in Computational Neuroscience (CRCNS.org., <http://dx.doi.org/10.6080/K0MS3QXD>)[38]. Total 22 sessions of four rats data used.

Chapter 4

Application of the edit similarity-based cell assembly detection framework

This chapter discusses performance evaluation under various conditions with comparison among existing frameworks and three examples in which the proposed framework is applied to datasets of real multi-neuronal activity in rat prefrontal cortex[33], hippocampus CA1 region[86], and amygdala [38] during the behavioral period. The first case employs the data used to show memory traces in PFC with template matching and shows the scalability of the proposed method. In addition, the chapter reports unobserved patterns in the amygdala data that cannot be detected with the template matching.

4.1 Performance evaluation with artificial data

We compared the performance of our method with that of PCA- [68, 102] and ICA-based method [69] by using synthetic population activity data. We embed-

ded 5 non-overlapping cell assemblies into background activity of 100 simulated neurons firing independently at a rate of 2 [Hz] (Figure 4.1A, top panel). PCA and ICA do not take the time structure of cell assemblies into account, but these methods should be good at detecting the assemblies of synchronously firing neurons. On the other hand, our method treats synchronously firing as a special case of sequential firing with zero time lags between spikes. In fact, we generally expect slightly better performance for synchronous firing than for sequential firing because the probability that the complete cell-assembly pattern falls within a time window will be higher for the former than for the latter. Therefore, we constructed synthetic data of length 300 s in which each cell assembly consisted of 20 synchronously firing neurons with certain timing jitters and appeared 50 times at randomly determined positions. The same size of time window (200 ms) was used in all the methods and no spikes of each cell assembly occurred across different time windows.

For performance evaluation, we searched the set of parameter values that maximizes the detection performance. Each parameter was tested within the following range the number of points for a minimal cluster in OPTICS MinPts 2 to 20 [5]; parameter for COPRA v 2 to 20 [41]. We evaluated the performance of each method in terms of F_{US} -score (see subsection 3.2.15 for details). We generated 40 artificial data with different background activity. We then analyzed each data by the three methods (i.e., PCA-based, ICA-based and the proposed methods) and calculated F_{US} -score for each trial. The resultant score was significantly larger in the proposal method (mean \pm s.d., 0.89 ± 0.09) than in the PCA-based (0.61 ± 0.07) and ICA-based (0.59 ± 0.11) methods (Figure 4.1B). In fact, our method correctly detected all target cell assemblies.

In Figure 4.1B, the value of timing jitters is relatively small (± 10 ms) and cell assemblies may be regarded as groups of synchronously firing neurons. For shorter timing jitters, the results would not change significantly. However, for longer jitters

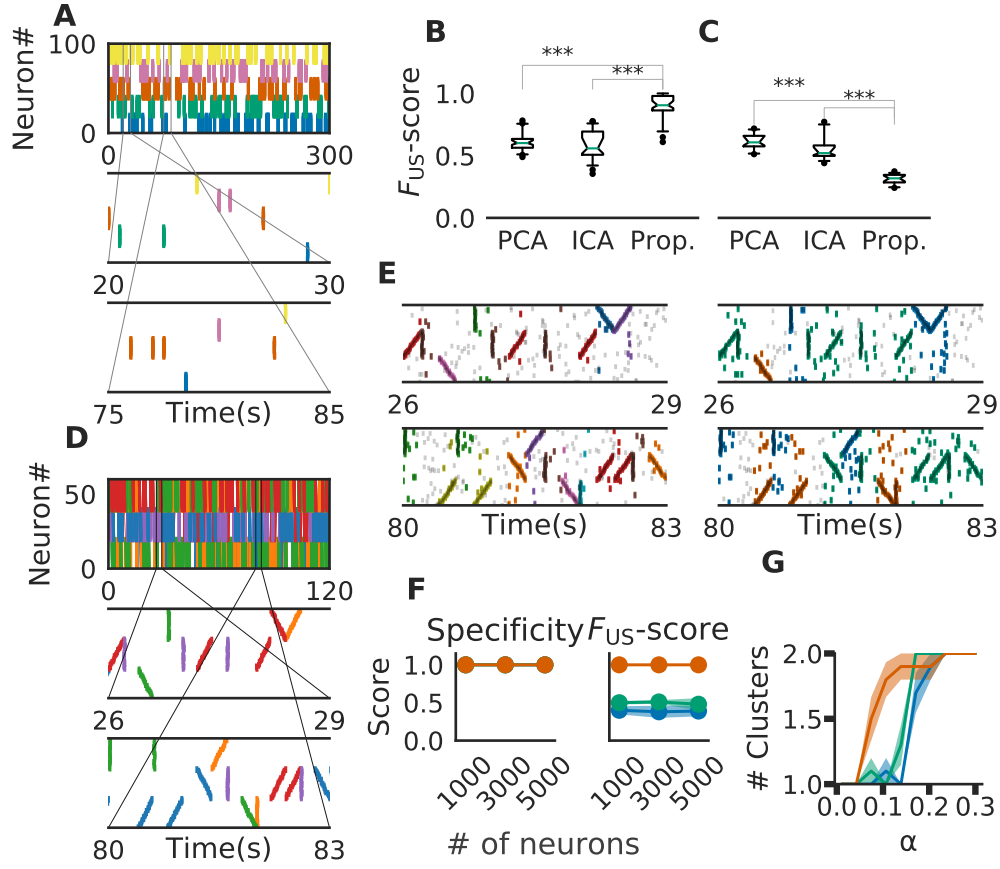


Figure 4.1: Comparison between different methods. (A) An example of the embedded artificial cell assemblies used for the comparison. In the raster plot, each dot is a spike. Each color indicates a cell assembly. For clarity, noisy spikes are not shown. (B) Timing jitters were within ± 10 ms and cell assemblies represented synchronously firing neuron ensembles. F_{US} -score was significantly higher for the proposed method than for PCA- and ICA-based algorithms (p values were less than $2.2e-16$ for both cases: Wilcoxon rank sum test). The time window used was 200ms. (C) Timing jitters were within ± 50 ms and cell assemblies may not be regarded as synchronously firing neuron ensembles. Our method is sensitive to the serial order of firing and hence the score is lowered (p values were less than $2.9e-11$ for both cases: Wilcoxon rank sum test). (D) Nine artificial spike sequences (middle and bottom) were embedded into noisy spike trains. Noise spikes are not shown here. (E) Sequences detected by our method from the artificial data shown in D are presented in two intervals together with noisy spikes (gray). The nine embedded sequences were successfully detected. The results are shown for different values of parameters: $\alpha = 2.0$, MinPts = 20, $v = 2$ in left panels; $\alpha = 0.5$, MinPts = 10, $v = 12$ in right panels. (F) Robustness of cell-assembly detection against background noise. Two scores, Specificity (left) and F_S (right), in the detection of a single cell-assembly are shown against the number of background Poisson spike trains: our method (orange), PCA (green), and ICA (blue). Shaded areas indicate standard errors. (G) Effect on different α for cluster formation. Different shrinkage rates tested: 10 (orange), 5 (green), and 3 (blue). Shaded areas indicate standard errors.

our method will exhibit degraded F_{US} -score because the method is sensitive to the serial order of firing. In contrast, PCA/ICA-based methods do not take the order of firing into account and therefore will exhibit no significant differences, as far as neurons belonging to each cell assembly fire within the same time windows. We studied this large-jitter case by using timing jitters of ± 50 ms without changing the discount factor (i.e., $\alpha = 0.1$) in Figure 4.1C. As expected, PCA/ICA-based methods show seemingly better performance compared to our method. These results indicate that our method is more sensitive to timing jitters than PCA-/ICA-based method. However, as we discuss in the next paragraph, sensitivity of our model can be controlled by changing α .

We then investigated if our method is able to extract spike sequences in noisy artificial data. Sequential firing of three non-overlapping cell assemblies each consisting of 20 neurons was embedded into background activity of 60 neurons (including the twenty) at a rate of 1 Hz in both forward, synchronous and reverse orders with ± 10 ms jitter (Figure 4.1D). Each sequences appeared 20 times. The time window was 200 ms and bin size was 10 ms. Our method detected the groups of cells firing sequentially as well as synchronously, but the way our method categorized these cells depended on the values of parameters used and noise level in input data. In the left panels of Figure 4.1E, the same group of neurons firing in different temporal orders were categorized as separate clusters (e.g., green and purple clusters). Namely, we can see three separate groups of cells that fire with an upward ramp, a downward ramp, or all simultaneously. In contrast, such a group of cells was classified into a single cluster in the right panels of Figure 4.1E. Although some spikes were misidentified, the following scores quantify the performance of our algorithm in detecting the three clusters: Precision = 0.75, Recall = 0.85, and the F_S -score = 0.8 in Figure 4.1, left; Precision = 0.63, Recall = 0.89 and the F_S -score = 0.73 in Figure 4.1E, right. The scores used are explained in subsection 3.2.15.

We further investigated the robustness of performance of our method at different signal-to-noise ratios. Here, we varied the ratio by modifying the number of background firing neurons. We embedded a single cell assembly consisting of 100 neurons firing synchronously without jitters into a sizable neural population while maintaining each neuron firing rate of 5Hz. The cell-assembly pattern appeared 20 times at random temporal positions in neural activity data of the total length of 60 sec. We generated ten instantiations for each total number of neurons ([1000, 3000, 5000]). We then analyzed these data sets by our method with the time windows of 200 ms and calculated F_S -score for supervised clustering for each set (subsection 3.2.15). The results are shown in Figure 4.1F, which proves the robustness of performance against changes in the magnitude of background noise. We used the parameter $\alpha = 1.0$, MinPts 5 and v 5 in this analysis.

Finally, we tested that the method's ability to detect a cluster of assembly sequences that were activated on two different time scales. Each assembly activation consists of 30 different neurons which were sequentially activated in 200 ms and embedded 20 times. In addition, we added the same numbers of compressed patterns. We used three different shrinkage factors, three, five, and ten times (illustrated in Green, Orange, and Blue in Figure 4.1G). These cell assemblies were embedded in the background Poisson firing at the rate of 1 Hz in 60 seconds. We have clustered the data with different alpha and same clustering parameter. The result indicates that our method with $\alpha = 0.1$ detects both types of cell assemblies as identical when shrinkage factor is three and five. We used MinPts 5 and v 5 for the clustering .

4.2 Place-cell firing sequences in the hippocampus

We now demonstrate that our method enables an automatic detection of firing sequences of rat hippocampal neurons during spatial exploration [86]. Our method

extracted 60 distinct clusters of data segments (i.e., time windows), each of which appeared repeatedly at a different location in the maze and in a specific movement direction (Figure 4.1A). We introduced three labels (Go, Back, Stop) for categorizing these clusters in terms of their dominant relationships to behavior, allowing each cluster to have multiple behavioral labels (see subsection 3.2.15 for details). Twenty clusters appeared primarily when the rat ran from the start to the goal (we call it the Go cluster), 20 clusters appeared primarily during the opposite movement (called the Back cluster). There are five overlapping clusters between the two types (#6, #10, #21, #40, #48). Another 20 clusters mainly appeared during immobility (Stop cluster). The relationship between each cluster and a behavioral state indicates that our method successfully detected behaviorally relevant cell assemblies, which likely consist of hippocampal place cells. Some clusters (e.g., clusters #4 and #10) were detected during both locomotion and the resting state. These patterns presumably correspond to place-cell firing phase-locked to theta oscillation and their ripple-associated replays, respectively [35, 89]. It is notable that our method automatically extracted these sequences in spite of the different time scale. Such reactivation was also observed in [86]. Figure 4.1B shows visualization of the feature space with t-SNE, which defines a mapping from high-dimensional data space to a low-dimensional space for visualization such that the spatial relationships between data points are optimally preserved [73]. We constructed the core temporal structure, which termed profile, of highly variable activity patterns of each cell assembly (see 3 for details). Figure 4.1C displays four sample profiles of activity patterns for the clusters detected, where only the first 10 neurons are shown. Three examples (#4, #16, #53) show clusters each of which was observed just once along the maze (see the rightmost color diagram) whereas cluster #10 appeared at two slightly different places during movements in both directions.

Figure 4.3 shows four examples of spike rasters from the extracted cell assemblies corresponding to two clusters (cluster 4 and cluster 10) together with the

position and velocity of the animal. In each cluster, the spatiotemporal activity patterns vary from segment to segment, but they also resemble each other (see Figure AD for the statistics of within-cluster and between-cluster similarity of activity patterns). Thus, our method is robust against changes in the temporal scale of sequences. In addition, each of the two clusters include an example of replays (at 1479 s in cluster 4 and at 1868 s in cluster 10) of a cell assembly in the immobile state of the animal. For the parameter values used here, these sequences were grouped into the same cluster because our method allows a certain degree of spike timing jitters. The larger the value of α , the more strict the penalty for spike timing jitters. We quantitatively evaluated clustering performance at different values of α to find that both the number of clusters (N_c) and the total number of time windows (N_{tw}) in the clusters decreased as the value of α was increased: at $\alpha = 0.1$, $N_c = 58$ and $N_{tw} = 21162$; at $\alpha = 1.0$, $N_c = 46$ and $N_{tw} = 17858$; at $\alpha = 10.0$, $N_c = 33$ and $N_{tw} = 8028$.

Figure 4.4A shows the receptive fields of neurons (top panels) and clusters (bottom panels) when the rat was running forward, backward and stopping with kernel density estimation [100, 111]. It is suggested that a cluster detected at a given spatial location consists of neurons having similar receptive fields around the location. To examine whether the detected sequences have significant relationships to behavior, we generated shuffled neural data in which spikes of each neuron were redistributed at randomly chosen temporal locations according to a homogeneous Poisson process. This manipulation preserved the average firing rates of individual neurons. Edit similarity score for cell-assembly sequences at some locations was significantly higher than the score calculated from the shuffled data (Figure 4.4B) (see Bayesian modeling section in chapter 3). The result suggests that the cell-assembly sequences and their profiles actually captured the behaviorally relevant characteristics of neural population activity. Chapter 3 explains the details of the statistical model used for the analysis.

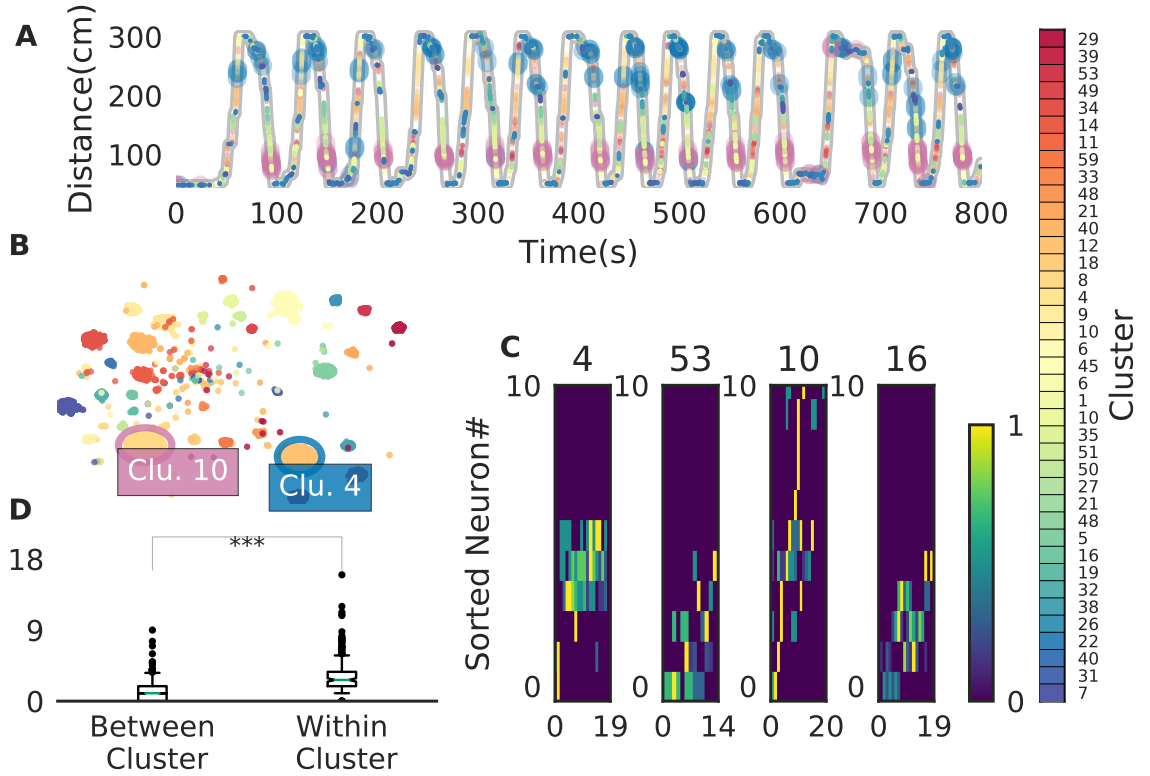


Figure 4.2: Cell assemblies extracted from hippocampal CA1. (A) The spatial locations in the linear maze are shown for the detected cell-assembly sequences. The x-axis shows time and y-axis shows the position of the rat. Twenty Go clusters and 20 Back clusters of time windows are shown, and the spatial positions at which they were detected were indicated by the rightmost color diagram. Each colored region shows where each segment was observed. Cluster four and ten were additionally colored in blue and pink. Cluster indices are shown on the right, which were sorted and colored according to the order of appearance during the going and returning along the maze. We sorted cluster indices according to their mean kernel density estimate (which is shown in Figure 4.4). The window size was 100ms. (B) t-SNE visualization of the feature space. Note that most of the neighboring colors in A are also adjacent in B, suggesting that two neuronal activities observed at contiguous spatial positions have similar temporal patterns, but are still separate enough in the feature space. (C) Profiles of cell-assembly sequences are shown for four cluster (left four panels). The first 10 neurons for four profiles are shown with number indicating cluster identity (Profile 4: [41, 0, 85, 53, 59, 68, 76, 5, 54, 4], Profile 53: [4, 15, 68, 49, 26, 84, 77, 12, 82, 13], Profile 10: [26, 54, 84, 17, 76, 68, 60, 36, 15, 0], Profile 16: [82, 39, 83, 61, 48, 13, 77, 28, 26, 85]). Color indicates the firing rate of each neuron after a normalization across neurons within the profile: from lowest (dark blue) to highest (yellow). The cells (y-axis) were sorted according to the relative temporal order note that the first ten neurons represent different firing order in the different profiles. (x-axis) of the peak activity of each cell in each profile. Note that the absolute length of the x-axis in each profile does not necessarily represent the actual temporal length of sequences, though the approximate length coincides the width of temporal windows (100ms in this case). (D) Between-cluster and within-cluster edit similarity scores. Edit similarity was compared between time windows belonging to the same cluster and those belonging to different clusters. Box plot with whiskers from 5 percentile to 95 percentile are shown with outliers (filled circles).

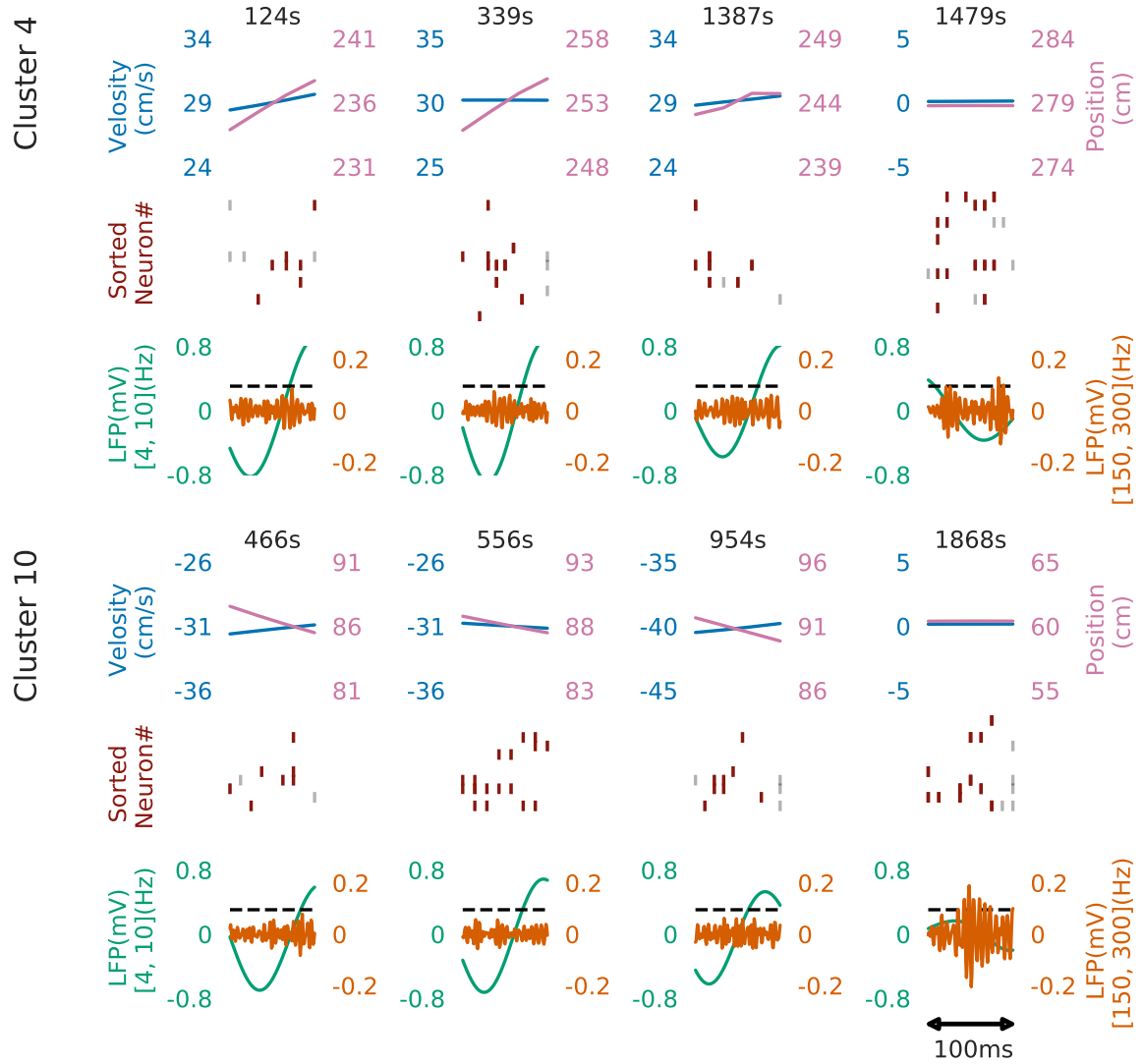


Figure 4.3: Cell assemblies detected from CA1. Four examples are shown from cluster 4 (top) and cluster 10 (bottom). The top, middle and bottom panels display the velocity and spatial position of the rat, spike raster, and local field potentials band-passed at 4-10Hz (theta band) and 150-200Hz (sharp-wave ripples). We calculated criteria (0.095) of the ripple detection with the method described in [28] to confirm our detected patterns during immobility accompanied with sharp wave ripple. In the middle panel, gray vertical bars show noisy spikes and red bars represent the core spikes of the corresponding profile. Neurons are sorted according to their firing position within the average profile.

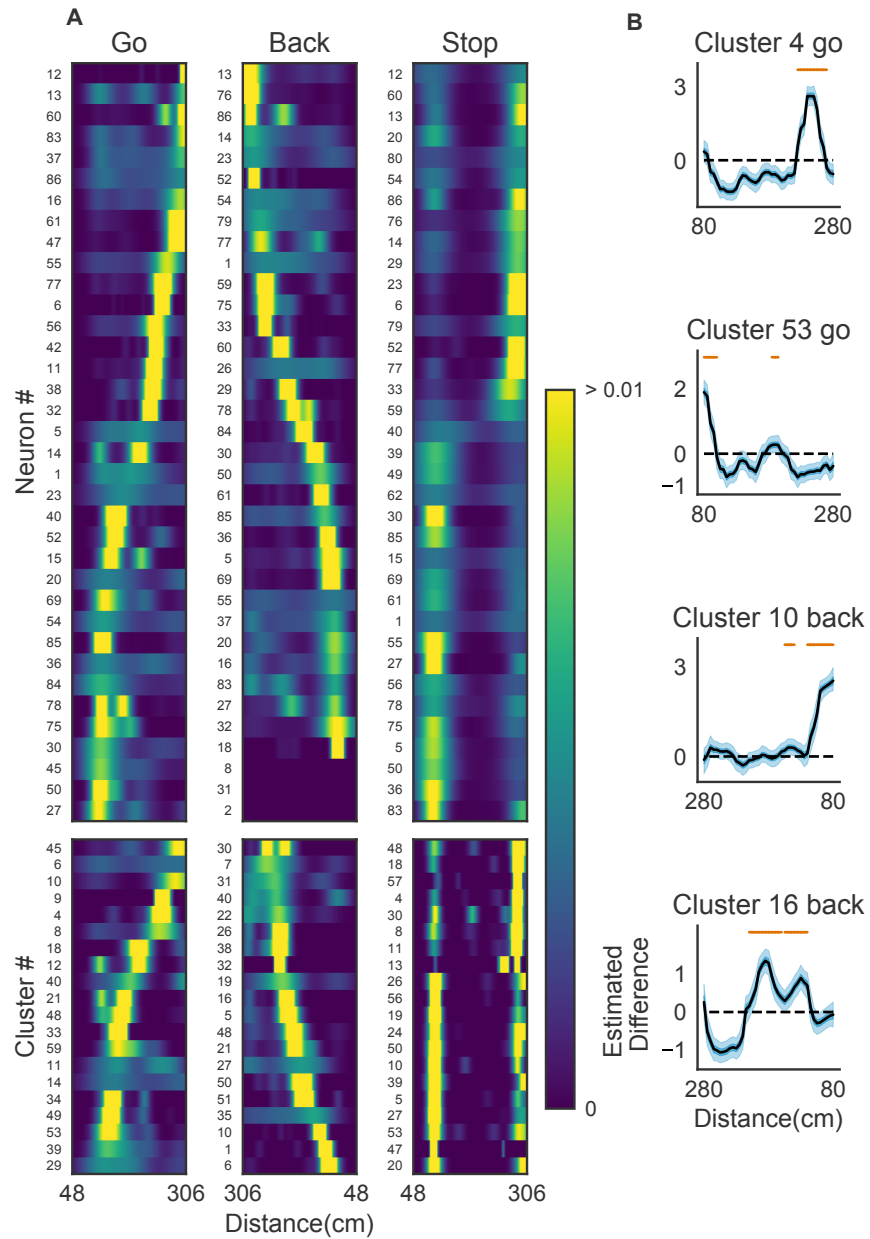


Figure 4.4: The relationships between hippocampal cell-assembly sequences. (A) The spatial receptive fields are shown for 36 hippocampal neurons (top) and cell assemblies belonging to the 20 clusters (bottom). The pseudo-color code indicates the probability of firing. (B) Estimated edit similarity difference between the original and shuffled data. Each similarity scores were calculated by using the profiles of cell assemblies and spike trains of hippocampal neural population using a sliding time window. Solid lines correspond to the mean of the estimated differences and blue shaded regions to 95% credible interval. Orange bars designate the spatial locations at which the lower bound of the credible interval is positive.

4.3 Cell assemblies in the prefrontal cortex

We further validated the method in neural ensemble activity recorded from the medial prefrontal cortex of rats performing a memory-guided spatial sequence task [33]: see the paper for experimental details). Briefly, the rats were trained to visit eight locations equally spaced around the perimeter of a circular arena in a prescribed sequential order with electrical brain stimulation as a reward. Our method detected 11 clusters of prefrontal cell-assembly sequences in total (Figure 4.5A). The previous analysis based on template matching revealed a sequence and its replay pattern in the same rat as we analyzed here [33]. Though some of the detected clusters are overlapped, the larger number of detected clusters indicate that the method extracted activity patterns without any reference to events or positions on the track. These clusters were detected in both behaving state and sleep state, and some clusters were frequently replayed during sleep (Figure 4.5A). During the behavior, cell assemblies were typically found when the rats were approaching or leaving a reward zone (Figure 4.5B). The profiles of three cell assemblies are shown in Figure 4.5C. Each sequence usually appeared just once in a 250 ms window during behaving state, whereas they were repeated multiple times during sleep state (Figure 4.5D). Thus, the detected sequences were time compressed during sleep. These results are consistent with the previous findings [33].

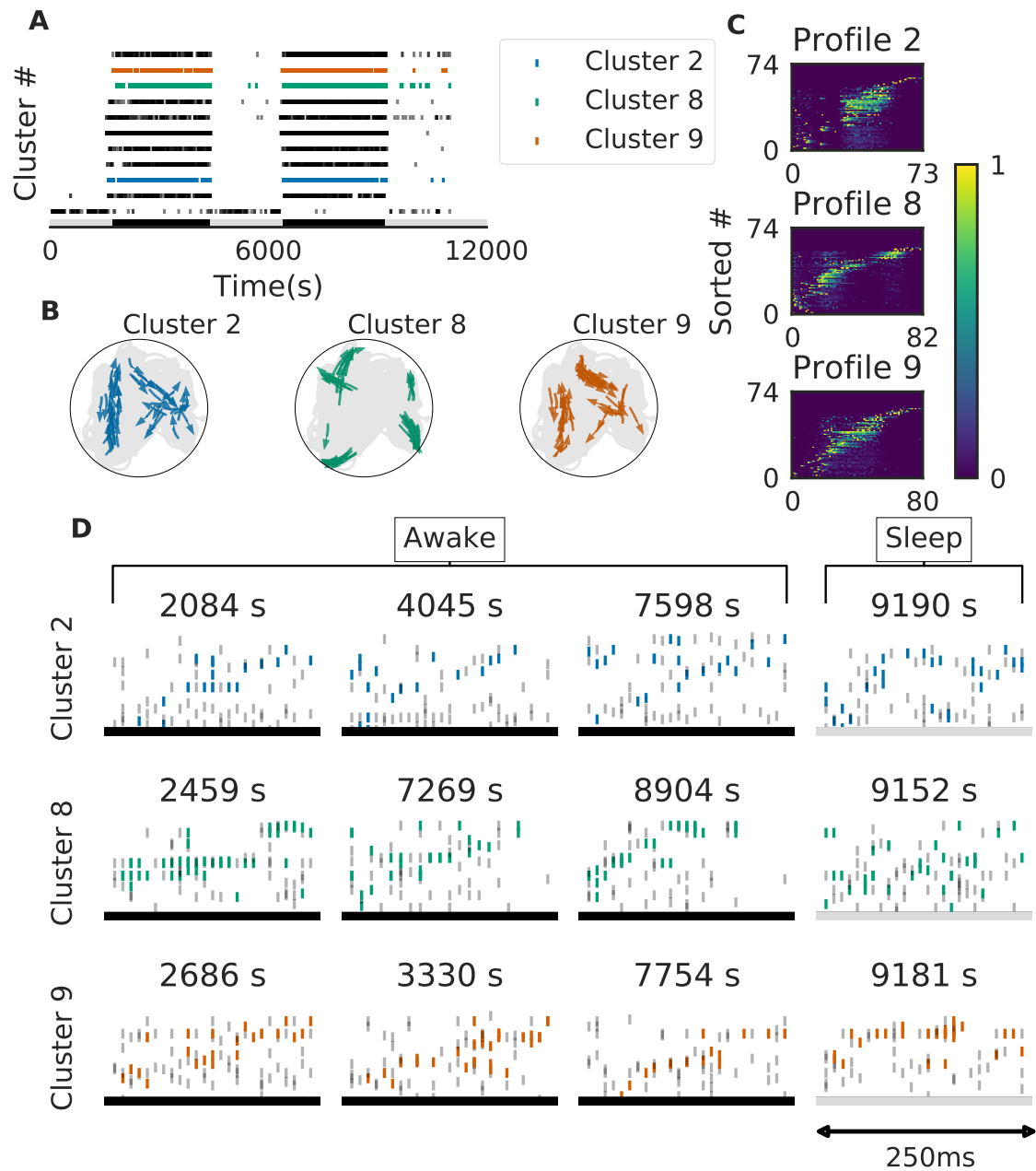


Figure 4.5: Cell assemblies detected from the prefrontal cortex. The width of time windows was 250ms. (A) The onset times of detected time windows are shown for all the clusters. (B) The spatial positions and movement directions of a rat are shown at the onset times of detected time windows belonging to three clusters by arrows. Only ten percent of randomly sampled elements are drawn for visualization. (C) Profiles are shown for three prefrontal cell assemblies in terms of the sorted neuron id and relative temporal order. The approximate length of the x-axis coincides the width of temporal windows (250 ms). (D) Cell assembly sequences detected in awake (left three panels) and sleep (rightmost panel) are shown for the three clusters. From top to bottom, each row corresponds to the profile 2, 8 and 9, respectively. Some sleep replay events showed evidence of multiple replays within the 250 ms window. This is most apparent in the first row, where the upward ramp is seen twice.

4.4 Anxiety coding cell-assembly sequence in basolateral amygdala

Lastly, we applied the proposed method to neural ensemble activity data recorded from the left and right basolateral amygdala of rats as the rats performed an aversive stimulation task [38].

It is believed that the amygdala stores the emotional components of experience, but the coding scheme of the neuronal population of the region remains unknown. We investigated the coding scheme by using a publicly available dataset that contains multiple single-unit recordings from the basolateral amygdala of rats as the rats learned the relationship between an aversive air puff and its location on a linear track, followed by a post-run test session without the stimulus. We found cell-assembly sequences that were activated when the aversive stimulus was applied by using the edit similarity-based method for cell-assembly sequences. Notably, such patterns were reactivated during the post-run session, and the reactivation was anti-correlated to the animals' running acceleration, suggesting that the pattern encoded the "anxiety" of the animals. Additionally, pattern reactivation was stronger along the run direction that involved the air puff than that in the "safe" direction. These results suggest that the observed replay may influence the processes of aversive memory consolidation and maintenance.

Distributed representation of a single event is beneficial because the number of possible patterns grows exponentially as the number of neurons increases, as has been discussed widely in the fields of theoretical neuroscience and machine learning [50, 85]. In such a coding scheme, the time lag between the spikes of different neurons is crucial in biological systems because it regulates synaptic connections between neurons via spike-timing-dependent plasticity (STDP); pre-synaptic activation followed by post-synaptic activation results in synaptic potentiation and vice-versa [80]. Hence, recurring sequences may contribute toward the consolida-

tion and maintenance of memories.

Several experimental results pertaining to various regions of the brain support this hypothesis. Cell-assembly sequences of cortical neurons play active roles in primates [1, 46, 124]. Place cells in the rat hippocampus exhibit precisely timed, repetitive firing sequences representing the behavioral trajectory of rats, subsections of which are repeated during each theta cycle of local field potential in the region [83, 95, 137]. These sequences are replayed over compressed temporal scales during awake immobile and sleep states [16, 20, 35, 65], presumably for memory consolidation [38, 54]. In the rat somatosensory and auditory cortices, spontaneous and stimulus-evoked activities exhibit repetitive sequences of neuronal firing [70, 71]. Similar replay events have been observed in the rat prefrontal cortex as well [33].

It is believed that the emotional components of an experience are processed by the amygdala [38, 52, 79, 98, 103, 134, 149], but the collective temporal coding protocol in the region has not been elucidated thus far. We hypothesized that contextual emotional experiences are represented in the form of temporal coding. To test the hypothesis, we used a publicly available dataset that contains recordings from rat basolateral amygdala [38].

We evaluated the profiles of $n = 4$ animals over 17 sessions of 500-ms neuronal activities recorded just after the timing of air puff application. Example profiles, templates, and neuronal activity at the timings are shown in Figures 4.6, 4.8, and 4.10. Thereafter, we considered the relationship between profile(/template)-related activity patterns and acceleration of the rats 4.7, 4.9, 4.11 and found that the profile-related activity patterns showed a strong negative correlation with the activity. This tendency was consistent across different sessions, as in Figure 4.12, which shows the data of all sessions.

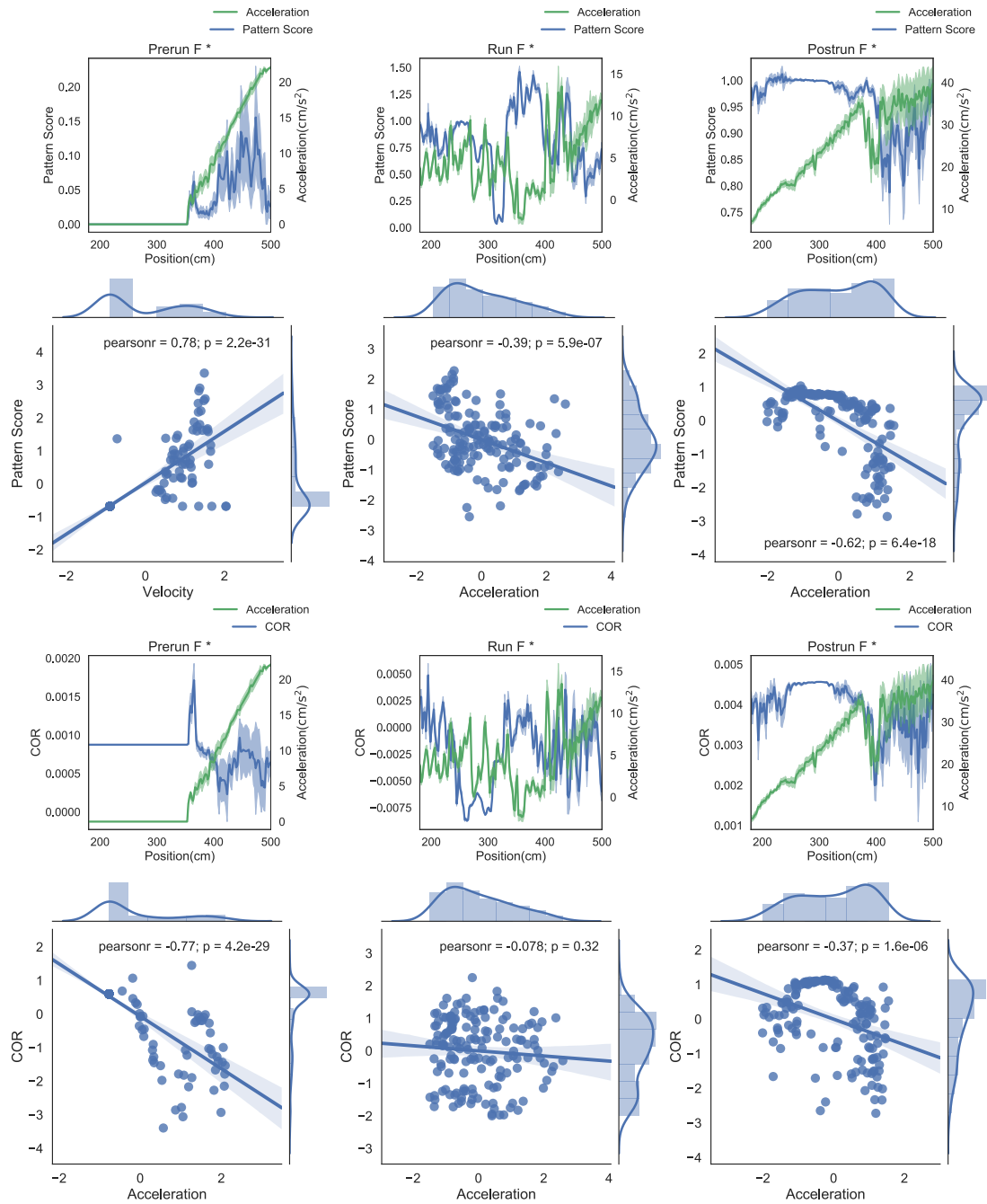


Figure 4.7: Relationship between correlation of neuronal activity versus patterns (profile/template) and acceleration of rat 1. Top half shows a comparison between profile and neuronal activity/movement of the rats, and bottom half shows the same comparison with the template.

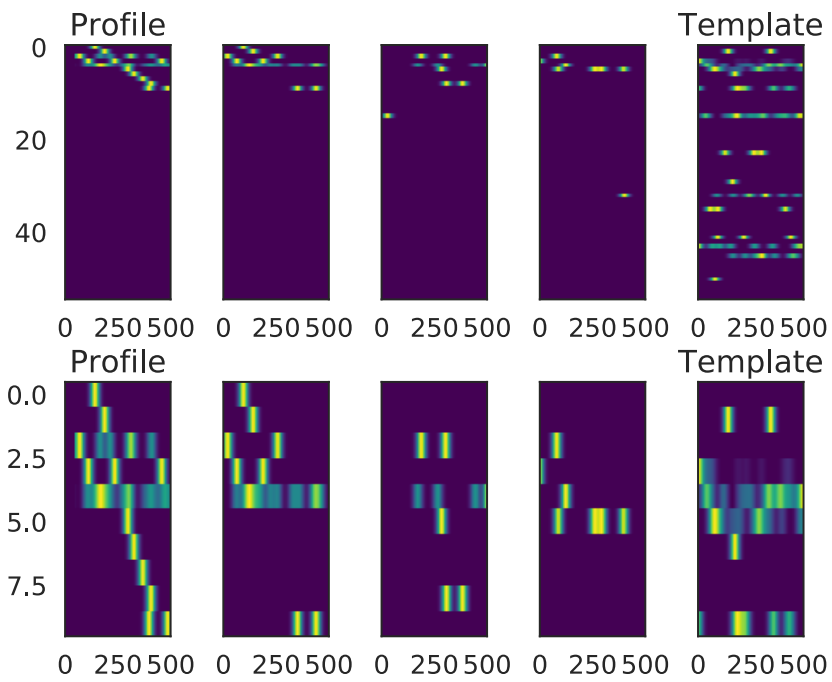


Figure 4.8: Profile and the template of an illustrative session 4.9 Top: The template and the profile are shown with three examples of 500-ms neuronal activity after the onset of air puff.

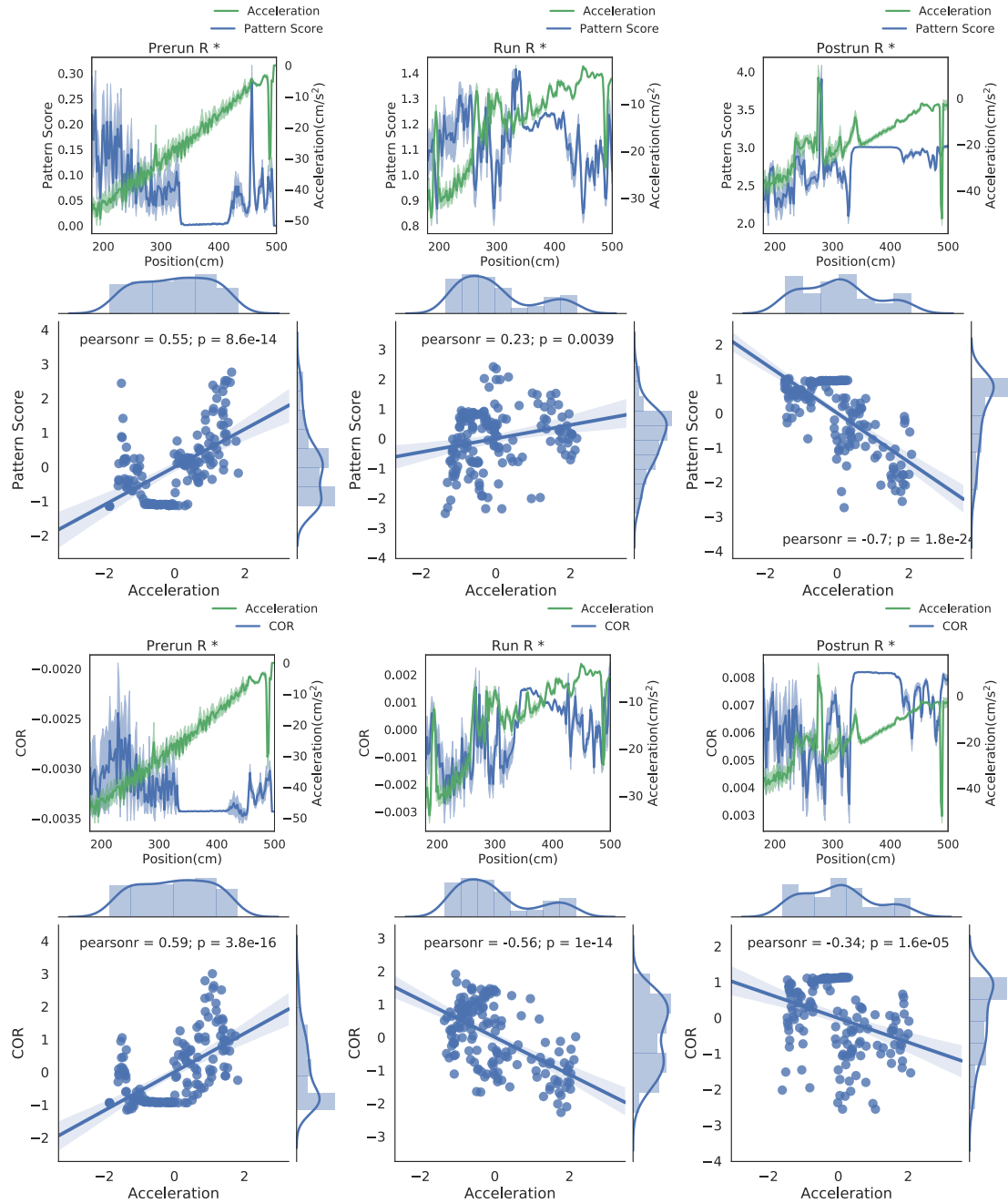


Figure 4.9: Relationship between correlation of neuronal activity versus patterns (profile/template) and acceleration of the rat 2. The top half shows a comparison between profile and neuronal activity/movement of the rats, and the bottom half shows the same comparison with the template.

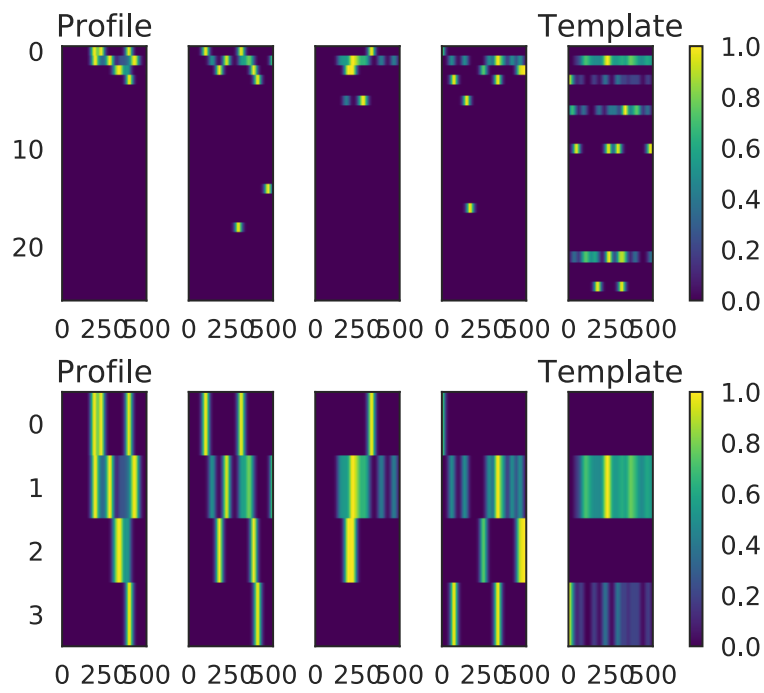


Figure 4.10: Profile and template of an illustrative session 4.11 Top: The template and the profile are shown with three examples of 500-ms neuronal activity after the onset of air puff.

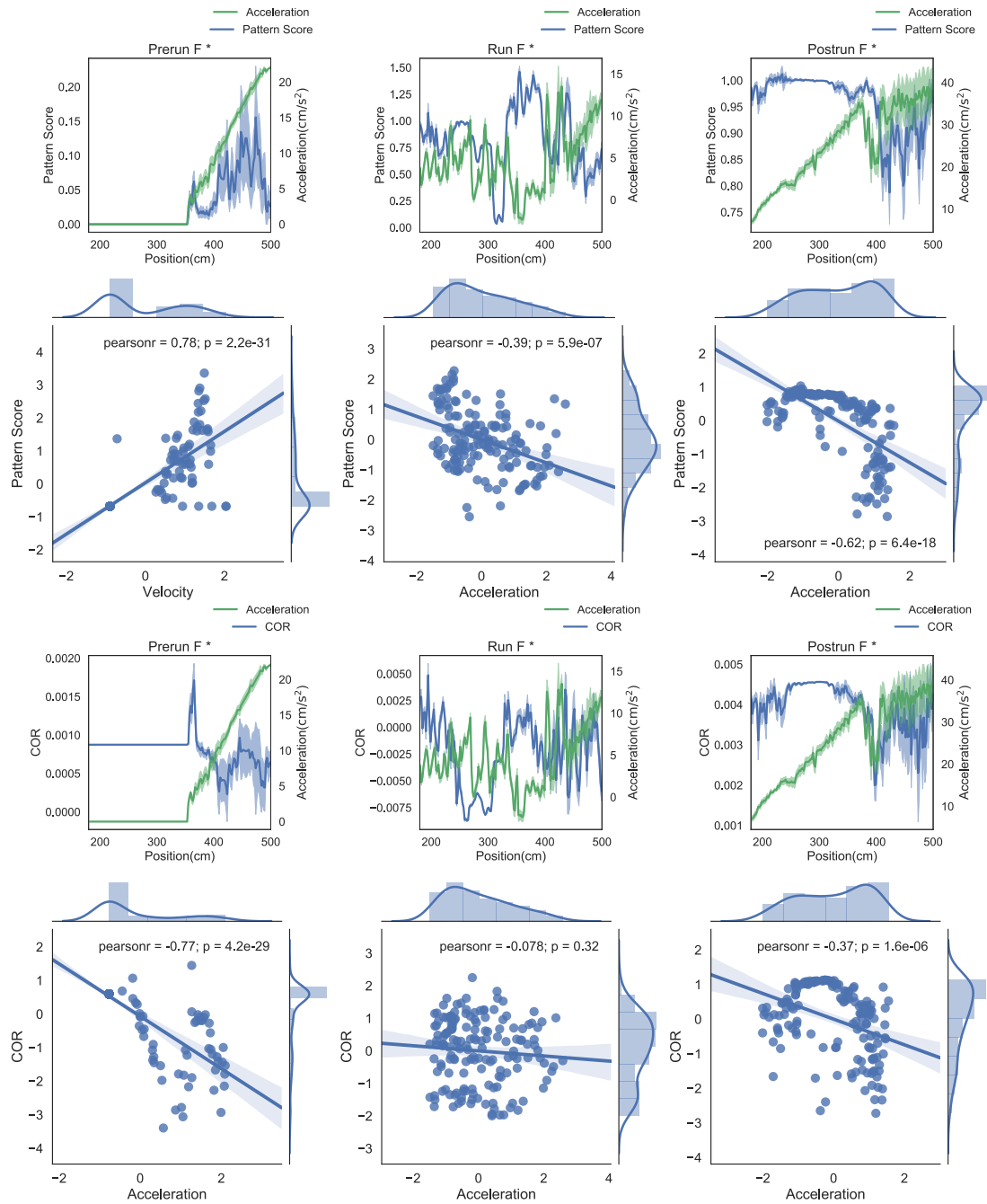


Figure 4.11: Relationship between correlation of neuronal activity versus patterns (profile/template) and acceleration of the rat 3. The top half shows a comparison between profile and neuronal activity/movement of the rats, and the bottom half shows the same comparison with the template.

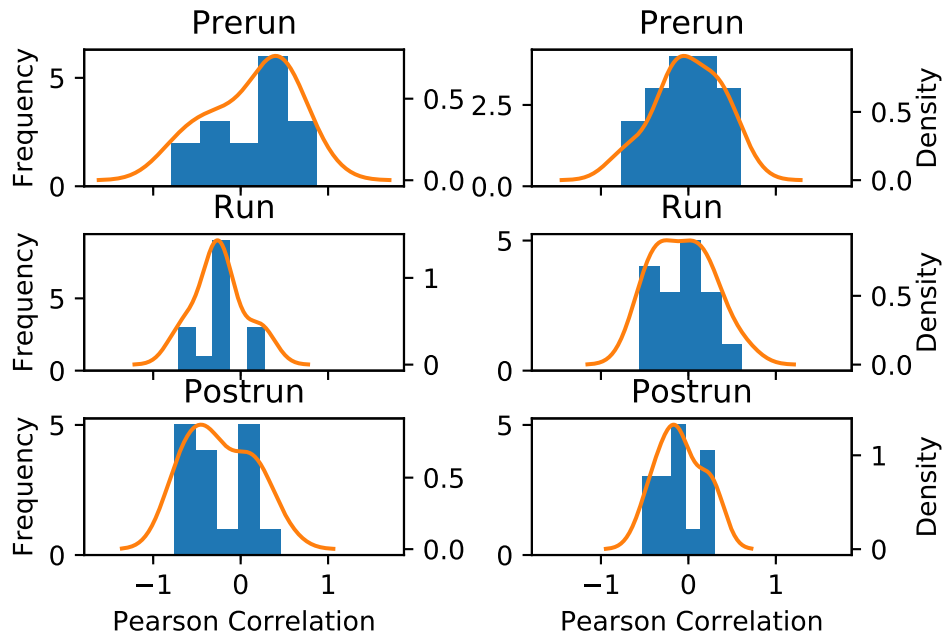


Figure 4.12: Pearson's r for all sessions Left: Pearson's correlation coefficient between the animal's acceleration and the similarity score calculated using the profile. Right: Comparison with the template did not show any strong correlation.

Chapter 5

DISCUSSION

In this thesis, we proposed a novel method for self-similar spatiotemporal activity pattern detection by using the extended edit similarity score. The method calculates the profile and common spatiotemporal patterns in a cluster of segments to extract typical firing sequences embedded in the data. This chapter discusses a few characteristics of the proposed framework and discusses possible extensions. Moreover, it presents a comparison of the proposed approach with that of [112].

5.1 Profile convergence

Because the profiling method is essential for inspecting the activity patterns of cell assemblies, we examined whether the method works robustly. For the applications reported in this study, $10x$ to $100x$ times update, where x is the number of members in the cluster to be profiled, maximized the average edit similarity score between the individual cluster members and the corresponding profile. Convergence of the profiling procedure for the experimental data is shown in Figure 5.1.

5.2 Comparison with a recent method for sequence detection

Recently, a statistical method to extract assembly structures with arbitrary constellations of time lags was proposed [112]. We compared our method with this method (the code is available at <https://github.com/DurstewitzLab/Cell-Assembly-Detection>). The recent method recursively combines neurons into larger sets based on significant statistical relationships between their activities. We first compared the performance of the two methods by applying them to artificial spike data in which a spike sequence of 10 neurons was embedded into background Poisson spike trains of 100 neurons (including the 10). The target sequence pattern occurred 60 times, and the background firing rate was 1 Hz. We generated two sets of 50 independent datasets: one with a sequence duration of 100 ms and the other with that of 500 ms. The length of each dataset was 60 s.

While our method robustly showed near-perfect detection, the performance of the recent method was contingent on the input sample. On some samples, the method worked almost perfectly, but on other samples, it worked rather poorly. In summary, the two methods worked equally well for the temporally shorter sequences of 100 ms (Figure 5.2A, left), but they exhibited significantly different performance for the temporally long sequences of 500 ms (Figure 5.2A, right). Specificity was significantly better in case of the recent method than in case of our method for both data lengths (left, $p = 0.001174$; right, $p = 4.353e-07$). This result suggests that the recent method is more conservative and it produces fewer false positives. However, the differences were subtle: 99% confidence interval was $(-0.0333, 0.0000)$ and $(-0.0166, 0.0000)$ for the lengths of 100 ms and 500 ms, respectively. In addition, we analyzed the spike data obtained from rat hippocampus (CRCNS.org., <http://dx.doi.org/10.6080/K09G5JRZ>), which highlighted another

difference between the two methods. As shown in Figure 5.2B, our method detected multiple cell assemblies that covered the entire linear track. In contrast, the recent method apparently failed to merge various subsequences into a small number of core sequences. This excessive division was presumably due to jitters and failures in spike generation. Although pruning solutions called "biggest" and "distance" were described in [112], the method could not avoid this challenge altogether at least in our analysis of real data.

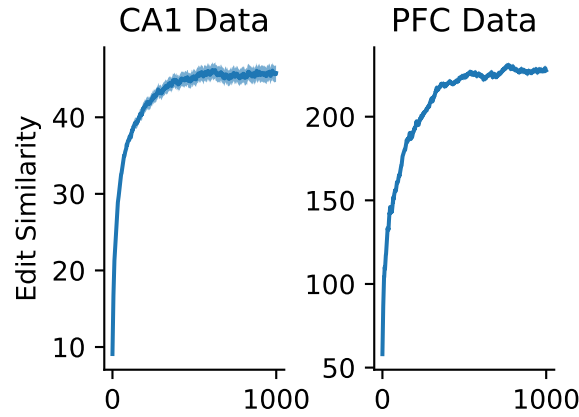


Figure 5.1: Convergence of profile evaluation in CA1 and HPC data analysis. Profiles were calculated for all clusters in the activity data from CA1 and PFC. The abscissa shows the number of iterations, and the ordinate shows edit similarity between the current profile and the preceding one. The thick lines represent the means over all clusters, and the shaded areas represent the standard errors.

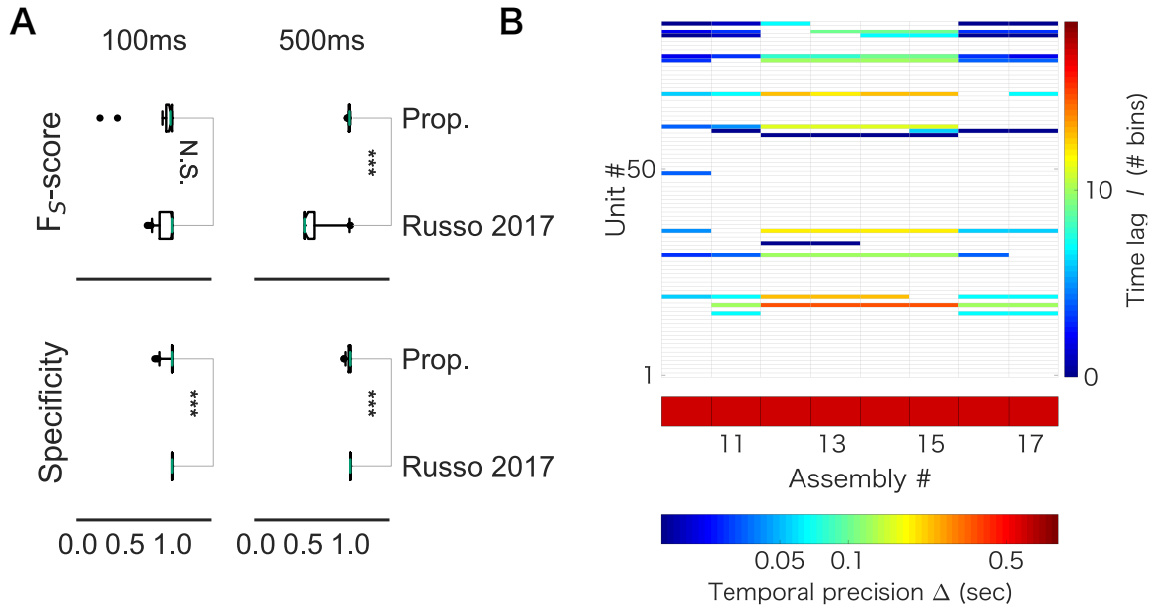


Figure 5.2: Comparison with Russo 2017 (A) F_5 -score (top) was significantly higher in case of the proposed method than in case of [112] for the artificial data of length 500 ms (right, p-value is $1.831e-12$); however, difference between the two methods was not significant for the artificial data of length 100 ms (left, p-value is 0.6386). (B) The output of Russo 2017 is shown. The distributed computer code of Russo 2017 analyzes spike train data with various temporal precisions (i.e., correlation time scales) ranging from milliseconds to several seconds. Only a part of the results is shown to clarify the characteristic property of Russo 2017: the eight cell assemblies classified as number 10 to number 17 look very similar to each other.

5.3 Requirements checklist

Four methods were considered for this thesis: template matching, PCA/ICA-based method, statistical significance based method, and the proposed edit similarity based method. This section reviews the methods and discusses their applicabilities in different situations. The overview of the discussion is shown in Table 5.1.

5.3.1 Template matching

By definition, the template matching is not applicable when no external behavioral variable is given. Also, as discussed in the section 4.4, it was not robust against variability of the onsets of the pattern.

5.3.2 PCA-/ICA-based method

Artificial data analysis in the section 4.1 revealed that these methods were not robust against background noise, in addition to the limitations of the detectable pattern discussed in [69].

5.3.3 Statistical significance-based method

Consistent with the previous reports [14, 45, 74], this method was not robust against pattern noise (c.f., section 5.2). In relation, the method was also suffered from a divergence of patterns where pattern noise existed.

5.3.4 The proposed method

(i) By definition, the method does not relies on external variable . (ii) Computational time remained in a practical range, thanks to the tips for the reduction of computation (c.f., subsection 3.2.11 and 3.2.12). (iii), (iv) Section 4.1 showed that the method is more robust against background noise than the PCA-/ICA-based method. That was also tested through real data application (c.f., chapter 4). (v) Section 4.1 showed that, the method could be robust against pattern duration. The robustness enabled the simultaneous detection of awake patterns and its replays during immobility/sleep (c.f., chapter 4) .

| Criteria | The template matching | The PCA-/ICA-based method | Statistical Significance based method | Proposed method |
|--|-----------------------|---------------------------|---------------------------------------|-----------------|
| (i) Without behavioral data | No | Yes | Yes | Yes |
| (ii) Realistic computational complexity | Yes | Yes | No | Yes |
| (iii) Robust against Background noise | Yes | No | No | Yes |
| (iv) Robust against Pattern noise | No | Yes | No | Yes |
| (v) Robust against Extension/Shrinkage of patterns | Yes | Yes | No | Yes |
| (vi) Shared neurons among different patterns | Yes | Yes and No | Yes | Yes |

Table 5.1: Requirements checklist for the cell assembly sequence detection. In this thesis, four methods: template matching, PCA-/ICA-based method, statistical significance based method, and the proposed edit similarity based method are considered.

5.4 Expansion of the framework

There are two remaining points to be improved in the proposed framework: further reduction of computational complexity and automatic hyperparameter tuning. This section describes two possible extensions of the method to mitigate this problem. In the first subsection, the artificial neuronal network is used to skip computation of the similarity matrix, which requires computational complexity proportional to the square of the number of segments. In the second part, a Gaussian process is used to model the goodness of the current parameter set.

5.4.1 Neural-network-based approach for edit-similarity-based feature learning

Although the Minhashing algorithm [24] helps reduce the computational burden of the proposed method significantly, calculation of the similarity matrix remains computationally intensive. A neural-network-based embedding approach could be one remedy for the issue. In machine learning, various types of artificial neural networks (ANNs) have been employed to model feature spaces of words [84, 85], sentences [67, 97], documents [64, 85], and so forth [25, 117, 144]. Moreover, several researchers have attempted to learn the similarity function by using ANNs. Based on these reports, we have developed an ANN model that “learns” the edit similarity function based on the Siamese network[88, 144, 147] with bi-directional[118] LSTM [47] cells [40](Figure 5.5).

We trained the network with the following data.

- Number of neurons 30
- Three different types of sequences appear 100
- 2.5 Hz noisy firing

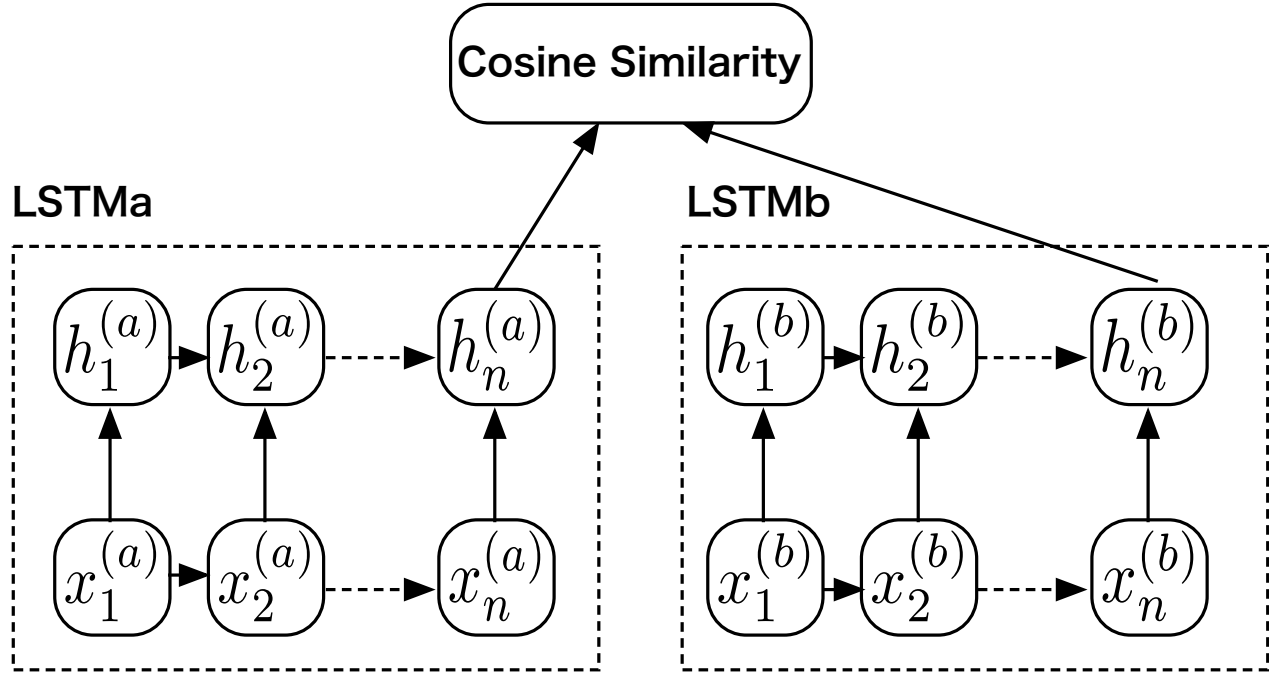


Figure 5.3: Network architecture.

Area Under the Receiver Operating Characteristic Curve (ROC AUC) [123] from the prediction scores is shown below.

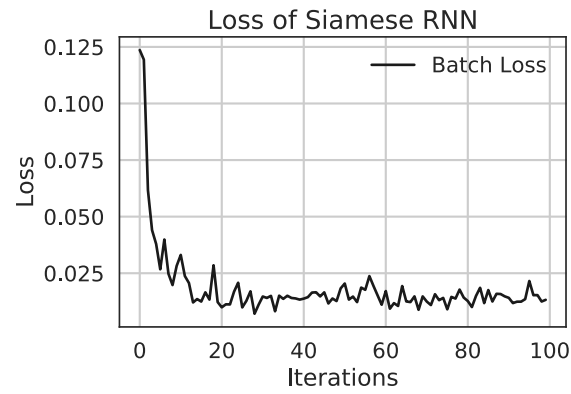


Figure 5.4: Training loss during learning. Example training iterations are shown.

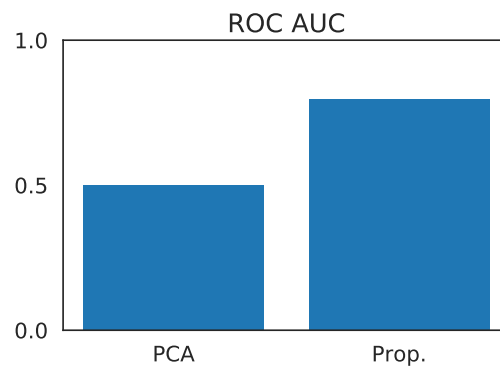


Figure 5.5: Performance of learned network prediction compared with PCA-based approach.

5.4.2 Gaussian Process Regression to find optimal hyper parameters

Another problem associated with the method developed herein is the lack of any warranty that the hyperparameter α would be optimal. One possible solution for the optimization problem is the use of a Gaussian process [10, 108]. Gaussian processes provide a probabilistic approach to learning in kernel machines. Therefore, the use of a Gaussian process regression model that predicts animal behavior from the similarity matrix would solve the problem and provide additional information about the importance of the activity for the behavior. The hyperparameter can be optimized using a stochastic gradient descent [109], L-BFGS [4] or a scaled conjugate gradient [87].

Chapter 6

Conclusion

In this study, we developed a method for extracting multiple repeated sequences of cell assemblies from multi-neuron activity data. The proposed method is based on edit similarity, an index used in computer science to measure similarity between strings. Edit similarity compares the serial order of common elements appearing in two strings with or without discounting variations in inter-element intervals. Therefore, it is a flexible and efficient metric for comparing highly noisy spatiotemporal activity patterns of cell assemblies. We validated the method first by using artificial data and then by using neural activity data recorded from the hippocampus and the prefrontal cortex of behaving rats.

6.1 Summary of Thesis Achievements

In the assessment with artificial data, we showed the superiority of our method over PCA- [68, 102] and ICA-based methods [63, 69] in detecting an assembly of synchronously firing cells when the cell-assembly structure is clear (i.e., small timing jitters of ± 10 ms). However, when timing jitters were large (i.e., ± 50 ms), our method tended to categorize such a cell assembly into multiple clusters, and

its performance was lower relative to that of PCA/ICA-based methods. However, this does not suggest that the proposed method is weak because it is constructed as such: the method is specialized for sequence detection. Dynamic programming-based methods have been proposed in the literature to quantify the similarity between spike trains of neurons[135, 136]. To our knowledge, no method has been developed on the basis of edit similarity to detect similar sequences of cell assemblies from noisy population neural data in an unsupervised manner. Other methods exist, and they can detect the activation of specific neuron ensembles [21, 65, 94]. These methods, however, are generally not effective when the data has a low signal-to-noise ratio, for instance, when most of the recorded neurons do not participate in sequences. In addition, with the previous methods, it is difficult to distinguish partially overlapping cell assemblies.

Specifically, our method enables blind detection of cell-assembly sequences without referring to external events, such as sensory stimuli and behavioral responses. Recently, a novel statistical approach was proposed for the detection of cell-assembly structures with multiple timescales [112]. Starting from pairwise correlations in neuron pairs, the method finds significantly correlated neurons within the set of cell assemblies detected in the previous step. The analysis was accelerated by discarding statistically less significant combinations in the next step. However, in successive statistical tests, detection of long sequences with that method proved to be rare and time consuming. By contrast, the proposed method is computationally more efficient when searching longer cell-assembly sequences. It may also be inappropriate to discard long sequences simply because they are statistically less significant. For instance, place-cell sequences spanning several seconds of behavior emerge in the hippocampus during spontaneous activity after spatial experience [30, 44]. We propose that behaviorally relevant cell-assembly sequences should be addressed after all possible candidates have been identified. Our method enables such an analysis of cell-assembly sequences.

We note that the three data examples analyzed herein [33, 38, 86] were recorded during stereotyped, repeated behaviors, which presumably entrained similar repeated patterns in neural activity. Whether the present algorithm can be used to detect spontaneous (as opposed to stimulus- or activity-driven) patterns, such as those reported by [70, 71], remains to be tested. Other intriguing extensions of this method include the detection of hierarchically organized cell assemblies over multiple spatiotemporal scales. Such an extension requires flexible online tuning of time windows, which is currently a challenge. One area where time-scaling would be especially relevant is in the detection of replay events, which often occur over a compressed timescale during slow-wave sleep. Our method might detect a considerably greater number of reactivation events if we adjust the temporal scaling between behavior and sleep epochs. Moreover, in principle, our method can be applied to optical imaging data if we adequately tune the sizes of the time window and the temporal discount factor.

In sum, we proposed a novel method for blind detection of cell-assembly sequences based on the edit similarity score and an exponential discount for timing jitters. This method does not rely on external references, and therefore, it is useful for detecting not only externally driven firing sequences but also internally driven sequences that emerge from arbitrary mental procedures.

Bibliography

- [1] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia. Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of neurophysiology*, 70(4):1629–1638, October 1993.
- [2] C. Allen and C.F. Stevens. An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, 91(22):10380–10383, October 1994.
- [3] O. Alvarez, C. Gonzalez, and R. Latorre. COUNTING CHANNELS: A TUTORIAL GUIDE ON ION CHANNEL FLUCTUATION ANALYSIS. *Advances in Physiology Education*, 26(4):327–341, December 2002.
- [4] G. Andrew and J. Gao. Scalable training of L1-regularized log-linear models. *ICML*, 2007.
- [5] M. Ankerst, M.M. Breunig, H.P. Kriegel, and J. Sander. OPTICS: Ordering points to indentify the clustering structure. *ACM Sigmod Record*, 28(2):49–60, June 1999.
- [6] J. Artiles, J. Gonzalo, and S. Sekine. The semEval-2007 WePS evaluation. In *Proceedings of the 4th International Workshop on Semantic Evaluations - SemEval '07*, pages 64–69, Morristown, NJ, USA, 2007.
- [7] J. Bahlmann, T.C. Gunter, and A. Friederici. Hierarchical and linear sequence

- processing: An electrophysiological exploration of two different grammar types. *Journal of Cognitive Neuroscience*, 18:1829–1942, 2006.
- [8] G.J. Barton and M.J. Sternberg. A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons. *Journal of molecular biology*, 198(2):327–337, November 1987.
- [9] K. Berlin, S. Koren, C.S. Chin, J.P. Drake, J.M. Landolin, and A.M. Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature Biotechnology*, 33(6):623–630, June 2015.
- [10] C.M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.
- [11] J.M. Bland and D.G. Altman. Multiple significance tests: the Bonferroni method. *BMJ*, 310(6973):170–170, January 1995.
- [12] A. Broder. *Original implementation of MINhash. On the Resemblance and Containment of Documents*. IEEE Computer Society, June 1997.
- [13] A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher. Min-Wise Independent Permutations. *Journal of Computer and System Sciences*, 60(3): 630–659, June 2000.
- [14] C.D. Brody. Correlations Without Synchrony. *Neural Computation*, 11(7): 1537–1551, March 2006.
- [15] G. Buzsáki. Large-scale recording of neuronal ensembles. *Nature neuroscience*, 7(5):446–451, May 2004.
- [16] G. Buzsáki and E.I. Moser. Memory, navigation and theta rhythm in the hippocampal-entorhinal system. *Nature neuroscience*, 16(2):130–138, February 2013.

- [17] G. Buzsáki, E. Stark, A. Berényi, D. Khodagholy, D.R. Kipke, E. Yoon, and K.D. Wise. Tools for probing local circuits: high-density silicon probes combined with optogenetics. 86(1):92–105, April 2015.
- [18] R.C. Cannon, C. O'Donnell, and M.F. Nolan. Stochastic Ion Channel Gating in Dendritic Neurons: Morphology Dependence and Probabilistic Synaptic Activation of Dendritic Spikes. *PLOS Computational Biology*, 6(8):e1000886, August 2010.
- [19] B. Carpenter, A. Gelman, M.D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A Probabilistic Programming Language. *Journal of statistical software*, 76(1), 2017.
- [20] M.F. Carr, S.P. Jadhav, and L.M. Frank. Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nature neuroscience*, 14(2):147–153, February 2011.
- [21] Z. Chen and M.A. Wilson. Deciphering neural codes of memory during sleep. *Trends in neurosciences*, 40(5):260–275, May 2017.
- [22] O. Chum, J. Philbin, M. Isard, and A. Zisserman. *Scalable near identical image and shot detection*. ACM, New York, New York, USA, July 2007.
- [23] B.A. Clegg, G.J. DiGirolamo, and S.W. Keele. Sequence learning. *Trends in Cognitive Sciences*, 2(8):275–281, August 1998.
- [24] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, and C. Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering, IEEE Transactions on*, 13(1):64–78, 2001.
- [25] P. Covington, J. Adams, and E. Sargin. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, pages 191–198, New York, New York, USA, 2016.

- [26] Y. Cui, S. Ahmad, and J. Hawkins. Continuous Online Sequence Learning with an Unsupervised Neural Network Model. *Neural Computation*, 2016.
- [27] A.S. Das, M. Datar, A. Garg, and S. Rajaram. *Google news personalization: scalable online collaborative filtering*. scalable online collaborative filtering. ACM, New York, New York, USA, May 2007.
- [28] T.J. Davidson, F. Kloosterman, and M.A. Wilson. Hippocampal Replay of Extended Experience. *Neuron*, 63(4):497–507, August 2009.
- [29] T. Deneux, A. Kaszas, G. Szalay, G. Katona, T. Lakner, A. Grinvald, B. Rózsa, and I. Vanzetta. Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations in vivo. *Nature Communications*, 7:12190, July 2016.
- [30] G. Dragoi and S. Tonegawa. Distinct preplay of multiple novel spatial experiences in the rat. *Proceedings of the National Academy of Sciences of the United States of America*, 110(22):9100–9105, May 2013.
- [31] G.T. Einevoll, F. Franke, E. Hagen, C. Pouzat, and K.D. Harris. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Current Opinion in Neurobiology*, 22(1):11–17, February 2012.
- [32] V. Emiliani, A.E. Cohen, K. Deisseroth, and M. Häusser. All-optical interrogation of neural circuits. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 35(41):13917–13926, October 2015.
- [33] D.R. Euston, M. Tatsuno, and B.L. McNaughton. Fast-forward playback of recent memory sequences in prefrontal cortex during sleep. *Science (New York, N.Y.)*, 318(5853):1147–1150, November 2007.
- [34] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968.

- [35] D.J. Foster and M.A. Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, March 2006.
- [36] A.P. Georgopoulos, A.B. Schwartz, and R.E. Kettner. Neuronal population coding of movement direction. *Science (New York, N.Y.)*, 233(4771):1416–1419, September 1986.
- [37] R. Giegerich and S. Kurtz. From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction. *Algorithmica*, 19(3): 331–353, November 1997.
- [38] G. Girardeau, K. Benchenane, S.I. Wiener, G. Buzsáki, and M.B. Zugaro. Selective suppression of hippocampal ripples impairs spatial memory. *Nature neuroscience*, 12(10):1222–1223, October 2009.
- [39] Gotoh, O. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708, December 1982.
- [40] A. Graves, S. Fernández, and J. Schmidhuber. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, pages 799–804. Springer, Berlin, Heidelberg, September 2005.
- [41] S. Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, October 2010.
- [42] A. Grinvald and C.C.H. Petersen. Imaging the dynamics of neocortical population activity in behaving and freely moving mammals. In *Membrane potential imaging in the nervous system and heart*, pages 273–296. Springer International Publishing, Cham, 2015.
- [43] K. Grochow, S.L. Martin, A. Hertzmann, Z. Popović, K. Grochow, S.L. Mar-

- tin, and Z. Popović. Style-based inverse kinematics. *ACM Transactions on Graphics (TOG)*, 23(3):522–531, August 2004.
- [44] A.D. Grosmark and G. Buzsáki. Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences. *Science (New York, N.Y.)*, 351(6280):1440–1443, March 2016.
- [45] L. Grossberger, F.P. Battaglia, and M. Vinck. Unsupervised clustering of temporal patterns in high-dimensional neuronal ensembles using a novel dissimilarity measure. *PLOS Computational Biology*, 14(7):e1006283, July 2018.
- [46] N.G. Hatsopoulos, C.L. Ojakangas, L. Paninski, and J.P. Donoghue. Information about movement direction obtained from synchronous activity of motor cortical neurons. *Proceedings of the National Academy of Sciences*, 95(26):15706–15711, December 1998.
- [47] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *dx.doi.org*, 9(8):1735–1780, March 2006.
- [48] M.D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *arXiv.org*, November 2011.
- [49] W.K. Hon, T.W. Lam, K. Sadakane, and W.K. Sung. Constructing Compressed Suffix Arrays with Large Alphabets. In *Algorithms and Computation*, pages 240–249. Springer, Berlin, Heidelberg, Berlin, Heidelberg, December 2003.
- [50] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, April 1982.
- [51] S.A. Huettel, P.B. Mack, and G. McCarthy. Perceiving patterns in random series: dynamic processing of sequence in prefrontal cortex. *Nature neuroscience*, 5(5):485–490, May 2002.

- [52] Y. Ikegaya, H. Saito, and K. Abe. Attenuated hippocampal long-term potentiation in basolateral amygdala-lesioned rats. *Brain Research*, 656(1):157 – 164, 1994.
- [53] Y. Ikegaya, G. Aaron, R. Cossart, D. Aronov, I. Lampl, D. Ferster, and R. Yuste. Synfire chains and cortical songs: temporal modules of cortical activity. *Science (New York, N.Y.)*, 304(5670):559–564, April 2004.
- [54] S.P. Jadhav, C. Kemere, P.W. German, and L.M. Frank. Awake hippocampal sharp-wave ripples support spatial memory. *Science (New York, N.Y.)*, 336(6087):1454–1458, June 2012.
- [55] P. Janata and S.T. Grafton. Swinging in the brain: shared neural substrates for behaviors related to sequencing and music. *Nature neuroscience*, 6(7): 682–687, July 2003.
- [56] J.J. Jun, N.A. Steinmetz, J.H. Siegle, D.J. Denman, M. Bauza, B. Barbarits, A.K. Lee, C.A. Anastassiou, A. Andrei, Ç. Aydın, M. Barbic, T.J. Blanche, V. Bonin, J. Couto, B. Dutta, S.L. Gratiy, D.A. Gutnisky, M. Häusser, B. Karsh, P. Ledochowitsch, C.M. Lopez, C. Mitelut, S. Musa, M. Okun, M. Pachitariu, J. Putzeys, P.D. Rich, C. Rossant, W.L. Sun, K. Svoboda, M. Carandini, K.D. Harris, C. Koch, J. O’Keefe, and T.D. Harris. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, November 2017.
- [57] E. Kandel, T. Jessell, J. Schwartz, S. Siegelbaum, and A. Hudspeth. *Principles of Neural Science, Fifth Edition*. Principles of Neural Science. McGraw-Hill Education, 2013.
- [58] J.N.D. Kerr, D. Greenberg, and F. Helmchen. Imaging input and output of neocortical networks in vivo. *Proceedings of the National Academy of Sciences*, 102(39):14063–14068, September 2005.

- [59] T.H. Kim, Y. Zhang, J. Lecoq, J.C. Jung, J. Li, H. Zeng, C.M. Niell, and M.J. Schnitzer. Long-Term Optical Access to an Estimated One Million Neurons in the Live Mouse Cortex. *Cell reports*, 17(12):3385–3394, December 2016.
- [60] T. Knöpfel, Y. Gallero-Salas, and C. Song. Genetically encoded voltage indicators for large scale cortical imaging come of age. *Current opinion in chemical biology*, 27:75–83, August 2015.
- [61] S. Kullback and R.A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.
- [62] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.
- [63] M. Laubach, M. Shuler, and M.A. Nicolelis. Independent component analyses for quantifying neuronal ensemble interactions. *Journal of Neuroscience Methods*, 94(1):141–154, December 1999.
- [64] Q.V. Le and T. Mikolov. Distributed Representations of Sentences and Documents. *CoRR*, 2014.
- [65] A.K. Lee and M.A. Wilson. Memory of sequential experience in the hippocampus during slow wave sleep. *Neuron*, 36(6):1183–1194, December 2002.
- [66] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.
- [67] Z. Lin, M. Feng, C.N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A Structured Self-attentive Sentence Embedding. *CoRR*, 2017.
- [68] V. Lopes-dos Santos, S. Conde-Ocazonez, M.A.L. Nicolelis, S.T. Ribeiro, and A.B.L. Tort. Neuronal assembly detection and cell membership specification by principal component analysis. *PLOS ONE*, 6(6):e20996, June 2011.
- [69] V. Lopes-dos Santos, S. Ribeiro, and A.B.L. Tort. Detecting cell assemblies in large neuronal populations. *Journal of Neuroscience Methods*, 220(2):149–166, November 2013.

- [70] A. Luczak, P. Bartho, S.L. Marguet, G. Buzsáki, and K.D. Harris. Sequential structure of neocortical spontaneous activity in vivo. *Proceedings of the National Academy of Sciences*, 104(1):347–352, January 2007.
- [71] A. Luczak, P. Bartho, and K.D. Harris. Spontaneous events outline the realm of possible sensory responses in neocortical populations. *Neuron*, 62(3):413–425, May 2009.
- [72] A. Luczak, P. Bartho, and K.D. Harris. Spontaneous Events Outline the Realm of Possible Sensory Responses in Neocortical Populations. *Neuron*, 62(3):413–425, 2009.
- [73] L.v.d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [74] E.L. Mackevicius, A.H. Bahle, A.H. Williams, S. Gu, N.I. Denissenko, M.S. Goldman, and M.S. Fee. Unsupervised discovery of temporal sequences in high-dimensional datasets, with applications to neuroscience. page 273128, March 2018.
- [75] K. Malde, E. Coward, and I. Jonassen. Fast sequence clustering using a suffix array algorithm. *Bioinformatics*, 19(10):1221–1226, July 2003.
- [76] M.S. Manasse. *On the Efficient Determination of Most Near Neighbors: Horse-shoes, Hand Grenades, Web Search and Other Situations When Close is Close Enough*. Morgan & Claypool Publishers, November 2012.
- [77] U. Manber and G. Myers. *Suffix arrays: a new method for on-line string searches*. Society for Industrial and Applied Mathematics, January 1990.
- [78] V.A. Marčenko and L.A. Pastur. DISTRIBUTION OF EIGENVALUES FOR SOME SETS OF RANDOM MATRICES. *Mathematics of the USSR-Sbornik*, 1(4):457–483, 1967.

- [79] S. Maren and M.S. Fanselow. Synaptic plasticity in the basolateral amygdala induced by hippocampal formation stimulation in vivo. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 15(11):7548–7564, November 1995.
- [80] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science (New York, N.Y.)*, 275(5297):213–215, January 1997.
- [81] N. Matsumoto, M. Okada, Y. Sugase-Miyamoto, S. Yamane, and K. Kawano. Population Dynamics of Face-responsive Neurons in the Inferior Temporal Cortex. *Cerebral Cortex*, 15(8):1103–1112, August 2005.
- [82] J.H. Maunsell and D.C. Van Essen. Functional properties of neurons in middle temporal visual area of the macaque monkey. I. Selectivity for stimulus direction, speed, and orientation. *Journal of neurophysiology*, 49(5):1127–1147, May 1983.
- [83] M.R. Mehta, A.K. Lee, and M.A. Wilson. Role of experience and oscillations in transforming a rate code into a temporal code. *Nature*, 417(6890):741–746, June 2002.
- [84] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv.org*, January 2013.
- [85] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, December 2013.
- [86] K. Mizuseki, A. Sirota, E. Pastalkova, and G. Buzsáki. Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. *Neuron*, 64(2):267–280, October 2009.

- [87] M.F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, January 1993.
- [88] J. Mueller, 2016, and A. Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. *AAAI Conferences on Artificial Intelligence*, 2016.
- [89] Z. Nádasdy, H. Hirase, A. Czurkó, J. Csicsvari, and G. Buzsáki. Replay and time compression of recurring spike sequences in the hippocampus. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 19(21): 9497–9507, November 1999.
- [90] S. Nakagawa. A farewell to Bonferroni: the problems of low statistical power and publication bias. *Behavioral Ecology*, 15(6):1044–1045, 2004.
- [91] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, 33(1):31–88, March 2001.
- [92] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, March 1970.
- [93] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03024, 2009.
- [94] K. Ohki, S. Chung, Y.H. Ch’ng, P. Kara, and R.C. Reid. Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex. *Nature*, 433(7026):597–603, February 2005.
- [95] J. O’Keefe. Place units in the hippocampus of the freely moving rat. *Experimental Neurology*, 51(1):78–109, January 1976.
- [96] B.D. Ondov, T.J. Treangen, P. Melsted, A.B. Mallonee, N.H. Bergman, S. Koren, and A.M. Phillippy. Mash: fast genome and metagenome distance estimation using MinHash. *bioRxiv*, page 029827, 2015.

- [97] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707, April 2016.
- [98] D. Paré. Amygdala oscillations and the consolidation of emotional memories. *Trends in Cognitive Sciences*, 6(7):306–314, July 2002.
- [99] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33:1065–1076.
- [100] E. Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, September 1962.
- [101] Peter, Sven, Durstewitz, Daniel, Diego, Ferran, and Hamprecht, Fred A. Sparse convolutional coding for neuronal ensemble identification. *31st Conference on Neural Information Processing Systems (NIPS 2017)*, June 2016.
- [102] A. Peyrache, K. Benchenane, M. Khamassi, S.I. Wiener, and F.P. Battaglia. Principal component analysis of ensemble recordings reveals cell assemblies at high temporal resolution. *Journal of Computational Neuroscience*, 29(1-2): 309–325, June 2009.
- [103] R.G. Phillips and J.E. LeDoux. Differential contribution of amygdala and hippocampus to cued and contextual fear conditioning. *Behavioral Neuroscience*, 106(2):274–285, 1992.
- [104] D. Picado-Muiño, C. Borgelt, D. Berger, G.L. Gerstein, and S. Grün. Finding neural assemblies with frequent item set mining. *Front. Neuroinform. (FINI)* 2013, 7:9, 2013.
- [105] G. Pipa, D.W. Wheeler, W. Singer, and D. Nikolić. NeuroXidence: reliable and efficient analysis of an excess or deficiency of joint-spike events. *Journal of Computational Neuroscience*, 25(1):64–88, January 2008.

- [106] E.A. Pnevmatikakis, D. Soudry, Y. Gao, T.A. Machado, J. Merel, D. Pfau, T. Reardon, Y. Mu, C. Lacefield, W. Yang, M. Ahrens, R. Bruno, T.M. Jessell, D.S. Peterka, R. Yuste, and L. Paninski. Simultaneous denoising, deconvolution, and demixing of calcium imaging Data. *Neuron*, 89(2):285–299, January 2016.
- [107] Quaglio, Pietro, Yegenoglu, Alper, Torre, Emiliano, Endres, Dominik M, and Grün, Sonja. Detection and Evaluation of Spatio-Temporal Spike Patterns in Massively Parallel Spike Train Data with SPADE. *Frontiers in Computational Neuroscience*, 11:83, May 2017.
- [108] C.E. Rasmussen. Gaussian Processes in Machine Learning. In *Advances in Knowledge Discovery and Data Mining*, pages 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
- [109] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, September 1951.
- [110] J.D. Rolston, D.A. Wagenaar, and S.M. Potter. Precisely timed spatiotemporal patterns of neural activity in dissociated cortical cultures. *Neuroscience*, 148(1):294–303, August 2007.
- [111] M. Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, September 1956.
- [112] E. Russo, D. Durstewitz, and M. Howard. Cell assemblies at multiple time scales with arbitrary lag constellations. *eLife*, 6:e19428, January 2017.
- [113] T. Sasaki, N. Matsuki, and Y. Ikegaya. Metastability of active CA3 networks. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 27(3):517–528, January 2007.

- [114] T. Sasaki, N. Takahashi, N. Matsuki, and Y. Ikegaya. Fast and accurate detection of action potentials from somatic calcium fluctuations. *Journal of neurophysiology*, 100(3):1668–1676, September 2008.
- [115] I. Sato and H. Nakagawa. *Topic models with power-law using Pitman-Yor process*. ACM, New York, New York, USA, July 2010.
- [116] J. Scholvin, J.P. Kinney, J.G. Bernstein, C. Moore-Kochlacs, N. Kopell, C.G. Fonstad, and E.S. Boyden. Close-Packed Silicon Microelectrodes for Scalable Spatially Oversampled Neural Recording. *IEEE transactions on bio-medical engineering*, 63(1):120–130, January 2016.
- [117] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823. 2015.
- [118] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [119] H. Shimazaki, S.i. Amari, E.N. Brown, and S. Grün. State-space analysis of time-varying higher-order spike correlation for multiple neural spike train data. *PLOS Computational Biology*, 8(3):e1002385, 2012.
- [120] A.M.S. Shrestha, M.C. Frith, and P. Horton. A bioinformatician ’ s guide to the forefront of suffix array construction algorithms. *Briefings in Bioinformatics*, 15(2):138–154, 2014.
- [121] D.K. Smetters and A. Zador. Synaptic transmission: Noisy synapses and noisy neurons. *Current Biology*, 6(10):1217–1218, October 1996.
- [122] Smith, T F and Waterman, M S. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, March 1981.
- [123] K.A. Spackman. SIGNAL DETECTION THEORY: VALUABLE TOOLS FOR EVALUATING INDUCTIVE LEARNING. In *Proceedings of the Sixth Inter-*

- national Workshop on Machine Learning*, pages 160–163. Morgan Kaufmann, January 1989.
- [124] P.N. Steinmetz, A. Roy, P.J. Fitzgerald, S.S. Hsiao, K.O. Johnson, and E. Niebur. Attention modulates synchronized neuronal firing in primate somatosensory cortex. *Nature*, 404(6774):187–190, March 2000.
- [125] I.H. Stevenson and K.P. Kording. How advances in neural recording affect data analysis. *Nature neuroscience*, 14(2):139–142, February 2011.
- [126] I. Sutskever, O. Vinyals, and Q.V. Le. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing System 27*, pages 3104–3112, 2014.
- [127] S. Tajima, T. Yanagawa, N. Fujii, and T. Toyoizumi. Untangling Brain-Wide Dynamics in Consciousness by Cross-Embedding. *PLOS Computational Biology*, 11(11):e1004537, November 2015.
- [128] O. Tange. Gnu parallel - the command-line power tool. *login: The USENIX Magazine*, 36(1):42–47, Feb 2011.
- [129] T. Taniguchi and S. Nagasaka. Double articulation analyzer for unsegmented human motion using Pitman-Yor language model and infinite hidden Markov model. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, pages 250–255. 2011.
- [130] M. Tatsuno, P. Lipa, and B.L. McNaughton. Methodological considerations on the use of template matching to study long-lasting memory trace replay. *Journal of Neuroscience*, 26(42):10727–10742, October 2006.
- [131] S. Terada, Y. Sakurai, H. Nakahara, and S. Fujisawa. Temporal and Rate Coding for Discrete Event Sequences in the Hippocampus. *Neuron*, 0(0): 1248–1262.e4, June 2017.

- [132] Torre, Emiliano, Canova, Carlos, Denker, Michael, Gerstein, George, Helias, Moritz, and Grün, Sonja. ASSET: Analysis of Sequences of Synchronous Events in Massively Parallel Spike Trains. *PLOS Computational Biology*, 12(7): e1004939, July 2016.
- [133] D. Vasquez, T. Fraichard, and C. Laugier. Incremental Learning of Statistical Motion Patterns With Growing Hidden Markov Models. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):403–416.
- [134] A. Vazdarjanova and J.L. McGaugh. Basolateral Amygdala Is Involved in Modulating Consolidation of Memory for Classical Fear Conditioning. *The Journal of Neuroscience*, 19(15):6615–6622, 1999.
- [135] J.D. Victor and K.P. Purpura. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of neurophysiology*, 76(2):1310–1326, August 1996.
- [136] J.D. Victor, D.H. Goldberg, and D. Gardner. Dynamic programming algorithms for comparing multineuronal spike trains via cost-based metrics and alignments. *Journal of Neuroscience Methods*, 161(2):351–360, April 2007.
- [137] V. Villette, A. Malvache, T. Tressard, N. Dupuy, and R. Cossart. Internally recurring hippocampal sequences as a population template of spatiotemporal information. *Neuron*, 88(2):357–366, October 2015.
- [138] J.T. Vogelstein, A.M. Packer, T.A. Machado, T. Sippy, B. Babadi, R. Yuste, and L. Paninski. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of neurophysiology*, 104(6):3691–3704, December 2010.
- [139] H.M. Wallach. *Topic modeling: beyond bag-of-words*. beyond bag-of-words. ACM, New York, New York, USA, June 2006.

-
- [140] C. Wang and D.M. Blei. *Collaborative topic modeling for recommending scientific articles*. ACM, New York, New York, USA, August 2011.
- [141] J.M. Wang, D.J. Fleet, and A. Hertzmann. Gaussian Process Dynamical Models. *NIPS*, 2005.
- [142] K. Watanabe, T. Haga, D.R. Euston, M. Tatsuno, and T. Fukai. Unsupervised detection of cell-assembly sequences with edit similarity score. *bioRxiv*, page 202655, oct 2017.
- [143] J. Wiest, M. Hoffken, U. Kresel, and K. Dietmayer. Probabilistic trajectory prediction with Gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium (IV)*, pages 141–146.
- [144] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston. StarSpace: Embed All The Things! *arXiv.org*, September 2017.
- [145] A. Yao, J. Gall, L.J. Van Gool, and R. Urtasun. Learning Probabilistic Non-Linear Latent Variable Models for Tracking Complex Activities. *NIPS*, 2011.
- [146] J. Ye, M.P. Witter, M.B. Moser, and E.I. Moser. Entorhinal fast-spiking speed cells project to the hippocampus. *Proceedings of the National Academy of Sciences*, 115(7):E1627–E1636, February 2018.
- [147] W.t. Yih, K. Toutanova, J.C. Platt, and C. Meek. *Learning discriminative projections for text similarity measures*. Association for Computational Linguistics, June 2011.
- [148] R.B. Zadeh and A. Goel. Dimension Independent Similarity Computation. *Journal of Machine Learning Research*, 14:1605–1626, 2013.
- [149] M. Zelikowsky, S. Hersman, M.K. Chawla, C.A. Barnes, and M.S. Fanselow. Neuronal Ensembles in Amygdala, Hippocampus, and Prefrontal Cortex Track Differential Components of Contextual Fear. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 34(25):8462–8466, June 2014.

- [150] A.X. Zhang, A. Noulas, S. Scellato, and C. Mascolo. Hoodsquare: modeling and recommending neighborhoods in location-based social networks. *2013 International Conference on Social Computing (SocialCom)*, pages 69–74, September 2013.

Appendix A

Statistical Table

Herein, we present statistical tables pertaining to the two statistical tests employed in this study. Detailed explanations can be found in the statistical analysis section within the Chapter 3.

Wilcoxon rank sum test for performance comparison between Prop. and PCA-/ICA-based method

We used the Wilcoxon rank sum test to determine the performance difference between the proposed method and the PCA/ICA-based methods (described in Fig.4.1) . For Fig.4.1(A):

| Comparison | Data Structure | Statistical Method | p-value | 99% Confidence Intervals |
|--------------|----------------|------------------------|-------------------|--------------------------|
| Prop. vs PCA | Not Assumed | Wilcoxon rank sum test | less than 2.2e-16 | (0.254, 0.348) |
| Prop. vs ICA | Not Assumed | Wilcoxon rank sum test | less than 2.2e-16 | (0.267, 0.360) |

For Fig. 4.1(B):

| Comparison | Data Structure | Statistical Method | p-value | 99% Confidence Intervals |
|--------------|----------------|------------------------|-----------|--------------------------|
| Prop. vs PCA | Not Assumed | Wilcoxon rank sum test | 1.451e-11 | (-0.3505697 -0.2521959) |
| Prop. vs ICA | Not Assumed | Wilcoxon rank sum test | 2.902e-11 | (-0.2761926 -0.1646134) |

Wilcoxon rank sum test for performance comparison between Prop. and (Russo 2017)

We used the Wilcoxon rank sum test to determine the performance difference between the proposed method and (Russo 2017) (described in Fig.5.2) . For the dataset consisting of 100-ms sequences (Fig.5.2, left):

| Comparison | Data Structure | Statistical Method | p-value | 99% Confidence Intervals |
|------------------------------------|----------------|------------------------|----------|----------------------------|
| F-score, Prop. vs (Russo 2017) | Not Assumed | Wilcoxon rank sum test | 0.6389 | (-0.017237664 0.008403361) |
| Specificity, Prop. vs (Russo 2017) | Not Assumed | Wilcoxon rank sum test | 0.001174 | (-0.033333333 0.000000000) |

For the dataset consisting of 500-ms sequences (Fig.5.2, right):

| Comparison | Data Structure | Statistical Method | p-value | 99% Confidence Intervals |
|------------------------------------|----------------|------------------------|-----------|---------------------------|
| F-score, Prop. vs (Russo 2017) | Not Assumed | Wilcoxon rank sum test | 1.831e-12 | (0.4470212 0.4730781) |
| Specificity, Prop. vs (Russo 2017) | Not Assumed | Wilcoxon rank sum test | 4.353e-07 | (-0.01666667 0.000000000) |

Wilcoxon rank sum test for between- and within-cluster edit similarity comparison

We used the Wilcoxon rank sum test to determine differences in edit similarity of time windows drawn from the same cluster/different clusters (the results are shown in Fig.D).

| Comparison | Data Structure | Statistical Method | p-value | 99% Confidence Intervals |
|--------------------|----------------|------------------------|-------------------|--------------------------|
| Inside vs. Outside | Not Assumed | Wilcoxon rank sum test | less than 2.2e-16 | (1.579333 1.999947) |

Bayesian modeling for edit similarity difference

The model described in the Chapter 3 chapter was used to calculate the credible intervals of differences in edit similarity (Fig.4.4B).

| Cluster | Data Structure | Statistical Method | Position(cm) | 95% Credible Intervals |
|---------|----------------|--------------------|------------------------|------------------------|
| 0 | 4 | Normal | Described in Chapter 3 | 80 (-0.078, 0.803) |
| 1 | 4 | Normal | Described in Chapter 3 | 85 (-0.208, 0.688) |
| 2 | 4 | Normal | Described in Chapter 3 | 90 (-0.901, -0.035) |
| 3 | 4 | Normal | Described in Chapter 3 | 95 (-0.969, -0.138) |
| 4 | 4 | Normal | Described in Chapter 3 | 100 (-1.289, -0.438) |
| 5 | 4 | Normal | Described in Chapter 3 | 105 (-1.55, -0.818) |
| 6 | 4 | Normal | Described in Chapter 3 | 110 (-1.508, -0.812) |
| 7 | 4 | Normal | Described in Chapter 3 | 115 (-1.61, -0.948) |
| 8 | 4 | Normal | Described in Chapter 3 | 120 (-1.623, -0.958) |
| 9 | 4 | Normal | Described in Chapter 3 | 125 (-1.634, -0.925) |
| 10 | 4 | Normal | Described in Chapter 3 | 130 (-1.542, -0.761) |
| 11 | 4 | Normal | Described in Chapter 3 | 135 (-1.22, -0.405) |
| 12 | 4 | Normal | Described in Chapter 3 | 140 (-0.97, -0.248) |
| 13 | 4 | Normal | Described in Chapter 3 | 145 (-0.996, -0.279) |
| 14 | 4 | Normal | Described in Chapter 3 | 150 (-1.12, -0.451) |
| 15 | 4 | Normal | Described in Chapter 3 | 155 (-1.223, -0.536) |
| 16 | 4 | Normal | Described in Chapter 3 | 160 (-1.32, -0.582) |
| 17 | 4 | Normal | Described in Chapter 3 | 165 (-1.192, -0.46) |
| 18 | 4 | Normal | Described in Chapter 3 | 170 (-0.929, -0.196) |
| 19 | 4 | Normal | Described in Chapter 3 | 175 (-0.816, -0.119) |
| 20 | 4 | Normal | Described in Chapter 3 | 180 (-0.814, -0.122) |
| 21 | 4 | Normal | Described in Chapter 3 | 185 (-0.924, -0.249) |
| 22 | 4 | Normal | Described in Chapter 3 | 190 (-0.904, -0.202) |
| 23 | 4 | Normal | Described in Chapter 3 | 195 (-1.01, -0.314) |
| 24 | 4 | Normal | Described in Chapter 3 | 200 (-1.206, -0.495) |
| 25 | 4 | Normal | Described in Chapter 3 | 205 (-1.149, -0.452) |
| 26 | 4 | Normal | Described in Chapter 3 | 210 (-0.957, -0.263) |
| 27 | 4 | Normal | Described in Chapter 3 | 215 (-0.984, -0.277) |
| 28 | 4 | Normal | Described in Chapter 3 | 220 (-0.934, -0.114) |
| 29 | 4 | Normal | Described in Chapter 3 | 225 (0.014, 1.07) |
| 30 | 4 | Normal | Described in Chapter 3 | 230 (0.827, 1.74) |
| 31 | 4 | Normal | Described in Chapter 3 | 235 (1.024, 1.937) |
| 32 | 4 | Normal | Described in Chapter 3 | 240 (2.203, 3.01) |
| 33 | 4 | Normal | Described in Chapter 3 | 245 (2.23, 2.971) |
| 34 | 4 | Normal | Described in Chapter 3 | 250 (2.218, 3.019) |
| 35 | 4 | Normal | Described in Chapter 3 | 255 (1.533, 2.572) |
| 36 | 4 | Normal | Described in Chapter 3 | 260 (0.466, 1.445) |
| 37 | 4 | Normal | Described in Chapter 3 | 265 (0.003, 1.044) |
| 38 | 4 | Normal | Described in Chapter 3 | 270 (-0.751, 0.163) |
| 39 | 4 | Normal | Described in Chapter 3 | 275 (-0.912, -0.126) |
| 40 | 4 | Normal | Described in Chapter 3 | 280 (-1.0, -0.151) |

| Cluster | Data Structure | Statistical Method | Position(cm) | 95% Credible Intervals |
|---------|----------------|--------------------|------------------------|------------------------|
| 0 | 53 | Normal | Described in Chapter 3 | 80 (1.543, 2.269) |
| 1 | 53 | Normal | Described in Chapter 3 | 85 (1.401, 2.135) |
| 2 | 53 | Normal | Described in Chapter 3 | 90 (0.554, 1.341) |
| 3 | 53 | Normal | Described in Chapter 3 | 95 (0.251, 1.023) |
| 4 | 53 | Normal | Described in Chapter 3 | 100 (-0.497, 0.324) |
| 5 | 53 | Normal | Described in Chapter 3 | 105 (-0.768, -0.147) |
| 6 | 53 | Normal | Described in Chapter 3 | 110 (-0.784, -0.167) |
| 7 | 53 | Normal | Described in Chapter 3 | 115 (-1.056, -0.443) |
| 8 | 53 | Normal | Described in Chapter 3 | 120 (-0.908, -0.365) |
| 9 | 53 | Normal | Described in Chapter 3 | 125 (-0.908, -0.342) |
| 10 | 53 | Normal | Described in Chapter 3 | 130 (-0.818, -0.192) |
| 11 | 53 | Normal | Described in Chapter 3 | 135 (-0.517, 0.077) |
| 12 | 53 | Normal | Described in Chapter 3 | 140 (-0.482, 0.091) |
| 13 | 53 | Normal | Described in Chapter 3 | 145 (-0.547, 0.035) |
| 14 | 53 | Normal | Described in Chapter 3 | 150 (-0.745, -0.164) |
| 15 | 53 | Normal | Described in Chapter 3 | 155 (-0.759, -0.215) |
| 16 | 53 | Normal | Described in Chapter 3 | 160 (-0.902, -0.265) |
| 17 | 53 | Normal | Described in Chapter 3 | 165 (-0.657, 0.04) |
| 18 | 53 | Normal | Described in Chapter 3 | 170 (-0.342, 0.276) |
| 19 | 53 | Normal | Described in Chapter 3 | 175 (-0.272, 0.308) |
| 20 | 53 | Normal | Described in Chapter 3 | 180 (-0.073, 0.493) |
| 21 | 53 | Normal | Described in Chapter 3 | 185 (0.013, 0.542) |
| 22 | 53 | Normal | Described in Chapter 3 | 190 (0.035, 0.549) |
| 23 | 53 | Normal | Described in Chapter 3 | 195 (-0.006, 0.57) |
| 24 | 53 | Normal | Described in Chapter 3 | 200 (-0.25, 0.353) |
| 25 | 53 | Normal | Described in Chapter 3 | 205 (-0.342, 0.259) |
| 26 | 53 | Normal | Described in Chapter 3 | 210 (-0.455, 0.196) |
| 27 | 53 | Normal | Described in Chapter 3 | 215 (-0.798, -0.154) |
| 28 | 53 | Normal | Described in Chapter 3 | 220 (-0.847, -0.279) |
| 29 | 53 | Normal | Described in Chapter 3 | 225 (-1.051, -0.457) |
| 30 | 53 | Normal | Described in Chapter 3 | 230 (-0.892, -0.368) |
| 31 | 53 | Normal | Described in Chapter 3 | 235 (-0.896, -0.388) |
| 32 | 53 | Normal | Described in Chapter 3 | 240 (-0.83, -0.336) |
| 33 | 53 | Normal | Described in Chapter 3 | 245 (-0.796, -0.314) |
| 34 | 53 | Normal | Described in Chapter 3 | 250 (-0.802, -0.307) |
| 35 | 53 | Normal | Described in Chapter 3 | 255 (-0.763, -0.267) |
| 36 | 53 | Normal | Described in Chapter 3 | 260 (-0.813, -0.266) |
| 37 | 53 | Normal | Described in Chapter 3 | 265 (-0.643, -0.095) |
| 38 | 53 | Normal | Described in Chapter 3 | 270 (-0.597, -0.041) |
| 39 | 53 | Normal | Described in Chapter 3 | 275 (-0.83, -0.215) |
| 40 | 53 | Normal | Described in Chapter 3 | 280 (-0.706, -0.016) |

| Cluster | Data Structure | Statistical Method | Position(cm) | 95% Credible Intervals |
|---------|----------------|--------------------|------------------------|------------------------|
| 0 | 10 | Normal | Described in Chapter 3 | 80 (2.202, 2.985) |
| 1 | 10 | Normal | Described in Chapter 3 | 85 (2.168, 2.769) |
| 2 | 10 | Normal | Described in Chapter 3 | 90 (2.078, 2.661) |
| 3 | 10 | Normal | Described in Chapter 3 | 95 (1.981, 2.593) |
| 4 | 10 | Normal | Described in Chapter 3 | 100 (1.798, 2.534) |
| 5 | 10 | Normal | Described in Chapter 3 | 105 (0.961, 1.979) |
| 6 | 10 | Normal | Described in Chapter 3 | 110 (0.52, 1.438) |
| 7 | 10 | Normal | Described in Chapter 3 | 115 (0.163, 1.117) |
| 8 | 10 | Normal | Described in Chapter 3 | 120 (-0.256, 0.495) |
| 9 | 10 | Normal | Described in Chapter 3 | 125 (-0.345, 0.339) |
| 10 | 10 | Normal | Described in Chapter 3 | 130 (-0.154, 0.437) |
| 11 | 10 | Normal | Described in Chapter 3 | 135 (-0.077, 0.482) |
| 12 | 10 | Normal | Described in Chapter 3 | 140 (0.018, 0.574) |
| 13 | 10 | Normal | Described in Chapter 3 | 145 (0.042, 0.593) |
| 14 | 10 | Normal | Described in Chapter 3 | 150 (0.028, 0.595) |
| 15 | 10 | Normal | Described in Chapter 3 | 155 (-0.069, 0.488) |
| 16 | 10 | Normal | Described in Chapter 3 | 160 (-0.118, 0.446) |
| 17 | 10 | Normal | Described in Chapter 3 | 165 (-0.261, 0.323) |
| 18 | 10 | Normal | Described in Chapter 3 | 170 (-0.341, 0.226) |
| 19 | 10 | Normal | Described in Chapter 3 | 175 (-0.352, 0.184) |
| 20 | 10 | Normal | Described in Chapter 3 | 180 (-0.295, 0.232) |
| 21 | 10 | Normal | Described in Chapter 3 | 185 (-0.205, 0.324) |
| 22 | 10 | Normal | Described in Chapter 3 | 190 (-0.189, 0.36) |
| 23 | 10 | Normal | Described in Chapter 3 | 195 (-0.31, 0.227) |
| 24 | 10 | Normal | Described in Chapter 3 | 200 (-0.331, 0.191) |
| 25 | 10 | Normal | Described in Chapter 3 | 205 (-0.388, 0.149) |
| 26 | 10 | Normal | Described in Chapter 3 | 210 (-0.394, 0.141) |
| 27 | 10 | Normal | Described in Chapter 3 | 215 (-0.561, 0.023) |
| 28 | 10 | Normal | Described in Chapter 3 | 220 (-0.636, -0.019) |
| 29 | 10 | Normal | Described in Chapter 3 | 225 (-0.524, 0.066) |
| 30 | 10 | Normal | Described in Chapter 3 | 230 (-0.476, 0.144) |
| 31 | 10 | Normal | Described in Chapter 3 | 235 (-0.292, 0.342) |
| 32 | 10 | Normal | Described in Chapter 3 | 240 (-0.151, 0.417) |
| 33 | 10 | Normal | Described in Chapter 3 | 245 (-0.079, 0.466) |
| 34 | 10 | Normal | Described in Chapter 3 | 250 (-0.1, 0.424) |
| 35 | 10 | Normal | Described in Chapter 3 | 255 (-0.094, 0.435) |
| 36 | 10 | Normal | Described in Chapter 3 | 260 (-0.037, 0.504) |
| 37 | 10 | Normal | Described in Chapter 3 | 265 (-0.047, 0.527) |
| 38 | 10 | Normal | Described in Chapter 3 | 270 (-0.006, 0.694) |
| 39 | 10 | Normal | Described in Chapter 3 | 275 (-0.361, 0.382) |
| 40 | 10 | Normal | Described in Chapter 3 | 280 (-0.573, 0.278) |

| Cluster | Data Structure | Statistical Method | Position(cm) | 95% Credible Intervals |
|---------|----------------|--------------------|------------------------|------------------------|
| 0 | 16 | Normal | Described in Chapter 3 | 80 (-0.395, 0.262) |
| 1 | 16 | Normal | Described in Chapter 3 | 85 (-0.371, 0.196) |
| 2 | 16 | Normal | Described in Chapter 3 | 90 (-0.425, 0.119) |
| 3 | 16 | Normal | Described in Chapter 3 | 95 (-0.486, 0.057) |
| 4 | 16 | Normal | Described in Chapter 3 | 100 (-0.553, -0.011) |
| 5 | 16 | Normal | Described in Chapter 3 | 105 (-0.591, -0.016) |
| 6 | 16 | Normal | Described in Chapter 3 | 110 (-0.518, 0.139) |
| 7 | 16 | Normal | Described in Chapter 3 | 115 (-0.265, 0.555) |
| 8 | 16 | Normal | Described in Chapter 3 | 120 (0.34, 1.04) |
| 9 | 16 | Normal | Described in Chapter 3 | 125 (0.466, 1.091) |
| 10 | 16 | Normal | Described in Chapter 3 | 130 (0.612, 1.235) |
| 11 | 16 | Normal | Described in Chapter 3 | 135 (0.454, 1.069) |
| 12 | 16 | Normal | Described in Chapter 3 | 140 (0.351, 0.951) |
| 13 | 16 | Normal | Described in Chapter 3 | 145 (0.192, 0.79) |
| 14 | 16 | Normal | Described in Chapter 3 | 150 (0.09, 0.683) |
| 15 | 16 | Normal | Described in Chapter 3 | 155 (-0.025, 0.564) |
| 16 | 16 | Normal | Described in Chapter 3 | 160 (0.075, 0.662) |
| 17 | 16 | Normal | Described in Chapter 3 | 165 (0.175, 0.792) |
| 18 | 16 | Normal | Described in Chapter 3 | 170 (0.307, 1.005) |
| 19 | 16 | Normal | Described in Chapter 3 | 175 (0.563, 1.314) |
| 20 | 16 | Normal | Described in Chapter 3 | 180 (0.974, 1.627) |
| 21 | 16 | Normal | Described in Chapter 3 | 185 (1.049, 1.663) |
| 22 | 16 | Normal | Described in Chapter 3 | 190 (0.926, 1.55) |
| 23 | 16 | Normal | Described in Chapter 3 | 195 (0.692, 1.418) |
| 24 | 16 | Normal | Described in Chapter 3 | 200 (0.216, 1.003) |
| 25 | 16 | Normal | Described in Chapter 3 | 205 (0.031, 0.723) |
| 26 | 16 | Normal | Described in Chapter 3 | 210 (-0.171, 0.478) |
| 27 | 16 | Normal | Described in Chapter 3 | 215 (-0.32, 0.416) |
| 28 | 16 | Normal | Described in Chapter 3 | 220 (-0.983, -0.224) |
| 29 | 16 | Normal | Described in Chapter 3 | 225 (-1.164, -0.541) |
| 30 | 16 | Normal | Described in Chapter 3 | 230 (-1.246, -0.689) |
| 31 | 16 | Normal | Described in Chapter 3 | 235 (-1.195, -0.65) |
| 32 | 16 | Normal | Described in Chapter 3 | 240 (-1.228, -0.689) |
| 33 | 16 | Normal | Described in Chapter 3 | 245 (-1.292, -0.761) |
| 34 | 16 | Normal | Described in Chapter 3 | 250 (-1.321, -0.791) |
| 35 | 16 | Normal | Described in Chapter 3 | 255 (-1.343, -0.802) |
| 36 | 16 | Normal | Described in Chapter 3 | 260 (-1.296, -0.731) |
| 37 | 16 | Normal | Described in Chapter 3 | 265 (-1.283, -0.674) |
| 38 | 16 | Normal | Described in Chapter 3 | 270 (-1.096, -0.395) |
| 39 | 16 | Normal | Described in Chapter 3 | 275 (-0.897, -0.086) |
| 40 | 16 | Normal | Described in Chapter 3 | 280 (-0.195, 0.723) |