Spike2Vec: Converting Spike Trains to Vectors to Analyse Network States and State Transitions:

Dr Russell Jarvis

Mr Pablo de Abreu Urbizagastegui

Mr Yeshwanth Bethi

***Abstract*— A scalable algorithm that can detect fine grained repetitions quickly across large spiking datasets is desirable, as such a frame work would provides a means to test for the tendency of neuronal activity to revisit states.**

Quickly identifying repeated states in large scale neuronal data and simulation is important, as the degree of repitition should influence the mindset of the scientists undertaking the analysis. For instance many theoretical Neuroscientists believe assumptions that valid cortical neuronal simulations should consist of Asynchronous Irregular activity (AI), and they may often simulate such activity using [1] Brunel's balanced model of cortex. However, new data sets prominently capture replayed states, and previously collected of spike trains may have been misleading, as too their limited ability to capture replay, and detect it in analysis.

Although the dynamic systems view of the brain is old, a survey of spiking datasets which can detect and labels network attractor states in large spike count data is merited, as this would bolster the dynamic systems view of the neuronal learning.

By quantifying repetitions large spiking datasets, using geometric representations of complex spike patterns, we can quantify the frequency of repitition, and achieve a better understanding of a networks ability to revisit states. To this end we represented time bound neural activity as simple geometric coordinates in a highdimensional space. Working with geometric representations of chaotic spike train recordings may enable researchers to interrogate the state-fullness of both biologically recorded spike trains and their digitally simulated counterparts. Furthermore, there is reason to believe that when mammal brains enact visual object recognition encoded memories guide cortical neurons to "replay" previously observed neural states, as replayed spiking states may cohere with the visual brains perceptual recognition of a familiar scene.

I. Introduction

There is a great demand for a scalable algorithm that can detect repeating temporal spatial features in biological and synthetic data sets of cortical neuronal networks.

Multivariate approaches to spike train network analysis often involves the computation of some kind of statistic between each possible pair of neurons in the network. To analyse causality in networks, spike train recordings are divided into time windows, and analysis compares previous (lagged time), with current time. Exhaustive pairwise iteration of multivariate statistics is not computationally tractible at the scale of billions of neurons, and adding time lagged analysis of network cross-correlation, or transfer entropy makes the prospoect of scaled temporal analysis even worse. Autocovariance acts on anolog signals (dense vectors), however autocovariance analysis of continuous membrane potentials would be another way to arrive at a network state description.

Two common models of cortical spiking networks are the, Potjan's and Diesmon model and the Brunel model, both of these models are said exist within a fluctuation driven regime, when these are simulated, observed spike times are typically chaotic and random, but some fine grained recognizable repeating patterns also occur. Under the dynamic systems view of the brain neuronal memories are analogous to attractor basins [Hopfield,Lin, Hairong, et al]. If the view of memories as basins is correct then it should be possible to demonstrate synaptic learning as the mechanism that encodes memories as basins. Network attractor basins may be derived from the interleaved application of Spike Timing Dependent Plasticity (STPD) and sleep when synapses are able to change in a way that strongly biases some future spiking activities towards stereotyped patterns.

The application of STDP learning within the fluctuation driven regime necessitates a simple method to optimise network parameters a way that maximises the networks capacity to encode and revisit attractor states. A spike2vec algorithm will enable researchers to investigate the state-fullness of spike trains, the corruption of information caused by STDP in the

absence of sleep and resistance to the degradation of memories that may be concomitant with neuronal death and synaptic pruning, as many of these network level phenonemana can be re-construed as network parameters: for example neuronal death relates to synaptic count and neuron count.

A. Discussion

Each conscious recall of a memory may involve the brain approximately repeating a pattern; ie recall may mean re-visiting a previous pattern of neuronal activity. Each recalled memory may retain significant traces of activity that is well correlated with the brain experience that caused the memory to be encoded. Such "replay" is has been observed in the hippocampus and prefrontal cortex in rats during sleep.

When replayed events are detected a sequential map of states can be derived from a spike train, and unrecognized state transitions and anomolies can also be sorted and labelled in discrete chunks.

Under spike2vec framework, spike patterns which are approximately the same as patterns in previous windows are detected because, in the geometric coordinate representation of vectors, spike trains which are close together will have be seperated by a small Euclidean Distance in the vector space.

The spike2vec frame work can be used to convert spike trains to markov transistion matrices, or simply state transition network maps. In such a state network, we can see that not all spike trains are equally stateful, some emperical recordings may have few replay events. When a spike recording is particularly stateful, there may be certain states which are only entered into given the appropriate sequence of prior states.

II. Theoretical Framework

1) *Algorithm Details:* The spike2vec frame work exists at the meta level. It is a novel mashup of pre-existing algorithms, its steps are as follows:

1. Spike trains are divided into N equally sized windows time windows.
2. In each window spike times are converted by subtracting the window start time, such that spike time variability is now mapped onto the smaller scope of each window (ie the time each window occured is subtracted from each window, as window times obscure the analysis). Each of the converted time, time windows is stored in an array.
3. The maximum firing rate of all the windows is found.
4. A single surrogate spike window is constructed by taking the maxmimum firing rate from step 3. And constructing a spike train that has regular Inter Spike Intervals (ISIs) occuring at the maximum firing rate. We call this the reference window.
5. For every N windows sampled in 1, the observed spike times is compared to the uniform reference window using the Thomas Kreuz spike Distance algorithm implemented in Julia by George Datseris. https://github.com/JuliaNeuroscience/SpikeSynchrony.jl/commits?author=Datseris
6. The Kreuz spike distance is a way of measuring the cost of converting observed spike train A, to a different spike train B. By measuring the Kreuz spike distance between a variation free regular spiking window, and a window with observed spike time variability, we get a picture of each neurons current unique local variability at each window (note that the method for comparing reference to observed doesn't have to uniquely encode unique spike sequences, it just has to be unique enough). There is M number of neurons we can build a vector of coordinate of **M** dimensions, at each of N time windows. An M by **N** matrix consists of M neurons and N time windows.
7. Since each column vector encodes a time window, we get the euclidian distance between each column vector and every other column vector, across the columns of the whole matrix.
8. We take these new distance values we fill a new matrix, between every window, and every other window at row and column location of the matrix. It's important to recognize that here we are not comparing spike distances between neurons (as has occured in established work, we are commparing spike train distance vectors within the same neurons along time).
9. We perform unsupervised clustering on this temporaly encoded dissimalirity matrix.
10. We discard all cluster labels that correspond to just a single time window, and retain the set of cluster labels, that have at least one repeating label. We regard these duplicated cluster labels as repeated temporal spatial patterns.

III. Methodological Framework

IV. Result Analysis

V. Statement of Need

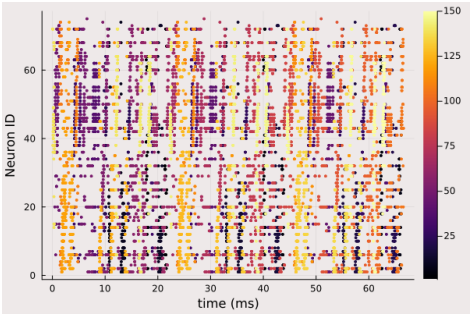
Scalable methods for representing the transient behavior of large populations of neurons are needed. The spike2vec algorithm will enable researchers to track the trajectory of the

network between familiar and unfamiliar states using a high-dimensional coordinate scheme. A network's ability to revisit an encoded coordinate is testable, and so a spike2vector test of object recognition could be construed as a formal hypothesis test.

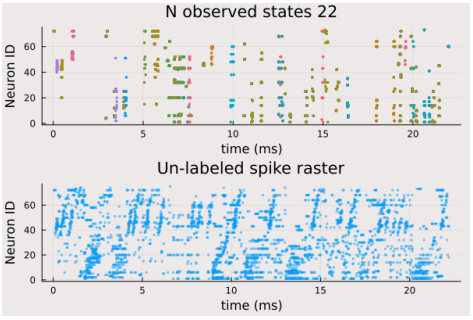
VI. Reproducibility

Some preliminary code that performs the Spike2Vec analysis is avaialble at the following link. the code is implemented in Julia, a modern language alternative to Python that makes large-scale model visualization and analysis more computationally tractable. A docker file is included.

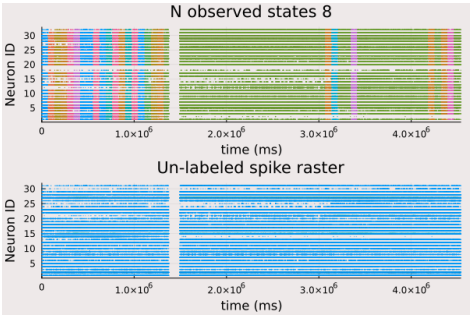
Herein lies a heatmap of dis-simarity matrices constructed using the NMNIST dataset, ie the heatmap above, comes from analysing spike train distance across the NMNIST data set numbers: 0-9 represented as spiking events. There are 300 total presentation number presentations. All nine numbers are incrementally cycled through. Number presentations within the one number are contiguous, (the data set isn't shuffled), and this contiguity is reflected in the heatmap too.



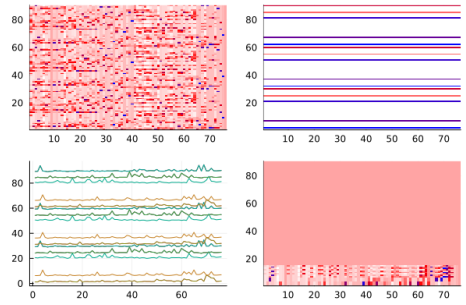
In order to test if the spike2vec framework worked as expected, we downloaded a alcium imaging recording from Zebra finch (a song bird's) High Vocal Centre (brain region) source [2]. Although the actual data source was from (https://github.com/lindermanlab/PPSeq.jl/blob/master/demo/data/songbird_spikes.txt) The downloaded data set was then simply augmented, by duplicating the spike time raster plot in a manner that appended the full repeated recording to the end of the first recording, the process was iterated 3 times yielding a highly repititive data set 4 times the length of the original. The intention of this exercise was simply to show that spike2vec could identify and label such obvious repeating patterns.



We take the single un augmented Zebra finche data and label repeating patterns using the discussed frame work, since there are no obvious explicit repitions caused by duplicating spike patterns, the algorithm is forced to consider the similarity between more disparate population level patterns.

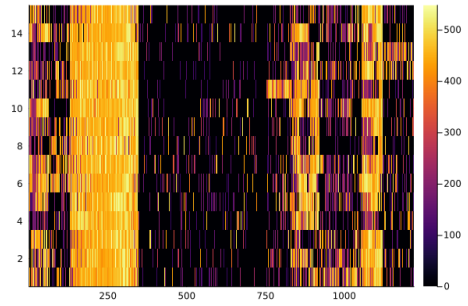


We took a data set from rat prefrontal cortex. The data set was captured, under experimental conditions explicitly designed to maximise the probability of recording replay ie dreaming and slow wave sleep.

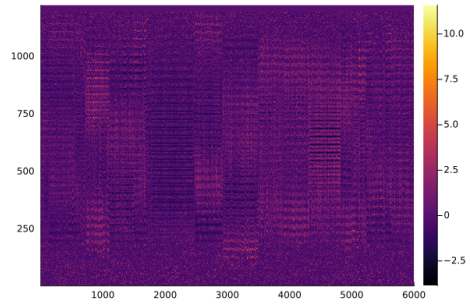


Figures from left top to bottom right:A: Top left: 75 NMNIST channels were recorded and time binned in a manner which yielded 85 vectorized time bins. Bottom right: Once the vectorized time bins had been vectorized, a clustering algorithm was applied to the entire matrix of vector coordinates. Cluster centres could then be used as reference points, such that it was possible to compare all

We used data augmentation to demonstrate that this labelling technique can trivially recognize patterns.

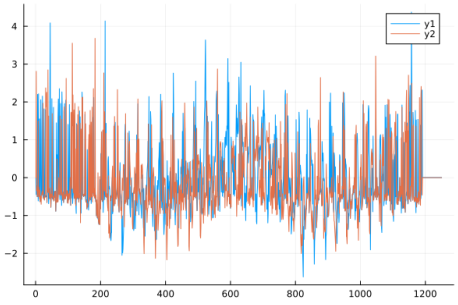


Normalized PFC vectors for 1000 neurons over 10 channels



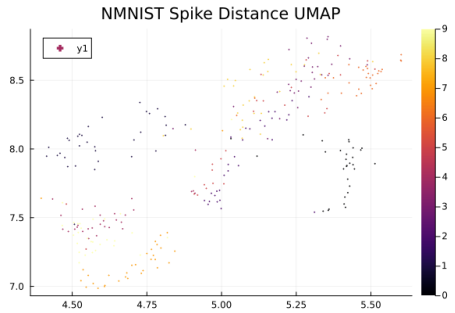
We sampled the NMNIST data set more broadly using 6, 000 samples to create vectors. The entire data set consists of 60, 000 samples.

align(center + bottom)[



Two spike time encoded numerals, where read in to Julia, then the spiking neuromorphic data were converted to vectors over 1200 channels. Orange and Blue plots are vectors corresponding to two distinct NMNIST data labels. Rapid positive alternating deflections are visible in both vectors, because the NMNIST data is caused by pixel activations, when $2D$ pixel derived data sources are converted into a $1D$ vector, sparse clusters of activated pixels, have regular gaps between them.]

align(center + bottom)[



As a matter of routine UMAP dimensional embedding of spike distances was applied to all spike differences]

cluster_sort_MNMIST.png cluster_sort_song_birds.png didit_work_NM.png didit_work.png everything_includinding_repeated_pattern_pablo.png everythin_includinding_repeated_pattern_pablo.png heatmap_after.png heatmap_before.png heatmap.png just_two_pablo_raw_vectors.png just_two_song_bird_raw_vectors.png labelled_mat_of_distancesNMINST.png labelled_mat_of_distancesNMINST_test_train.png labelled_mat_of_distances_pablo.png labelled_mat_of_distances.png LabelledSpikes18.png LabelledSpikesPartition18.png Levine13-CD4.png Normalised_heatmap_pablo.png Normalised_heatmap_song_bird.png normal.png not_cluster_sort_MNMIST.png pablo_umap.png reference_labelled_mat_of_distances_pablo.png relative_to_uniform_referenceNMMIST.png repeated_pattern_pablo.png repeated_pattern.png repeated_pattern_song_bird.png scatternmnist_angles.png scatternmnist_distances.png scatternmnist.png slice_one_window.png slice_three_window.png slice_two_window.png sorted_train_map.png test_map.png train_map.png umap_of_NMNIST_Data.png UMAP_song_bird.png UniformSpikes.png Unormalised_heatmap_pablo.png Unormalised_heatmap.png Unormalised_heatmap_song_bird.png vector_differences_another.png

A. *References*

References

[1] N. Brunel, "Hebbian learning of context in recurrent neural networks," *Neural Computation*, vol. 8, no. 8, pp. 1677–1710, 1996.

[2] E. L. Mackevicius, A. H. Bahle, et al., "Unsupervised discovery of temporal sequences in high-dimensional datasets, with applications to neuroscience," *Elife*, vol. 8, 2019.