Crabsort: spike-sorting for small circuit networks

Alec Hoyland
Center for Systems Neuroscience
Boston University
Boston, MA 02215
entropyvsenergy@posteo.de

Cosmo Guerini
Department of Biology and Volen National Center for Complex Systems
Brandeis University
Waltham, MA 02453
machinelearning@cosmo.red

Abstract

Crabsort.

1 Introduction

Electrophysiological recording by extracelullar electrode is one of the simplest and most reliable methods for recording local electrical activity around the electrode tip. Action potentials produce characteristic voltage deflections, known as a spike waveform. The extracellular recordings take the form of voltage time series, with one waveform for each electrode, from which spikes must be identified, as well as from which cell the spike originates (Quiroga 2012). The first task is well-studied, and can be managed effectively by high-pass filtering and counting threshhold crossings (Quiroga 2012; Rossant et al. 2016). However, it is extremely difficult to tell by eye which spikes originate from which cells. In larger networks with up to 1,000 simultaneous recordings, algorithms exist which can cluster based on PCA, SVD, and stochastic k-means matching (Pachitariu et al. 2016; Rossant et al. 2016).

In smaller networks, of about 30 cells, where each cell is named and well-studied, these algorithms break down. With paucity of data and similarity between waveforms of differing cell types, it is not feasible to use algorithms designed for large multi-channel recordings on smaller circuit data. Furthermore, methods such as Granger causality can report spurious causal relationships if the network is strongly oscillatory (Kispersky, Gutierrez, and Marder 2011). One solution is to use a mixed linear point process model (Gerhard et al. 2013), but this strategy is not extensible to unknown systems, or for use with unlabeled data.

Another strategy, implemented in the software package `crabsort` (S. Gorur-Shandilya *et al.*, unpublished), uses pre-sorted data to train a neural network model which can automatically sort the remaining data. Briefly, this process involves dimensionally-reducing the spike trains to a 2-dimensional manifold and manually clustering a subset of the data. A neural network is trained on this dataset to identify which spikes belong to which clusters, where each cluster is a different neuron.

We extend the `crabsort` toolkit to include several backends implementing fast Fourier transform-accelerated interpolation-based t-SNE (FIt-SNE) (Linderman et al. 2019; Van Der Maaten 2014) and uniform manifold approximation and projection (McInnes, Healy, and Melville 2018).

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Crabsort

`crabsort` is a toolkit for visualizing and sorting electrophysiological data. Its name originates in its use on electrophysiological data from the stomatogastric ganglion in crustaceans.

Raw electrophysiological data are loaded into MATLAB (Mathworks, Newton, MA) and spikes are found using the built-in peak finding algorithm with parameters that can be adjusted in real-time. The training data typically consists of less than 1 percent of the total data to be spike-sorted.

Snippets around each spike waveform are taken. The membrane potential at each time step is counted as the value in a different dimension, so that the number of dimensions is usually around 70.

The data are then dimensionally-reduced to a 2-dimensional manifold. This is one of the most crucial steps, as the dimensional reduction method determines the spatialization of the clusters and therefore the representativeness of the training data given to the neural network. The choice of dimensionality reduction algorithm, along with the noise in the data and distinguishability of the classifications (different neurons from whence the spike originated) determine the separation of clusters in the low-dimensional space.

A neural network consisting of two fully-connected layers, interspersed with reLU layers, is trained on the clustered data. A layer with a 10 percent dropout rate leads to a softmax-activated classification layer.

After training, `crabsort` can be used to classify the remaining spikes in the dataset.

If the spike does not clearly fit into one of the classes, `crabsort` flags it. If spikes are manually reclassified after viewing the prediction, `crabsort` retrains the network to encompass those new data.

2.1 Aims of this project

We add to the extant `crabsort` framework, by extending the dimensionality reduction options beyond naive PCA and t-SNE. We implement a faster and more accurate version of t-SNE that uses fast-Fourier transform interpolation and parallelized approximate nearest neighbors, as well as uniform manifold approximation and projection, a very new dimensionality reduction method that is very accurate for subsets of data.

2.2 Current dimensionality reduction algorithms

Currently there are three dimensionality-reduction algorithms implemented in `crabsort`. The first separates by spike amplitude, which can sometimes be sufficient. The second uses a naive principle component analysis (PCA) algorithm. PCA is an eigenvector-based multivariate analysis that linearly transforms a dataset onto orthogonal principle components which account for as much variance in the data as possible. It is a linear transformation, and is fast to use, making PCA a popular first step in exploratory data visualization. Unfortunately, it performs worse at clustering than more recently-developed algorithms.

`crabsort` also implements t-distributed stochastic neighborhood embeddings (t-SNE), a type of k-nearest dimensionality reduction algorithm (Van Der Maaten 2014). t-SNE first computes the conditional probabilities that are proportional to the similarity of each data point. In the original algorithm, Gaussian kernels are used with the Euclidean distance. The probability of x_j conditioned on x_i indicates the probability that x_j would be chosen as a neighbor to x_i, if neighbors were picked in proportion to their probability density under a Gaussian distribution centered at x_i. The bandwidth of the Gaussian kernels (i.e. the variance) is set so that the perplexity of the conditional distribution equals a predefined hyperparameter value. Student's t-distribution is used in the low-dimensional representation to measure similarity, which is compared to the high-dimensional conditional distributions by minimizing the Kullback-Leibler divergence between the two, by gradient descent. t-SNE provides much better clustering than PCA, but is much slower, acting in $\mathcal{O}(n^2)$ time.

3 New dimensionality reduction algorithms

Since dimensionality reduction is crucial to generating an accurate training dataset, we implement backends for two improved dimensionality reduction algorithms: FIt-SNE and UMAP.

Since both the FIt-SNE and UMAP wrappers create structs which contain modifiable parameters, we modify the `crabsort` framework to expose plugin object parameters, for real-time tuning of algorithm parameters to better cluster the data.

3.1 FIt-SNE

The first, fast Fourier transform-interpolated t-SNE (FIt-SNE) provides accelerated t-SNE on $\mathcal{O}(n)$. Common implementations of t-SNE use the Barnes-Hut approximation to simplify the n-body problem which must be solved during optimizing the embedding, yielding time-complexity $\mathcal{O}(n \log n)$ (Van Der Maaten 2014). Linderman et al. 2019 interpolate an equispaced grid rather than compute the repulsive forces in the n-body problem directly. Computing the objective function amounts to performing a convolution over the grid. Since the grid is translation-invariant with respect to the interpolating spline functions, and the matrix associated with the convolution is Toeplitz, the convolution can be simplified to multiplication in the Fourier domain, by taking the fast Fourier transform (FFT).

FIt-SNE comes equipped with a MATLAB wrapper that runs the compiled C-code as a MEX file. We improve the wrapper by through minor speed improvements, and implement FIt-SNE as a dimensionality reduction plugin for `crabsort`.

3.2 UMAP

In addition, instead of taking k-nearest neighbors, FIt-SNE uses an approximate nearest-neighbors algorithm, `ANNOY` (Bernhardsson 2019), which can take advantage of multithreading for a speed increase. FIt-SNE results in time-complexity $\mathcal{O}(n \log n)$, but converges 15-30x times faster than the Barnes-Hut accelerated t-SNE implementation.

Like FIt-SNE, uniform manifold approximation and projection (UMAP) (McInnes, Healy, and Melville 2018) is based on a k-nearest neighbors algorithm, however unlike t-SNE, UMAP is not stochastic. It relies on three core assumptions:

- The data are uniformly distributed on a Riemannian manifold.
- The Riemannian metric is locally constant (or can be approximated as such).
- The manifold is locally connected.

While the theoretical underpinnings are satisfactory for any fuzzy simplicial sets, the algorithm is best implemented over a weighted graph. The k-nearest neighbors weighted graph can be computed by any suitable algorithm. UMAP then performs spectral clustering on the Laplacian matrix constructed from the graph. Optimization of the fuzzy simplicial set cross entropy follows to optimize the embedding. UMAP is stable under subsampling and must faster than even FIt-SNE, though the clustering is more opaque, and therefore sometimes less desirable than PCA or t-SNE. Furthermore, UMAP is deterministic and therefore allows new data to be mapped to the low-dimensional space without recomputing the entire embedded representation.

Since UMAP is written in Python, we used `condalab` and a UMAP wrapper for MATLAB to configure a virtual environment for use with `crabsort`.

4 Benchmarking

Data were collected from the stomatogastric ganglion of *Cancer borealis* crabs by Daniel Powell in the Marder Laboratory at Brandeis University. Electrophysiological recordings were taken at a 44.4 kHz sample frequency, and stored in multiple `.abf` files. The dataset used for benchmarking, `901-046`, was segmented into 126 files. The first of these files was used for clustering and training the neural network.

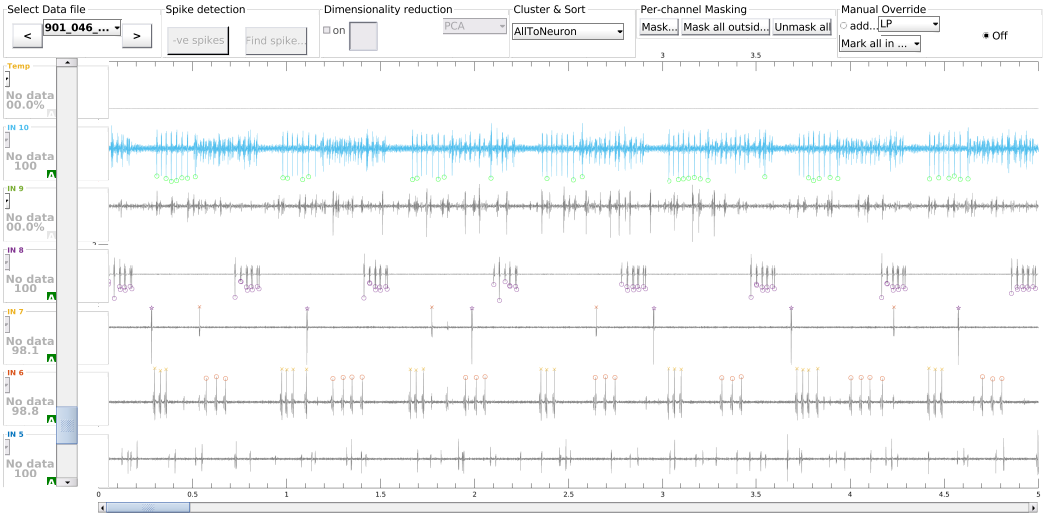

Figure 1: Snapshot of the `crabsort` GUI. The top trace was used for benchmarking. Spikes are denoted by circles.

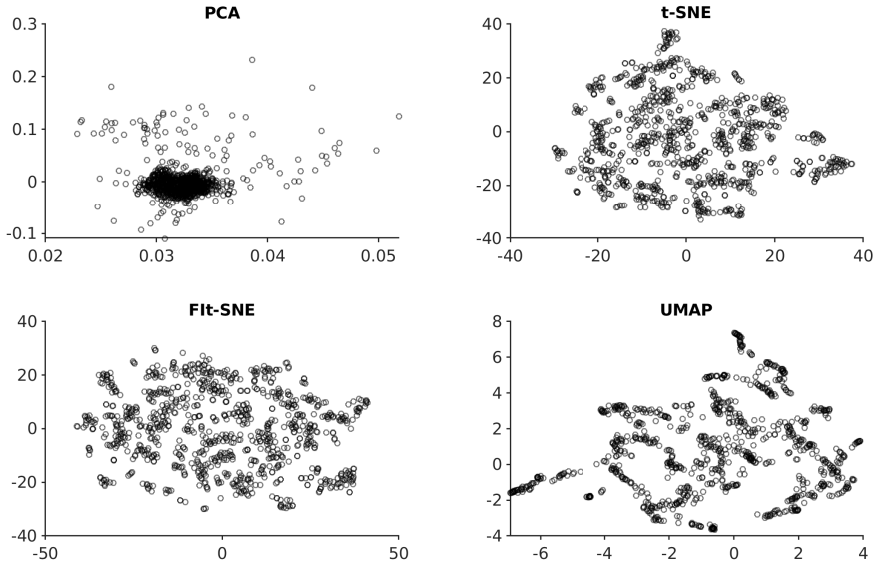

Figure 2: Four dimensionality-reduction schemes run on the same data.

180 spikes were found using the `findpeaks` algorithm and 947 time steps were taken around each spike.

Four dimensionality-reduction schemes were tested: PCA, t-SNE, FIt-SNE, and UMAP. PCA performs the worst, as it can only account for linear combinations. t-SNE and FIt-SNE are almost indistinguishable, modulo randomness. The long chains that appear in the UMAP-reduced presentation are a result of the simplicial complexes used to construct the topological space.

Surprisingly, FIt-SNE performed terribly in time benchmarking. We suspect this is due to the more rigorous optimization algorithm used to determine the best t-SNE fit. For example, using the ANNOY library for the approximate k-nearest-neighbors search was parallelized, and took only 0.037 s with $k = 90$. The algorithm terminated the gradient descent optimization after 1,000 iterations.

Table 1: Time elapsed during dimensionality reduction.

Algorithm Name	Time Elapsed (s)
PCA	0.0228
t-SNE	5.2076
FIt-SNE	21.3535
UMAP	4.5879

Though FIt-SNE should be theoretically faster than t-SNE, for a small dataset, it proves to be much slower without a significant increase in cluster separation.

UMAP performed as well as promised, acting in a competitive amount of time with t-SNE, and proved to be the best algorithm among the four tested for this particular application.

5 Discussion

We implemented two backends for dimensionality reduction to the `crabsort` spike-sorting toolkit. Both backends are integrated into the GUI, and exist as methods of the `crabsort` class.

The first backend, for fast Fourier transform-interpolated t-SNE, proved to be the simpler to implement, since the source code is in C, and calling C code from MATLAB is trivialized by use of MEX functions. To our surprise, however, FIt-SNE proved much slower than the native t-SNE algorithm. We suspect that this limitation would be mitigated for larger datasets, as the FIt-SNE algorithm uses a very conservative optimization procedure to account for the interpolation effect. In small datasets, early exaggeration, another feature of FIt-SNE, does not strongly affect the time to converge. For random, uncorrelated datasets, FIt-SNE embeds faster than native t-SNE for datasets with more than 10,000 datapoints.

UMAP performed very well, making explicable clusters in comparable time to native t-SNE. While both UMAP and t-SNE take much longer than PCA, the PCA embedding is unsuitable for manual clustering.

However, using the UMAP algorithm in MATLAB requires some workarounds, since the source code is in Python. MATLAB does not work well with Anaconda. Solutions which used to work very well, such as `condalab`, to switch conda environments in MATLAB, no longer work. In fact, `condalab` may actually break working solutions. To use UMAP in MATLAB with Anaconda, it is necessary to set the conda environment, then run MATLAB from the same terminal session. Furthermore, it is imperative that `.mat` files containing options or data are saved as version 7.3 files *MAT-File Level 5 File Format* 2015, or modules for reading HDF5 files, such as `h5py` will not be able to load the files.

Acknowledgments

Thank you to Dustin Clark at Research Computing Services, Boston University, for his help debugging Python-MATLAB compatibility, and to Srinivas Gorur-Shandilya at Brandeis University, for helming the `crabsort` project.

References

Bernhardsson, E (June 18, 2019). *Approximate Nearest Neighbors in C++/Python Optimized for Memory Usage and Loading/Saving to Disk: Spotify/Annoy*. Spotify. URL: `https://github.com/spotify/annoy` (visited on 06/19/2019) (cit. on p. 3).

Gerhard, Felipe et al. (July 11, 2013). "Successful Reconstruction of a Physiological Circuit with Known Connectivity from Spiking Activity Alone". In: *PLOS Computational Biology* 9.7, e1003138. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003138. URL: `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003138` (visited on 06/18/2019) (cit. on p. 1).

Kispersky, Tilman, Gabrielle J. Gutierrez, and Eve Marder (May 25, 2011). "Functional Connectivity in a Rhythmic Inhibitory Circuit Using Granger Causality". In: *Neural Systems & Circuits* 1.1, p. 9. ISSN: 2042-1001. DOI: 10.1186/2042-1001-1-9. URL: https://doi.org/10.1186/2042-1001-1-9 (visited on 06/18/2019) (cit. on p. 1).

Linderman, George C. et al. (Mar. 2019). "Fast Interpolation-Based t-SNE for Improved Visualization of Single-Cell RNA-Seq Data". In: *Nature Methods* 16.3, p. 243. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0308-4. URL: https://www.nature.com/articles/s41592-018-0308-4 (visited on 06/18/2019) (cit. on pp. 1, 3).

MAT-File Level 5 File Format (Dec. 16, 2015). URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000440.shtml (visited on 06/28/2019) (cit. on p. 5).

McInnes, Leland, John Healy, and James Melville (Feb. 9, 2018). "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: arXiv: 1802.03426 [cs, stat]. URL: http://arxiv.org/abs/1802.03426 (visited on 06/18/2019) (cit. on pp. 1, 3).

Pachitariu, Marius et al. (June 30, 2016). "Kilosort: Realtime Spike-Sorting for Extracellular Electrophysiology with Hundreds of Channels". In: *bioRxiv*, p. 061481. DOI: 10.1101/061481. URL: https://www.biorxiv.org/content/10.1101/061481v1 (visited on 06/18/2019) (cit. on p. 1).

Quiroga, Rodrigo Quian (Jan. 24, 2012). "Spike Sorting". In: *Current Biology* 22.2, R45–R46. ISSN: 0960-9822. DOI: 10.1016/j.cub.2011.11.005. URL: http://www.sciencedirect.com/science/article/pii/S0960982211012541 (visited on 06/18/2019) (cit. on p. 1).

Rossant, Cyrille et al. (Apr. 2016). "Spike Sorting for Large, Dense Electrode Arrays". In: *Nature Neuroscience* 19.4, pp. 634–641. ISSN: 1546-1726. DOI: 10.1038/nn.4268. URL: https://www.nature.com/articles/nn.4268 (visited on 06/18/2019) (cit. on p. 1).

Van Der Maaten, Laurens (Jan. 2014). "Accelerating T-SNE Using Tree-Based Algorithms". In: *J. Mach. Learn. Res.* 15.1, pp. 3221–3245. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2697068 (visited on 06/18/2019) (cit. on pp. 1–3).