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Another strategy, implemented in the software package `crabsort` (S. Gorur-Shandilya *et al.*, unpublished), uses pre-sorted data to train a neural network model which can automatically sort the remaining data.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`.

3.3 Style

Papers to be submitted to NeurIPS 2018 must be prepared according to the instructions presented here. Papers may only be up to eight pages long, including figures. Additional pages *containing only acknowledgments and/or cited references* are allowed. Papers that exceed eight pages of content (ignoring references) will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2018 are the same as since 2007, which allow for $\sim 15\%$ more words in the paper compared to earlier years.

Authors are required to use the NeurIPS LaTeX style files obtainable at the NeurIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

3.4 Retrieval of style files

The style files for NeurIPS and other conference information are available on the World Wide Web at

`http://www.neurips.cc/`

The file `neurips_2018.pdf` contains these instructions and illustrates the various formatting requirements your NeurIPS paper must satisfy.

The only supported style file for NeurIPS 2018 is `neurips_2018.sty`, rewritten for LaTeX 2_ε. **Previous style files for LaTeX 2.09, Microsoft Word, and RTF are no longer supported!**

The LaTeX style file contains three optional arguments: `final`, which creates a camera-ready copy, `preprint`, which creates a preprint for submission to, e.g., arXiv, and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

New preprint option for 2018 If you wish to post a preprint of your work online, e.g., on arXiv, using the NeurIPS style, please use the `preprint` option. This will create a nonanonymized version of your work with the text "Preprint. Work in progress." in the footer. This version may be distributed as you see fit. Please **do not** use the `final` option, which should **only** be used for papers accepted to NeurIPS.

At submission time, please omit the `final` and `preprint` options. This will anonymize your submission and add line numbers to aid review. Please do *not* refer to these line numbers in your paper as they will be removed during generation of camera-ready copies.

The file `neurips_2018.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in Sections 4, 5, and 6 below.

4 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points. Times New Roman is the preferred typeface throughout, and will be selected for you by default. Paragraphs are separated by ½ line space (5.5 points), with no indentation.

The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow ¼ inch space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in Section 6 regarding figures, tables, acknowledgments, and references.

5 Headings: first level

All headings should be lower case (except for first word and proper nouns), flush left, and bold.

First-level headings should be in 12-point type.

5.1 Headings: second level

Second-level headings should be in 10-point type.

5.1.1 Headings: third level

Third-level headings should be in 10-point type.

Paragraphs There is also a `\paragraph` command available, which sets the heading in bold, flush left, and inline with the text, with the heading followed by 1 em of space.

6 Citations, figures, tables, references

These instructions apply to everyone.

6.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated…

If you wish to load the `natbib` package with options, you may add the following before loading the `neurips_2018` package:

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{neurips_2018}
```

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous."

6.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[2]

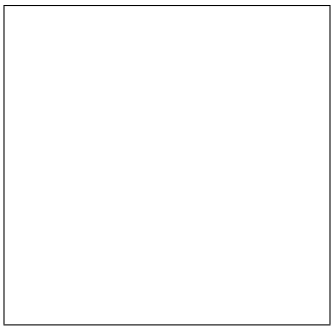

Figure 1: Sample figure caption.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

6.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

6.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 1.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 1.

7 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

[1] Sample of the first footnote.

[2] As in this example.

8 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NeurIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- `xfig` "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

8.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can use more than eight pages as long as the additional pages contain *only* cited references.**

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).

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).