Lee Lab Data Processing Pipeline

Hal Rockwell 6/4/21

June 7, 2021

Abstract

A guide to the data processing pipeline used to convert recorded neural data into a practically usable state.

1 Basic overview of file types

The general structure of an electrophysiology experiment (as usually performed in the Lee Lab) is that the experimenter shows stimuli to the monkey in a series of trials, and records the neural activity from a number of electrodes in response to those stimuli. Most of the work is in processing that recorded neural activity, but you also need to align it with the correct stimuli, so files involving both types are necessary.

First, the stimuli. For a given recording session, there is a `.ITM` file, which is a text file listing (among other things) the filenames of each stimulus presented, and an ID number for each stimulus (ranging from 1 to the total number of stimuli). This is necessary for aligning the stimuli with their corresponding neural responses, when the neural data is processed. For the work that I have done, this has been the only stimulus-relevant file I need. There are also `.CND` and `.SET` files, which are text and binary respectively—you shouldn't need them for basic work.

Second, the neural data, which is more complicated. The recording equipment produces at least 3 "raw data" files for a given recording session: a `.NEV` file, a `.NS2` file, and a `.NS6` file. The `.NS*` files are less processed, containing the complete voltage trace for the whole recording session for each electrode (`.NS2` typically at 1 KHz, and `.NS6` typically at 30 KHz). The `.NEV` file, containing just the voltage traces around threshold passings (spike candidates), is all you will need, unless you are trying to do something fancy like extract LFPs. The `.NEV` file is spike sorted, remaining in `.NEV` form, and then converted to a CDTTable data structure in a `.mat` file in Matlab. This is where you can stop processing, or you can go further and convert it to an N-dimensional array, also saved in a `.mat` file (but easily convertable to Python data formats).

To convert the sorted `.NEV` file to a `CDTTable`, two additional files are needed, in order to align the recorded spikes with the proper stimulus IDs and trials. These are both text files with the extention `.prototxt`, the "parameter" file and the "template" file. The parameter file contains information on how to extract each trial from the `.NEV` file, and the template file contains information on how to parse the data corresponding to a particular trial from the `.NEV` file. These files should be provided along with the neural data, as they are necessary for the processing to work. In the event that they aren't, it may be possible to construct them with some trial and error (and reuse of template files from recordings around the same time), but this is quite difficult, and potentially error-prone.

2 Detailed walkthrough of neural data processing

2.1 Sorting the `.NEV`

There are a few methods of automatic spike sorting one can use (and it can even be done by hand), but this guide will focus on using the "batch sort," as has been fairly standard in the lab. The code for the batch sort is on the lab Github[1], and that repository contains some documentation in the file `SAC_doc.pdf`. However, for basic usage, you should only need to call `sac_batch` with two arguments:

[1] https://github.com/leelabcnbc/batch-sort

the first a cell list of the filenames of the `.NEV` files you are sorting, and the second a string containing the path to the directory that holds those files. This can be done interactively in the Matlab console, or with a short script. Importantly, the batch sort overwrites the original `.NEV` files with the sorted version, so you should make sure to keep a copy of the data somewhere else that you don't run the sort on.

There is more functionality in the repository, including a GUI that allows hand-sorting. However, the code is very old, and it might need slight modifications to work in recent versions of Matlab—I haven't tested it. It shouldn't be necessary for basic work, though.

2.2 From `.NEV` to `CDTTable`

Once you have a sorted `.NEV` file, you need to convert it to a Matlab data structure that's easier to work with. The lab has been using the `CDTTable` since around 2015, and a fair amount of Matlab code (particularly Summer's) is written to work with them. First, you need to install Yimeng's neural analysis toolbox[2], following the instructions in the README file there. Then, you need to locate your `.NEV` file(s), parameter file, and template file, load them in in your Matlab script, and call `import_NEV_files` with the appropriate arguments. On the next page is an example script (from the `gaya-data` repository on the lab Github[3]) that does the conversion on some files—you can use its basic structure for your own processing, and just modify the filepaths used to be appropriate to your situation.

2.3 From `CDTTable` to Matlab arrays

This step is optional, but you might find it more convenient to work with data in a different format than the `CDTTable`. Particularly, a lot of neural data lends itself to a multi-dimensional array format, with 4 axes: stimuli, trials, neurons, and time (the entries at each point represent the neural response, usually a 1 or 0 indicating the presence of a spike). If you want, you can use the Matlab function `ArrayFromCDT`, in the `data_processing` folder of the `gaya-data` repository on the lab Github, to convert your `CDTTable`s to arrays. This function is quite straightforward, just needing two arguments: the `CDTTable` object returned by the previous step, and the time (in milliseconds) of each trial.

3 Summary

To reiterate: the basic pipeline is a raw `.NEV` file, sorted by batch sort, then converted to a `CDTTable` (and possibly further into a Matlab array) using its corresponding parameter and template files and Yimeng's neural data analysis toolbox. This results in a data structure that is convenient for use in Matlab (and it's not hard to use the arrays in Python either). The order of the stimuli in the processed data will correspond to the stimulus files laid out in the corresponding `.ITM` file.

[2] https://github.com/leelabcnbc/yimeng_neural_analysis_toolbox
[3] https://github.com/leelabcnbc/gaya-data

```
%%% convert the Tang NEV files to CDTTables

% first setup the path
% couldn't get MATLABPATH env variable to work right
% so you might need to modify this if your toolbox is in a weird spot
import import_NEV.import_params_dir
import import_NEV.import_NEV_files

% for loading and saving
base_path = mfilename('fullpath'); % full path to this file
base_path = base_path(1:end-12); % get rid of the filename itself
base_nev_path = fullfile(base_path, '..', 'data', 'tang', 'batch', 'raw')
    ;
base_cdt_path = fullfile(base_path, '..', 'data', 'tang', 'batch', 'cdts'
    );

param_path = fullfile(base_path, 'gaya_params.prototxt');
template_path = fullfile(base_path, 'gaya_template.prototxt');


% collect the path of each NEV file into a cell array
nev_list = dir(fullfile(base_nev_path, '*.nev'));
nev_list = {nev_list.name};
% full filepath for each one
full_nev_list = {};
for i = 1:length(nev_list)
    full_nev_list{i} = fullfile(base_nev_path, nev_list{i});
end

% this is mostly copied from the import_NEV demo
% but first we copy over the template files to the right spot
copyfile(param_path, fullfile(import_params_dir(), 'gaya_params.prototxt'
    ));
copyfile(template_path, fullfile(import_params_dir(), '..', '
    trial_templates', 'gaya_template.prototxt'));

% these are needed to get the start and end times of each trials, which
    are successful, etc.
proto_class = 'com.leelab.monkey_exp.ImportParamsProtos$ImportParams';
params_prototxt = fullfile(import_params_dir(), 'gaya_params.prototxt');
import_params = proto_functions.parse_proto_txt(params_prototxt,
    proto_class);

% now convert to CDT
% (should print number of successful trials for each file as it's
    processed)
cdt_tables = import_NEV_files(full_nev_list, import_params);

% and save the results (in each own file)
for i = 1:length(nev_list)
    table = cdt_tables{i};
    save([base_cdt_path '/' nev_list{i}(1:end-3) 'mat'], 'table');
end
```