Using Kilosort, Phy, and MClust

# Setting up Kilosort

### Installation

Download and install:

1) MATLAB 2017a

Need the Parallel Computing Toolbox (for mexcuda)

Older versions might work too? This is just the version I’m using.

Versions <2016 probably won’t work though without a patch.

2) Microsoft visual studio 2015

<https://my.visualstudio.com/Downloads?q=visual%20studio%20community%202015>

You have to make a Microsoft account and sign up for a Visual Studio Dev Essentials membership

Otherwise you’re not able to download versions older than 2017

3) CUDA Toolkit 8.1

<https://developer.nvidia.com/cuda-downloads>

Make sure to get version 8.1

4) Kilosort

<https://github.com/cortex-lab/KiloSort>

Download the repo as a .zip and extract to MATLAB/Kilosort/

Such that the files are in MATLAB/Kilosort/KiloSort-master

5) npy-matlab

<https://github.com/kwikteam/npy-matlab>

Download the repo as a .zip and extract to MATLAB/Kilosort

Such that the files are in MATLAB/Kilosort/npy-matlab-master

### Compilation

1) Open Matlab.

2) Add MATLAB/Kilosort and subfolders to the path.

3) Run

**mexcuda -setup**

And select

Microsoft Visual C++ 2015 Professional (C)

4) Run

**mexGPUall**

You’ll probably get the following warning:

Warning: The selected C++ compiler is not supported for CUDA compilation. Searching for a supported compiler.

And that’s OK as long as it then says:

Building with 'NVIDIA CUDA Compiler'.

If you get errors at this point see KiloSort\Docs\readme\_win\_linux.txt, which has some suggestions.

And if that doesn’t help then ¯\\_(ツ)\_/¯

Kilosort should now be ready for use.

### Test

To test that Kilosort was correctly installed and compiled, they have a script which generates some test data and runs Kilosort on it.

First, test that matlab can work with the GPU at all by running the following command (It shouldn’t take more than 2 min to run and shouldn’t throw any errors. Again if it does see KiloSort\Docs\readme\_win\_linux.txt for suggestions)

**gpuDevice(1);**

Next edit the script which runs the Kilosort test to match your path.

1) In Matlab, open MATLAB\KiloSort\KiloSort-master\eMouse\master\_eMouse.m

2) Change fpath to a directory where you would like to have the script store the test data (will be around 1.65 GB).

3) On line 7, change the directory to the location of the KiloSort folder

(MATLAB\KiloSort\KiloSort-master)

4) On line 8, change the directory to the location of the npy-matlab folder

(MATLAB\KiloSort\npy-matlab-master)

5) On line 9, change the directory to the location of the eMouse folder within the KiloSort folder

(MATLAB\KiloSort\KiloSort-master\eMouse)

Finally, run the script to test Kilosort by running (should take 2-5 min):

**master\_eMouse**

# Setting up Phy

### Installing Phy

1. Install Anaconda
   1. The most recent version here should work (I think? Haven’t tried it)
      1. <https://www.continuum.io/downloads>
   2. But if it doesn’t, I used Anaconda 4.2.0 w/ Python 3.5.2, which can be downloaded here:
      1. <https://repo.continuum.io/archive/>
2. Download Phy as a .zip and extract it (to whatever folder you want)
3. Make a Phy environment within Anaconda
   1. Open an Anaconda Prompt
   2. Navigate to the directory where you downloaded Phy
   3. Run the following commands at the prompt:

**conda env create -n phy**

**activate phy**

**pip install phy phycontrib**

# Setting up MClust + ADRLAB code

1) In the Kilosort\KilosortToMClust directory, compile SpikeFilter by running in Matlab:

**mex SpikeFilter.c**

2) Move LoadIntanSpikes.m from the Kilosort\KilosortToMClust directory to MClust\LoadingEngines

3) Move feature\_Peak15to25.m from the Kilosort\KilosortToMClust directory to MClust\Features

4) Add the entire Kilosort directory to your Matlab path.

To run Kilosort on your data:

1. Make a channel map for your rat (see next section)
2. Open the IntanToKilosort\Kilosort\_script.m file and
   1. Set the raw data directory (pathIn),
   2. the output data directory (pathOut),
   3. what sensors to do (32Si, 24TT, etc),
   4. the location of your configuration file,
   5. and the SSNs to run on.

For more detailed instructions, see the following sections.

# Sorting Intan Tetrode data w/ Kilosort, Phy, and MClust

### Run Kilosort on Tetrode data

First make sure you have the correct channel map for your rat.

See the “ADRLAB Channel Map File Format” section below,

Also example channel maps in the Kilosort\ChannelMaps folder.

Put your rat’s channel map in

Path\To\PromotedDataDir\R###\R###\_channel\_map\_<sensor>.txt

Where <sensor> can be 24TT, 32Si, etc

In Matlab, run Kilosort on the raw Intan data with:

**run\_KiloSort(‘Path\To\RawIntanData’, ‘Path\To\OutputDataDir’, …**

**‘Path\To\OutputDataDir\R###-YYYY-MM-DD-KilosortRaw-24TT.dat’, …**

**‘Path\To\channel\_map\_24TT.txt’, …**

**‘Path\To\KilosortConfigFile\_24TT.m’);**

See KiloSort\IntanToKilosort\Kilosort\_script.m for an example. This script also allows you to easily run Kilosort on multiple days/sensors. Be sure to use the Kilosort config file for tetrodes, not for Si probes. This will take an hour or two to run.

### Use Phy to sort clusters into cells/noise

Open an Anaconda Prompt

Navigate to the data directory for KiloSort-ed data

If your data directory is in a different drive than your OS drive (e.g. it’s in D:\ or F:\ or something instead of C:\), then at the command prompt, type the letter of the drive followed by a semicolon, e.g.:

**F: <enter>**

At the prompt, run

**activate phy**

**phy template-gui params.py**

Select clusters in the ClusterView panel (upper left). Work your way through all the clusters, classifying them as either good (Alt-G) or noise (Alt-N) or multi-unit activity (Alt-M). For each cluster, you can:

* View the waveform in the WaveformView panel. Zoom/scale w/ right mouse button + click+drag. Toggle showing several example waveforms and the average waveform with the ‘w’ key.
* Compare to other clusters by selecting clusters in the SimilarityView panel (sorted by clusters which are most similar to the one you’ve selected in the ClusterView panel).
  + Ensure the two waveforms look different (to ensure they are two separate cells)
  + Ensure the two clusters are not overlapping in the FeatureTemplateView or FeatureView windows.
  + Ensure the cross-correlogram between the two clusters doesn’t have a huge valley @ 0 (which indicates the two clusters may be the same cell)
* Merge clusters which appear to be the same unit by pressing the ‘g’ key. This will merge the cluster you have selected in the ClusterView panel and the cluster you have selected in the SimilarityView window.
* Split clusters in either the FeatureTemplateView or FeatureView panels by Ctrl-clicking around the points you want to separate, then pressing the ‘k’ key.

The full Phy documentation is at <http://phy-contrib.readthedocs.io/en/latest/template-gui/>

TraceView

Alt+DownArrow - shrink the waveforms on the Y axis (might need to use this)

Alt+LeftArrow - move viewing window left in the recording

Alt+RightArrow - move viewing window right in the recording

### Save .spikes and .clu files from Kilosort/Phy output

To save spike times and waveforms (.spikes) and what cluster they belong to (.clu), in Matlab run:

**ExtractSpikesFromKilosort(...**

**‘Path\To\OutputDataDir\R###-YYYY-MM-DD-KilosortRaw-24TT.dat’, …**

**‘Path\To\RawIntanData’, …**

**‘Path\To\OutputDataDir’, ...**

**‘Path\To\chanMap.txt’);**

See KiloSort\KilosortToMClust\KilosortToMClustScript.m for an example. This will take 2-3 hrs to run.

### Manually adjust clusters and save .t files from MClust

To manually adjust the clusters found by Kilosort, add the MClust directory to your path, then open MClust by running in Matlab:

**MClust**

Load the spikes and clusters by:

1. Select “LoadIntanSpikes” from the drop-down box in the upper left of the MClust window
2. Click the “Create/Load FD files” checkbox
3. Select and open the .spikes file for the tetrode you want to load
4. Click the green “Select from KKwik” button
5. Select and open the corresponding .clu file for the tetrode you want to load
6. Toggle the keep/toss button for each cluster depending on whether you want to include it
7. Click “Exit (Export)”

Click the green “ManualCut” button to touch up the clusters if desired.

# Sorting Intan Si Probe data w/ Kilosort + Phy

### Run Kilosort on Si Probe data

First make sure you have the correct channel map for your rat. See the “ADRLAB Channel Map File Format” section below, and example channel maps in the Kilosort\ChannelMaps folder.

In Matlab, run Kilosort on the raw Intan data with:

**run\_KiloSort(‘Path\To\RawIntanData’, ‘Path\To\OutputDataDir’, …**

**‘Path\To\OutputDataDir\R###-YYYY-MM-DD-KilosortRaw-32Si.dat’, …**

**‘Path\To\channel\_map\_32Si.txt’, …**

**‘Path\To\KilosortConfigFile\_32Si.m’);**

See KiloSort\IntanToKilosort\Kilosort\_script.m for an example. Be sure to use the Kilosort config file and channel map for Si probes, not for tetrodes. This will take an hour or two to run.

### Use Phy to refine clusters

Use Phy in the same way as described above in the Tetrodes section.

Again, the full Phy documentation is at <http://phy-contrib.readthedocs.io/en/latest/template-gui/>

### Save .t files from Kilosort/Phy output

After sorting/merging/splitting the clusters from the silicon probe in Phy, write the spike times from each cluster to a .t file by running the following in Matlab:

**SaveSpikeTimesFromKilosort(‘Path\To\RawIntanData’, ‘Path\To\OutputDataDir’);**

Now you can load the spike times from the Si probe with the LoadSpikes function.

# 

# 

File Formats

# ADRLAB Spikes File Format for Intan system (.spikes)

Stores spike times and waveforms from Intan recordings in a binary file with the following format:

Note: “double” means float64

-------------------------------------

uint32 - Number of spikes

uint16 - Number of channels

uint16 - Number of samples per spike (length of spike waveform window)

double - spike1 time

...

double - spikeN time

int16 - spike1 channel1 sample1

int16 - spike1 channel1 sample2

...

int16 - spike1 channel1 sampleN

int16 - spike1 channel2 sample1

...

int16 - spike1 channelN sampleN

int16 - spike2 channel1 sample1

...

int16 - spikeN channelN sampleN

-------------------------------------

# ADRLAB Channel Map File Format

A matlab code file (can be saved as .txt or .m) which stores the mapping from Intan port/channels to TT (or Si Probe) numbers. Below, a “sensor” refers to a recording device with multiple channels (e.g. a single tetrode is a sensor, or one shank of a silicon probe is a sensor). The file specifies the following variables:

**fs** - sampling frequency in Hz (scalar)

**tets** - list of tetrode/probe numbers (Ntetrodes-length vector)

**channel\_map** - Mapping from tetrodes to Intan channels, such that if you want data for TT t, chan c, get amp-{port(t,c)}-{channel\_map(t,c)}.dat (Nsensors-by-NchannelsPerSensor array)

**port** - Mapping from tetrodes to Intan ports, such that if you want data for TT t, chan c, get amp-{port(t,c)}-{channel\_map(t,c)}.dat (Nsensors-by-NchannelsPerSensor array of chars)

**shank** - Mapping from tetrodes to shank number (Nsensors-by-NchannelsPerSensor array)

**headstage** - (optional) Mapping from sensors to which headstage they were recorded on (Nsensors-by-NchannelsPerSensor array)

**xcoords** - X-coordinates of each recording site (Nsensors-by-NchannelsPerSensor array)

**ycoords** - Y-coordinates of each recording site (Nsensors-by-NchannelsPerSensor array)

**connected** - boolean array specifying what channels were not dead. Channels which were dead should have connected(thatChannel)=false. (Nsensors-by-NchannelsPerSensor array)

Example channel map file:

--------------------------------------------------------------------------

fs = 30000; %30kHz  
tets = 1:24;  
channel\_map = [ ...   
 24 25 26 27; ...% TT 1 (1.1 1.2 1.3 1.4) PORT A (HS 1)  
 28 29 30 31; ...% TT 2  
 ...  
 47 46 45 44; ...% TT 23  
 43 42 41 40]; % TT 24  
port = [ ...  
 'A' 'A' 'A' 'A'; ... % TT 1  
 ...  
 'B' 'B' 'B' 'B']; % TT 24

shank = ones(24,4).\*(1:24)';  
headstage = [ones(8,4); 2\*ones(8,4); 3\*ones(8,4)];

xcoords = zeros(24,4);  
ycoords = ones(24,4).\*(1:24)'; ycoords = ycoords + repmat(0.025\*[-1 0 1 2], 24, 1);  
connected = true(size(channel\_map)); %default is none are dead  
connected(15:17,:) = false; %TT15-17 were dead  
--------------------------------------------------------------------------

# ADRLAB Spike Time File Format (.t)

Stores spike times for a single cell. A binary file containing just uint32s, each of which is the time (in 10s of microseconds) of a spike.

-------------------------------------

uint32 - (Time of spike 1 in seconds) \* 10000

uint32 - (Time of spike 2 in seconds) \* 10000

...

uint32 - (Time of spike N in seconds) \* 10000

-------------------------------------

# KiloSort File Formats

### Channel Map

Stored in fpath\chanMap.mat, it’s a matlab .mat file storing the following variables:

* chanMap
  + 1 x Nchannels vector
  + Which channel corresponds to each row in the raw data file
* Connected
  + 1 x Nchannels logical vector
  + Which channels are valid / not dead / in use
  + E.g. w/ 2 TTs where the 3rd channel on the 1st TT is dead: logical([1 1 0 1 1 1 1 1])
* Fs
  + Scalar
  + The sampling frequency of the raw data
  + E.g. 30000
* kcoords
  + 1 x Nchannels vector
  + Which shank/tetrode each channel belongs to
  + E.g. w/ 2 TTs: [1 1 1 1 2 2 2 2]
* xcoords
  + 1 x Nchannels vector
  + X-coordinates of each recording site
  + E.g. w/ 2 TTs: [0 1 1 0 0 1 1 0]
* ycoords
  + 1 x Nchannels vector
  + Y-coordinates of each recording site
  + E.g. w/ 2 TTs: [1 1 0 0 1 1 0 0]

### Raw Data for KiloSort

Binary file with the raw data which is read in by KiloSort. Just a sequence of int16s with the data in

int16 t=1 channel1

int16 t=1 channel2

...

int16 t=1 channelN

int16 t=2 channel1

…

int16 t=N channelN

### Rez

Output Clusters file from KiloSort (rez.mat)

rez.st3

Nspikes-by-5 matrix, where each row is a different spike

rez.st3(:,1) - **Spike Times**

First column of rez.st3

rez.st3(:,2) - **Spike templates**

Which spike template was used for each spike. Aka the cluster IDs.

Don’t understand how that’s different from spike cluster IDs, below, but the Spike template values are exactly the same values as Spike Cluster IDs except for about 16% of spikes in the simulation example which have different values...

rez.st3(:,3) - **Spike Amplitudes**

rez.st3(:,4) - ?????

rez.st3(:,5) - **Spike Cluster IDs**

This is only added in the example code, where it’s the auto-merged clusters. St3 usually only has 4 columns.

rez.Wraw

Template waveforms (Nchannels x Nsamples x Ntemplates)

### All the .npy files

**spike\_clusters.npy**

A nSpikes-length vector with the cluster identities of every spike. Load by:  
 cid = readNPY(fullfile(folderName, 'spike\_clusters.npy'));

**spike\_times.npy**

An nSpikes-length vector with the spike time of every spike. Load by:

ss = readNPY(fullfile(folderNames{f}, 'spike\_times.npy'));

**cluster\_groups.csv**

cids is a length nClusters vector specifying the cluster IDs that are used  
 cgs is a length nClusters vector specifying the "group" of each cluster:  
 0 = noise; 1 = MUA; 2 = Good; 3 = Unsorted

[cids, cgs] = readClusterGroupsCSV(fullfile(folderNames{f}, 'cluster\_groups.csv'));

Useful docs:

<http://phy-contrib.readthedocs.io/en/latest/template-gui/>

<http://data.cortexlab.net/dualPhase3/>

<http://data.cortexlab.net/dualPhase3/data/script_dualPhase3.m>