1. Use gen\_train\_data.py to generate training data. Adjust parameters, such as

PEAK\_NPNTS\_THRESH

PEAK\_POLARITY

PEAK\_PRE\_CROP\_WIN

PEAK\_POST\_CROP\_WIN

Adjusting the display range using (DISPLAY\_WIN and yrange ) properly is crucial because this affects our perception of what is or isn’t a peak quite drastically. Output files are

cropped\_peaks.csv (reused if already exists)

peak\_vars.csv (reused if already exists)

analysis\_pars.txt

1. Append peak\_vars to cropped\_peaks to generate a combined file. Use classify\_peaks.py to cross-validate and predict whether a peak is selected or rejected and additional features (such as threshold\_rel\_x). classify\_peaks.py can be used for both but with different parameters. In particular, the following will need to be changed.

* label\_column
* clf (classifier). Peak selected or rejected is a classification task while features like threshold\_rel\_x are regression tasks. With the former one can use something like *ExtraTreesClassifier.* With the latter one can use svm.SVR.

The two can be done sequentially. It’s ok to predict thresh\_rel\_x (or other features) for peaks that are not selected because they don’t have target values anyways and so are not used for training.

**Note: The input file for classification of thresholds is the predicted file (output of classifying selection of peaks)**

*The low accuracy in prediction can also happen if the training data is noisy. Some very similar peaks might get labeled as peaks some times and not at other times by the same user.* **It is useful to have consistent criteria when generating training data.**

Output file are

* predicted.csv
* \*\_classific\_pars.txt where \* = label\_col.

1. Use cal\_abs\_thresh.py to calculate, thresh\_x, and thresh\_y thresh\_rel\_x. These are needed for plotting threholds as well as for calculating peak amplitudes. Overwrites on original file.
2. Use plot\_peaks to visualize selected and rejected peaks. Set same filter pars and display range.

**Calibration:**

Cross validation accuracy assumes that the training data is correct. But it's possible that multiple labeling of the same training data leads to different labels. The cross-validation should be compared against this human accuracy.

**Estimation of accuracy of training data:**

1. Use generate\_training\_calib\_datasets.ipynb to generate multiple training data sets from combined file obtained by combining cropped peaks and peak\_vars. Choose how many peaks to relabel and how many times. generate\_training\_calib\_datasets.ipynb generates 'num\_repeats' data sets the first of which has original selection values but the rest have NaN. The selection value of other peaks can be assesed using plot\_peaks.py in which detect\_features is set to empty list. For now the 0 or 1 for selection are entered manually in the sample data sets.
2. Use calibrate\_training\_accuracy.ipynb to get the accuracy tested against the original training data set.

**Following steps are specific to analysis of synaptic currents**

1. Use merge\_genotype\_info.ipynb to merge mouse info. Merged file overwrites data file.
2. Use pt\_psc\_freq\_and\_amp.ipynb to calculate peak amplitude and peak freq averaged for each cell. Uses only selected peaks. Outputs 2 files

* peak\_amp\_cell\_avg.csv
* peak\_freq\_cell\_avg.csv