Guide to Paul’s code and NDT use

The root folder for all code, data and figures is: /ldata/lspace/plevy

My code is divided into two folders: /analysis\_code and /helper\_code

The (modified) NDT code is in the folder: /ndt.1.0.2

Data (firing rates with labels, binned data, decoding results, etc.) is in: /data

Figures are in: /figures

# Brief Neural Decoding Toolbox (NDT) help

There are three important parts of the object that is used in the NDT:

* Data: A [#trials X #time measurements] matrix of firing rates
* Labels: The labels that describe the conditions present for each trial
* Site\_info: Extra information like RF position, from what reference point the data is aligned, etc.

The data is organized in subdirectories by within the folder data by monkey 🡪 the experiment conducted (attention always, for the foreseeable future) 🡪 area\_cell OR Population\_area\_alignment

The data is either in raster format (before processing) or binned format (after processing). Separate files are created for each (binned files are put in their own subfolder, /binned)

The result of decoding is also put in its own subfolder, /results

# My code

**Paths**: Make sure that /analysis\_code and /helper\_code (and any subdirectories) are added to path

## Helper Code:

This folder mainly contains functions for error checking, cleaning and saving the neural data as well as scripts for cleaning up folders.

### Folders

* attention\_code is a folder of untouched code for grabbing and processing the data (like dhfun an dh\_calc functions, for example). You shouldn’t ever have to edit this code
* bipolar\_colormap.zip\_FILES contains code for a nice color map, used in 2-D matrix-style plots. Again, this code shouldn’t need to be edited.
* Save\_072514 contains a backup of code. If anything ever goes horribly wrong, this will be a decent starting point!

### Functions (less important)

* check\_analysis\_validity: Called during plot/analysis functions to check if the data grabbed has the properties expected (based on parameters from function call).
* debug\_catalog\_trials: For debugging purposes, simply prints how many trials of the various conditions appear for a particular cell. Input is the raster\_labels struct (part of NDT objects)
* fix\_align\_case: Simple error check and case-fixing used in functions. Alignment refers to when data is aligned on saccade/stimulus: do the times used refer to the *start* of that data or the *onset* of the saccade/stimulus. Shouldn’t need to be changed/called.
* fix\_area\_case: Same as above, but for PITd/LIP
* fix\_monkey\_case: Same as above, but for Quincy/Michel
* fix\_ref\_case: Same as above, but for stimulus/saccade
* fix\_task\_case: Same as above, but for attention/MGS\_file
* monkey\_area\_cells: Should be currently unused, but can be useful; given a monkey and area, tells how many cells are in that recorded population.

### Functions (important but behind the scenes)

* create\_population\_folder\_func: Moves the pseudo-population cells from the folder of the parent cell to a folder for a cell population (population cells must have already been created!)
* get\_clean\_data\_name: Given the time reference point, alignment, time start, and length of time window, generates the file name for (cleaned) data (often used as directory name)
* get\_decoder\_labels: Generates the train and test labels used by the decoder given what to decode, the label to use (ex. rel\_phi\_brt, meaning relative coordinates where we specifiy phi AND brt), etc.
* get\_decoder\_labels\_separated: Same as above, but rather than grouping multiple labels together in cell(s) for use by decoder, each label (for ex. 0\_180, meaning phi=0, brt=180) can be indexed separately from the others in a cell containing all of the labels for the specified parameters
* get\_file\_name\_conditions: Generates file name used to save analyses for given conditions as well as for the title of plots.
* permute\_cell\_labels: Used when generating a population of cells, finds a given cell and saves a duplicate cell with the labels shifted. Change line 53 of this function if you want to generate a “full” population (i.e. RFs at 0, 90, 180, 270 for a given cell) since currently set to just generate a “mirror” neuron (RF at 180 w.r.t. original RF)
* permute\_cell\_labels\_func: Permutes labels of all cells for given monkey
* prepare\_cell\_labels\_for\_pop: Used for generating population of cells, simply replaces the absolute labels of all cells with the relative labels before these original cells are copied with the absolute labels shifted.
* prepare\_for\_populating: Called by the above to actual do the copying for a given cell.
* save\_clean\_data\_batch: Called by the “main” script (discussed later), this cleans a set of original data of all trials containing NaNs within the specified interval
* save\_clean\_neural\_window: Called by the above function, does the actual “cleaning” of one given cell
* save\_data\_batch: Called by the “main” script (discussed later), this grabs and saves a set of original data from the hd5/hdf files
* save\_neural\_data: This is to the above function as save\_clean\_neural\_window is to save\_clean\_data\_batch
* write\_analysis\_parameters: Given a struct containing the parameters of a given decoding, this writes a text file with information on the decoding. Called by analysis functions (for ex., the function that plots confusion matrix results)

### Scripts

* create\_population\_folder: Same as create\_population\_folder\_func but as script
* delete\_pop\_cells: Deletes all of the population cells (i.e. ones that had labels shifted) for a specified monkey
* permute\_cell\_labels\_script: Same as permute\_cell\_labels\_func but as script
* save\_clean\_data\_script: Same as save\_clean\_data\_batch, but required to specify the parameters (monkey, area(s), start time, etc.)
* save\_data\_script: Same as save\_ data\_batch, but required to specify the parameters (monkey, area(s), start time, etc.)

### control\_all\_steps\_loop: most important script

This script allows you to control all aspects of the decoding. It has seven sections:

* Save data from original files
* Clean data that has already been saved (and create a cell population)
* Run decoding (including normal, “special” (i.e. if decoding PHI, fix BRT 0 or 180 rather than allowing for both) and NULL decodings, controlled *separately*)
* Plot confusion matrix analysis (either normal or special)
* Plot analysis of mean decoding results (either normal or special)
* Plot above analyses, but comparing between LIP and PITd (either normal or special)
* Compute statistical values (currently just PITd vs. LIP significance, but should include “are the results for (area) above chance?”)

Before section 1, there are variables that allow you to control which of the steps are run. In each step, you’ll need to specify a few parameters (always the monkey). The decoding has the most parameters to set (like #resample\_runs, which classifier to use, if decoding a population, etc.) All analysis sections also need you to specify the monkey and whether or not population or single-cell decoding should be analyzed.

## Analysis Code:

This folder contains functions that compute values for plots, actually make and save plots as well as the code for running the actual decoding analysis. The functions here won’t need to be edited on a day-to-day basis, but if large changes are made or new decodings desired, you will have to edit here.

### Functions (confusion matrix)

* analyze\_confusion: Computes and plots the results of the confusion matrix for a certain monkey/area/train condition. Currently saves a .jpg AND a .svg
* analyze\_confusion\_population: Same as above, but for a population (ex. Quincy LIP)
* analyze\_confusion\_slice: Called by the two functions below, this actually determines the fraction that a certain condition was met (i.e. class i correctly guessed when presented) for a given confusion matrix (fixed time, fixed resample)
* analyze\_resample\_confusion: Sets up the confusion matrix analysis when resamples are used, analyzing the confusion matrix at all times in the trial
* analyze\_resample\_standard: Same as above, but for decodings without resampling
* get\_confusion\_analysis: Actually processes entire confusion matrix; Called in analyze\_confusion and make\_comparative\_conf\_plots
* get\_confusion\_analysis\_pop: Same as above, but for population (called in analyze\_confusion\_pop)

### Functions (mean decoding)

* mean\_decoding: Gets and plots the mean decoding for a given decoding
* make\_NDT\_mean\_plot: Calls NDT’s mean decoding plot; requires that NULL distributions have been generated, since the plot expects to compute and show where significance occurs
* make\_TCT\_plot: Plots cross-time confusion matrix by calling NDT function
* quick\_mean\_decoding: Plots just a single mean decoding curve

### Functions (comparative)

* make\_comparative\_conf\_plots: For a given condition, plots the decoding accuracy of LIP vs. PITd split for the classes of a given decoding (from confusion matrix)
* make\_comparative\_decode\_plots: Same as above, but just plots mean decoding results

### Functions (decoding)

* run\_single\_decoding\_analysis: Runs the actual decoding analysis for non-generalized decodings. NOTE: non-generalized in the NDT lingo means “I haven’t specified the train/test labels, but just indicated what labels (i.e. rel\_phi or abs\_phi\_brt) I will draw from”. Though it remains in code, it is effectively not used and you may run into problems if you do use it. But I hope that you don’t…
* run\_single\_decoding\_batch: Calls the functions directly above and below but for all cells/populations specified in the main script (control\_all\_steps\_loop)
* run\_single\_decoding\_generalization\_analysis: Runs the actual decoding when train/test labels are specified. Very important function that shouldn’t need to be edited unless large changes are made.

### Functions (help)

* determine\_num\_valid\_sites: For a given cell population (ex. Michel LIP), determines for the given train/test conditions and decoding parameters (namely num\_CV\_splits) how many valid cells there are
* get\_confusion\_matrix: Grabs the confusion matrix for the requested decoding
* get\_confusion\_matrix\_pop: Same as above, but for population rather than cell
* get\_decoder\_labels\_plot: Gets the decoder labels to be used when the confusion matrix plots are made for a given area.
* get\_decoding\_results: Same as get\_confusion\_matrix, but returns entire decoding results structure rather than just the confusion matrix
* shadedErrorBar: Downloaded Matlab function that creates plots with shaded error bars

### Functions (statistics)

* comptue\_significance\_across\_area: Determines if there is a statistically significant difference between the decoding accuracy of LIP and PITd.
* create\_resampled\_diff\_distr: Actually computes the vector of differences between LIP and PITd mean decodings for a given resample. Called by above function

# General

If naming/saving conventions are changed, a lot may go wrong! There are certain expectations as far as directory and file names go, and these assumptions are in functions that grab the decoding results for analysis/plotting. If you change these conventions, make sure you change code in all affected places: the actual saving/decoding, functions that grab decoding results, functions that save plots

Currently, the data (un-cleaned, cleaned, pseudo-population, binned [an intermediate step of decoding] and decoding results) are in:

/data/“monkey”/attn/”area\_cell\_no OR population\_alignment”/

Note that attn is the task (very early code was made such that MGS\_task could’ve been worked with, but I soon realized that we wouldn’t use this, so code just makes this assumption!)

The saccade/stimulus aligned un-cleaned data are in this directory. Then, there are subdirectories with the name: “align”\_”ref”\_”time\_start”\_”time\_length”\_”time\_type”\_clean, where

* align = saccade or stimulus (or anything new you make)
* ref = onset or start (w.r.t. alignment, though onset means that time 0 refers to the actual event, so start is sort of a useless refence point)
* time\_start is when, w.r.t. the reference, the given data starts
* time\_length is how long the time window is. If this value is 0, that means that a “slice” rather than a window of data was saved/cleaned
* time\_type: Unless the above is 0, this is window (otherwise slice)
* clean refers to this data being cleaned or being decoded from cleaned data

In *this* sub-directory, the cleaned data is saved directly. Then there is a folder for binned data (always just holds the binning from the most recent decoding) and a results folder. The results files always contain the most recent decoding. So, for ex. if you decoded PHI on absolute coordinates with train and test having BRT out, any previous decoding will be overwritten.

Of course e-mail me with any questions! [paul.gerald.levy@gmail.com](mailto:paul.gerald.levy@gmail.com) (in case it’s after I graduate from UM)