1 Introduction

In this programming project, we will deal with 1-D signals. We are going to look at the single-neuron recordings of responses to different tastes, and see if we can extract meaningful dependence of the neural response to a sensory stimulus. In other words, we will conduct a classification task to classify neural response to its corresponding stimulus (taste).

This assignment is quite open that you can try any models you want. You can design your own feature space, either by hand or by using a dimensionality reduction package. Then pick up a classification algorithm and run it on the dataset. You are encouraged to attempt novel approaches to do this project. However, repeating Di Lorenzo et al.'s work [1] is acceptable is you don't have any idea on it.

This assignment can be done individually or in pairs, though we strongly encourage you to work in pairs. You can use any language you want on any platform. (Although we suggest Matlab, C/C++, Java, Python, etc. on Linux, Mac OS or Windows, you can definitely choose your own environment for this assignment. The only requirement is you must make sure we can easily reproduce your result.)

2 Dataset

2.1 Overview on the Dataset

The data consist of single-neuron recordings from the nucleus tractus solitarius of the rat [1], obtained by Pat Di Lorenzo, Professor of Psychology, SUNY Binghamton. The recorded neurons convey signals from the taste receptors ("taste buds") in the tongue to the brain. The experiment consists of application of solutions that elicit the elementary tastes on the tongue, while recording the activity of a single neuron conveying information from a single taste bud.

The stimuli are labeled:

- 'N'=salty (NaCl)
- 'Q'=bitter (quinine)
- 'H'=sour (HCl)
- 'S'=sweet (Sucrose)

and also pairwise mixtures (NH, NS, NQ, HS, HQ, SQ), so there are a total of 10 "categories" of responses for each experiment (4 primary tastants, 6 pairwise mixtures).

There are 35 neurons represented in this dataset. The data have already been pre-processed, i.e., all the spikes have been identified and the dataset only contains spike times which are given in units of seconds. The stimulus is presented at time $t = 20$ sec and rinsed away at $t = 25$ sec. **But we will ONLY focus on the first 2 seconds after the stimulus, i.e. spikes between 20–22 sec in our project.**

For further details, you are (strongly) encouraged to see Lorenzo et al.'s freely available paper [1].

2.2 Data File Format

The original data are stored in a '.mat' file with Matlab structure. You can import the '.mat' file directly when you use Matlab to do this project. For other programming languages, we also provide a textual file and you need to read them from file and organize your own way to store them in memory.

There are 35 neurons (35 independent experiments) represented in the data file. Each neuron is tested with all 10 stimuli. Considering factors like noise, we repeated the recording for the same neuron and stimulus several time. But the number of trials may vary with different neurons and stimulus pairs. (The average number of trials is around 10) And the data file only contains spike times.

See more details about the data format below.

2.2.1 Matlab File

The data are contained in the matlab structure taste_data in 'CS5540_taste.mat'.

The label for neuron *icell*, category *c* can be found in: *taste_data{icell}.categories(c).label*. The labels can be one of N, H, S, Q, NH, NS, NQ, HS, HQ, SQ.

length(taste_data{icell}.categories(c).trials) is the total number of trials for the neuron *icell* and category *c*.

Finally, *taste_data{icell}.categories(c).trials(i).list* stores the time of spikes for the *i*-th trial of neuron *icell* and category *c*. It's a 1-D array only records the time in units of seconds. They are floating point numbers between 20 and 25 in ascending order, however, just as we mentioned before, we ONLY need to take the spikes between 20 to 22 into consideration.

For example, part of the data may look like:

```
>> taste_data{1}.categories(2).label
ans =
    'Q'
>> taste_data{1}.categories(2).trials(1).list
ans =
  Columns 1 through 9
   20.1848 20.2269 20.4848 20.5262 20.7015 20.7113 20.8100 20.9952 22.1983
  Columns 10 through 13
   22.7219 23.5990 23.7182 24.3811
```

2.2.2 Textual File

The data are stored in the textual file 'CS5540_taste.txt'.

Since we have 35 different neurons, there are 35 parts in the file. Each part starts with one line "Neuron # *icell*" to indicate the head of *icell*-th neuron followed by the data of that neuron.

There are 10 sections for each neuron data corresponding to different stimuli. Each section starts with one line of the label and the number of trials separated by a space, e.g., "N 9" means there are 9 trials for this neuron and salty stimulus. Then we describe one trial on one line, starts with the number of spikes n followed by n floating point numbers between 20 to 25 in ascending order recording the spike times.

For example, part of the data may look like (first few lines of a data section for one neuron and one stimulus):

```
Q 16
13 20.1848 20.2269 20.4848 20.5262 20.7015 20.7113 20.8100 20.9952 22.1983 22.7219 23.5990 23.7182 24.3811
5 20.3792 20.4081 20.4913 20.5614 22.7948
..... data for other 14 trials
```

Please see the data file for more details.

3 Assignment

3.1 Task I

Before our classification task, we will have a warm-up exercise. Note that we have 35 different neurons with different behavior, i.e., they will not respond in the same way to all the 10 tastes. Therefore our first task is to determine which tastes each neuron is sensitive to.

We observe that if a particular neuron is sensitive to one particular taste, it will generate many more spikes as response. (This does not imply 0 spikes for a taste that it is insensitive to!) We don't need to work out a

very fancy model here since this task is relatively easier. A threshold on the number of spikes is OK, or you can come up with something more elaborate.

In the report you turn in, describe the method you applied in this task and your result (e.g., a 35×10 table indicating whether each neuron is sensitive to each taste). Your experiments in the following task are based on these active neuron-taste pairs.

3.2 Task II

In this part, we are going to do a classification task, i.e., given the 1-D signal (in this dataset, time of spikes), determine what the stimulus is. This is a somewhat open-ended problem, but you will need to (1) design a classification technique, based on some feature representation computed from the data, and (2) come up with an evaluation technique and (assuming your solution has parameters) a parameter-tuning method.

A minimal solution to the problem would be to re-implement the work of Lorenzo et al., though we hope you will do something interesting on your own. They defined a kind of edit distance to quantify similarity between spike times. Then they applied standard multidimensional scaling (MDS)[1] to this distance to embed the 1-D signal responses (times of spikes) into a new space (feature space). While Lorenzo et al. didn't do this, it would be natural to apply a nearest neighbor classifier in this space.

While the above would provide a minimal solution, we strongly encourage you to try your own approach. Basically, this task can be done in three steps: feature representation, classification and evaluation. For example, in the first step, we can use a binning strategy to get the histogram of the spikes in a trial, or we can use MDS to project each trial into a feature space, etc. In the second step, we can use k-NN, SVM, etc.[2]

And please don't hesitate to show your progress in your report. For example, you might start from a naive approach, e.g., classifying signals just based on the number of spikes. Maybe this model is not good enough, and then you will work out a more sophisticated model with better performance. It's good to show it in your report.

Although we hope to put as few constraints as we can, here are some key points to keep in mind for this project:

- Each neuron should be considered **separately**, i.e., we need to build 35 models and evaluate them separately for these 35 neurons.
- We only work on the **active** neuron-taste pairs we obtained from the first task.
- We ONLY need to focus on the **first 2 seconds** of the response (although the data consist of 5 seconds).
- The project needs to be done in accordance with the obvious design principles that will prevent overfitting. We view the spikes in each trial as a data instance, and we need to split all the instances of the same neuron into two **disjoint** set, one training set and one test set. All the information you have to build the model comes from the training set[3]. We conduct the evaluation on the test set. If there is any model selection (parameter tuning) phase in your approach, you need to split the whole into 3 disjoint parts, training set, validation set and test set and conduct the model selection on validation set. Finally, please provide the quantitative evaluation result (e.g., accuracy on the test set) of the task.
- Since this dataset is a very small dataset, we strongly suggest you to conduct a **k-fold cross validation** (or even leave-one-out cross validation) in the third step to make the evaluation result more accurate

[1] You may find the build-in function 'mdscale' or 'cmdscale' in Matlab or an open source in Python at `http://orange.biolab.si/doc/modules/orngMDS.htm` helpful.

[2] Please note the choice of classification algorithm may depends on your feature representation. Furthermore, for some algorithms, like k-NN, we only need to define the distance, e.g., edit distance, between trials so that we don't have a clear boundary between the first and second step.

[3] The only exception may be when you want to utilize the semi-supervised framework, you can access the test data without a label.

and reliable. Please see Wikipedia (`http://en.wikipedia.org/wiki/Cross-validation_(statistics)`) for more technical details, though we will discuss this briefly in lecture.

You are supposed to describe your approach, the evaluation methods, the experimental results and your analysis in your report.

4 Assignment Submission

Please submit an archived file through CMS including your report, source code and any other necessary files to run your program (or links to some open sources and public tools) and, if applicable, an executable.

The requirements of report are described in each task specification. It is highly recommended a README file including build instructions about your code and everything else you want to tell us. And detailed building instructions are required if you believe it's not easy for us to run your code and reproduce your results.

5 Assignment Evaluation

The actual performance (accuracy) of your method is only a small part of our evaluation. So please don't feel worried if your model cannot get a very high accuracy. Basically, we will take the workload, quality of the report, novelty of the method, etc. into consideration. Simply repeating Lorenzo et al.'s work [1] will get you the majority of the points, and exploring a number of natural extensions will get you almost all the points. We will reserve a small number of points for a solution that exhibit some creativity as well.

6 Academic Integrity

Academic integrity is important in this course. You must follow the school's code (`http://cuinfo.cornell.edu/Academic/AIC.html`).

Since this is a programming project, we would like to emphasize the following rules:

- Having discussions with other people, using open sources and public tools, getting ideas from research papers is allowed, but proper citations and acknowledgements are required. Otherwise, any direct or indirect copy from other's work, Internet, etc. is strictly forbidden.
- All the results you reported in this project must be generated by your submitted programs.

Violations of academic integrity are taken very seriously. Please feel free to contact Professor Zabih if you have any questions or concerns about this topic, or if you feel there is any possibility that you may be violating the code of academic integrity.

References

[1] P. Di Lorenzo, J.-Y. Chen and J.D. Victor, *Quality time: Representation of a multidimensional sensory domain through temporal coding*, J.Neurosci **29**, 2009, pp. 9227–9238. `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766857/pdf/nihms136261.pdf`