BMI Spring 2020
Neural Decoding Competition Rulebook

1. Description of the Competition

In the coming weeks you will design a neural decoder designed to drive a hypothetical prosthetic device. This is a realistic and difficult task in brain-machine interfacing. The task will take the form of a competition to make it even more enjoyable and to encourage creative solutions.

You will be provided with spike trains recorded from a monkey's brain, as it repeatedly performs an arm movement task. Your algorithm will estimate, from these data, the precise trajectory of the monkey's hand as it reaches for the target.

This is a continuous estimation task. Over the course of each spike train, you must estimate the X & Y position of the monkey's hand at each moment in time (while the training data includes the Z position, you are not asked to estimate it). Caveats: "time travel" is banned – i.e. your decoder must be causal, it may not use information from the future to estimate the position of the hand at the present. When your algorithm is tested, it will be given data in chunks of increasing length in order to enforce this (see *5. Implementation*).

2. Competition Scoring & Prizes

There will be a small prize in this contest, which the team with the lowest error will win. The prizes are highly desirable though as-yet unspecified.

The following error metric will be used to evaluate the performance of the submissions: **your error will be the root mean squared error (RMSe) of the trajectories you compute.** The mean will be taken across both dimensions, over all trajectories. The units of this error are cm.

If you don't submit a code by the end of the 3rd lab session, your final RMSe score will be increased by 10% on the leader board for the Competition.

3. Teams and Deadlines

• Teams

This competition will be a team effort. Each team will consist of 3-4 students (we will accept exceptionally groups of 5 if well justified). You are encouraged to form the teams on your own, but we are happy to provide help in forming the teams. Any serious grievances or disputes regarding teams should be addressed to the GTAs or lecturers as early as possible.

• Report Guidelines

Each team will submit only ONE final report. A template is available on Blackboard. The file will have ALL of the team members' names written at the very TOP of the file. Group reports due on the day of the BMI exam. They need to be uploaded on Turnitin. BlackBoard will not accept submission after this time. Feedback to be provided within two weeks.

Structure the report as described in the attached Template: maximum 4 sides of two- column A4 paper. This includes references. All team members should co-author the report: you should meet to plan how you will coordinate the writing. We encourage groups to use the free online collaborative Latex editing tool Overleaf, which includes the template we are using. To keep your report within 4 pages, you will need to write concisely: this is good practice. Do not go over length – 10% will be deducted from your marks if you do so. Also, please stick to the recommended font and font size. A 10% penalty mark will be taken off the report if the team doesn't participate to the competition.

The neural data have been generously provided by the laboratory of Prof. Krishna Shenoy at Stanford University. The data are to be used exclusively for educational purposes in this course.

The contributions of each member should be concisely noted in a section at the end. If team members do not contribute, note this as well. Order your authors alphabetically in the author list. Team members will normally receive the same, joint, mark for the report, however we reserve the right to bump individual members marks up or down. All members should upload a copy of the group report to Blackboard: the feedback will be marked up on the first author of the report (author with surname closest to 'A'): that person should share it with the other members of the group.

Describe the approach you took to solving the problem. Describe your methods and results, including figures and/or tables describing the performance of your algorithm. In the Discussion, critically discuss the performance of your algorithm and how it could be improved. How does it compare against other approaches? A First-Class report would be, essentially, of a standard acceptable for submission to a conference such as the International IEEE EMBS Conference on Neural Engineering.

- Algorithm submission process
  - During the lab sessions, please submit your training function to the GTAs along with your position estimator function (i.e. two separate functions) as two separate files, preferably in a zip file. Please name the .zip file with your team name.
  - You will need to make sure your functions work. You can test this by running it through the test functions provided on Blackboard. The format required is further described in the comments of the template functions.
  - Your algorithm must take no longer than 5 minutes to run, as tested.
- Deadlines
  - Homework (see *A. Pre-Comp Homework*). – To finish by the end of the first session, 20th of February 2020.
  - SUBMISSION: Algorithm – 11 am GMT, Thursday 19th March 2020.
  - SUBMISSION: Group report – 5pm on the day of the BMI exam.

4. Data

Overall Content

The data contains spike trains recorded from 98 neural units while the monkey reached 182 times along each of 8 different reaching angles, as well as the monkey's arm trajectory on each trial. On each trial, both the neural data and arm trajectory are taken from 300 ms before movement onset until 100 ms after movement end. The trials of different reaching angles were interleaved; and the neural data includes both well-isolated single-neuron units (~ 30% of all units), as well as multi-neuron units. The set has been divided into a training dataset of 100 trials per reaching angle and a test file of 82*8 trials. The former can be found on Blackboard as a *.mat* file, while the latter is not accessible by students.

.mat File

The *.mat* file has a single variable named *trial*, which is an array of dimensions 100(trials) x 8(reaching angles). The *trial* array contains 100*8 *struct* data structures which contain 3 fields: *trialId*, *spikes*, *handPos*. The first is the trial ID number and is therefore a of little consequence. The latter two are arrays containing the neural data and the hand position traces respectively.

Neural Data

The spike train data takes the form of a time-discretized sequence of zeros and ones with time steps of 1 ms. Each value in the array therefore takes the binary form 0/1 and represents the presence or absence of a spike in its neural unit in the 1 ms time window, or *bin*. A zero indicates that the unit did not spike in the 1 ms bin, whereas a one indicates that the unit spiked once in the 1 ms bin. Thus, a spike train of duration T ms is represented by a $1*T$ vector. The spike train recorded from the i^{th} unit on the n^{th} trial of the k^{th} reaching angle is accessed as: *trial(n,k).spikes(i,:)* where $i = 1, \ldots, 98$, $n = 1, \ldots, 100$, and $k = 1, \ldots, 8$.

- ### Trajectory Data

In this task, the monkey reached to targets on a fronto-parallel screen. Most of the arm movement was in the plane of the screen along the horizontal directions: *handPos(1,:)* and *handPos(2,:)*. The movement perpendicular to the plane of the screen, *handPos(3,:)*, was relatively small.

The three-dimensional arm trajectory recorded on the n^{th} trial of the k^{th} reaching angle is contained in *trial(n,k).handPos*, which is a $3*T$ matrix of the hand position (in mm) at each 1 ms time step. On each trial, the data in *spikes* and *handPos* are aligned in time.

The indices $k = 1, \ldots, 8$ correspond to the reaching angles (30/180π, 70/180π, 110/180π, 150/180π, 190/180π, 230/180π, 310/180π, 350/180π) respectively. The reaching angles are not evenly spaced around the circle due to experimental constraints that are beyond the scope of this problem set.

5. Implementation

The competition will be run in Matlab. While you are welcome to use other programming languages for your own edification, non-Matlab submissions will not be graded.

Your submission for each task will take the form of a MATLAB function. The function will accept two arguments: a set of training data (that includes reaching angles and trajectory data), and a set of test data (which only has spike trains and are not included in your training data). Your function will then return the appropriate angle classification and position estimation.

You will be given a template (accessible through Blackboard), including a function declaration with a description of the arguments it must accept and the data structure it must return. You may change the variable names but not the function name nor the order of the arguments.

The model must be causal. To enforce this, the model will be scored at regular intervals of 20 ms: to start, the first 320ms are fed in, then the first 340 ms, then the first 360, 380, 400, 420… until the whole recording is eventually fed in to your model. Reading the code and comments in the template will clarify the implementation significantly. The test script run by the GTAs has the same format and feeds data into your algorithm in the same way but uses the test data to evaluate your model.

6. Inviolable Laws

- You MUST cite every resource you use.
- Your submissions MUST implement the interfaces in the Matlab templates exactly as specified.
- You MUST put the names of ALL team members on the top of EVERY file you submit.
- While you may use existing (cited) algorithms, all submitted Matlab code MUST be entirely your own. Exceptions are the Matlab template files, any code posted by the lecturers or GTAs on Blackboard, and the calling of standard Matlab functions. Calling functions from Matlab toolboxes such as the Statistics and Machine Learning Toolbox or the Deep Learning Toolbox is not allowed.
- Late submissions will be ignored. Your submission may be updated online, so it is suggested that you submit early and then continue updating your submission until the deadline if you so desire.

7. Final bit of Advice

Feel free to use whatever algorithms you like, as long as you write the code yourself. In particular, you are encouraged to attempt to design your own. You have also been exposed to several algorithmic ideas throughout the lectures, and you may mime research papers, professors, textbooks, etc. The sky is the limit. **All code submitted must be your own and plagiarism checks are implemented.**

Be warned, however, that more mathematical artillery doesn't necessarily result in a better solution. Consider implementing a simple idea early on and then taking time to improve on it, rather than using up all your time designing a complicated but untested idea. Remember also that the ideas described in research papers rarely work exactly as specified.

A. Pre-Competition Homework

Before you start with the BMI competition you will try to get comfortable with the dataset by playing around with it. You will visualize parts of the dataset by writing and performing calculations in Matlab.

First, download the *monkeydata_training.mat* file from \Course Content\The Competition location in Blackboard. Start Matlab and set the working directory to the location of the *monkeydata_training.mat* file. Load the data set by either dragging the file into the Matlab workspace or by using the following command:

```
>> load('monkeydata_training.mat');
```

Read the section 4 'Data' from this document to familiarize yourself with the fields in your workspace.

Once you are comfortable with the data fields in your workspace, try the following exercises:

1) Familiarize yourself with raster plots and for a single trial, compute and display a population raster plot. A population raster would have time (bins) on the x-axis and neural units on the y-axis. A sample raster plot can be found here:

   http://en.wikipedia.org/wiki/File:Sample_raster_plot_from_Brian_neural_network_simulator.jpg

2) Compute and display a raster plot for one neural unit over many trials (suggestion: try using different colours for different trials to help with the visualization).

3) Familiarize yourself with what peri-stimulus time histograms (PSTHs) are and compute these for different neural units. Refer to http://icwww.epfl.ch/~gerstner/SPNM/node7.html for more information on PSTHs. Try smoothing your histogram to get a continuous rate estimate.

4) Plot hand positions for different trials. Try to compare and make sense of cross-trial results.

5) For several neurons, plot tuning curves for movement direction. Tuning curves measure directional preference of individual neural units. Different neural units will have different directional preferences.

   Get the tuning curve, by plotting the firing rate averaged across time and trials as a function of movement direction. To get an idea of the variability, compute the standard deviation across trials of the time-averaged firing rates. You can use the Matlab command '*errorbar*' to plot the tuning curve with error bars indicating the variability.

6) Think about other ways that you can play around and visualize the data in order to understand the nature of the experiments.

7) If you are feeling clever: implement the population vector algorithm and use it to predict arm movements.