BE 521: Final project part 1

Spring 2021

Adapted by Thomas Campbell Arnold & Adam Rayfield

32 points

Due: Wednesday, 4/2/2021 10:00 PM

Objective: Predict finger flexion from ECoG recordings.

Project Overview

This final project involves predicting finger flexion using intracranial EEG (ECoG) in three human subjects. The data and problem framing come from the 4th BCI Competition (Miller & , 2008). For the details of the problem, experimental protocol, data, and evaluation, please see the original 4th BCI Competition documentation (included as separate document). The remainder of the current document details your deliverables for part 1 of the project. You should submit 6 files: the five .m files specified in the "writing your code" section below, along with this document filled out and compiled as (pennid)_part1.pdf

Important Deadlines

Date	Requirement	Points
Friday, 4/2, 10:00pm	final project part 1	32
Friday, 4/5, 10:00pm	team registration	5
Friday, 4/9, 10:00pm	pass testing set checkpoint 1	20
Wednesday, 4/19, 10pm	pass testing set checkpoint 2	15
Friday, 4/23, by 10pm	end of competition, submit algorithm (Canvas)	15
Wednesday, 4/28, 10pm	hand in final report	60
Wednesday, 4/28, 1:30-3pm	competition results (Final class session)	

The grading is structured so that going the extra mile is definitely rewarded. We want you to show what you've learned this semester, and to have some fun!

Dataset

The dataset has been uploaded to the IEEG Portal and can be accessed as follows:

- Subject 1
  - `I521_Sub1_Training_ecog` - Training ECoG
  - `I521_Sub1_Training_dg` - Training Data Glove
  - `I521_Sub1_Leaderboard_ecog` - Testing ECoG
- Subject 2
  - `I521_Sub2_Training_ecog` - Training ECoG

  - `I521_Sub2_Training_dg` - Training Data Glove
  - `I521_Sub2_Leaderboard_ecog` - Testing ECoG
- Subject 3
  - `I521_Sub3_Training_ecog` - Training ECoG
  - `I521_Sub3_Training_dg` - Training Data Glove
  - `I521_Sub3_Leaderboard_ecog` - Testing ECoG

Your task is to develop an algorithm to use the ECoG to predict finger flexion that is captured by the Data Glove.

Writing your code

To get you started with the final project we have provided a series of method stubs for you to fill out. Your job for part 1 of the final project is to build a prediction pipeline that takes in the ECoG and dataglove finger angle recordings (serving as the data and labels respectively), then uses machine learning methods to generate predicted finger angles from the ECoG signals. The files you will develop in this assignment are as follows:

- $final\_project\_part\_1.m$. This script should run the entire prediction pipeline. Here you will load the data, call the $getWindowedFeats$ function, as well as ultimately train and test your machine learning classifiers.
- $getWindowedFeats.m$. This function will take in raw ECoG data, and use the three following helper functions to filter the data, calculate sliding-window features, and ultimately return a response matrix to use in model training and testing (see section on optimal linear decoder).
- $filter\_data.m$. This function will apply a filter to the raw data and return cleaned data.
- $get\_features.m$. This function will take in a window of cleaned data and return a vector of features for that window.
- $create\_R\_matrix.m$. This function will take in a feature matrix and return a response matrix as an adaptation of the optimal linear decoder method (see following section).

Optimal Linear Decoder

You will use the *optimal linear decoder* method as described in Warland et al., 1997. We will recapitulate the method in this section [1], but consult the paper for more details. Our ultimate goal is to predict the angle of each finger as it moves over time using data recorded from the ECoG channels. The position data is captured for 300 seconds, which you will split up into M total time bins, and the number of ECoG channels, ν, is 62, 48, and 64 for subject 1, 2, and 3 respectively. The paradigm we adapt here tries to predict finger angle at a given time window using ECoG features calculated over the preceding N time windows, using the following steps: First features will be calculated across all ν ECoG channels x M total time windows. Then, following the approach that Warland et al., 1997 takes, we will construct a row vector corresponding to each time bin, that contains features for all the ECoG channels over the preceding N time bins (in the paper, spike counts are their features and they index neurons instead of ECoG channels). Thus, there will be a good amount of redundancy between row vectors of adjacent time bins, but that is okay.

[1] We have changed minor notation in a few places for clarity. Our subscripts start at 1, theirs at 0. Also, here our index j refers to ECoG channels from 1 to ν.

Let r_i^j be the value of the feature of channel j in time bin i. Let the response matrix $\mathbf{R}$ be defined as

$$\mathbf{R} = \begin{bmatrix} 1 & r_1^1 & r_2^1 & \dots & r_N^1 & r_1^2 & r_2^2 & \dots & r_N^2 & \dots & \dots & r_1^\nu & r_2^\nu & \dots & r_N^\nu \\ 1 & r_2^1 & r_3^1 & \dots & r_{N+1}^1 & r_2^2 & r_3^2 & \dots & r_{N+1}^2 & \dots & \dots & r_2^\nu & r_3^\nu & \dots & r_{N+1}^\nu \\ 1 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & r_M^1 & r_{M+1}^1 & \dots & r_{N+M-1}^1 & r_M^2 & r_{M+1}^2 & \dots & r_{N+M-1}^2 & \dots & \dots & r_M^\nu & r_{M+1}^\nu & \dots & r_{N+M-1}^\nu \end{bmatrix}$$

This is also referred to as the design or feature matrix, with each column being a predictor, or feature. The column of 1's accounts for the intercept term in linear regression/decoding. Make sure you understand what this matrix means before moving on. We denote the target matrix [2] (e.g. the M x 5 matrix of finger angles) as $\mathbf{Y}$ and the reconstruction (e.g. the predicted finger angles) as $\hat{\mathbf{Y}}$. Solving for the minimum least-squares difference between the stimulus and reconstruction, $(\mathbf{Y} - \hat{\mathbf{Y}})^T(\mathbf{Y} - \hat{\mathbf{Y}})$, we get the analytic form for the optimal filter,

$$\mathbf{f} = (\mathbf{R}^T\mathbf{R})^{-1}(\mathbf{R}^T\mathbf{Y}) \tag{1}$$

This equation should take a familiar form. Warland et al., 1997 don't refer to it as such, but this is *exactly* the same as linear regression, one of the most commonly used algorithms in practical machine learning. Not only is this algorithm remarkably powerful, but it has a beautiful analytic form for learning the "weights" (here, the $\mathbf{f}$ matrix), a rarity in a field where almost all optimizations involve some sort of iterative algorithm. After learning the filter weights $\mathbf{f}$, we can calculate the optimal predictions as

$$\hat{\mathbf{Y}} = \mathbf{R}\mathbf{f} \tag{2}$$

1 Getting Started (4 points)

The following sections will walk you through the development of the prediction pipeline.

1. Extract dataglove and ECoG data for each subject in the associated section of *final_project_part_1.m*, then split the data into a testing set and training set (at least 50% of the data should be a part of the training set). How many samples are there in the raw ECoG recording (it is the same for all subjects)? (2 points)
   Answer Here

2. Next, complete the *filter_data* function. Test it using the raw data extracted in the prior step. What filter type(s) and cutoff frequencies did you use? (2 points)
   Answer Here

2 Calculating Features (12 points)

Here you will complete the *getWindowedFeats* and *getFeatures* functions.

1. We will calculate features across sliding time windows. If we use a suggested window_length of 100 ms with a 50 ms window_overlap, how many feature windows, M, will we have if we computed features using all the data in a given subject (1 point)?
   Answer Here

2. Now complete the *getFeatures* function. Please create 4 OR MORE different features to calculate for each channel in each time window. Features may include the average time-domain voltage, or the average frequency-domain magnitude in consecutive 15Hz frequency bands (N.B., the *conv* and *spectrogram* functions, respectively, may be helpful for this.) (8 points).

3. Now finish the *getWindowedFeats* function by putting the *filtered_data* and *getFeatures* functions together to return a feature vector for each time window (3 points).

[2] In Warland et al., 1997, this quantity is referred to as the stimulus vector since they are talking about decoding the stimulus from neural data after it. We, on the other hand, are trying to decode finger positions using the ECoG data before it, but we can conveniently use the same method.

3 Creating the Response Matrix (6 points)

In this section, you will develop code for your *create_R_matrix* function.

1. For our set of 62 channels in subject 1, what would the dimensions of the R matrix be if we calculated 6 different feature types per channel, and $N = 3$ time bins where the number of total time bins M is the number you calculated in 2.1? (1 pt)
   Answer Here

2. We do not have feature data to fill out the first N-1 data rows in the R matrix that will be used to predict the first N-1 finger angles. One way to work around this is to append a copy of the first N-1 rows of your feature matrix to the beginning of your feature matrix before calculating R. Make this adjustment in *create_R_matrix.m*, then compute the response matrix R. You can test whether your function is running correctly by running *create_R_matrix* with data from *testRfunction.mat* and verifying that the quantity mean(mean(R)) is 25.4668 (5 points).

4 ML Training and Testing (10 points)

Here we will use the optimal linear decoder framework to predict finger angles, and additionally you will use one or more classifiers of your own choosing to make the prediction.

1. Update *final_project_part_1.m* to include outputs from *getWindowedFeats* and *create_R_matrix*. Next, calculate the linear filter (i.e. the weights matrix) $\mathbf{f}$ as defined by Equation 1 for all 5 finger angles using features calculated from your training data. You will have to first down-sample the finger flexion data so that your feature matrix, $\mathbf{R}$, and your flexion data have the same number of time windows. (Note: instead of using the matrix inverse of $\mathbf{R}^T\mathbf{R}$, use Matlab mldivide function, which is generally more stable.) (4 points)

2. Try one other machine learning classifier using your features and finger angle labels. Look back through previous homeworks to get some ideas. (4 points)

3. Produce predictions on your test set for each finger angle. Calculate the correlation coefficient ρ between predicted and test finger angles for each finger separately. (Hint: see Matlab's corr function). Report your correlations here for each finger when using the linear filter, and when using the other classifier(s) that you tried (2 pts).
   Answer Here