BE 521: Final Project Part 2– Competition!

Spring 2021

Adapted by T. Campbell Arnold

March 26, 2021

Objective: Predict finger flexion from ECoG recordings.

Important Deadlines

Date	Requirement	Points
Friday, 4/2, 10:00pm	final project part 1	30
Monday, 4/5, 10:00pm	team registration	5
Friday, 4/9, 10:00pm	pass testing set checkpoint 1	20
Monday, 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	Prizes

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!

1 Competition

Registration, submission and the leaderboard will be at `http://be521.pythonanywhere.com/`

1.1 Registration

1. You must form a team one to three people. If you do not have a team, use the Piazza team building function to reach out for team members.

2. Each team will register by filling out a registration form at the website above. Please use your Penn email addresses. Spaces and other normal characters are allowed in the team name, but be creative!

1.2 Leaderboard

1. You will use the "Testing ECoG" data for each subject (`leaderboard_data.mat`) found in the Final Project folder on canvas to create your Data Glove predictions to submit to the leaderboard. Submissions can be made on the website. The leaderboard will show whether you've passed the checkpoints as well as the current ranking.

2. Any member of the team can make submissions to the leaderboard using the registered email.

3. Leaderboard submissions must be: Matlab *.mat file with one variable: `predicted_dg`. This variable is a 3 x 1 cell array, where `predicted_dg{i}` corresponds to the predictions for subject i. [1]. Thus, `predicted_dg{1}` is a 147,500 x 5 dimensional matrix for the 147,500 time points and 5 fingers predicted from subject 1's testing ECoG.

4. The submission results will be refreshed every 6 hours at the beginning of the competition, and more frequently as deadlines approach. This is to help reduce overfitting to the supplied test set.

1.3 Evaluation

Performance will be measured by the average correlation coefficient across the 3 patients on fingers 1, 2, 3, and 5. [2]

1.3.1 Checkpoints

We will use your maximum achieved correlation to evaluate checkpoints.

1. Checkpoint 1 corresponds to achieving an average testing set correlation coefficient of $\mathbf{r \geq 0.33}$. The points for this checkpoint are all-or-nothing, i.e., you get all points if you pass it and 0 points if you do not.

2. Checkpoint 2 corresponds to achieving an average testing set correlation coefficient of $\mathbf{r \geq 0.45}$. The points for this checkpoint are broken down as follows: 5 points for $\mathbf{r \geq 0.40}$ and the remaining 10 points for $\mathbf{r \geq 0.45}$.

1.3.2 Final ranking

We will run your model on an independent test set for the final rankings and results will be announced at the final presentation.

1.4 Prizes

The prizes for the top three winners are

1st place: an automatic A in the **class** for each member of the team

2nd place: +5 points on the final **class grade**

3rd place: +3 points on the final **class grade**

Other awards will be given for creativity and the "Medal of Valor Award" for that team which kept trying in the face of adversity (however we define it).

2 Deliverables

In addition to the registration and checkpoints above, you will have to submit your algorithm and report. Only 1 of each needs to be submitted per team.

1. **Final Submission Script and Auxiliary files**
   (due ON CANVAS 1 hour after competition closes)
   Your team will have to complete and submit the provided `make_predictions.m` script that takes in all subject's ECoG and outputs a predicted data glove for each subject. See `make_predictions.m`

[1] These submission variables are slightly different from those required in the original BCI Competition. Be sure to follow our variable specifications.

[2] We do not test on the 4th finger because it is highly correlated with the 3rd finger

for more details. In this script, you should load your models from a `.mat` containing your model(s) for each subject, do the necessary testing, and return the predictions. You should not do any model training. Any libraries (e.g. LibSVM) that you use should also be included.

Any late submissions or errors that are encountered when the TAs run each team's code will result in automatic disqualification of the team from the competition. Yes, this means that even if you come in 1st place, if you submitted late or we had problems running your code, you will not get an automatic A. TA's will provide a test script and hold a session where you can make sure your algorithm runs.

2. **Final Report**
   Due: Wednesday, May 6th, 12:00 PM to Canvas by the final zoom meeting.
   NO LATE SUBMISSIONS ACCEPTED
   Teams must hand in a final report in person, 1 report per team. Make sure your team members are listed. The report should be 3-5 pages, 1.5 spaced (or similar) and should describe your final algorithm in detail. In particular, your report must have the following

   - a summary paragraph describing your final algorithm (5 pts)
   - a step-by-step, detailed explanation of your final algorithm (15 pts)
   - a flow chart summarizing the general steps of your algorithm (5 pts)
   - at least one interesting figure that illustrates part of your algorithm or motivates a particular decision you made along the way (5 pts)
   - a discussion of what methods you tried that did not work (10 pts)
   - some thoughts about why the fourth (ring) finger's flexion was generally so correlated with the third (middle) and fifth's (little) (5 pts)
   - a conclusion about your overall experience with the project and how well you felt you did on it (5 pts)
   - a references section listing the papers and other resources you used (5 pts)
   - an appendix with the code used in your final version (i.e. please do not include all the code you ever wrote, just what your finished solution used) (5 pts)

   In addition to these points, the report will be graded on the clarity of the writing and thoroughness of the explanation.

3 Honor code

While you are allowed to consult other sources, your team must write your own code. External libraries such as LIBSVM and all Matlab toolboxes are allowed.

4 Guidance

In this section, we first try to give you some general advice and then a description of a straightforward method that should get you well on your way to passing at least the first checkpoint.

General advice

1. Get started early! The sooner you pass the checkpoint(s), the sooner you can relax a little bit and have fun with the competition. Submit early and often as you make progress. Do not wait until the last minute to start submitting results.

2. Figure out a good system within your team to share and (if possible) version-control your code. Emailing versions back and forth can quickly become cumbersome and confusing. If you really want to do it right, you should set up a shared repository using (hyperlinks underneath)

   - SVN via CETs at UPenn
   - Github
   - BitBucket

   a repository not only provides a central home for your code, but it also provides easy version control, so you can look back in time and/or roll back the state of your code if need be.

   A simpler yet still good alternative would be to use some sort of shared network drive or folder so that at least your code is in one place. Free software like Dropbox is easy and even provides a bit of version control (through their website) if you need it. But, this can also lead issues such as conflict files when two people are working on the same file at the same time.

3. Document your code well (i.e. comments!). Good documentation will be invaluable to the rest of your team. It is also imperative when you're always going back to old code and trying to improve it. This habit will always serve you well. If your code is unreadable we will deduct points.

4. Organize your code well. Do not simply put it all in one big script. Split out small parts (like feature extraction, training, and making the predictions) into functions that you can easily call for each subject and/or each finger. You may want to have one main script (with code sections) which you can run that goes through all the steps, from loading the data to making the predictions. Talk to TA's if you'd like advice on code organization.

5. Save the results of intermediate processing if it makes sense. For example, you may want to save the results of your feature extraction (which can be time-intensive, depending on your features) to a file so that you don't have to do it again for the near future. That way you can easily share with your other team members and/or come back to your work another day (or Matlab session) and pick up in the middle where you left off.

6. Design. Build. Test. Iterate. Do not try to solve this problem all at once. Start simple and get something working, even if it's not working as well as you want. Once you have your general algorithm flow down (scripted!, see previous point) it's much easier to improve small pieces of it. You may want to keep a notebook or text-file log of current attempts and final performance metrics so it's easier to compare "versions" of your algorithm. Trust us, all of your ideas will not necessarily make your algorithm better, so you will want some quantitative way to compare versions.

7. Cross-validation is your friend. If you're doing any learning that has some tunable parameters involved, you will probably want to cross-validate. Even if you're not tuning parameters, cross-validation will give you a better sense of what your generalization performance (i.e., testing performance) will be. Training performance can be deceptively good.

8. Be creative and clever. Remember, everyone else is probably thinking of the same obvious things to do as you. Keep in mind some of the areas/skills you have at your disposal, including (but by no means limited to)

   - neuroscience: timescales of motor planning
   - feature extraction: sliding windows, interpolation
   - signal processing: time and frequency-domain features, convolution, smoothing
   - unsupervised learning: clustering, dimensionality reduction (e.g., PCA)
   - supervised learning: classification

- prediction: linear regression/prediction

The best methods will almost certainly combine a number of these tools.

9. Consult outside references. You're probably not the first person who has tried to do motor prediction from ECoG data (although there aren't a ton!), so see what other people have tried as a good grounding and/or jumping off point. Some recent papers of Gerwin Schalk (who gave the P300 lecture and was one of the designers of the BCI Competition) might be a good place to start.

10. Start working on a small chunk of data, perhaps a half or even a quarter of the training data for one subject. Things will go faster that way when you're getting your algorithm up and running. Once you think you have a reasonable algorithm, then expand what data you are working with.

11. You may find it useful to sometimes have multiple Matlab sessions running. Running Matlab in the terminal is much more lightweight than a big graphical session and so might be an option if your OS is having trouble with multiple graphical sessions.

12. You may also find it useful to print status messages to the console (using `disp` or `fprintf`) to tell you what your code is currently doing, e.g. where it is in a loop. For example, when our code is doing feature extraction, it prints out the current subject ("Subject 1", "Subject 2", etc) and then a little dot (.) after processing each channel. Such feedback can sometimes make longer computations more bearable because you have a better idea of where you are in them and when they will finish.

A Simple Method To help get you on your way with this problem, below we outline a simple method adapted from Kubanek *et al.* (J Neural Engineering, vol 6, 2009). This is the method we have prepared you to use in Final Project Part 1
For each subject individually,

1. We used a moving window 100 ms in length with 50 ms overlap and extracted the same 6 features over each of the EEG channels. The features were the average time-domain voltage, and the average frequency-domain magnitude in five frequency bands: 5-15 Hz, 20-25 Hz, 75-115 Hz, 125-160 Hz, 160-175 Hz. (N.B., the `conv` and `spectrogram` functions, respectively, were very helpful for this.) Thus, the total number of features in a given time window was $62(5+1) = 372$ for a subject with 62 EEG channels.

2. We downsampled the dataglove traces (using the `decimate` function) so that each sample was separated by 50 ms, to keep them on the same time scale as the features.

3. We used linear prediction (linear regression) to predict each (downsampled) finger flexion from all the EEG features from the previous three time windows (so 150 ms lag). This process was basically the same as what you did in the motor prediction homework. Note, for matrix
boldsymbolX representing the EEG data (same as the
boldsymbolR matrix from the motor-prediction homework) and target matrix $\boldsymbol{Y}$ (where each column is a different finger flexion trace), the formulae for the filter coefficients is still the same: $\boldsymbol{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$. The weights $\boldsymbol{\beta}$ are now a matrix instead of a vector, but that means that since we have the same $\boldsymbol{X}$ for each finger, we can predict all finger positions at once with $\hat{\boldsymbol{Y}} = \boldsymbol{X}\boldsymbol{\beta}$.

4. We then interpolated the prediction using a cubic spline (see the `spline` function) back up to the original 1000 Hz sampling frequency, making sure that the first and last points in the data interpolation were values we know (i.e., not values we needed to interpolate, since the cubic spline often blows up at the edges if not constrained). The interpolation was zero-padded and the beginning and end to time-align with the original flexion trace.

This method achieves an average correlation coefficient of at least $r = 0.33$ on the testing set.