BME 6710 Project 1 P300 Speller

2/21/2024

Lexi Reinborough and Ashley Heath

Part I. Introduction

The P300 speller is a brain-computer interface designed to take readings of brainwave data using an electroencephalogram, or EEG, and turn it into letters and numbers. An EEG records electrical signals in various parts of the brain, and that data can be used to determine when each part of the brain is active. The P300 speller displays a 6x6 grid of letters and numbers and periodically flashes rows and columns. By looking at the desired character, the subject being measured will have a neurological response whenever that character flashes, and not so much when others do, and we can correlate these responses to the flashes based on when they happen. This type of interface could be very useful for people with limited motor function. This code analyzes data recorded by subjects who were instructed to look at specific characters, and uses statistical methods to determine whether the data are reliable enough to use in practical applications. Understanding how robust a system like this is can inform our design choices surrounding the tradeoffs between speed and accuracy. We want our speller to be able to accurately identify the correct character near 100% of the time, but not be so infrequent as to be unusable.

Part II. Methods

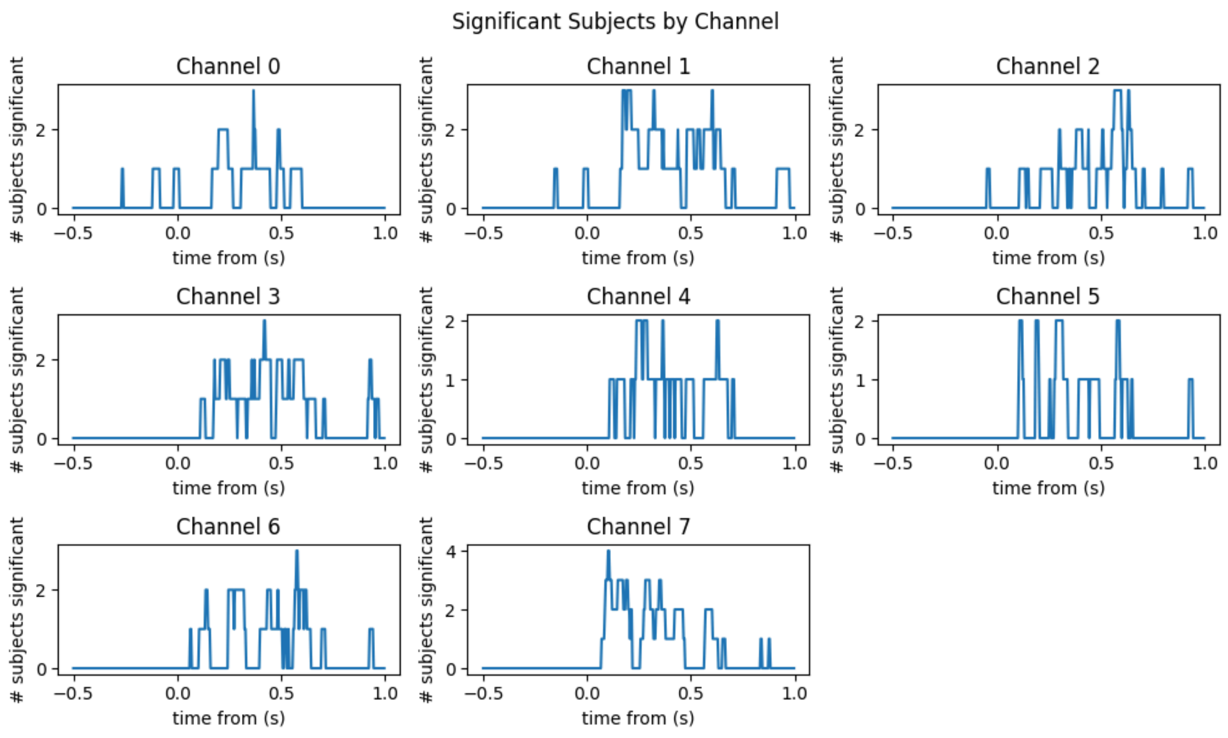
Our data is raw eeg readings from 10 subjects. Subjects numbered 3-10 are the ones using our variation of the speller. Each subject has 8 channels of data, which represent 8 signals from different parts of the brain. We must collect data surrounding each flash of any row and column, so we get a sense of what was happening in the brain before and after. These time periods are called ‘epochs.’ We separate the data into ‘target’ (epochs where the flashed row or column was the correct one) and ‘nontarget’ (epochs where it wasn’t.) by taking the mean and the standard error around each time across all epochs, we can get a sense of what all the targets and nontargets looked like as a whole.

The first function is erp_by_subject, which is a convenience function which takes a subject index and returns epoched eeg data, as well as the time offsets where each data point in each epoch occurred, and the means and standard deviations of the eeg data. This is a lot of return values, but they all turn out useful.

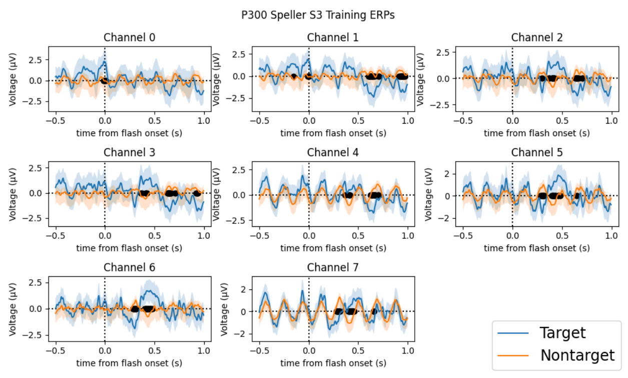
The next three functions are related to bootstrapping. Bootstrapping is a method where we learn about our data by randomly sampling from it repeatedly, and creating a table of synthetic data which is representative of the possibilities if our targets and nontargets were to exhibit no differences.
bootstrapERP is the base function, which does the random sampling. It takes data and a size parameter, so that we can simulate the number of targets and nontargets using the whole dataset. It returns the mean across the samples (but maintaining the time and channel data.)
bootstrap_iter takes the means of the target/nontarget epochs, and calls bootstrapERP twice to simulate target and nontargets, then subtracts them to find the difference (again maintaining the time and channel axes.) It returns the absolute value of this difference, because our investigation is into how different they are, not which one is bigger.
bootstrap is the top-level function, taking in the means of the epochs and the epochs themselves and running 3000 iterations of bootstrap_iter. We attempted to speed up the runtime, but attempting to do the bootstrapping all in one step used a surprising amount of memory for the random indices. This function determines where in the distribution of synthetic instances the real differences fall, then corrects for the false discovery rate and returns those as p values.
plot_erps_and_stats plots the means and 95% confidence intervals of the erps, and places a dot wherever the corrected p value is less than 0.05 and saves it to a file. It uses the means of the nontarget/target erps, the std errors of each, the time offsets, the p values, and a subject number for file naming purposes. It returns nothing.
evaluate_across_subjects bootstraps all subjects from 3 to 10, and does this internally, requiring no input or output. It plots each subject’s stats, and adds up the number of subjects for which the target and nontargets were significantly different for each time point and each channel, then plots those totals in a file.

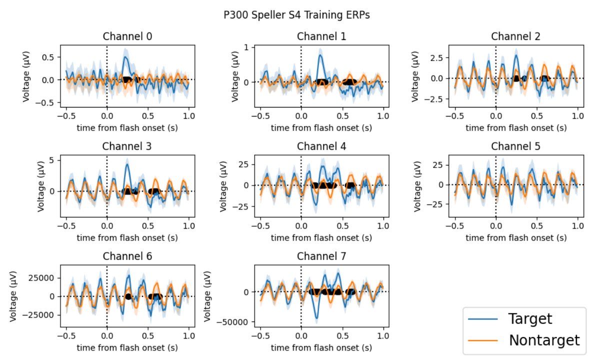
spatial_map also needs no inputs or outputs, as it simply creates an image with the n2 and p3b scalp plots for each subject 3 to 10 and saves it to a file.

Part III. Results

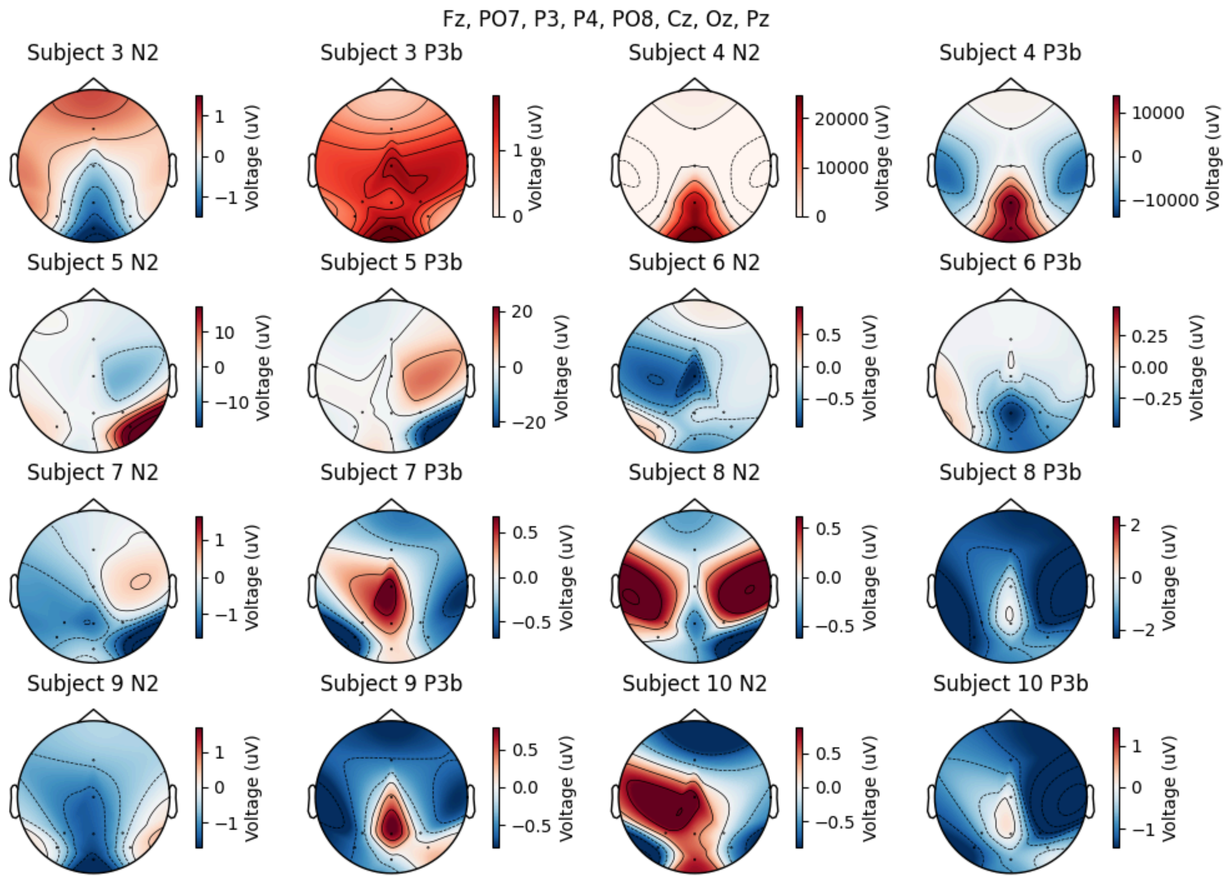


From this we can see that across subjects, their brainwaves exhibited similarly significant responses across channels. There are several channels with very clear spikes of significance. There are also some anomalies, such as some channels having some ‘statistically significant differences’ before the onset of the flash, but this plot shows us that there are patterns in the data, even if some of it is noisy or faulty.





These examples of subjects show that not all the data are reliable. Subject 4 in particular exhibits some large anomalies, but other subjects can be subtly inconsistent with one another.



We chose to plot targets only, as non-targets should theoretically not have n2 or p3b responses. This is with the channel numbers assigned to 'Fz', 'PO7', 'P3', 'P4', 'PO8', 'Cz', 'Oz', and 'Pz', respectively. We arrived at this by looking at the median values we wanted to plot for each p3b, and tried to balance them out, pairing up similar values on opposite hemispheres, giving the largest value to Pz, (which should have the largest response,) and the lowest to Fz, and doing my best to create smooth interpolation around that. We used time ranges of 390ms to 520ms for p3b and 210 to 270 ms for n2. Subjects 4 and 6 seem to be anomalies, and different time ranges seem better for different subjects, but this is the best we could come up with.

Part IV. Discussion

Channels 0, 2, 3, and 5 (hypothesized to be Fz, P3, P4, and Cz) appear to me, based on surveying the scalp maps and the significance plots, to be the most consistent in their behavior following a target. And the time periods surrounding 275 and 450 ms after the event seem to have the clearest distinct behavior. We would argue that this doesn’t warrant throwing the other channels away, as more data is almost always more useful in painting a clear picture of what is going on, and this information can be gleaned with machine learning or simpler models which weigh more information than just certain channels at certain time points. If we are set on choosing channels and time points, We would say it is imperative to mold that to the user, as every person is different, everyone’s brain responds differently, and these spellers will not have a large enough target market to justify the kind of mass production which might warrant creating a one-size-fits-all approach. It might be good to have a default setting, but any product will ideally learn its user’s tendencies, and calibrate accordingly.