Contents

Introduction

The P300 is a characteristic waveform in the human EEG, occurring as a response to rare task-relevant stimuli in a series of task-irrelevant stimuli [3]. This paper explores various feature engineering and machine learning techniques for achieving high accuracy in identifying P300 signals from EEG trials.

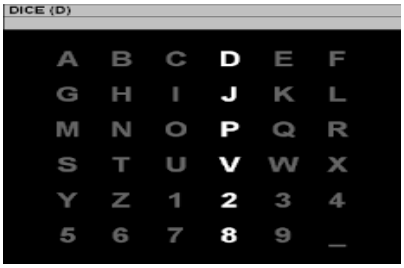
In this paper, powerful Boosting and Bagging ensemble methods like Gradient Boosting and Random Forest are described for detecting the P300 from single EEG trials. Paper also focuses on feature engineering techniques like PCA and median filtering which coupled with above mentioned machine learning algorithms significantly increase the overall classification accuracy.

Objective

The goal is to detect the P300 from EEG trials by using appropriate machine learning algorithms.

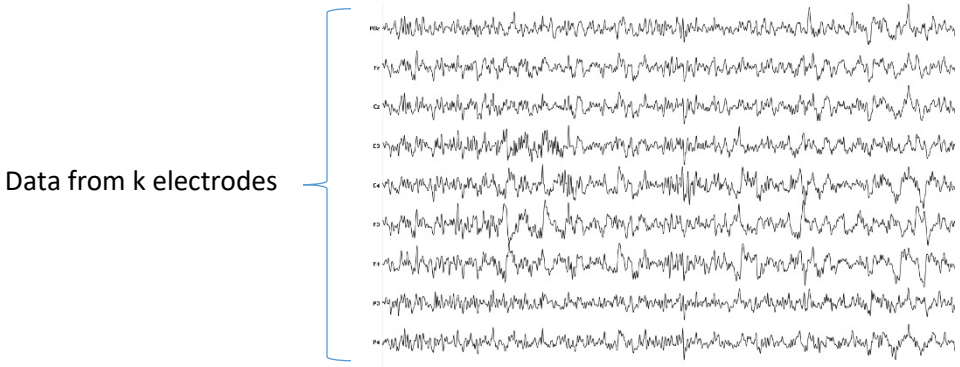
Experiment Setup

A 6x6 matrix of symbols is presented to the user. Rows and columns of the matrix are flashed in random order. Subject focuses on a symbol in the matrix and counts the number of times a specific character flashes. Each time the row or column containing the desired character flashes, a P300 signal is elicited.

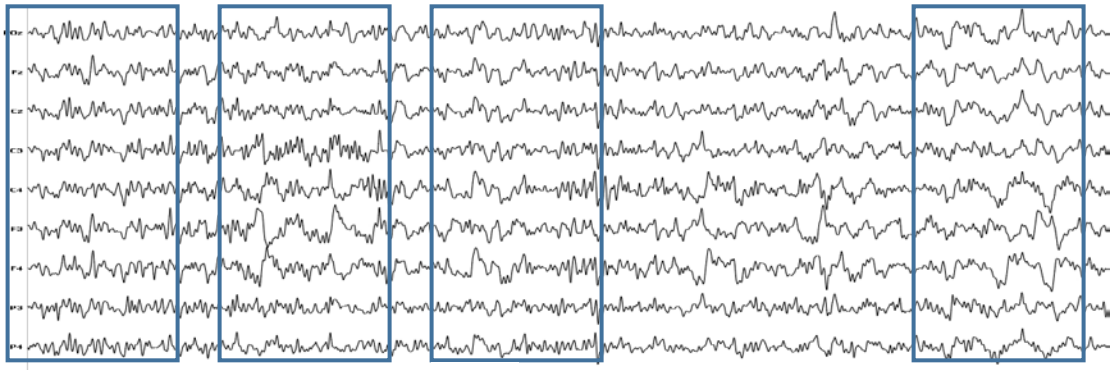


Data Acquisition

From the test subject, we get data in the form of electrical signals from k channels. Data is recorded from k channels at 2048 Hz. The data obtained from EEG is shown below as an example-



Starting at the onset of each flash (shown to the subject) and lasting 1 sec we extract a chunk/sample of data. For each flash, we have the information on whether the subject's desired character is flashed or not. Thus, we have a response variable indicating whether or not the flashed row/column contains the character, subject focused on.



Sample	Response
Sample 1	1
Sample 2	0
…..	
Sample n	1

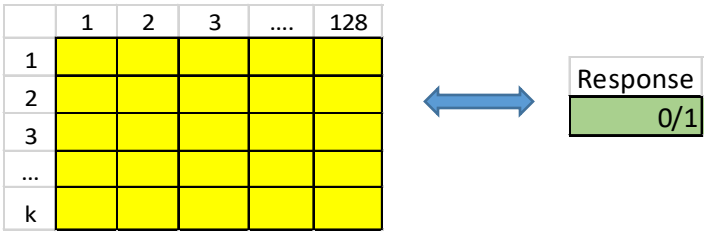
Response 1 indicates that the flashed row/column contains the character, the subject was focusing on. This, in turn, means that a P300 signal elicited in these cases.

Data Preprocessing

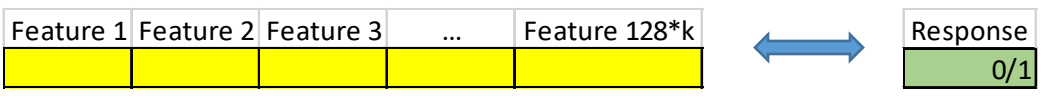
1. Very Low and Very High frequencies can be removed using an appropriate band pass filter (1-12Hz)
2. Data is down sampled to 128Hz (twice the signal frequency is sufficient to capture information) (as in [3])
3. Artifacts due to muscles (EMG) produce large amplitude outliers. These outliers can be removed by replacing top 5% amplitude values with 95 percentile values and bottom 5% values by 5 percentile values.

Feature Extraction

Data corresponding to each sample is in the form of a matrix with k rows and 128 columns k is the number of channels or electrodes used. Each sample has a response value of zero or one associated with it.



A feature vector is created by concatenating samples from each channel as shown below –



Submitted By: Deepak Kumar Sisodia

Structure of Data

Following is the final training data format where n is the number of samples taken and k is the number of channels or electrodes used.

n samples
(1 sample per flash
for 1 sec)

Feature 1	Feature 2	Feature 3	…	Feature 128*k	Response
…					

Feature Engineering

We can use two approaches for creating new useful features-

Approach 1: Noise removal using median filtering

For each flash, data from each electrode can be filtered using the median filtering technique. Based upon the frequency of the noise signals and frequency of P300 signals, we can decide an appropriate number of points (say x).

Step 1: Take x points around each value and take median value of all these points. This median value would represent the background noise present in the signal.

Example: Consider a signal from an electrode 1 corresponding to a flash as below (5 signal values are taken per second for illustration)-

	1	2	3	4	5
1	6	2	14	25	20

Suppose x=3

Median filtered output signal ‘S’ will be:

S[1]=Median(6 , 6 , 2) = 6

S[2]= Median(6,2,14)=6

S[3]=Median(2,14,25)=14

S[4]=Median(14,25,20)=20

S[5]=Median(25,20,20)=20

S = [6,6,14,20,20]

Step 2: Subtract median filter output from the signal value. This would result in a signal without any background noise.

Submitted By: Deepak Kumar Sisodia

Cleaned Signal = Input Signal – Median filter output
Cleaned Signal = [6,2,14,25,20] – [6,6,14,20,20] = [0,-4,0,5,0]

	1	2	3	…	128
1	6	2	14		17
2					
3					
…					
k					

→

	1	2	3	…	128
1	0	-4	0		10
2					
3					
…					
k					

Approach 2: Principal Component Analysis

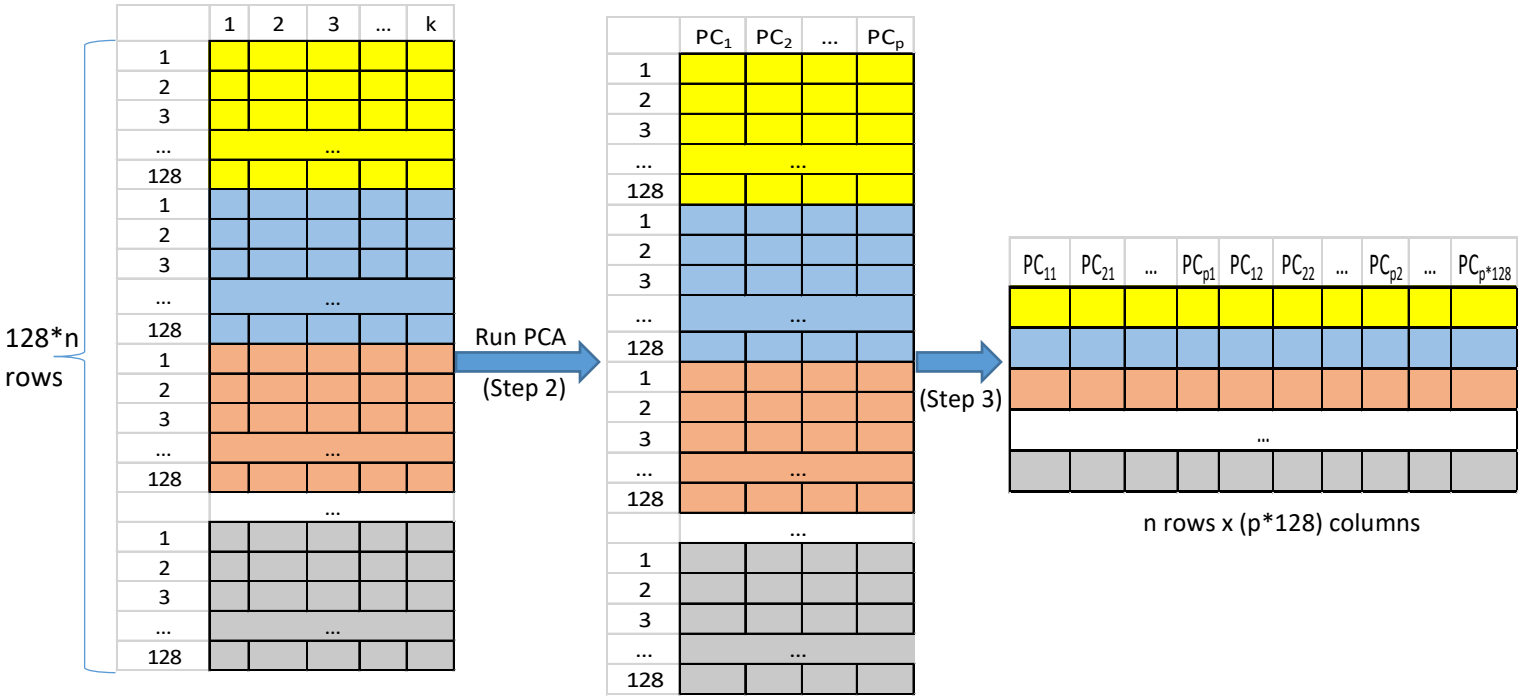
In our data, we have 128*k features or dimensions. We can reduce the number of dimensions by using the principal component analysis.

Data in the form of signals is being collected from k electrodes. It is likely that at any particular point in time, signals from different electrodes are correlated. We can reduce the number of electrodes in our data by using principal component analysis.

Step 1: Transpose all ‘n’ data samples and bind them row wise as illustrated in figure below. Transposed matrix of each sample is represented in a different color. Dimensions of each sample matrix after transposing is 128 * k.

Step 2: Run principal component analysis on this data of k dimensions and 128*n rows. After PCA, we select ‘p’ principal components which capture most of the variance in the data (p << k)

Step 3: Concatenate (i.e. column-bind) all principal component values of a sample as a single row. Each column in the transformed row is a feature.



By means of PCA, total features count has been reduced from 128*k to 128*p where p<< k. For model building, the new table with principal components as features can be used.

Modeling Techniques

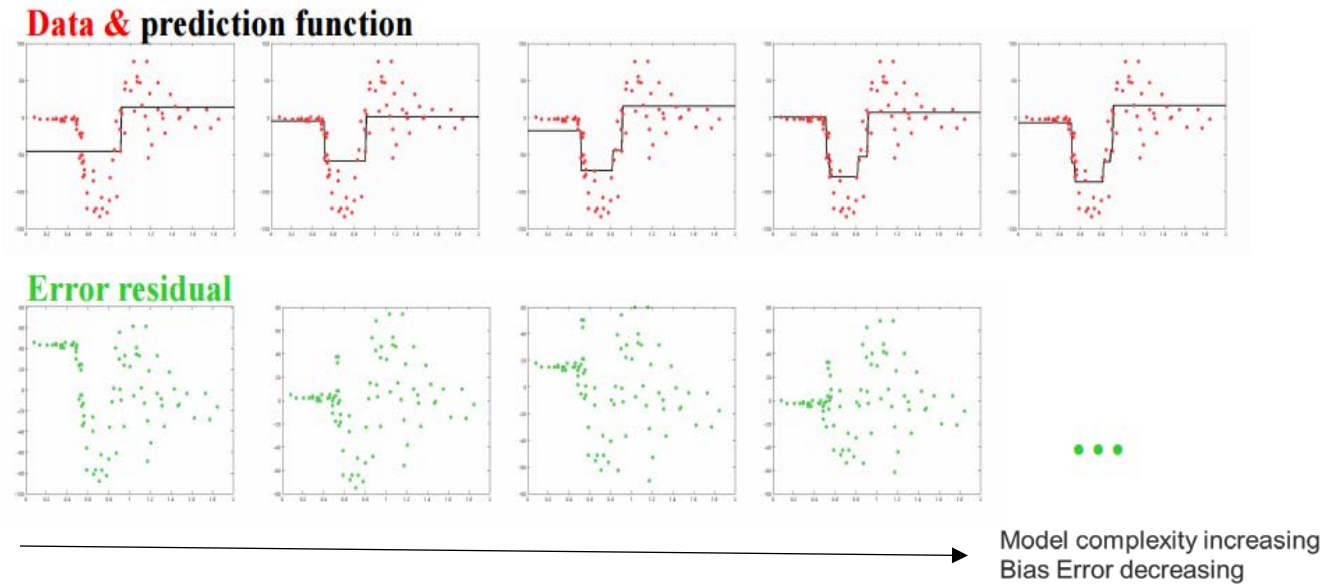
To classify observations in training data into whether or not P300 signal is detected in EEG trial, we can use several machine learning techniques:

Gradient Boosting

Gradient Boosting is an ensemble boosting technique where weak learners (decision trees) are sequentially built in such a way that each new model correct errors made by the previous weak learner.

Gradient boosting is illustrated in picture below. With each iteration, the model complexity increases and bias decreases. In order to avoid overfitting on training data, we stop at the point where misclassification rate increases on validation data set.

Boosting technique has high classification accuracy and it is simple to implement. For solving our problem, we can use gradient boosting algorithm.



Pros	Cons
Often best possible model	Overfits (Need to find proper stopping point)
Robust	Sensitive to noise and outliers
Directly optimizes cost function	Several hyper-parameters
	Lack of Transparency

Results:

Built GBM model on a small data set (Data file 'data.mat' taken from [1]). Modeling was done on the raw data i.e. no feature engineering was done prior to running these models. This is being done to check how powerful these models are.

Tool Used: R for data processing and model building.

Submitted By: Deepak Kumar Sisodia

Random stratified 80:20 split of data set into Training and validation was done.
Package Used: H2o package in R was used to build gradient boosted model to decrease the run time.

GBM Output without parameter tuning is shown below.
Misclassification rate = 24/188 = 12.77%
AUC = 0.94

```
** Reported on validation data. **

MSE:  0.09345728
RMSE:  0.3057078
LogLoss:  0.2996289
Mean Per-Class Error:  0.1273345
AUC:  0.9449915
Gini:  0.889983

Confusion Matrix for F1-optimal threshold:
        0  1    Error     Rate
0      80 15 0.157895   =15/95
1       9 84 0.096774    =9/93
Totals 89 99 0.127660  =24/188
```

After parameter tuning, GBM output is as shown below-
Misclassification rate = 19/188 = 10.11%
AUC = 0.96

```
** Reported on validation data. **

MSE:  0.07957545
RMSE:  0.2820912
LogLoss:  0.2657672
Mean Per-Class Error:  0.1004527
AUC:  0.9606112
Gini:  0.9212224

Confusion Matrix for F1-optimal threshold:
        0   1    Error     Rate
0      80  15 0.157895   =15/95
1       4  89 0.043011    =4/93
Totals 84 104 0.101064  =19/188
```

Tuned parameters values

```
Hyper-Parameter Search Summary: ordered by decreasing auc
   col_sample_rate learn_rate max_depth   sample_rate   model_ids        auc
1              0.5        0.1         3         1.0  gbm_grid.1_model_1 0.9606
```

Random forest

Random Forest is an ensemble bagging technique where multiple decision trees are independently built on different samples of the data. The forest chooses the classification having the most votes.

Random Forest reduces variance and helps to avoid overfitting. Bagging is a special case of the model averaging approach and is relatively robust against noisy data and outliers

Our data has noise in the form of muscle artifacts, thus, to avoid overfitting the noise we can use random forest model.

Pros	**Cons**
Reduces variance and avoid overfitting	Slow to score
Robust to noisy data and outliers	Lack of Transparency
Competitive Accuracy	
Can handle interactions very well	
Easy to use (Few parameters)	

Results:

Built GBM model on a small data set (Data file 'data.mat' taken from [1]). Modeling was done on the raw data.

Tool Used: R for data processing and model building.
Package Used: H2o package in R was used to build Random Forest model

Random Forest Output with 200 trees is as shown below.
Misclassification rate = 35/188 = 18.6%
AUC = 0.90

```
MSE:  0.1436971
RMSE:  0.3790741
LogLoss:  0.4584557
Mean Per-Class Error:  0.1852292
AUC:  0.9035088
Gini:  0.8070175

Confusion Matrix for F1-optimal threshold:
        0   1    Error     Rate
0      69  26 0.273684   =26/95
1       9  84 0.096774    =9/93
Totals 78 110 0.186170  =35/188
```

Conclusion

Gradient Boosting and Random Forest are powerful modeling techniques with high classification power. Accuracy of these models can be further increased by implementing feature engineering techniques on raw data as discussed in this paper.

References

[1] http://mmspg.epfl.ch/BCI_datasets

[2] https://infoscience.epfl.ch/record/101093/files/manuscript.pdf

[3] https://infoscience.epfl.ch/record/87218/files/Hoffmann2005_1207.pdf

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3398470/