P300-Speller Error Correction Using EEG Data

Felix Boyeaux
fboyeaux@stanford.edu

Nehan Chatoor
nchatoor@stanford.edu

Stanford University
Department of Computer Science
CS 229

1 Introduction

Brain-computer interfaces (BCI) seek to enable computers to predict what action a person wants to accomplish based solely on his or her brain waves as he or she thinks about the task at hand. If the computer can be trained to attain a high degree of accuracy in such predictions, this technology can be applied to a variety of uses and will benefit many, including the disabled and speech-impaired by enabling them to communicate without needing assistance. However, deriving information from brain wave data is quite challenging due to the presence of considerable amount of noise, which arises as the brain strives to support other functions of our body in addition to thought. Thus, computers' predictions can be fairly erroneous in such interfaces. This paper seeks to use machine learning algorithms to detect when a specific BCI, the P300-Speller, erroneously predicts the next letter in a word that the person is trying to spell. Being successful in detecting erroneous predictions can in turn be used to improve the performance of the speller by outputting the second-best guess in case of error.

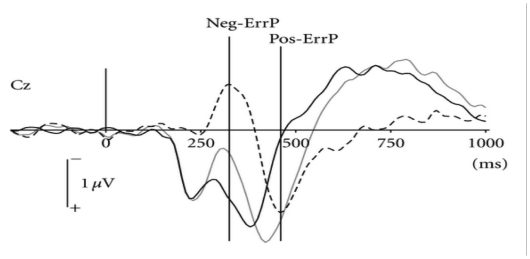

Figure 1: Sample EEG reading

The P300-Speller is a non-intrusive BCI based on the concept of electroencephalography (EEG), and consists of displaying 36 characters in a 6-by-6 matrix. When the subject focuses on one of the characters in the grid, the electrodes placed of the patient's scalp record the brain-wave patterns and recognizes which particular character the subject is trying to choose.

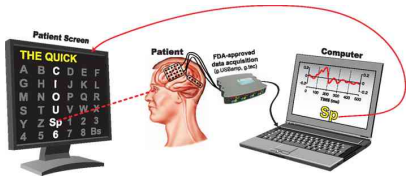

Figure 2: P300 setup (courtesy of Brunner et al.)

2 Dataset

For the purposes of this paper, we drew inspiration and obtained our dataset from the study conducted by Perrin et al. (2012). Their study was comprised of 26 participants, each of whom were asked to spell twelve five-letter words in each of four sessions and twenty five-letter words in a fifth session. Each participant spelled a word by thinking about a letter at a time while attached to scalp electrodes. The electrical activity in the neurons generated by their thoughts was captured by EEG, which were then analyzed by the P300-Speller which then displayed a predicted letter on the computer screen for 1.3 seconds.

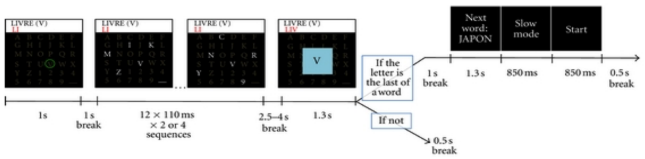

Figure 3: Experimental setup

The data for each individual and each session originally recorded the EEG data for each of the 57 electrodes at a frequency of 600Hz. Additionally, the dataset included a flag for the beginning of each feedback event (displaying the computer's best guess). Finally, for each feedback event, the dataset included an indicator variable of whether the computer's guess was correct or not. This allowed us to use the tools of supervised earning when deriving our models.

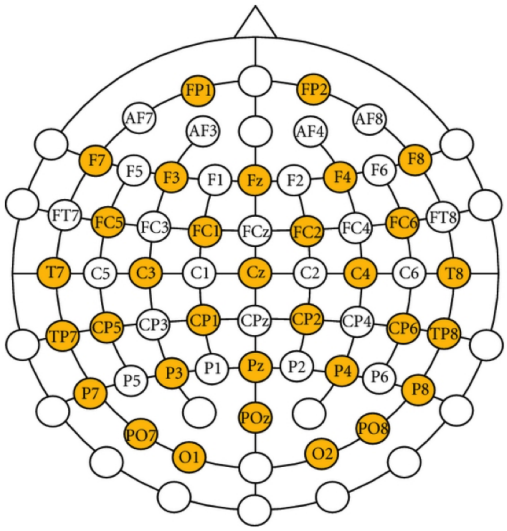

Figure 4: EEG electrodes setup

3 Features and Preprocessing

For the purposes of this paper, we used a dataset downsampled to 200Hz for computational tractability, and focused the analysis on the 1.3 seconds after the start of a feedback event, during which the P300 Speller displays its prediction. We hypothesized that it is during this period that the relevant brain activity for error detection occurs. The reasoning behind this hypothesis is that the participant would have a certain response to the accuracy or the inaccuracy of the computer's output, which will be reflected in the EEG readings. At 200Hz, these 1.3 seconds resulted in 260 readings per electrode, for each of the 57 electrodes, or 14,820 features for each of the 5440 feedback events in our training set. Motivated by Onton and Makeig (2006), we preprocess the data by running independent component analysis on the EEG recordings in order to filter out blink and heartbeat artifacts. Through ICA, we also hoped to isolated a few sources that best predict errors of the speller. Moreover, to reduce the dimensionality of the training set, we conducted principal component analysis on both the data preprocessed with ICA and the original data for comparison. We found that in both cases, first principal component explained up to 90% of the total variance of the data.

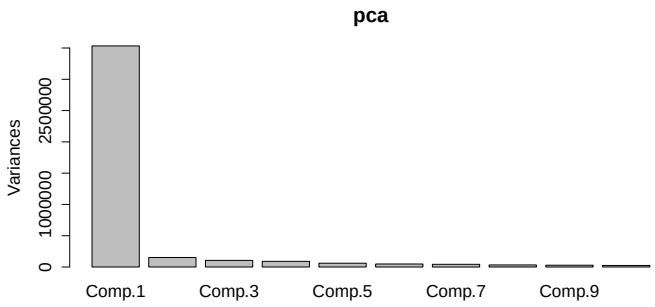

Figure 5: Results of PCA for non-ICA data

For the sake of computational efficiency, we therefore based our models on the the first principal component of our data, which allowed us to reduce the dimensionality of the features from 14,820 to 260. Out of the 5540 observations, we withheld 1632 (30% of observations) randomly selected training examples for the purpose of hold-out cross-validation.

4 Models

In Perrin et al. (2012), the error detection function uses a Gaussian Discriminant Analysis (GDA) algorithm, which, based on an area under the receiver operating characteristic curve (AUROC) metric, is not statistically significant from the trivial predictor which randomizes between

the two classes with probability 1/2. We similarly implement GDA for the purpose of having a benchmark predictor with which to compare our algorithms. This classifier predicts an error ($y = 1$) if the posterior probability $p(y = 1 \mid x) \geq 0.5$, where x is the training set, and assuming the priors $y \sim \text{Bernouilli}(\phi)$, $x \mid y = 0 \sim \mathcal{N}(\mu_0, \Sigma)$ and $x \mid y = 1 \sim \mathcal{N}(\mu_1, \Sigma)$. We estimate the parameters ϕ, μ_0, μ_1 and Σ using Maximum Likelihood Estimation.

To improve on the GDA benchmark, we implement three state-of-the-art machine learning classification algorithms: support vector machines (SVM), random forests (RF) and gradient boosting machines (GBM). The latter two both use a decision tree as the base learner and are examples of ensemble methods, which have shown considerable performance in the classification for EEG-based brain-computer interfaces, since they can consistently provide higher accuracy results compared to conventional single strong machine learning models (Lotte et al., 2007). All three algorithms are trained on both the first principal component of the ICA-preprocessed and of the non-preprocessed data to compare performance.

4.1 Support vector machines

Support vector machines are considered among the best off-the-shelf learning algorithms given their ability to map data to infinite-dimensional feature space using kernel methods. Intuitively, SVMs aim to find the best separating hyperplane to the data projected in high-dimensional space, which yields a highly non-linear decision boundary in the original feature space. Given a classifier $h_{w,b}(x) = g(w^T x + b)$, where $g(z) = 1$ if $z \geq 0$ and -1 otherwise, the SVM algorithm finds the optimal decision boundary by maximizing the margins and imposing a cost on every misclassified training example (using ℓ_1 regularization).

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m \end{aligned}$$

In practice, the optimization problem above is solved by Lagrange duality and applies the kernel method for better performance. In our algorithm, we use a cost $C = 1$, and a Gaussian kernel, which achieved consistently better performance than a linear kernel.

4.2 Random forests

On a high level, random forests, as developed in Breiman (2001), grow a multitude of classification trees. To classify a new object for a feature vector, each tree in the forest classifies the object, and the the classification with most votes is the classification chosen by the forest.

Random forests are based on the concept of bootstrap aggregating (bagging). To predict the class of a new input, one selects B bootstrap samples from the training set, and for each bootstrap sample generate a decision tree. At each split in the tree, $\sqrt{p}$ features out of the p features in the training set are chosen to form the basis of the split. Randomly choosing features in such a way decreases the correlation between each tree and therefore makes random forests less prone to overfitting and high variance problems. For this particular problem, we choose $B = 500$ since improvements were only marginal past that point. Below is the a description of the random forest algorithm.

Algorithm 1: Random forest classification

Data: n observed data points $\{(x^{(i)}, y^{(i)})\}$

Input: Number of bootstrap samples B, number of features p

Result: Predicted classification $\hat{y}$ for x

`/* Main bagging training loop */`

for $b \in \{1, \ldots, B\}$ **do**

  $(X_b, Y_b) \leftarrow$ bootstrap sample of n training examples from $\{(x^{(i)}, y^{(i)})\}$;

  $f_b \leftarrow$ decision tree trained on (X_b, Y_b) using $\sqrt{p}$ randomly selected features at each split;

end

`/* Prediction */`

return $\hat{y} \leftarrow \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x)$;

4.3 Gradient boosting machines

Friedman (2001) considers the problem of estimating the functional dependence $x \overset{f}{\mapsto} y$ such that some specified loss function $\Psi(y, f)$ is minimized:

$$\hat{f}(x) = \arg\min_{f(x)} \Psi(y, f(x))$$

Suppose we parametrize the function estimate $\hat{f}(x) = \sum_{i=1}^{M} \hat{f}_i(x)$, where each $\hat{f}_i$ is called a boost, we can formulate a greedy stagewise approach that at each iteration estimates $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \theta_t)$ where $h(x, \theta)$ is the base learner (here a decision tree), and $(\rho_t, \theta_t) = \arg\min_{\rho,\theta} \sum_{i=1}^{N} \Psi(y^{(i)}, \hat{f}_{t-1}) + \rho h(x^{(i)}, \theta)$. While this optimization problem is hard for general loss function and base learners, Friedman suggest using a new function $h(x, \theta_t)$ to be the most parallel to the negative gradient along the observed data, whereby the optimization task becomes a classic least-square minimization.

Algorithm 2: Gradient boosting algorithm

Data: n observed data points $\{(x^{(i)}, y^{(i)})\}$

Input: Number of iterations M, loss function $\Psi(y, f)$ and base-learner model $h(x, \theta)$

Result: Predicted classifier $\hat{f}(x)$ for x

Initialize $\hat{f}_0$ with a constant;

for $t \in \{1, \ldots, M\}$ **do**

Compute the negative gradient $g_t(x)$;

Fit a new base-learner function $h(x, \theta_t)$;

Find the best gradient descent step-size ρ_t:

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{N} \Psi[y^{(i)}, \hat{f}_{t=1}(x^{(i)}) + \rho h(x^{(i)}, \theta_t)]$$

Update the function estimate $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \theta_t)$;

end

return $\hat{f} \leftarrow \hat{f}_M$;

5 Results

The results reported here are based off the cross-validation sample of 1632 observations that were omitted from the training set. The p-value reported below is the p-value testing the null hypothesis that the predictor is not different from the trivial predictor that outputs one label 50% of the time and has AUROC = 0.5. This statistic is based on the Wilcoxon test.

Method	**AUROC**	***p*-value**
Gaussian discriminant analysis	0.512	0.215
with ICA	0.573	0.001
Support vector machines	0.543	0.003
with ICA	0.633	0.000
Random forests	0.546	0.002
with ICA	0.788	0.000
Gradient boosting machines	0.542	0.004
with ICA	0.717	0.000

Table 1: Summary of testing results

As we see from the result, even though the learning algorithms based of a non-ICA-preprocessed sample are statistically different from a useless predictor, the margin by which they improve the baseline GDA algorithm is very small, on the order of 3 percentage points.

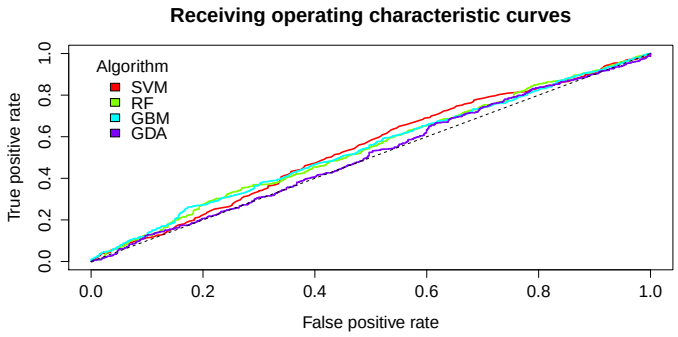

Figure 6: ROC curves for non-preprocessed data

In contrast, the ICA-preprocessed data significantly improves each algorithm. The best performance is from the ICA-preprocessed random forest algorithm with an area under the ROC curve of 0.79. This indicated that the probability that a classifier will rank a randomly chosen positive instance (error) higher than a randomly chosen negative one (correct guess) is 0.79.

6 Future

This paper offers a glimpse at the quality of error correction that can be achieved using EEG data. We suppose that one of the main reasons random forests performed particularly well is because of their tendency to avoid overfitting. Given the high-dimensional feature space, we suspect that both SVM and GBM suffer from a high variance problem. However, more careful analysis of the results would need to be performed in the future to confirm this. Should it be true, more effort should be spent on careful feature selection in order to avoid such problems. Preprocessing the data using ICA allows us to isolate independent sources from the EEG data, each one related to a specific function of the brain which in turn helps us to filter out artifacts, and also to use the most relevant sources in our prediction. Using this, we therefore suggest exploring using features other than just the first principal component. We would like to isolate the most important features and base a classifier on those, and would also consider adding more principal components in order to capture more variance in the data.

7 Conclusions

To conclude, despite limited computing power available and the necessity to use principal components analysis for dimension-reduction in order to have a tractable problem, the algorithms presented here, especially random forests trained on data that had previously been unmixed using independent component analysis, perform well on detecting errors made by the P300-Speller. Therefore, while the spelling accuracy is fairly poor due to the noisy nature of electroencephalography data, implementing an error-detecting algorithm such as the one described in this paper, coupled with an error-correction algorithm that corrects a prediction identified as wrong to the second-best guess, could dramatically improve the performance of such a speller. This leaves us confident that it is possible to facilitate communication for people suffering from speech-impairment by increasing the rate at which they can form words correctly.

References

[1] Breiman, L (2001). Random Forests. *Machine Learning* 45 (1): 5-32.

[2] Perrin, M, et al. (2012). Objective and Subjective Evaluation of Online Error Correction during P300-Based Spelling. *Advances in Human-Computer Interaction* 2012 (2012).

[3] Onton, J. and Makeig, S. (2006). Information-based modeling of event-related brain dynamics. *Progress in Brain Research* 159 (2006): 99 - 120.

[4] Lotte, F., et al. (2007). A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of Neural Engineering* 1 (13): 1-24

[5] Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29 (5): 1189-1232

[6] Natekin, A. and Knoll, A. (2013). Gradient Boosting Machines, A tutorial. *Frontiers in Neurorobotics* 7 (21)

[7] Miltner, B. and Coles, M. (1997). Event-related brain potentials following incorrect feedback in a time-estimation task: evidence for a "generic" neural system for error detection. *Journal of Cognitive Neuroscience* 9(6): 788-798