# Introduction

The p300 Brain-Computer Interface (BCI) is a brain-operated keyboard first presented in [1]. BCIs offer communication and environmental control to people with the most severe forms of paralysis or neurodegenerative diseases. Some of these individuals are completely without voluntary movements (except perhaps for their eyes), so even a poor system of communication can have a tremendous impact on quality of life.

In the p300 BCI, shown in Figure 1, the subject is shown a grid of flashing letters or symbols and asked to attend to the character he or she wishes to type. Since the attended letter is different from the others by virtue of importance and flashes only rarely relative to the other characters, the attended stimulus usually produces a "p300 response" in surface electro-encephalogram (EEG) recordings. This response is named for its most prominent feature, a positive deflection about 300 ms post stimulus onset. The BCI must detect the p300 response to decode which letter is attended. The shape and exact timing of the p300 response are subject-specific. This work is focused on the p300 classification problem. Because the typing speed with a p300 BCI is on the order of characters per minute, even marginal gains in accuracy or speed would be worthwhile for those who depend on a BCI for communication.

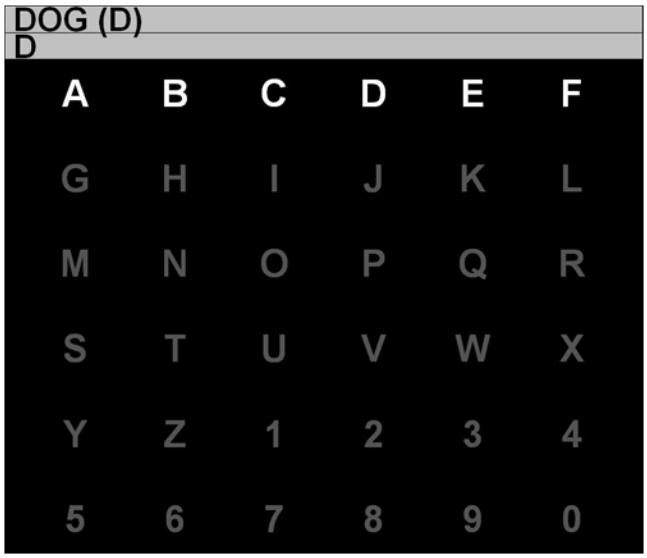


**Figure 1 - A p300 BCI matrix in copy spelling mode (used to produce data with known labels). The EEG recordings from 0 to 800 ms after stimulus onset are typically considered one pattern. Figure from http://www.etsu.edu/cas/bcilab/.**

Early work (e.g. [1]) investigated a very simple classifier (choosing the maximum row and column) based on simple features of the resulting EEG waveforms, such as area under the curve, peak amplitude, and weighted time features with weights derived from stepwise linear discriminant analysis (SWLDA). Recently, more advanced algorithms have been applied. However, comparison between algorithms is difficult. Comparisons published by the same group often display obvious differences in familiarity between techniques (e.g. [2]). Perhaps the best comparison available is from two BCI competitions, where data was publicly available and labs competed to classify the results. However, nearly every group used a different feature set. Some of the leading results from these competitions (as compiled by [3]) were voting with linear support vector machines (SVMs), boosting with ordinary least squares (OLS), and Gaussian-kernel SVMs. Yet, similar results were achieved with linear discriminant analysis (LDA) applied to a feature space of t-Continuous Wavelet Transforms. Also, though it is possibly the best comparison available, the organizers acknowledge that it is in no way objective, due to the variation in effort applied to the problem by the competing groups [4].

Despite the success of SVMs in competition, some of the pioneering BCI labs (such as the inventors of BCI2000 – an open-source BCI platform used by many researchers) feel that the performance and requirements of SWLDA are at an ideal tradeoff point [2]. One main reason

SVMs and other more complicated methods have not seen widespread acceptance is that the competition datasets had very limited numbers of subjects (3 total), and there is skepticism about their generalization. Additionally, although the p300 competitions were won by an advanced machine-learning algorithm, similar complex algorithms placed last or next to last in nearly every category of the competition.

In this work, we aim to improve classifier performance based on investigation of several areas of the classification problem, including feature selection, label error, score combination, and alternative classifiers. The motivation for each of these areas is listed in the following section. Additionally, we hope to validate machine learning methods, in particular SVMs, by testing them on an independent dataset collected by the University of Michigan Direct Brain Interface (UM-DBI) group. Although time constraints did not allow full testing on every subject prior to writing, the UM-DBI group will continue testing those methods that show promise.

## Background / Focus

EEG signals are high-dimensional, noisy, and may include user or perceptual errors. While the UM-DBI EEG setup has only 16 channels (small relative to the 64 used in other labs), each channel contains many time samples. The "state of the art" feature set for p300 is just a concatenation of all channels' data from 0-600 or 800 ms after each stimulus onset. This feature set became standard based on results from [5], which used only SWLDA (which has inherent feature selection) for classification. The number of features is often reduced by applying a moving average (a form of low-pass filtering) and decimation procedure, also from [5]. This is based on the fact that the frequency content of the p300 response is much lower than the Nyquist frequency of the sampling rate of almost any EEG system.

This feature set is extremely inclusive – in fact, according to univariate analysis, entire channels of the EEG data contain little or no useful information (see figures 5 and 6 of [6]). As can be seen in [6], the most useful features appear in the 400-600 ms range and the features before 100 ms probably have little discriminatory value. Of course, it is possible that there is discriminatory information contained in the interaction of these features, but many of these features have no known physiological reason for their inclusion. Even with algorithms such as SVMs that are often cited as being "resistant to the curse of dimensionality," these extra features add computational burden. Thus, part of our effort went toward feature selection based on SVMs (our chosen classifier for most of this project).

Noise in EEG recordings can easily overwhelm the p300 signal because the p300 is a very small response (on the order of microvolts). Single-sequence binary classification results are typically poor (~80-90%) even for the best subjects. Signal averaging is commonly applied to improve signal-to-noise ratio. This is based on the assumptions that the noise is zero-mean and additive, and that the signal is time-locked to stimulus onset. In practice, these assumptions hold well enough that the number of sequences can be varied to produce an optimal speed-accuracy tradeoff. However, large artifacts are often present in EEG recordings, which may cause the sample mean of the noise at particular features to be very far from zero.

An important note is that with any linear classifier, averaging the outputs of the classifier is equivalent to signal averaging. This is not the case for non-linear classifiers, and so the question of how to combine scores from different sequences is not easily answered. Part of our effort went toward investigating a methodology of combining these scores to address these issues.

User and perceptual errors include the user attending the wrong target or blinking during a flash. Also, if the attended letter is flashed twice in a row, the second p300 may not look like a

normal p300 response. According to [7], SVMs are sensitive to labeling errors. Based on this information, we decided to spend some effort investigating automated ways to determine if mislabeled patterns were present.

Finally, we spent a part of our effort investigating alternative classifiers, particularly ensemble and boosting methods, to see if these classifiers could improve system performance. This investigation is also hoped to form the basis of a meaningful comparison between these classifiers, as in all cases the same feature set was used.

# Methodology

## Feature Selection

The misdetection rate $m^{(j)}$, as defined below, was used to rank the channels.

$$m^{(j)} = \frac{1}{n}\sum_{i=1}^{n} [\![1\{ y_i \neq f_j(x_i^{(j)})\}]\!]$$

In this formulation, n is the number of patterns, $x_i^{(j)}$ is all timepoints from the $j^{th}$ channel of the $i^{th}$ pattern, $y_i$ is the $i^{th}$ pattern label, and $f_j$ is the classifier built trained on only the appropriate channels of training data. The channels were ranked by misdetection rate, and then a classifier was built by recursively eliminating the worst channel and training on the remainder. This procedure resulted in 16 classifiers, which were applied to the test set for comparison.

## Score combination

Suppose that classifier scores are sorted for each sequence, so that each possible choice has a vector of scores $s^j$. We define a weighting methodology (applied to every set of scores) such that the final scores are a vector $f$ where $f^{(j)} = \sum w^T s^j$ . The final prediction is stimulus with maximal $f$.

Normally, the total score are the average of the scores along each row and column, thus the weights are equal ($w = \underline{1}$). We call these weights **normal weights**. We propose four additional kinds of weights. **Exclude-one weights** exclude (set weight to 0) only one score, the one that is farthest from the mean (other elements are weighted equally). **Exclude-two weights** exclude the biggest and the smallest scores. **Triangle weights** are in a triangular shape. **Raised-cosine weights** are in a raised cosine shape.

To investigate how these methods respond to feature selection, the included channels were randomly selected during analysis. Since the results obviously depend heavily on channels chosen, the procedure was performed multiple times (50) per number of included channels, and the resulting accuracies were averaged.

## Label error

The methodology in [7] is to iterate through all patterns, flip the associated label for each pattern, and then perform Leave-One-Out (LOO) cross-validation to build a perturbation matrix. In this matrix, the entry in the *i*th row and *j*th column is the classification of pattern *j* by a classifier built with the label for pattern *i* flipped. Then a row of the matrix corresponds to prediction accuracy with a particular label flipped, whereas a column of the matrix contains information about the sensitivity of a particular point to any single-label perturbation.

Unfortunately, [7] gives no information about how to choose SVM hyperparameters (C, gamma). Using cross-validation to choose the parameters seems to break the assumptions leading to LOO cross-validation being an effective predictor of generalization error.

Additionally, it quickly became evident that this procedure was intended for a very small number of patterns (*n*). LOO cross-validation is expensive for large *n*; when the procedure must be repeated *n* times, the computational requirements are extreme.

In this study, then, the following approximations of the method were taken. First, only the standard linear SVM kernel was used, to reduce the number of hyperparameters. Second, cross validation was performed once for a range of values of C. The cross-validation error was observed to be a fairly monotonically decreasing function of C, while computation time increased nonlinearly. C was then chosen such that a single cross-validation operation could be performed in 5-7 minutes. Third, because SVM-Light has a convenient, efficient way to calculate LOO errors (but not individual LOO predictions), only the sum of the rows of the matrix were calculated. This is unfortunate, because in [7] the column sensitivity was shown to be a better predictor of mislabeling. However, the factor of $n^2$ in computational time was too large to be ignored in our study.

Finally, only the rows which corresponded to support vectors in the original analysis were calculated. This was based on the following reasoning: if a point is not a support vector, it lies on the correct side of the margin for its label and thus it is probably correctly classified. Of course, if the original model was overfit due to mislabeling, it is possible that such a point could itself be mislabeled. However, it seems unlikely that a flipped label would result in something sitting on the "correct" side of the margin. This again limited computational burden, by a factor of three. After all of these simplifications, the CPU time to run a single subject was reduced to approximately 100 hours.

## Alternative Classifiers

The purpose of this section is to investigate the various classifiers with several different strategies in order to compare their performances and to find an optimal classifier. The following five classifiers were investigated using the BCI datasets from several subjects. These methods were selected mostly based on a review paper [8].

1) Ensemble SVMs strategy: This method is the winning method for BCI competition III [9]. By using the ensemble of classifiers approach, the classifier variability becomes reduced and obtains enhanced performances [10]. Each single classifier is a SVM with non-homogenous polynomial kernel. Each classifier is independently trained on a partitioned data set and assigns a real-valued score, $f_k(x_{r|c})$, k = 1, … , K. K is the number of classifiers and $x_{r|c}$ is a post-stimulus vector associated to a given row or column. For a given sequence number J, the most probable row and column is the one that maximize the following score function

$$S_{r|c} = \frac{1}{K}\sum_{k=1}\sum_{i\in P_k} y_i \alpha_i^{(k)} \left\langle \frac{1}{J}\sum_{j=1}^{J} x_{r|c}^{(j)}, x_i \right\rangle + b^{(k)}$$

where $\alpha_i^{(k)}$ and $b^{(k)}$ are the parameters for $k^{th}$ SVM. For this classifier, the channel selection method based on recursive channel elimination was used. The channel elimination method selects channels by a greedy strategy (recursively removing channels based on classifier performance after removing each remaining channel at each step).

2) Boosting OLS (Ordinary Least Squares): In this method, gradient boosting was used to stepwise maximize the Bernoulli log-likelihood of a logistic regression model. Ordinary Least

Squares regression was used as weak learner [11]. The pseudocode for the algorithm is shown in Figure 2, where $F_m$ is the ensemble of classifiers after step m, $f_m$ is the weak classifier at step m, and $\tilde{y}_i$ is the first derivative of the likelihood function with respect to F. The number of iterations was180. See [11] for details.

< Boosting with OLS algorithm >

Step 1 : $p_0(y_i = 1 \mid \mathbf{x}_i) = 0.5, \forall i$

Step 2 : $F_0(\mathbf{x}_i) = 0, \forall i$

Step 3 : For m = 1 to M do

$a) \tilde{y}_i = 2(y_i - p_{m-1}(y_i = 1 \mid \mathbf{x}_i)), \forall i$

$b) f_m = \arg\min_f \sum_{i=1}^{N} (\tilde{y}_i - f(\mathbf{x}_i))^2$

$c) \gamma = \arg\max_\gamma L(F_{m-1} + \gamma f_m)$

$d) F_m = F_{m-1} + \varepsilon \gamma f_m$

$e) p_m(y_i = 1 \mid \mathbf{x}_i) = \frac{e^{F_m(\mathbf{x}_i)}}{e^{F_m(\mathbf{x}_i)} + e^{-F_m(\mathbf{x}_i)}}, \forall i$

End for

< Gentle AdaBoost Algorithm >

Step 1 : initialize weights $\omega_i = 1/N$,

i = 1,2,..., N, $F(x) = 0$

Step 2 : Repeat for m = 1,2,..., M :

a) Fit regression function $f_m(x)$ by

weighted least - squares of $y_i$

to $x_i$ with weights $\omega_i$.

b) $F(x) \leftarrow F(x) + f_m(x)$

c) $\omega_i \leftarrow \omega_i \exp(-y_i f_m(x_i))$, remormalize

Step 3 : Output the classifier

$sign[F(x)] = sign[\sum_{m=1}^{M} f_m(x)]$

**Figure 2 - Pseudocode for Boosting with Ordinary Least Squares (OLS) and Adaboost.**

3) Gentle Adaboost: This method is a more robust and stable version of Adaboost introduced in [12,13]. The algorithm for gentle Adaboost is shown in Figure 2; this is a modified version of Adaboost using Newton stepping rather than exact optimization at each step. The number of iterations was 180. See [12,13] for details.

4) Other methods: The following methods were also used to classify the data: Linear SVM [14], Fisher linear discriminant, normal densities based linear classifier (LDA), and logistic linear classifier. PRTools [15] was used to simulate these methods.

5) Combination classifier: The combination of SVM and logistic linear classifiers using several different combination methods were investigated.

For every case, the data (from 0 to 600 ms) was preprocessed by moving average filter and then decimated by a factor of 10.

# Results

## Feature Selection

The results of the channel selection algorithm are shown in Figure 3. There is a very slight (probably not statistically significant) accuracy enhancement after removing the first few channels, but the computational burden is decreased significantly. After a certain point (about 5 channels), however, performance begins to fall off dramatically.

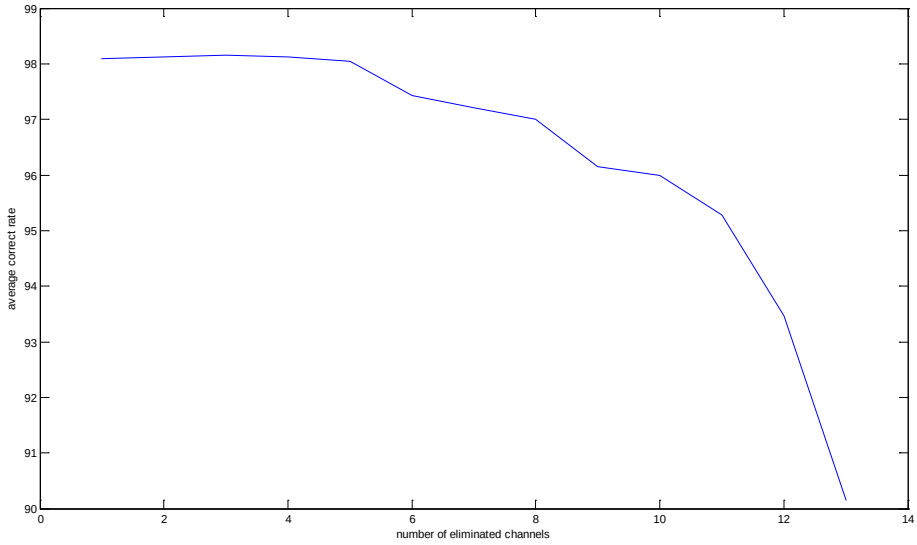


**Figure 3 - Accuracy of a classifier trained after removing the "worst" n channels.**

## Score Combination

For each type of weights described above, the average error rates versus the number of channels selected graphs are plotted in Figure 4 (left). There were 50 trials for each number of channels, and their average computation times for each number of channels are plotted in Figure 4 (right).

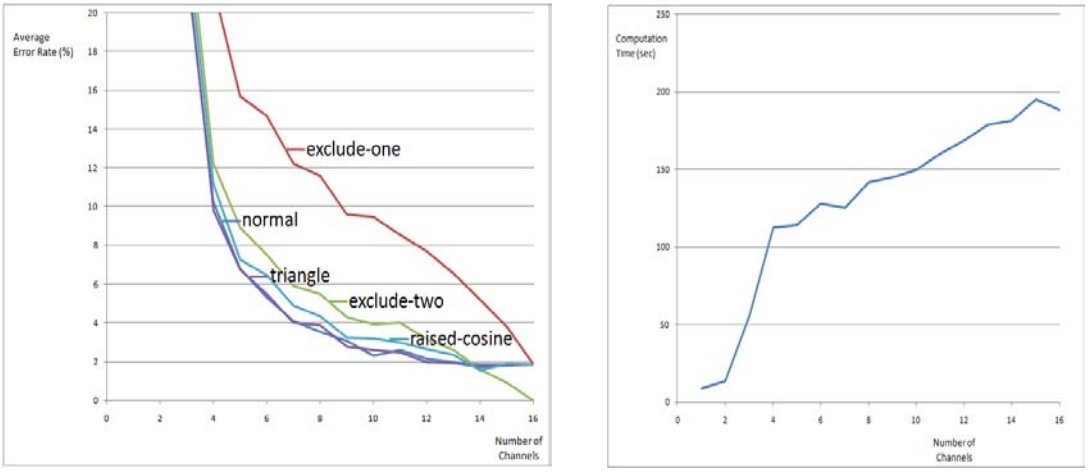


**Figure 4 – (Left) Average error rate versus number of channels used. (Right) Average computation time versus number of channels used.**

## Label Error

Due to the computational burden of this technique (see the Methods section), only two subjects were analyzed in detail. Unfortunately, the results were not conclusive. Approximately 281 of the 933 support vectors for subject H114 were identified as possibly mislabeled (candidates). Unfortunately, due to the granularity of LOO validation results, several of the candidates tied for performance. For H114 there was a clear winner, while five candidates tied

for second. For H115, 17 tied for first and 50 tied for the next group. Testing with the most important group, the first and second groups, and all of the candidates removed led to per-flash and character accuracies as shown in Table 1. Although some letters were predicted differently by each model, overall accuracy was very similar for each case. Removing all candidates is not suggested based on the results.

**Table 1 - Accuracies based on removing suspicious patterns. Note that the original results of pattern accuracies are unavailable, because only characters were classified.**

| | H114 pattern accuracy | H115 pattern accuracy | H114 characters correct | H115 characters correct |
|---|---|---|---|---|
| Original | ? | ? | 221 | 226 |
| SVM | 0.9023 | 0.8339 | 224 | 253 |
| "Worst" pattern group removed | 0.9028 | 0.8349 | 224 | 252 |
| "Worst 2" pattern groups removed | 0.9017 | 0.8393 | 224 | 253 |
| All candidate patterns removed | 0.8935 | 0.8439 | 221 | 210 |

## Alternative Classifiers

Table 2 shows the results obtained by applying various methods to UM-DBI data. Data from two subjects was used for the evaluation. H114's dataset is well organized and does not suffer much from motion artifacts or mislabels. In contrast, H106 contains a lot of movement artifacts, and the subject was not entirely concentrating to the experiment. This affected the results, as shown in Table 2.

**Table 2 - Performance of various classifiers with different strategies. UM-DBI data of two different subjects were used for classification. (* Results from all three sessions were averaged.)**

| Subject | Sessions | Methods | Classifier combination methods | Channel selection (Total 16 channels) | Classifier performance in % of correct classification* |
|---|---|---|---|---|---|
| **H114** | 001<br>002<br>003 | Ensemble SVM Method | Voting | Channel elimination | 91.33 |
| | | | | Use all | 92.43 |
| | | | | Random 4 | 80.86 |
| | | Boosting OLS | Boosting | Use all | 90.82 |
| | | Gentle Adaboost | Boosting | Use all | 71.54 |
| | | Linear SVM | Voting | Use all | 86.58 |
| | | Logistic linear | Voting | Use all | 85.19 |
| | | Normal densities based linear classifier (LDA) | Voting | Use all | 87.10 |
| | | Fisher linear discriminant | Voting | Use all | 71.65 |
| | | Logistic linear + | Maximum | Use all | 87.13 |

| | | | | | |
|---|---|---|---|---|---|
| | | linear SVM | Combining | | |
| | | | Averaging Combining | Use all | 87.54 |
| **H106** | 001 | Ensemble SVM Method | Voting | Channel elimination | 70.27 |
| | | | | Use all | 87.02 |
| | | | | Random 4 | 9.89 |
| | 002 | Boosting OLS | Boosting | Use all | 83.33 |
| | | Gentle Adaboost | Boosting | Use all | 13.56 |
| | 003 | Logistic linear | Voting | Use all | 70.13 |
| | | LDA | Voting | Use all | 70.27 |
| | | Fisher linear discriminant | Voting | Use all | 59.46 |

# Discussion

## Feature Selection

Feature selection was successful in reducing computations. We could eliminate up to 7 worst channels with slight performance degradation. It will lead to 43.75% less computation with the increase of the 1% error rate. By removing the worst 2 channels we could gain slight performance improvement. By using this simple method for feature selection, we could substantially reduce computation burden with almost no loss in performance.

## Score Combination

Our work in this area was not very successful. While some specific cases showed that the proposed methodology can reduce the error rate, none of the weighing methods produced reliable enhancements of the error rates. Among the four kinds of proposed weighing methods, the triangle weights showed the best and the exclude-one weights showed the worst performance, but all of them were no better than the normal weighing method.

However, it is notable that triangle weights and raised-cosine weights showed comparable error rates to the normal weighing. Since most of the test data have only 4 or 5 sequences, it is not enough to apply weights. In some test data with more sequences we expect that this method may perform better.

## Label Error

The results of the label error investigation were disappointing. However, several statements need to be made. First, the investigation was a dramatic simplification of the algorithm from the original reference, so its failure is not a condemnation of the full method. Second, however, investigation of label sensitivity on some easily-separable, artificial 2-D data has not shown the same conclusions as [16]. The sensitivity to mislabeling appears to be highly dependent on the hyperparameters of the SVM, but [16] did not describe the method used to set them. In order to cause the linear SVM to respond sensitively to mislabeling, the C parameter had to be set very high. Gaussian SVM overfitting required a large gamma and a fairly large C. In most of the circumstances tested, however, both forms of SVMs did a remarkable job of

ignoring mislabelings (often even if >10% of the data was mislabeled). Perhaps these problems are exacerbated by the small sample sizes in microarray data, or the data in that field requires very high C values. In all honesty, if we were performing this analysis without prior knowledge of the alleged sensitivity, we might have reported that SVMs were resistant to labeling errors. This investigation brings into question the need for addressing mislabeled data. However, if C or gamma values are large, it may be worth looking into.

### Alternative Classifiers

Classification performance of each classifier for dataset H114 is much better than that of dataset H106. This implies that, without proper label correction or feature selection, the classification of p300 data largely depends on the subject.

Ensemble SVM and Boosting OLS methods performed very well for both datasets, showing relative robustness when compared to other methods, but the performance of Ensemble SVM method with fixed channels was very poor. Ensemble SVM shows slightly better quality than Boosting OLS, but it required much more computation time. Therefore, selecting a classifier will depend both on operational tradeoffs of the final environment.

Adaboost showed the worst performance and its sensitivity to mislabel [17] seems to be the reason. In [17], performance of Adaboost was critically deteriorated when applied to a poorly constructed dataset, so it is not recommended to use Adaboost when the dataset is suspected to have a lot of artifacts.

The performance of each classifier can be further improved by using different feature selection or channel selection methods. The channel selection method proposed in the previous section can be applied to some of these classifiers, and may give improved results.

## Conclusion

Unfortunately, few of the individual investigations produced good results. We had hoped to combine findings from each area of investigation to produce a system to compare with the Ensemble SVM method, but were unable to do so because of the lack of conclusive results. The primary contribution of this work is then the comparison of different methods performed on the same feature set from new data, as done in the Alternative Classifiers sections. Some of the methodologies warrant further investigation (such as a fairer test of the mislabeling correction algorithm).

## References

[1] L. A. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," Electroencephalogr. Clin. Neurophysiol., vol. 70, pp. 510-523, Dec. 1988.

[2] D. Krusienski et al., "A comparison of classification techniques for the P300 Speller." Journal of Neural Engineering, vol. 3, pp. 299, 2006.

[3] Lotte, Congedo, Lcuyer, Lamarche and Arnaldi, "A review of classification algorithms for EEG-based brain-computer interfaces." Journal of Neural Engineering, vol. 4, pp. R1, 2007.

[4] See “Remarks” in the results of any competition at http://www.bbci.de/competition.

[5] D. Krusienski, E. Sellers, D. McFarland, T. Vaughan and J. Wolpaw, "Toward enhanced P300 speller performance." J. Neurosci. Methods, vol. 167, pp. 15, 2008.

[6] B. Allison and J. Pineda, "Effects of SOA and flash pattern manipulations on ERPs, performance, and preference: implications for a BCI system." International Journal of Psychophysiology, vol. 59, pp. 127, 2006.
[7] A. Malossini, E. Blanzieri and R. Ng, "Detecting potential labeling errors in microarrays by data perturbation." Bioinformatics, vol. 22, pp. 2114, 2006.
[8] F Lotte, M Congedo, AL´ecuyer, F Lamarche and B Arnaldi. A review of classification algorithms forEEG-based brain–computer interfaces. J. Neural Eng. 4 (2007) R1–R13.
[9] A. Rakotomamonjy and V. Guigue, "BCI competition III: dataset II- ensemble of SVMs for BCI P300 speller." IEEE Trans. Biomed. Eng., vol. 55, pp. 1147, 2008.
[10] Y. Grandvalet, “Bagging equalizes influence,” Mach. Learn., vol. 55, no. 3, pp. 251–270, 2004.
[11] Hoffmann, U., Garcia, G., Vesin, J.-M., Diserenst, K., Ebrahimi, T. A Boosting Approach to P300 Detection with Application to Brain-Computer Interfaces.2nd International IEEE EMBS Conference on Neural Engineering 2005, art. no. 1419562, pp. 97-100.
[12] GML AdaBoostMatlab Toolbox developed by Alexander Vezhnevets (avezhnevets@graphics.cs.msu.ru).
[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. The Annals of Statistics, 38(2):337–374, April 2000.
[14] M. Kaper, P. Meinicke, U. Grossekathoefer, T. Lingner, and H. Ritter. BCI Competition 2003—Data Set IIb: Support Vector Machines for the P300 Speller Paradigm. IEEE Transactions on Biomedical Engineering, vol. 51(6), pp. 1073, June 2004.
[15] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, S. Verzakov. PRTools4.1, A Matlab Toolbox for Pattern Recognition, Delft University of Technology, 2007.
[16] A. Malossini, E. Blanzieri, and R. Ng, “Assessment of SVM reliability for microarrays data analysis.” Proceedings of the Second European Workshop on Data Mining and Text Mining in Bioinformatics, pp. 42, 2004.
[17] R. Boostani and M. H. Moradi, “A new approach in the BCI research based on fractal dimension as feature and Adaboost as classifier.” Journal of Neural Engineering, 2004

## Individual Effort

Dave investigated label error, as well as providing .m files to allow the group members to read the datafiles and parse them into observations and labels. He also wrote the introduction, background, and conclusion sections of this report, as well as compiling it and editing.

Sanghoon investigated feature selection, and compiled the progress report.

Jong Wook investigated combination of scores, and also performed a method of channel selection based on hierarchical clustering which is not included in this report. The method showed advantages in performance versus randomly selecting the channels, but was not included due to space constraints and overlap with Sanghoon’s work (Dave’s executive decision, based also on the fact that random selection of channels is not a common method).

Jang Hwan performed the work under Alternative Classifiers.