BCI challenge: Error potential detection with cross-subject generalisation

Duncan Barrack
Nottingham, UK
duncan.barrack@nottingham.ac.uk

March 10, 2015

1 Summary

In this document I outline my solution to the brain computer interface (BCI challenge hosted on Kaggle [1] which was awarded first prize. The competition used data collected in the study conducted by Perrin *et al.* [6] and the aim was to form a statistical model to predict errors in the spelling task of the P-300 speller [3]. During a P-300 trial users are presented with letters and numbers and are tasked with spelling words. Using electroencephalogram (EEG) data the speller attempts to determine which letter the user is thinking of and presents him/her with this letter. If the letter coincides with the same letter the user was thinking about then the feedback is regarded as positive, otherwise the feedback is negative. The challenge is to predict, based on the user's response to the feedback event, whether feedback was positive or negative.

The EEG (56 channels) and Electrooculography (EOG, 1 channel) data of sixteen subjects who partook in 340 P-300 trials are provided as a training set. Ten other subjects make up the test set for which it is not known whether the feedback was positive or negative. The efficacy of the predictive models submitted by competition entrants are measured against this test set. Two subjects make up the score used for the public leader-board and eight for the private leader-board. It is the latter which is used for the final standings.

For the solution presented in this work, I focused on the 1.3s long EEG and EOG signals after feedback which contain the event related potentials (ERP) associated with the user's response to the feedback event. I engineered several features from the ERP signals including the means of the EEG values for each channel in windows of various lengths and lags, features based on template matching, as well as meta features such as trial time-stamp, trial session number etc. To predict the probability of a positive feedback event in the test set I took the arithmetic mean of the posterior probabilities of two regularised support vector machines which were trained on different features sets using linear kernels. This approach scored an area under the receiver operating characteristic curve on the private leader-board of 0.76921.

2 Features Selection / Extraction

Raw EEG and EEG signals were bandpass filtered to between 1 and 20 Hz using a fifth order Butterworth filter [5]. The features I used were predominantly based on material I came across in the academic P-300 speller literature and are described below.

2.1 Feature set A - Meta features

I used the trial number and trial session number as features as well as whether the trial was long or short. The motivation behind this was that as a user progressed through the trials they would begin to fatigue and make more mistakes. Session and trial number could act as proxies for this. Furthermore trials were spread over five sessions and session five differed from the other sessions in that if the P-300 speller incorrectly predicted

the feedback letter then it would present the user with the next most probable letter. It was possible to infer whether this had happened by looking at the times between feedback events (longer times tended to correspond with 'retrials' where the initial prediction of the P-300 speller was incorrect). I used this as a feature.

2.2 Feature set B - Mean EEG values in windows of different lengths and lags

Based on a video lecture given by Müller, Blankertz *et al.* [2] I took the mean of the EEG and EOG recordings after the feedback event over windows of a number different lengths (50ms to 650ms at increments of 50ms) and lags (0ms to 1250ms at increments of 50ms). This was done for each channel separately giving a total of 14820 features.

2.3 Feature set C - Template matching based features

Inspired by Smulders *et al.* [7] I averaged all ERP signals labelled as positive feedback in the training set to create a 'positive feedback template' and similarly to create a 'negative feedback template'. This was done separately for each channel. I then used the correlation coefficient, maximum cross-correlation coefficient at lags of 200ms, covariance, maximum cross-covariance at lags of 200ms and Euclidean distance between the templates and test ERP signals as features. Additionally I used the difference between the correlation coefficients obtained using the positive and negative feedback templates (and similarly for the maximum cross-correlation coefficients), the ratio of the covariance obtained using the positive feedback template to the covariance obtained using the negative template (and similarly for the cross-correlation covariance) as well as the ratio between the Euclidean distances obtained using the positive template and that obtained using the negative feedback template. This gave a total of 855 additional features.

3 Modeling Techniques and Training

3.1 Cross validation

I used 4 fold 'subject wise' cross validation (CV) and calculated the area under the receiver operating characteristic curve (AUC) [4] for the four subjects in the test fold. I repeated my CV procedure five times with different splits and took the average of the 20 AUC scores produced (5 repetitions $\times$ 4 folds). For the template based features (feature set C) I ensured that only the data within the training folds was used to form the templates so as to not bias the results.

3.2 Model selection

I tried a number of different machine learning algorithms including gradient tree boosting, random forests, support vector machines (SVM) and logistic regression with elastic net regularisation [4] using different sets of features. The two models which gave the highest CV scores were a SVM with a linear kernel using feature sets A and B and SVM with a linear kernel using feature sets A and C. For these models L_2 regularisation was used with regularisation parameters set via CV. Furthermore, rather than use all of feature set B a sub-set of 7410 features, accounting for half of the total generated, was used. This was determined using a logistic regression model with ridge regression (penalty term set via CV) [4] and taking the 7410 features which had the highest coefficient values. Although models using the subset had similar CV scores to models using all of the features, the variance in the AUC CV scores was greatly reduced when the the sub-set was used.

3.3 Model averaging

My final model was formed by taking a weighted average of the posterior probabilities produced by the best two linear SVM models described in Section 3.2. The weights were set via CV (see Figure 1).

4 Code Description

The code used is available at https://github.com/duncan-barrack/kaggle_BCI_challenge. The repository contains seven programs which are described below.

- **generate_meta_features.m** - generates all of the features in set A (Section 2.1) except the features based on session 5 retrials.
- **get_sess5_retrial_features.m** - generates the remaining features in set A.
- **get_ave_amplitude_features.m** - produces the features in set B (Section 2.2)
- **get_template_features1.m** - gives all of the features in set C (Section 2.3) expect those based on cross-correlation and cross-covariance.
- **get_template_features2.m** - generates the remaining features in set C.
- **train_model.py** - Trains the two SVM models described in Section 3.2 using the training data.
- **predict.py** - Combines the predictions of the SVM models by taking a weighted average and produces a **Submission.csv** file.

5 Dependencies

To generate the features I used Matlab R2014b. To fit and predict the model Python 2.7.6 was used with scikit-learn 0.14.1, numpy 1.8.2 and pandas 0.13.1.

6 How To Generate the Solution

After setting the paths in the **SETTINGS.json** file, to generate the features the following Matlab scripts will need to be run in order, **generate_meta_features.m**, **get_sess5_retrial_features.m**, **get_ave_amplitude_features.m**, **get_template_features1.m** and **get_template_features2.m**. To train the model run **train_model.py** and finally to make predictions on the test set run **predict.py**.

7 Figures

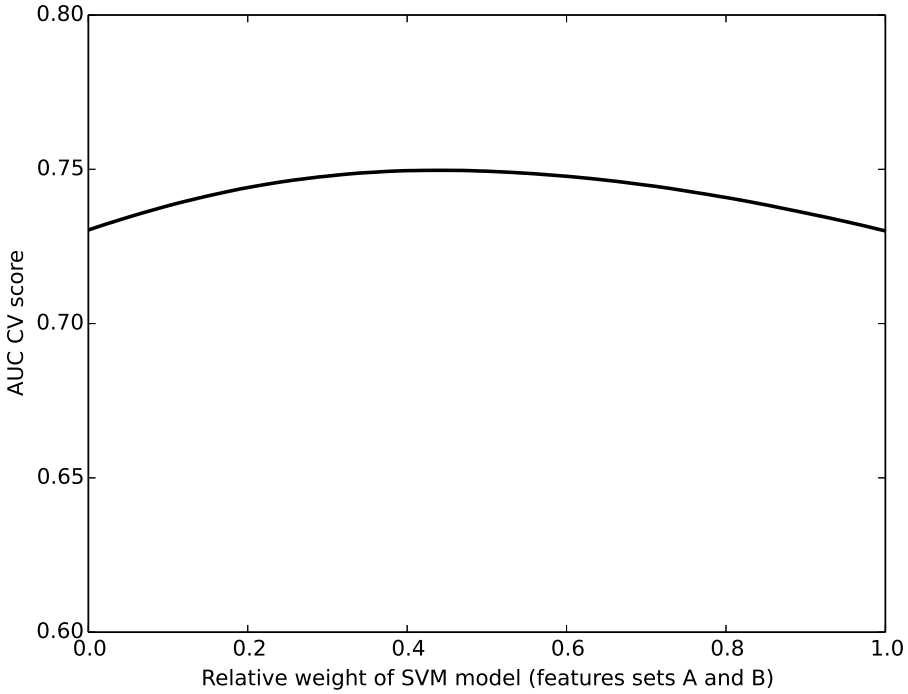

Figure 1: Plot showing the CV scores when the posterior probabilities from a linear SVM model which uses features sets A and B and a linear SVM model which uses features sets A and C (both described in Section 3.2) are averaged using different weights. Here the maximum CV score ≈ 0.75 is obtained by a applying a relative weight of 0.46 to the results of the SVM model which uses features sets A and B. This weighting was used to predict the labels of the test set.

References

[1] BCI Challenge @ NER 2015. https://www.kaggle.com/c/inria-bci-challenge. Accessed: January 2015.

[2] Machine Learning and Signal Processing Tools for BCI. http://videolectures.net/bbci09_blankertz_muller_mlasp/. Accessed: January 2015.

[3] Lawrence Ashley Farwell and Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988.

[4] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[5] B Latni. *Signal processing and linear systems*. Oxford University Press, USA, 1998.

[6] Margaux Perrin, Emmanuel Maby, Sébastien Daligault, Olivier Bertrand, and Jérémie Mattout. Objective and subjective evaluation of online error correction during p300-based spelling. *Advances in Human-Computer Interaction*, 2012:4, 2012.

[7] Fren TY Smulders, JL Kenemans, and A Kok. A comparison of different methods for estimating single-trial p300 latencies. *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section*, 92(2):107–114, 1994.