Error Detection based on neural signals

Nir Even-Chen and Igor Berman, Electrical Engineering, Stanford

Introduction

Brain computer interface (BCI) is a direct communication pathway between the brain and an external device. In Professor Shenoy's Lab primates are trained to use BCI to control 2D cursor to accomplish different tasks. The primate controls the cursor through its neural activity, which is being recorded by an array of electrodes implanted in the motor cortex.

Decoding neural activity is a challenging task prone to error. In this work, we propose a **binary classifier** that uses the same neural inputs from the motor cortex to **detect errors**.

This classifier can alert the system when an error occurs and enable auto-correction, as well as provide feedback regarding the primate's understanding of failure and success. Eventually, this method can be used to enable paralyzed patients to communicate with their environment.

In the first part of the project we've focused on reducing the data's dimensionality through preprocessing. In the second part, we compared various classification techniques on the data, eventually achieving a classification error rate of $5 \pm 1\%$.

Experiment Setup

In this project we will use data from the 'spelling' task, which includes time series of neural activity, and labeling of the trials (Success / Failure). In this particular task the primates were required to bring the cursor above a highlighted letter and hold it there within a set timeframe (500ms). If successful the primates received a reward. In the case of failure, a specific sound is produced (Figure 1).

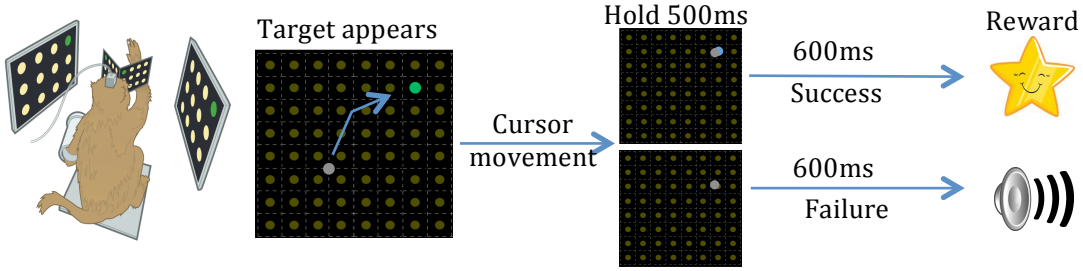

Figure 1: Experiment Setup. A Target appears and the primate moves the cursor using BCI. Selection is made when it holds the cursor on top of a circle for 500ms, than the target is vanishes. The target changes to blue in while it is being help. Feedback appears 600ms after selection.

For our initial (pre-milestone) study, we used existing data from previous experiments, where the reward was given immediately after the success. For the sake of this project, we then conducted a new experiment, where feedback was delayed for 600ms, so that the reward doesn't mask the primate's internal "error recognition".

The Data

The neural activity is recorded from the motor cortex (areas M1 and PMd) with a 1 kHz sampling rate from 192 electrodes. In neuroscience it is common to analyze neuron's spikes, so our raw data includes is discrete in time and binary in values, with $x(t) = 1$ if there was a spike during millisecond t.

Based on previous studies, we expect the "failure signal" to appear at most 200ms after the actual mistaken selection, so we use samples from 300ms before letter choice to 400ms after. The experiment was conducted on the same primate for two days; 1988 trials were recorded with a 30% failure rate.

Preprocessing

Dimensionality reduction

Each learning examples included 700 time samples in 192 channels, resulting in $700 \times 192 = 134{,}400$ dimensions. Before proceeding to run learning algorithms on the data, we'll take some steps to reduce dimensionality:

Time Dimension:

Based on prior studies, we expect the failure signal to last for tens of milliseconds. Furthermore, the usual firing rate for neurons in around 15-20Hz, meaning that the 1ms resolution produces very sparse data. Based on this evidence, and as is common in the field, we expect not to lose any critical information by downsampling the data and grouping the samples into 10ms bins of int values, resulting in only 70 time samples per channel per trial.

Number of Channels:

In the channel domain we applied per-time-frame PCA to reduce the number of channels. We decided not to normalize the variance, since all channels represented the same type of data (#spikes/10ms), and intuitively, active channels should have more influence than sparse ones.

We've tried two different approaches to performing the PCA, and we compared them by the % of variance represented in the first n components, using the rule of thumb that it's better to have fewer components that capture more variance:

Naïve PCA: Our first approach was to concatenate all the timeframes of all trials (success and failure), and extract the primary components from the resulting covariance matrix. The result is shown in Figure 2 below (red), demonstrating that $\sim$40 components are required to capture 50% of the variance.

Mean-Diff PCA: Following these results, we came to assume that the first primary components we capture actually represent the kinematic signals produced by the brain, which are strong and common to both failure and success trials. This model can be represented a sum of three random processes:

$$x_i(t) = v_k(t) + v_{s\backslash f}(t) + v_n(t) \quad (1)$$

Where $v_k(t)$ is kinematic component that controls the cursor, $v_{s\backslash f}(t)$ is the component caused by success\failure and $v_n(t)$ is noise. We assume that the three processes are independent and have zero mean. To extract the second component $v_{s\backslash f}(t)$, we applied the following method:

$$\Delta\mu(t) = \frac{1}{N_S}\sum_{i\in Suc} x_i(t) - \frac{1}{N_F}\sum_{i\in Fail} x_i(t) \approx \hat{v}_s(t) - \hat{v}_f(t) \quad (2)$$

The resulting mean-diff $\Delta\mu \in \mathcal{R}^{70\times 192}$ captures the average differences between successful and failed trials. Since the kinematic component is generally orthogonal to the {fail/success} labelling, subtracting the two averages reduces its effect. Eventually, we ran a PCA on the matrix $\Delta\mu$ and examined the resulting vectors. As shown in Figure 2 below, the 3 first PCs together capture 90% of the variance:

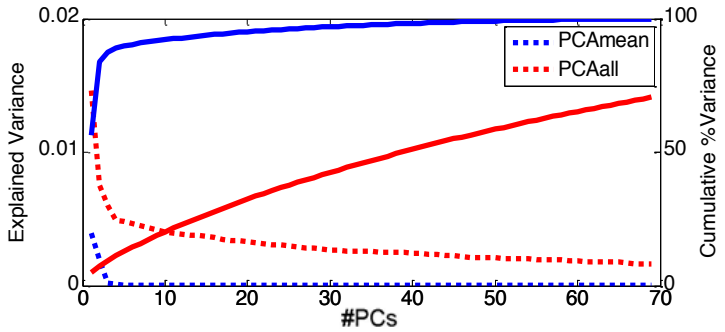

Figure 2: Per-vector(dotted line) and Cumulative (solid line) Variance for two sets of PCs: the naïve PCA (PCAall – red) and mean-diff PCA (PCAmean – blue)

Based on the above, we choose to proceed with the second set of PCs, generated from the mean-diff. To gauge their relevance to our classification problem, we project the successful and failed trials on the first three PCs we found (Figure 3), with the projection of failed results in blue and successful in red:

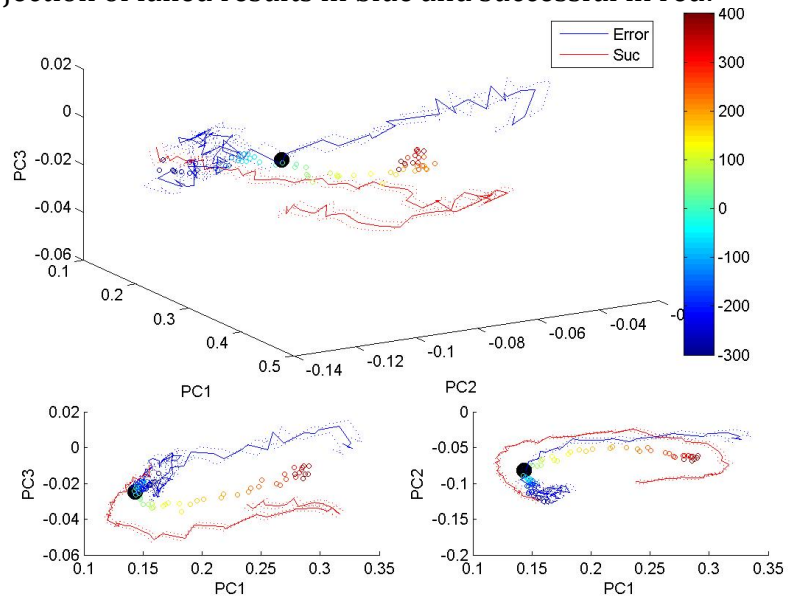

Figure 3: Projection of successes and failures on first three PCs. Failure in blue, success in red. Solid line represent the average projection for every timestamp, dotted line is the standard deviation of the mean. The circles represent the time evolvement where the black circle is target selection time (t=0). Top: 3D of 3 leading PCs, Bottom: 2D view of PC1/3 and PC1/2.

We can see that the two processes are differentiating from target selection time (t=0, black circle). In the bottom figures it is evident that the mean projection on PC1 and PC2 are common for both scenarios. However, PC3 contains the information that distinguishes between the two scenarios.

Classification

Algorithm Comparison

Using the preprocessed data (three channel with 70 samples each, meaning $R^{70\times3}$) we compared several learning algorithms: SVM (C=0.1), Logistic regression, GDA and K-NN (K=10) as shown in Figure 4. To compare the different methods we used 10-fold cross validation, maintaining in each random group the original success/failure rate. At this stage, we ignored the temporal information, treating our samples as vectors in 210-dim space.

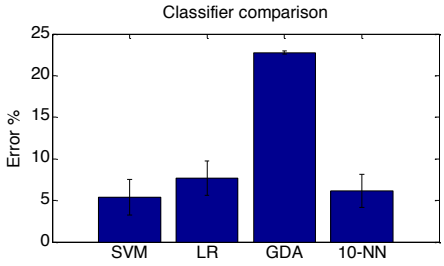

Figure 5: Four learning algorithms performance using 10-fold cross validation

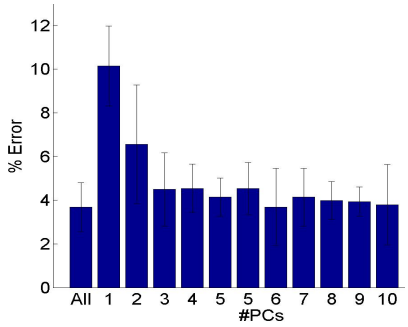

Figure 4: SVM performance as function of number of PCs.

We can see that SVM achieves the best performance (5% classification error). This can be explained by the fact that our data is still in a high dimension space, and learning algorithms are prone to overfitting in this case, especially where the number of examples is low. The exception is the SVM algorithm, that has a low Effective VC Dimension, and can produce robust classifiers. The GDA algorithm performs especially poorly, indicating that the data distribution can't be modeled as Gaussian.

Effect of Dimensionality Reduction

Based on the above results, we selected the SVM method, and revisited our decision to choose the three leading PCs to project the data on, as shown in Figure 5.

We can see that 3 PCs capture almost all of the relevant information of all 192 channels. Note that the other learning algorithms we applied would not be able to converge to a solution if run on the original data without preliminary PCA.

Effect of Time Evolution

Finally, we tried to answer this question - how soon does the primate know it made a mistake? To asses this, we trained per-time-window classifiers (100ms windows, every 50ms) and measured their performance, results are in Figure 6.

First of all, we see that 300ms before selection, error rate is ~30%, which is the expected performance of a classifier without information, since the failure rate is 30%. Progressing in time, we see that the trials are classified with better precision, and even before the selection, a prediction can be made with around 15% error rate. The minimum error of classification is around 200ms after the selection.

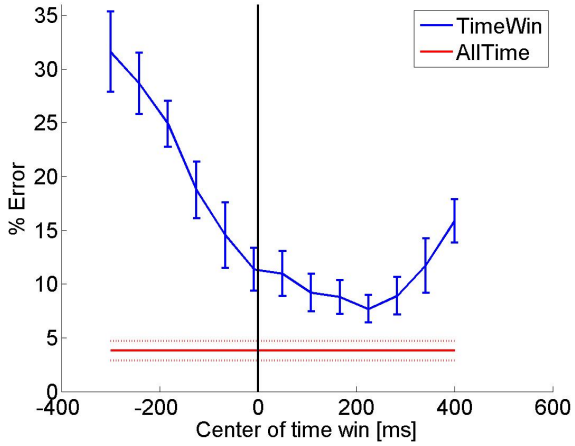

Figure 6: Performance of time-window classifiers, windows of 100ms (blue). Classification based on all time samples (red)

Summary

In this project we've shown that data related to the success or failure of a trial can be extracted and expressed in three primary components. Furthermore, an SVM classifier can be trained on this data to achieve and error rate of $5 \pm 1\%$. Indication about target selection error is available even before the error occurs, but is most evident 200ms after the selection.

Error detection can improve the speed and precision of speller-based BCI applications.

For future work, we would like to implement a real-time error detector that would prevent wrong selections. In addition, we would try training separate classifiers for each time window, and then bagging their results into a single output. For example, by using 10 SVMs aggregated by a logistic regression classifier.

Finally, given that false-positive and false-negative classifications can have different costs, we should train a weighted classifier, that takes these costs into account and minimizes the expected cost.

Acknowledgments

The experiments were done in prof. Shenoy's lab and with the help of Sergey Stavisky and Jonathan Kao.

Reference

1. Kao, J. C., Stavisky, S. D., Sussillo, D., Nuyujukian, P., & Shenoy, K. V. (2014). Information Systems Opportunities in Brain–Machine Interface Decoders.
2. Bottou, L., Cortes, C., & Vapnik, V. (1994). On the effective VC dimension.