Toward a NIRS Brain Computer Interface

Ilya Sherman, Nikil Viswanathan, and Tony Wu

Dr. Xu Cui and Daniel M. Bryant

Abstract

NIRS is a non-invasive spectroscopic method for measuring oxygenated and deoxygenated hemoglobin concentrations in cortical regions of the brain. In this report we investigate the use of machine learning techniques to train a model for online classification of a specific motor activity -- finger tapping -- using NIRS data. Thus, NIRS measurements of blood flow serve as an indirect proxy for the direct signal -- neural activity. However, changes in blood flow to and from the brain are relatively slow; and so there is a significant delay before the brain activity registers at peak level in NIRS output. Furthermore, the NIRS signal tends to be extremely noisy, as many factors may play a role in the concentration of blood in the brain. To tackle these problems, we first pre-filter the data to smooth out the noise, and then compute key features of the data. We use PCA and greedy feature selection to improve the robustness of our features and to reduce overfitting. Finally, we train classifiers using the resulting features to classify NIRS data as corresponding to periods of idleness or finger tapping.

1. Introduction

Imagine being able to turn on or off a TV not by pushing a red button on your remote control, but just by thinking "TV on" or "TV off". As a first step in the direction of brain computer interfaces, we are working on using the NIRS brain-imaging technique to detect a simple, strong brain signal corresponding to periods of active finger tapping. Data is gathered through a fairly simple experiment: the subject puts on a NIRS brain scanner and is periodically instructed to start or stop tapping her fingers.

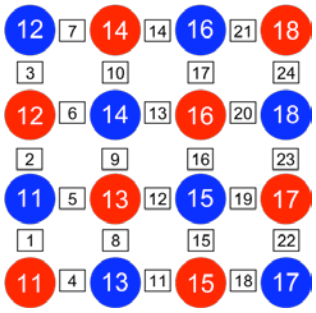

Figure 1. NIRS Sensors

The NIRS brain scanner is essentially just a head garment with sensors that can measure blood flow in different regions of the brain. Roughly, each pair of sensors corresponds to one channel; the diagram on the right shows the sensor and channel layout for the left hemisphere scanner. For each channel, the NIRS scanner detects the concentrations of oxygenated and deoxygenated hemoglobin. This signal is a pair of real numbers, and is collected with a time resolution of 0.1 seconds. The activity of cortical regions of the brain can also be directly detected by measuring electrical fields. However, the state-of-the-art technique for measuring such signals -- the EEG technique -- is far less practical as a signal source for a brain-computer interface; it is more expensive and less robust than the NIRS scanner, in that it does not allow subjects to move around.

The NIRS scanner measures a secondary or derivative signal rather than the direct source signal of interest. As a brain region becomes active, it requires energy to maintain its activity and blood flows into it, bringing oxygenated hemoglobin. As the oxygen is used up, the hemoglobin becomes deoxygenated and flows out from the brain region to be replenished. This is a fairly consistent signal, but it exhibits an inherent delay, due to the speed of blood flow, in measuring brain activity.

Our goal in this project is to design an online classifier that can detect brain activity accurately and with a minimal delay -- you wouldn't want to have to think "TV on" for ten seconds straight before the TV actually turned on! The time difference between when the patient begin tapping and when the classifier first correctly identifies this is called the onset delay. Previous work has succeeded in designing classifiers with very high accuracies, but onset delays in the 3- to 6-second range [1][2].

2. Baseline

As a very naive baseline, we ran an SVM-based classifier using only the raw NIRS measurements from channel 13 (the one that directly measures the cortical region that controls finger tapping). On the non-noisy data set, this resulted in a training accuracy of about 73% and an undefined onset/offset delay. By an "undefined delay", we mean that measuring the delay doesn't make sense for this baseline: it is unable to even identify *any* tapping, much less do it with any form of delay!

Delay Calculation

It is worth noting that establishing a good metric for the testing delay is itself a nontrivial task. The delays we present are computed as follows: For each period of finger tapping activity, we compute the delay to the earliest subregion L of duration at least three-quarters of a second, such that our

classifier correctly classified each sample in *L* positively. We then take the overall delay to be the mean of the delays over all periods of finger tapping activity.

3. Classifiers

3.1 SVM

After plotting the raw data following some pre-filtering (see Section 5), we predicted the feature space would form a approximately linearly separable data set. For this reason a majority of our trial runs were conducted using SVM with a radial basis function kernel, which resulted in a linear separating hyper-plane. In the end using the right combination of features History and Gradient we were able to get a test accuracy of 85.80% and a delay of 3.44 seconds. We used a greedy feature selection algorithm in order to get a locally optimal set of features (See Section 7).

3.2 AdaBoost

As an alternative to the SVM classifier, we tried using AdaBoost to see if we could achieve better performance. We ran two variations on the AdaBoost algorithm: Gentle AdaBoost and Modest AdaBoost [AdaBoost Toolbox], using decision tree stumps for each. Using only the features oxygenated and deoxygenated bloodflow as a baseline, we started off with an accuracy of around 72% (both variations performing similarly) and, again, and unbounded delay. However, using all of the reasonable features, filtered through PCA, gave much better results: Gentle AdaBoost is more finicky, prone to overtraining; but with careful parameter tuning we were able to attain 86.01% test accuracy. With less careful parameter selection, this variation achieved near-perfect training set classifcation, but test accuracies closer to 83%. Modest AdaBoost is a variation that is inherently resistant to overtraining, and pretty much any reasonable parameter setting gave a test accuracy around 85.5%, with a similar training accuracy. Both variations resulted in a delay of about 3.7 seconds. Thus, we found the peak performance of AdaBoost to be nearly identical to that of SVM. This offers some evidence that surpassing the present results likely requires a radically different approach, perhaps with more focus on pre-processing the data than on novel features or learning algorithms.

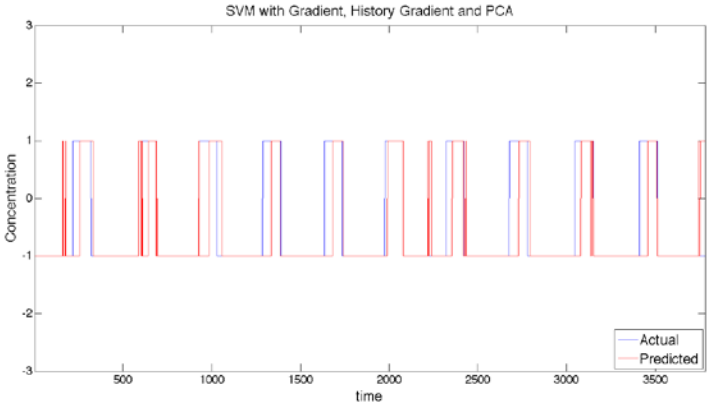

Figure 2. Optimal SVM Output

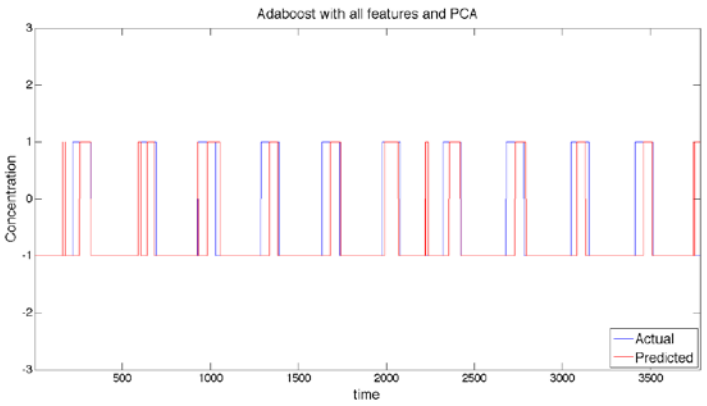

Figure 3. Optimal AdaBoost Output

4. Features

Channel Variation

The oxygenated and deoxygenated concentrations from channel 13 turned out to be the most relevant to predicting whether the subject was finger tapping, from both a empirical and theoretical standpoint. However, we found that by using also the channels adjacent to 13 -- 9, 10, 16, and 17 (see figure [1]) -- we were able to improve test accuracy from the baseline by 2% and reduce onset delay by 0.7 seconds.

Gradient Slope/Rate of Change

We looked at slope (rate of change, or first and second derivatives of the values). In general, changes in bloodflow through a region of the brain indicates "brain activity" in that region. By measuring the gradients of the blood flow, the classifier is able to identify critical transitions more quickly than it could using a simple threshold. Simply using the gradient of oxygenated and deoxygenated blood is not helpful at all on its own (without any pre-filtering).

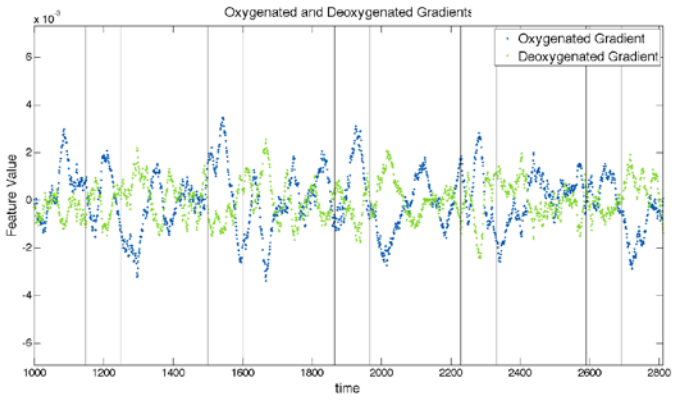

Figure 4. Gradient Features

Ratio of Oxygenated/Deoxygenated Bloodflow
We added this feature as a cross-term between the two signals we could directly measure, but it only resulted in dramatic overfitting to the training set (see results in Section 7).

Linear Combinations of Oxygenated and Deoxygenated Bloodflow
We chose linear combinations of oxygenated and deoxygenated bloodflow concentration as one of our features, which worked out fairly well on top of the pre-filtering that we did. We also took the gradient and second derivatives of these values and they made (see results in section 7.) somewhat of a small difference in test accuracy.

Classifying with Look-ahead
Though we are ultimately looking for an on-line algorithm, if we incorporate future data, we're able to get astounding accuracy. In fact doing something as simple as taking oxygenated data set and smoothing it so that the ith timestep is the average of the next 50 time-steps gives us astounding results: 90.5% accuracy and 1.5 seconds of delay. We did not include these results in the table as they're not usable to the end-product of developing an online algorithm.

Previous Predicted Label
We found that there was a tradeoff between attaining low delays and high false-positive rates. As one way to balance these, we tried adding a feature that would introduce a penalty for changing labels over time. Of course, if we use the actual label of the previous datapoint as a feature, this gives almost perfect accuracy, but is not a valid feature for the test set. As a replacement, we tried substituting the predicted label of the previous data point. For the training set, we computed this feature by initializing it to a constant, and then iteratively running SVM on the training data until the feature values converged. We then labeled the test data one at a time, and updated this feature for future data points based on the current prediction. However, this feature did not end up helping the performance of the classifier, most likely because it was just too noisy. A possible direction for further exploration is to instead look at the confidence values returned by AdaBoost, and reject label transitions that have a confidence value less than a learned threshold.

5. Pre-filtering

NIRS data is inherently noisy, as blood flow through the brain can be somewhat sporadic. This turned out to be problematic, as certain features like gradient, depend on a smoothed surface, and taking the gradient of raw data proved not to be effective. We explored several smoothing techniques that worked fairly well in reducing the noise. It turns out that smoothing the data was the singular most effective action in terms of increasing accuracy and reducing delay.

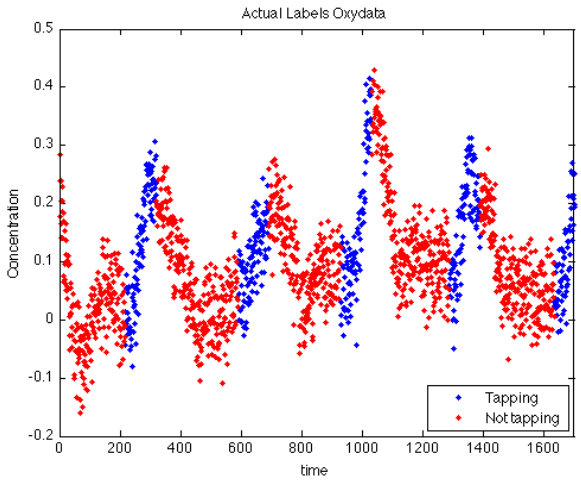

Figure 5. Output without smoothing

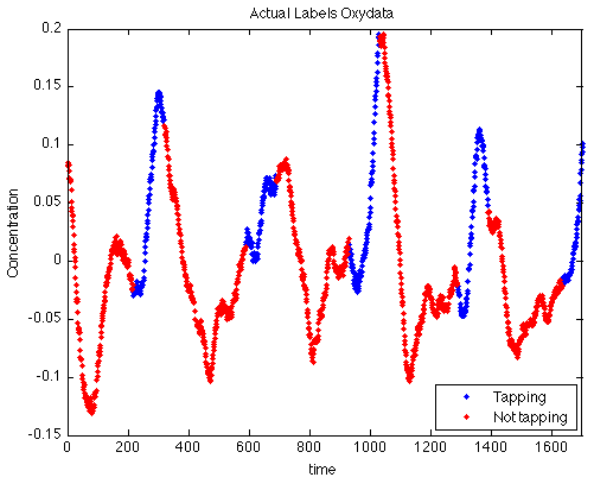

Figure 6. Output with smoothing

5.1 Exponential Moving Averages

Running classification techniques naively led to poor results, and as we examined the feature space, we realized the need for some type of smoothing of the data. We first smoothed the data by taking at each timestep *t* the exponentially weighted moving average of the previous timesteps and then normalized the data afterwards. This proved enormously helpful in smoothing the data and straightening out features such as gradient and history. Large fluctuations from timestep to timestep were removed and we were able to work with a more natural transformation of the data set. For virtually all of the features attempted, EMA shifted our accuracy from the 70% range to the 80% range.

5.2 Other Filtering

We also tried other smoothing techniques such as taking the average every *k* points, as well as running a low-frequency Chebyshev filter (both type 1 and type 2). Both filters had little additional improvement over EMA.

5.3 Reclassifying Training Data

Another approach we tried was converting a two-class problem into a three- or a four-class problem. This approach was motivated by the simple observation that all of the other approaches we tried either resulted in large delays or in high false-positive rates. This suggests that early finger tapping blood flows look qualitatively more like a passive state than an active state; but perhaps the best clustering of the data would label these early regions as a distinct transition state, qualitatively unlike either the passive or the active state. Thus, we tried relabeling the data so that early periods of finger tapping belonged to a different class rather than the same class. We then ran a one-against-one multiclass SVM classifier using these labels. Finally, we collapsed early finger tapping and sustained finger tapping into one class for evaluating the results. However, this approach proved ineffective: the accuracies decreased by several percentage points rather than increasing. One possible reason for this is that we tried a fairly naive approach for distinguishing between "early" and "sustained" finger tapping. One interesting possibility for further exploration is to run a clustering algorithm on the data corresponding to each label, and assign different sublabels to each cluster.

6. Principle Component Analysis

With 48 channels of raw data, we noticed an increase in generalization error when adding several features, indicating overfitting. Additionally when trying to decide which set of brain data channels to use, we initially used all 48 channels to figure out the best combination which resulted in each SVM run taking several minutes. When adding features, we needed the increase in speed from PCA. We applied PCA as a postprocessing step on the selected features, chose the first (sorted by decreasing) k component, which cumulatively accounted for ~90% of the variance, and then passed the features projected onto the reduced dimension space (typically down to 6-8 dimensions) into the classifer. As noted above, this merely improved the runtime of the SVM classifier, but actually contributed about 4% to the accuracy of the AdaBoost classifier.

7. Feature Selection

In order to discover the best set of features to use, we implemented a greedy feature selection algorithm, using a heuristic of best accuracy to choose the optimal feature to add. Though this might result in a local optimum, we noticed that several of the features had similar performance and that we achieved (at least locally) optimal performance with just a few features.

Feature (cumulative)	Accuracy (%)	Delay (seconds)
Gradient History	77.546	---
History	78.736	4.55
Gradient Scale	79.238	4.17
OxyData + DeoxyData	79.450	4.09
Oxygenated Data	79.344	4.09
OxyData – DeoxyData	79.450	4.09
Deoxygenated Data	79.503	3.99
Gradient	79.503	3.99
EMA'd Gradient	79.423	3.99
OxyData / DeoxyData	72.944	---

Table 1. Feature Selection Algorithm – With EMA

Feature (cumulative)	Accuracy	Delay
History	76.699%	5.82
Gradient	85.797%	3.69
Deoxygenated Data	85.745%	3.65
Gradient Sign	85.639%	3.55
Gradient EMA'd	85.480%	3.56
Gradient	85.216%	3.46
OxyData - DeoxyData	85.216%	3.45
OxyData + DeoxyData	85.083%	3.40
Oxygenated Data	85.083%	3.40
OxyData / DeoxyData	72.706%	---

Table 2. Feature Selection Algorithm – Without EMA

8. Remarks & Commentary

8.1 Which Specific Features

One major insight is that we get approximately the same accuracy and delay as long as we pick two or three good features (Gradient, gradient history, OxyData + DeoxyData, OxyData - DeoxyData, etc). Therefore for the purposes of reducing delay and/or improving accuracy, which features we select don't seem to make that big of a difference.

8.2 Which classifier

We also found that the specific learning algorithm did not seem to matter too much, except insofar as it can help avoid overfitting: both the SVM approach and the boosted decision trees approach indicated a tradeoff between low delays and low false-positive rates, with similar results attainable using either approach.

8.3 Pre-filtering makes a big difference

We found that pre-filtering the data was what made the large difference in jumping from a 72-73% accuracy (Adaboost and SVM, respectively) to the mid-80% range. Whether it's taking the fixed weighted average or an exponential moving average, the smoothing effect allows the classifier to find cleaner boundaries within the feature-space. Almost all the features (especially gradient, and gradient direction) suffer huge fluctuations as the NIRS data is very noisy. So it makes perfect sense that smoothing will help the classifier achieve higher accuracies and better delays.

Acknowledgments

We would like to thank Dr. Cui and Daniel Bryant for the data, advice, and guidance on this project.

Appendix

Definitions

NIRS - Near infrared Spectroscopy. Uses the transmittance of tissues to detect haemoglobin absorption. This is useful for tracking blood flow of oxygenated and deoxygenated blood in the brain. We use this data in order to train on whether the user is in a state of finger-tapping or not.
Channel - A location on the brain on the brain where NIRS records bloodflow in a particular region. In particular the channel we are most interested is **channel 13**, as that is the location of where the motor cortex (the region of the cerebral cortex that controls motor movements). See diagram above.
Onset delay - The difference in time from when the subject begins tapping to when the machine learning algorithm first classifies the time interval as tapping.
Offset delay - The difference in time from when the subject stops tapping to when the machine learning algorithm first classifies the time interval as not-tapping.
Accuracy - The percentage of time instances labeled (either tapping or not tapping) by the algorithm correctly

References

[1] Sitaram, Ranganatha. Caria, Andrea. Birbaumer, Niels. Hemodynamic brain–computer interfaces for communication and rehabilitation

[2] Sitaram, Ranganatha. Zhang, Haihong. Guan, Cuntai. Thulasidas, Manoj. Temporal classification of multichannel near-infrared spectroscopy signals of motor imagery for developing a brain–computer interface

[3] Adaboost Toolbox. http://graphics.cs.msu.ru/ru/science/research/machinelearning/adaboosttoolbox

[4] Libsvm Package. http://www.csie.ntu.edu.tw/cjlin/libsvm/

[5] Exponentially Weighted Moving Average. http://en.wikipedia.org/wiki/Moving_average#Exponential_moving_average

[6] Matlab Chebyshev Filter. http://www.mathworks.com/access/helpdesk/help/toolbox/signal/cheby1.html