EEG Data Analysis for BCI?

Introduction

The general idea behind this project is to work on some of the problems associated with creating a computer interface for the physically handicapped. Such an interface has been dubbed BCI (Brain-Computer Interface) because of its reliance on the measurement of neural activity through EEG (*E*lectro*e*ncephalo*g*raph). The goal of the project is to classify unknown EEG data as associated with either a left finger movement or a right finger movement. A successful classification algorithm could then theoretically be implemented in a BCI to give patients limited control over the computer by altering the electrical activity in their brain, or simply by thinking. Based on real EEG data, generously provided by Fraunhofer-FIRST, Intelligent Data Analysis Group (Klaus-Robert Müller), and Freie Universität Berlin, Department of Neurology, Neurophysics Group (Gabriel Curio) through the BCI Competition 2003 webpage, an attempt at such an algorithm is developed and presented.

Data Collection

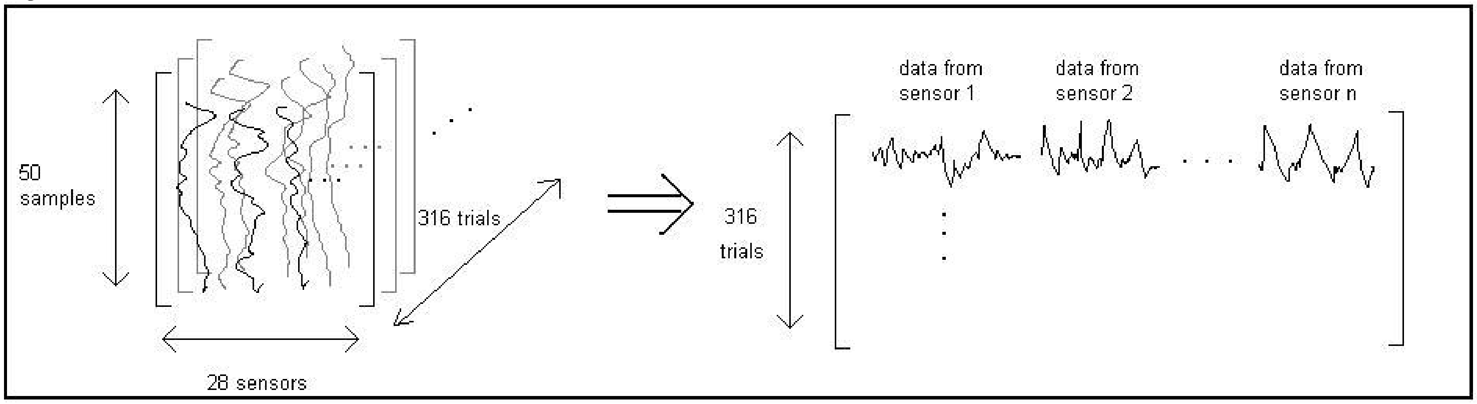
Data were collected from 28 EEG sensors as the subject moved either his left or right index finger in succession 416 times at his own pace. Only the data for the 500ms before the finger movement were used. This data is expressed as a 50x28 matrix for each trial. Only the first 316 of these trials are available for analysis, the remaining 100 are left unlabeled for classification.

Data Analysis

Part I

There are two matrices associated with the first 316 trials. One is a 50x28x316 matrix with all the raw voltage readings from the EEG from all the sensors for all 50ms. The second is a 1x316 matrix of labels for each trial. If the nth entry of this vector is 1, then the subject used a finger on his *right* hand on the nth trial; otherwise, he used a finger on his left hand.

Figure 1



To make this massive matrix easier to work with, it is converted to a 316x1400 matrix, where each row contains all the data from that trial. This transformation is schematized in Figure 1 (above). The process used to perform this transformation is shown below, where `x_train` is the original 50x28x316 matrix containing the raw data and `xnew` is the new matrix that is used for the remainder of the project.

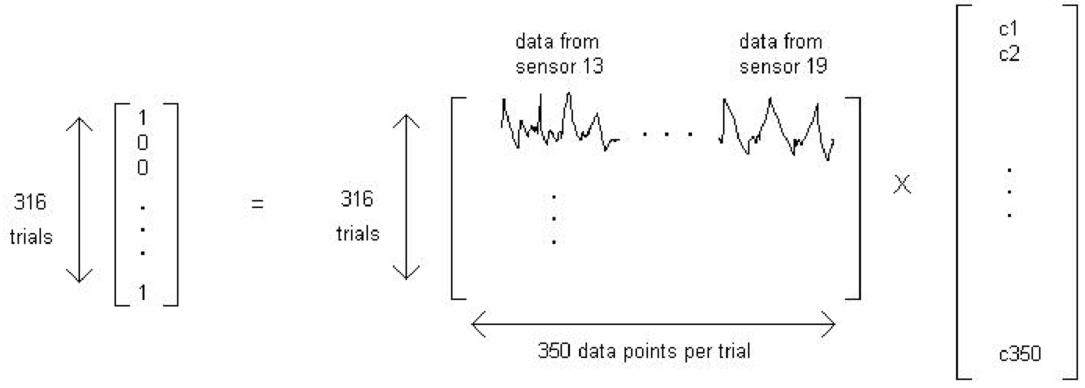
MATLAB code:

```
xnew = [];
xtrial = [];

for i=1:size(x_train,3)

for j=13:19
a = x_train(:,j,i);
xtrial = [xtrial a'];
end

xnew = [xnew; xtrial];
xtrial=[];

end
```

This process creates a 316x1400 matrix containing all the original EEG data, but such a matrix was found to be too large to work with in a computationally efficient way. Since part of this project involves making judgments about which sections of the data are more relevant, such a judgment was made at this point to reduce the computational load for the remaining sections. Hence, as shown above, only data from certain sensors were used (13 through 19). These sensors correspond to the area above the motor cortex, an area of the brain responsible for initiating motor movements. Previous research has implicated measurements from sensors C3 and C4 (14 and 18 in our notation) in this area as being the best predictors of movement of the fingers (Kelly et al, 2002).

Now the data is used to construct a linear fit. We are searching for a vector c such that when multiplied by our data matrix on the right, gives us the vector with the proper labels, y_train'. This relationship is illustrated below.

Figure 2



The error-minimized solution to the general expression y = Ac (where y is m x 1, A is m x n, and c is therefore n x 1) is given by c = inv(A'A)*(A'y). This relationship was used to generate a vector c, and this vector was then used to predict the classification of the unknown trials.

Part II

Once this vector is generated, it is interesting to get a measurement of how accurately it predicts the proper classification. One way to do this is to train the classifier on all but 1 data point, and then use the calculated vector to predict the classification of the remaining point. This process can be repeated for each point in the data set to provide a good picture of how accurate the classifier is. The code used to do this is shown below:

MATLAB code

```
for n=1:size(x_train,3)

xnewer = [xnew(1:n-1,:); xnew(n+1:end, :)];
y_train_newer= [y_train(:,1:n-1) y_train(:,n+1:end)];

%solve
c = pinv(xnewer'*xnewer)*(xnewer'*y_train_newer');

%test
testit = xnew(n,:)*c;
if testit<0.4637
testit=0;
else
testit=1;
end
if testit == y_train(:,n)
score = score + 1;
whichtrialsright(1,n)= 1;
else
whichtrialsright(1,n)= 0;
end
end
end
pscore=(score/n)*100
```

What the above code does is to create a matrix of all the rows except the nth row, and use this matrix to solve for the vector of coefficients, c. The excluded row is then multiplied by the vector c to create a 1 x 1 matrix corresponding to what would be the nth entry in the output vector. This value is lies near 0 or 1, but does not typically equal 0 or 1, so an arbitrary cutoff is created, above which the value of prediction is rounded to 1 and below which the prediction is rounded to 0. The value for the cutoff used in this code is the mean value of the predicted outputs from one of the attempts to use the algorithm. This value is then tested against the known value from `y_train`.

With this self-evaluation built into the algorithm, several variations on the original approach could be tested. For example, it seems possible that the accuracy of the algorithm can be increased by using only the time samples immediately preceding the event. To test this, the classifier was trained in two conditions: using all 50 time samples and using only the last 25. The results, as shown in Table 1, indicate that vital information may actually be lost using this approach: the classifier is 61% accurate in the first condition, but only 56% accurate in the second. The remainder of the techniques employ the full range of time values.

Part III

All the classification to this point has been done in the time domain. It is a well known fact that the brain's activity oscillates in certain frequency bands. This fact forms the basis of clinical EEG applications, in which the doctor can visually scan an EEG reading in search of irregularities in the frequency of the brain's activity. In our research paradigm, analysis in the frequency domain can be useful for much the same reason: certain frequencies may correlate better with certain activities. For example, lower frequencies in the 0-8 Hz range (Delta and Theta) are not thought to be particularly involved in motor movements, so would constitute some of the statistical noise in the frequency spectrum. By selecting which frequency bands are used to fit the classifier to the data, we can perhaps increase its accuracy. There are several mathematical tools for analysing which components of a frequency distribution are more useful for this purpose, but they are not reviewed here (see Eric's project). Instead, certain frequencies are eliminated on neurophysiological grounds; this somewhat a priori elimination is justified, however, by corroborating evidence from some of these more sophisticated techniques (Blankertz et al).

MATLAB is capable of calculating the frequency spectrum in a variety of ways; a few of them are investigated here. The results of this investigation are summarized in Table 1.

Table 1

Time Domain methods	**Percent Correct**
The full time series	61%
Last half of the time series	56%
Frequency Domain methods	
Pwelch, sampling rate = 20Hz, frequencies < 6Hz removed	65%
Pwelch, sampling rate = 1 Hz, frequencies < 6Hz removed	62%
Pwelch, sampling rate = 20 Hz, all frequencies	50%
Pcov, all frequencies	45%
Pburg, all frequencies	64%
Peig, all frequencies	42%
var(Pwelch)	46%

As can be seen, the calculation of the frequency spectrum that gives the most accurate results is the Pwelch command. This result is counterintuitive; according to the MATLAB help file, Welch's method is a nonparametric method and among the most simple of the methods MATLAB can use. An example of a parametric method is Pburg, which performed about as well as Pwelch. The counterintuitive result is the poor performance of Peig, which according to MATLAB's help file, is best for sinusoidal and noisy data. The data from the EEG is indeed noisy, but perhaps it was not periodic enough to be analysed well by Peig.

Another possibility that was investigated was the difference in the variance of the spectrum between left and right handed trials. Visual inspection of the spectrums suggests that the spectrum of the right handed trials had a higher variance. A variation on the classifier was written to train the data solely on the variance of the spectral density:

MATLAB code

```
xnew = [];
xtrial = [];

for i=1:size(x_train,3)

for j=13:19
b = pwelch(x_train(:,j,i),[],[],[],20);
a = var(b(65:end));
xtrial = [xtrial a'];
end

xnew = [xnew; xtrial];
xtrial=[];

end
```

This classifier (as shown in Table 1) did not perform especially well, essentially performing at chance. This is not surprising since a ttest revealed no significant difference between the variance of right handed trials and the variance of the left handed trials, and there is no physiologically parsimonious reason to believe that one hemispheres employs a bigger range of frequencies than the other.

Conclusions

From this analysis we can conclude several things about this approach to EEG data for a potential BCI interface. One is that a simple linear fit accurately classifies above chance levels, though not much above (61% correct). It is also apparent that using selective frequency bands can improve this accuracy (to at least 65%) by throwing out irrelevant data in low end of the frequency spectrum and that the accuracy of the classifier depends on the method by which the frequency spectrum is calculated. For more information about rigorous approaches to determining the ideal components for analysis, the reader is referred to Down 2003 (in press).

References

Blankertz, B, Curio, G, & Muller, K. "Classifying Single Trial EEG: Towards Brian Computer Interfacing."
Kelly, S, Burke, D, de Chazal, P, & Reilly, R. "Parametric Models and Spectral Analysis for Classification in Brain-Computer Interfaces." *Proceedings of 14th International Conference on Digital Signal Processing*.