34th Annual International Conference of the IEEE EMBS
San Diego, California USA, 28 August - 1 September, 2012

Asynchronous Brain Computer Interface using Hidden Semi-Markov Models

Gareth Oliver[1], Peter Sunehag[1] and Tom Gedeon[1]

Abstract— **Ideal Brain Computer Interfaces need to perform asynchronously and at real time. We propose Hidden Semi-Markov Models(HSMM) to better segment and classify EEG data. The proposed HSMM method was tested against a simple windowed method on standard datasets. We found that our HSMM outperformed the simple windowed method. Furthermore, due to the computational demands of the algorithm, we adapted the HSMM algorithm to an online setting and demonstrate that this faster version of the algorithm can run in real time.**

I. Introduction

Brain Computer Interfaces(BCI) make use of devices such as EEGs to read a person's brainwaves and classify them. This allows the BCI to be used as a control device. BCI has many interesting applications, both as a hands-free control device as well as being one of the few ways people suffering from full body paralysis can communicate [1].

The majority of BCI research to date has focused on synchronous classification, where segments with known boundaries are classified. Ideally a BCI would operate asynchronously, requiring the data to be segmented as well as classified. Most commonly this is done by selecting a window around each time step to perform the classification. Hidden Semi-Markov Models have been shown to be highly effective at segmentation of data in a variety of different fields, such as movement tasks[2] and functional MRI sequence analysis [3]. HSMM makes use of variable duration windows to allow for more accurate segmentation of time-series data.

This paper will first introduce the preprocessing and feature extraction methods that are to be used, as well as the HSMM and the Support Vector Machine classifiers. A method for applying the HSMM algorithm online will also be discussed. We compare the proposed HSMM method with a simple window based method. Finally the results indicating the speed issues with the HSMM will be discussed.

II. Preprocessing and Feature Extraction

The most common forms of preprocessing are Frequency Filtering and Channel Selection. Frequency Filtering is used to isolate frequency bands within which responses occur. It has been shown that different actions cause response in particular frequency bands. For motor based BCI, responses are known to occur in the α (8-12 Hz)and β (13-30 Hz) rhythms [4]. Additionally the θ rhythm (1-7 Hz) has been used successfully. The commonly used Butterworth filter was used to perform the band-pass filtering in this paper [5].

Channel selection is another important preprocessing step. This eliminates electrode channels that are not relevant to the separation of the classes, thus reducing the dimensionality and noise within the data. As channel selection was not the focus of this paper this was performed manually using expert knowledge.

A. Feature Extraction

Common Spatial Subspace Decomposition (CSSD)[8] and Common Spatial Patterns (CSP)[7] are among the most popular feature extraction techniques for motor imagery BCI. CSSD seeks to reduce the dimensionality of the data by selecting the channels that maximise the variance between the classes. This is done through the simultaneous diagonalisation of co-variance matrices. The algorithm used can be found in [8]. For completeness sake the algorithm used is given briefly below.

$$\begin{aligned} R_a &= X_a.X_a^T \\ R_b &= X_b.X_b^T \\ R &= R_a + R_b \\ &= U_0.\Sigma.U_0^T \\ P &= \Sigma^{-1/2}.U_0^T \\ S_a &= P.R_a.P^T \\ S_b &= P.R_b.P^T \end{aligned}$$

First the co-variance matrix of each class, R_a and R_b, are estimated from the training data. The eigenvalues, Σ and eigenvectors, U_0 of R are then found. These are used to obtain the whitening matrix P. The co-variance matrices are then whitened to give S_a and S_b. After ranking the vectors in descending order by eigenvalue, the first m eigenvectors are selected from S_a and combined to form U_a. Similarly the first m eigenvectors of S_b are selected and combined to form U_b.

$$\begin{aligned} SF_a &= U_a^T.P \\ SF_b &= U_b^T.P \end{aligned}$$

U_a and U_b are used to construct the corresponding spatial filter matrices SF_a and SF_b. The log of the variance of each trial is then selected as a feature. This gives a feature vector containing m values for each class, as given below for some input X.

[1]G. Oliver, P. Sunehag and T. Gedeon are with the Research School of Computer Science at the Australian National University. gareth.oliver at anu.edu.au

$$features = (var(SF.X))$$

III. Classification

In this section the classifiers for both the asynchronous and synchronous problems are presented. To allow synchronous testing to be transferred to the asynchronous problem we extend the synchronous support vector machine classifier to a hidden semi-markov model classifier. This makes parameter exploration much easier and quicker to perform.

A. Support Vector Machine

Support Vector Machines (SVMs) are among the most popular, and successful classifiers. They have been used to great success in the classification of motor tasks[9]. SVMs seek to maximise a decision boundary between the classes being classified. For simplicity a linear kernel was used. The decision boundary problem can be formalised as finding a discriminant function F(x,y) so that the prediction $\hat{y}(x)$ satisfies

$$\hat{y}(x) = \arg\max_{y \in C} \; F(x, y)$$

where

$$F(x, y) = \langle \phi(x, y), w \rangle$$

ϕ is the feature map and C is the set of classes. Given a set of *n* training pairs (x^i, y^i). A SVM choses a w so as to minimise

$$\lambda \frac{\|w\|^2}{2} + \frac{1}{n} \sum_{i=1}^{n} \max_{y' \in C} \left\langle \phi(x^i, y') - \phi(x^i, y^i), w \right\rangle + \delta_{y^i, y'} \quad (1)$$

Where $\lambda > 0$ is a regularisation constant. A solution to this convex optimisation problem can be found by performing stochastic gradient descent. From equation 1 is follows that the weight update law is

$$w_{t+1} = w_t - \rho_t(\lambda w_t + \phi(x^i, y^*(x^i, y^i)) - \phi(x^i, y^i))$$

where ρ is the learning rate. Convergence can be guaranteed by using $\rho_t = \frac{\tau}{1+t}$ as it satisfies the robbins-monro conditions required for convergence[13].

B. Windowed Segmentation

Windows are the standard method used for BCI segmentation [10]. It has the advantage of being fast, and easy to implement. It was used as a benchmark to evaluate the effectiveness of the HSMM method. The Windowed method works by taking a fixed window around each time step. It uses this window to extract features and assign a class to that timestep.

Inputs

obs[m] - observed data
W - model parameters for each state
S[m+1,c] - Matrices for storing dynamic solution
J[m+1,c] - stores previous class
L[m+1,c] - stores previous time point

Algorithm

$S[0,:] \leftarrow 0$
$J[0,:] \leftarrow 0$
$L[0,:] \leftarrow 0$
for (i = 1,$i < m + 1$, i++) **do**
  for (c : classes) **do**
    $g_{b,c} \leftarrow \phi(\Phi(obs[i-b:i]), w_c)$
    $L[i,c], J[i,c] \leftarrow argmax_{b,p_c}(g_{b,c} + S[i-b, p_c])$
    $S[i,c] \leftarrow max_{b,p_c}(g_b + S[i-b, p_c])$
  end for
end for
$j \leftarrow argmax(S[m,:])$
$i \leftarrow m$
$seq \leftarrow []$
while (i != 0) **do**
  for (k=0, k < i - L[i,j], k++) **do**
    seq.append(j)
  end for
  $j, i \leftarrow J[i,j], L[i,j]$
end while
return reverse(seq)

Fig. 1. Viterbi Algorithm

C. Hidden Semi-Markov Model

The Hidden Semi-Markov Model(HSMM) extends Hidden Markov Models(HMM) in the sense that HSMM assigns a single label for a series of observations, rather than for a single observation. We define all possible labels that can be transitioned to from a segment (n_i, l_i) as

$$(n_{i+1}, l_{i+1}) \in S(n_i, l_i) = \{(n, l) | l \in C \; n - n_i \in T(l)\} \quad (2)$$

where C is the possible class labels and T(l) is the possible durations for a given class. We also define Y(x) as the set of possible labeled segmentations that abide by the constraints of equation 2. The goal is now to find some descriminant function F(x,y) so that the maximiser of F(x,y) provides a good segmentation and classification of the data. We use a linear discriminant function

$$F(x, y) = \langle \Phi(x, y), w \rangle$$

Inputs
X - Data
N,L- List of Segmentations
i - Current Segment
m - Maximum Segment Duration

Algorithm
1: $No, Lo \leftarrow HSMM(X[n_{(i-1)} : n_i) + m])$
2: $N \leftarrow N \backslash N_i \cup No_0 \cup No_1$
3: $L \leftarrow L \backslash L_i \cup Lo_0 \cup Lo_1$
4: $i++$
5: **return** N, L, i

Fig. 2. Online HSMM

where

$$\Phi(x, y) = \sum_{n_i \in Y} \phi(x, n_{i-1}, n_i, l_i)$$

where Φ is the joint feature map that breaks down into a sum over the features of each segment. For a particular segment, this setting becomes the same as that which was discussed in section III-A. We perform parameter estimation for the HSMM by performing the described SVM training on a dataset that consists of all individual segments and their labels. For maximising the discriminant function the dynamic programming solution Viterbi algorithm was used [2], given below.

D. Online HSMM

Performing the above algorithm on a large segment of data would be very slow, and impossible to do in an online setting. However, if we make the assumption that the segmentation of the current segment (n_i, l_i) will be unaffected by segmentation of data after segment $(n_{(i+1)}, l_{(i+1)})$ then the result of (n_i, l_i) can be fixed after the segmentation $(n_{(i+1)}, l_{(i+1)})$ is found. This allows the HSMM algorithm mentioned above to be used in an online setting, and reduce the length of each continuous segment the HSMM is performed on. A simple way of performing this is as follows in Fig. 2.

Here m is the largest duration in T, and HSMM is the previously mentioned HSMM algorithm, returning the segmentation and classification of the data. A minimum of two segments is guaranteed due to m. This can be repeated until all of the data is classified.

Preprocessing	Feature Extraction
Filter 1-7 Hz Filter 8-12 Hz Filter 14-28 Hz	CSSD, m=3

TABLE I
PREPROCESSING AND FEATURE EXTRACTION

	IVb	I a	I b	I c	I d	I e	I f	I g
Windowed	90	73	69	74	78	79	72	75
HSMM	94	77	73	78	82	83	73	80
OHSMM	94	77	71	76	81	82	72	78

TABLE II
ASYNCHRONOUS ACCURACY (%)

IV. EXPERIMENTS

Parameter exploration for both preprocessing and feature extraction methods were first carried out in a synchronous setting using the SVM. Due to the structure of the HSMM it is reasonable to assume that the parameters explored in the SVM should work well for the SVM. Table:I gives the preprocessing and feature extraction used for both asynchronous methods.

The asynchronous tests were then carried out on several datasets. The first is an artificial segmentation problem created from the training data of BCI competition III dataset IVb[11]. The second dataset used was BCI competition IV dataset I[12]. It consists of 4 subjects and 3 artificial datasets (c, d and e). The relax segments were removed to give a two class asynchronous segmentation problem. CSSD was used as the feature extraction method for both windowed and HSMM. The HSMM was given a window range of 150 to 800 for each dataset, and the windowed method was 100 samples wide. OHSMM made use of the online HSMM algorithm described earlier to perform the classifications.

The accuracy was calculated on each time point to give an overall accuracy. The HSMM consistently outperformed the windowed method. The OHSMM had a very similar set of results to the HSMM, being equal or slightly worse in all cases. One thing that was noted was that the HSMM had a smoother output when compared with the Windowed method.

It was noted in initial tests that the HSMM appeared to classify very slowly, especially compared to the windowed method. It makes sense that HSMM would perform slower than the windowed method, as it will perform the feature extraction and classification process $classes^2 \times durations$ more times than the windowed method for any given segment. It will also perform it on significantly larger pieces of timeseries data.

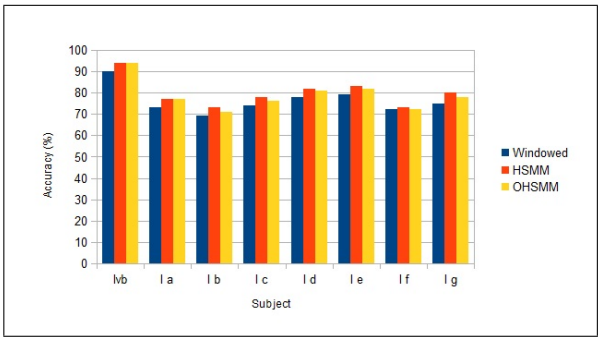

Fig. 3. Asynchronous Graph

Length	Viterbi Time (s)	Percent of Real time
200	0.01	0.5
400	0.03	0.8
600	0.74	12.4
800	3.44	42.9
1000	9.50	95.0
1200	16.31	135.9
1400	23.58	153.7
1600	30.90	193.1

TABLE III
SPEED TEST RESULTS

To determine if this would be a problem when attempting real time classification, the calculation time of the viterbi algorithm was measured for different segments of data. The code was written in inline C using the scipy.weave package[6] and compiled with MinGW g++ compiler. It was run on a i5-2400 processor. The data used was taken from the previous experiment.

The length is the number of entries in the data. For purposes of calculating the real time available it is assumed to be at 100 hz. The time the viterbi takes is given in seconds. The percent of real time is the percent of the total time available to perform the segmentation and classification that the viterbi consumes. The viterbi begins considerably faster as less calculation are done when not all durations are possible. The HSMM is sufficiently fast for segments up to 10s in length. For longer segments the Online HSMM to be used in real time in the given setting it needs to be sped up further. The easiest method for doing this is to limit the allowable durations so that only the n_th duration is calculated. This will reduce the number of calculation required to be performed at each time to $\frac{classes^2 \times durations}{n}$ (i.e. reducing it by a factor of n), at the cost of coarser segmentation.

An alternate solution is to make use of the highly parallelisable nature of the viterbi algorithm and make use of multiple processors. This could provide the required speed up, especially given the current trend of increasing the number of processors of CPUs while not improving the speed of each individual one.

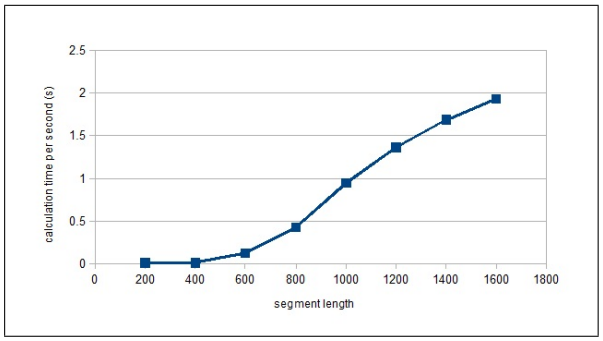

Fig. 4. Viterbi calculation time per second

V. CONCLUSION

A HSMM approach to segmentation of BCI data was proposed, and found to outperform a standard windowed method on several artificial and complex datasets. Additionally a method to use the HSMM Online was given. Finally the speed problems demonstrated by the Viterbi algorithm were investigated, and a variety of solutions to compensate for this were put forward.

REFERENCES

[1] A. Kubler, B. Kotchoubey, J. Kaiser, J R. Wolpaw, and N Birbaumer, Brain-computer communication: unlocking the locked in, *Psychol. Bull.*, 127:358-75, 2001.

[2] O. Thomas, P. Sunehag, G. Dror, S. Yun, S. Kim, M. Robards, A. Smola, D. Green, and P. Saunders, Wearable sensor activity analysis using semi-Markov models with a grammar, *Pervasive and Mobile Computing*, 6:342-50, 2010.

[3] S. Faisan, L. Thoraval, J-P. Armspach, and F. Heitz, Unsupervised learning and mapping of active brain functional MRI signals based on hidden semi-Markov event sequence models, *IEEE Transactions on Medical Imaging*, 24:263-76, 2010

[4] G. Pfurtscheller, and F. H. L. Da Silva, Event-related eeg/meg synchronization and desynchronization: basic principles, *Clinical Neuroscience* , 110:1842-57, 1999

[5] M. Grosse-Wentrup, M. Gramann, and M. Buss, Adaptive spatial filters with predefined region of interest for EEG based brain computer interfaces, *Advances in Neural Information Processing Systems* ,19:537-44, 2007

[6] E. Jones, T. Oliphant, P. Peterson and Others, SciPy: Open source scientific tools for Python. ¡http://www.scipy.org/¿, 2001.

[7] J. Muller-Gerking, G. Pfurtscheller, and H. Flyvbjerg, Designing optimal spatial filters for single-trial eeg classification in movement tasks. *Clinical Neurophysiology*, 101:787-798, 1998.

[8] Y. Wang, P. Berg, and M. Scherg, Common spatial subspace decomposition applied to analysis of brain responses under multiple task conditions: a simulated study. *Clinical Neurophysiology*, 110:604-614, 1999

[9] F. Lotte, M. Congedo, A. Lecuyer, F. Lamarche, and B. Arnaldi, A review of classification algorithms for EEG-based brain-computer interfaces, *J. Neural Eng.*, 4:R1-R13, 2007.

[10] G. Townsend, B. Graimann, and G. Pfurtscheller, Continuous EEG Classification During Motor Imagery and Simulation of an Asynchronous BCIs, *IEEE Trans on Neural Systems and Rehabilitiation Engineering*, 12:258-265, 2004.

[11] G. Dornhege, B. Blankertz, G. Curio, and K-R. Muller, Boosting bit rates in non-invasive EEG single-trial classifications by feature combination and multi-class paradigms. *Trans. Biomed. Eng.,* , 51:993-1002, 2004.

[12] B. Blankertz, G. Dornhege, M. Krauledat, K-R. Muller, and G. Curio, The non-invasive Berlin Brain-Computer Inteface: Fast aquisition of effective performance in untrained subjects. *NeuroImage*, 37:539-550, 2007.

[13] H.E. Robbins, and S. Monro, A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400-407, 1951.