Decoding directed covert attention in brain computer interfaces using hidden Markov models

Steffen Janssen
steffen.janssen@gmail.com
Radboud University of Nijmegen
Faculty of Science

Bachelor Thesis

August 1, 2011

Supervisors:

Marcel van Gerven
Ali Bahramisharif

Contents

Introduction

The topic of this thesis, *Decoding directed covert attention in brain computer interfaces using hidden Markov models*, is largely based on an experiment performed at the Technische Universität Berlin. This experiment was aimed at recording EEG brain signals from volunteers in order to research directed covert attention in brain computer interfaces (1). Brain Computer Interfaces (BCI) are systems designed to directly use brain activity to communicate with a computer, and to control a computer (2).

The brain controls motor functions in the body in order to perform and direct all body processes. Motor control in humans is directed by the brain, however not all people have full control over their motor functions due to a multitude of reasons. Paralysis can cause a person to be incapable of walking or even operating an electric wheelchair. Other people may be so severely paralyzed that besides moving, they cannot talk or communicate in any other way. These patients are classified as *locked in* (3) and belong to the most severe class of paralyzed patients that are not in a vegetative state. These people are a main target audience for Brain Computer Interfaces. A BCI system can improve their quality of life by offering another option to communicate with the outside world. Currently, most BCI systems are not sufficiently fast or accurate enough for all users, especially locked-in patients, to give an acceptable feeling of control over the system. A BCI system requires a minimum accuracy of 70% to give the feeling of control (4). The medium through which this communication takes place is typically a computer connected to a screen, in addition to the BCI specific apparatus physically connected to the subject in order to register the brain activity.

While the available technology enables us to register and record brain activity, these recordings cannot be directly translated back into thoughts. To do any kind of analysis on the recorded data, the embedded patterns which may be very hard to identify must be extracted using clustering and classification algorithms. Different types of algorithms can be better suited to this particular type of (BCI) information. This paper will focus on testing and experimenting with a particular kind of classification method: Hidden Markov Models (HMM). This classification method has been proven to yield results in similar classification problems (5, 6, 7), and could present itself to be useful for this particular problem of classifying BCI data.

I will be looking to find the answers to the following research questions:

- What is the classication performance of directed covert attention in BCIs using the average alpha power in a fixed time window and how does it compare to the method used in Treder *et al.*? (1)
- Does the inclusion of hidden Markov models contribute to the classication performance of directed covert attention in BCIs?

And the main research question:

- **What is the classication performance of directed covert attention in BCIs using a hidden Markov Model classifier with the data split into multiple time segments?**

Hidden Markov models

A hidden Markov model is a statistical model that extends the Markov chain model and can be interpreted as a Bayesian network structure based on Markov chains. It is used in many different pattern recognition problems.

Pattern recognition problems in bio-informatics (8), speech recognition (9), handwriting recognition (10), data mining (11), image and video technology (12) and AI are all examples where hidden Markov models have been successfully applied. HMM systems have also been applied to certain BCI problems (5, 6, 7). The BCI data obtained in the experiment that will be analyzed using these hidden Markov models is another example of a pattern recognition problem.

A Markov model or Markov chain is a way to model a stochastic process, in this stochastic process transitions between states are not pre-determined, the model can only predict future states based on probabilities. A deterministic model on the other hand, describes a process where each transition is guaranteed. The Markov model is defined to have the special property (Markov property) that its future state is independent of the past given its current state. That is to say, the probability distributions for the future states in the model are determined entirely by the properties of the current state it is in. A variation on the Markov model is a Markov chain with a limited memory. A Markov model of order m has a memory of the last m transitions that have occurred. This additional information can then be used in calculating the next transition probabilities for a Markov model.

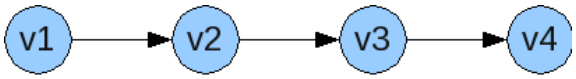

Figure 2.1: Graphical model for a first order Markov chain

There is a difference between Markov models and Markov chains. Markov chains are the simple variant where the system and the process being modeled are completely observable. In this scenario all the transitions between states are known and visible, they can be stored in a table and used to calculate path probabilities, future states and other statistical properties of the model in a straight forward fashion. Formally a Markov chain is defined as (13): $p(v_t|v_1, \ldots, v_{t-1}) = p(v_t|v_{t-L}, \ldots, v_{t-1})$ where the states are $v_1 \ldots v_t$ and L is the order of the Markov chain. For example in Figure 2.1 the joint probability distribution of the transition $p(v_1, v_2, v_3, v_4,)$ is given

by $p(v_1)p(v_2|v_1)p(v_3|v_2)p(v_4|v_3)$

The hidden Markov model (HMM) is a variant on the Markov Model, where besides the visible and observable variables $v_1, \ldots, v_T$, there is a Markov chain on hidden variables $h_1, \ldots, h_T$. The observable variables are influenced by the hidden layer of variables $h_1, \ldots, h_T$. The change of the observable variables through the hidden layer happens through an emission density $p(v_T|h_T)$.

The process of transmitting information through an HMM can be represented as in figure 2.2. In this figure the hidden layer contains states $h1...h4$ and the visible layer the states $v1...v4$. Information is transmitted through the hidden layer in a sequential fashion. Information is fed to each of the visible states through its corresponding counterpart in the hidden layer.

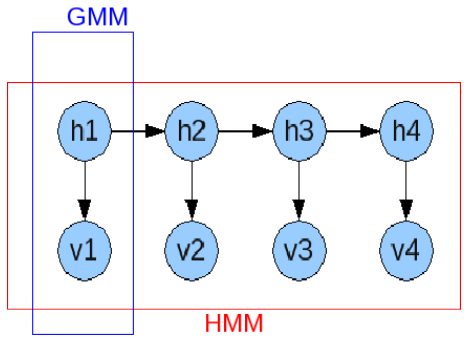

Figure 2.2: Graphical representation of a hidden Markov model. The Gaussian mixture model is the simplest type of HMM.

Formally a hidden Markov model is defined as (13):

$$p(v_{1:T}, h_{1:T}) = p(v_1|h_1)p(h_1)\prod_{2}^{T} p(v_t|h_t)p(h_t|h_{t-1})$$

The simplest type of HMM that utilizes only a single time slice is a Gaussian mixture model (GMM). In a Gaussian mixture model, the formal definition is $p(v_{1:T}, h_{1:T}) = p(v_1|h_1)p(h_1)$. In this paper the results of the HMMs are compared with the simplest case which is a GMM. The accuracies achieved with the GMM classifier are reported in section 6.1.

The distribution $p(h_1)$ is parameterized by a vector a defining the probabilities for each of the states. The probability distribution $p(h_t|h_{t-1})$ is parameterized by a matrix defining the probabilities of a transition occurring between the states. $p(v_t|h_t)$ is a conditional Gaussian distribution: $p(v_t|h_t) = N(v_t|\mu^{h_t}, \sigma^{h_t})$ where v_t is the visible state t. h_t is the hidden state t. N is the normal distribution, μ the mean and σ the covariance matrix.

A first order hidden Markov model can also be defined as a tuple M = (Q, $\sum$, a, t, e):

- Q is a finite set of states: $Q = \{1...n\}$.

- Finite alphabet $\sum = \gamma_1 ... \gamma_m$ defining the output possibilities.
- A vector a of size n defining the probability distribution of starting the process in state i.
- Transition matrix t of size $n \times n$.
- Emission density matrix e of size $n \times n$ indicating the probabilities of emitting output symbol γ_z in state i.

Given a hidden Markov model it is possible to calculate the likelihood of a new data point fitting the parameters of the existing model. The process of filtering in section 2.1 produces this likelihood value as a byproduct.

2.1 HMM training

When learning a hidden Markov model using a set of data $V = v_1, ..., v_n$ of n sequences, the goal is to try and find the transition matrix A, emission density B, and initial vector a most likely to have generated dataset V. It is assumed that each sequence is independently generated and that we know the number of hidden and visible states.

2.1.1 EM algorithm

In order to solve a problem with a hidden Markov model, one must first train this model, a way of doing this is by using a gradient descent algorithm. This, however, is very time consuming. The Expectation Maximization algorithm by Dempster, Laird and Rubin (14) is a faster and more elegant way of training a hidden Markov Model. Expectation Maximization is an iterative algorithm suitable in instances where data is missing or hidden, as is the case in HMMs. It aims to find the model parameters for which the observed data are the most likely. Each iteration of the Expectation-Maximization algorithm consists of two processes: The E- and M step. The expectation, or E-step, deals with estimating the hidden data in the model using the observable variables. In the Maximization, M-step, the likelihood function is maximized assuming that the hidden data are known. This assumption uses the estimate from the E-step to fill the unknown, hidden data.

The algorithm is iterative as it keeps executing the E and M steps over and over until a certain number of iterations is reached or the likelihood has not changed over a pre determined threshold value since the last iteration. The likelihood value is guaranteed to converge to a certain point, but the EM algorithm is not guaranteed to find a global optimum, only a local optimum (15). As such it is not guaranteed to find the best possible solution for the training of the HMM. Proper initializations are crucial in obtaining the optimal trained HMM.

Some of the types of problems that are typically solved using Markov models are: (13)

- Filtering (Inferring the present)
- Prediction (Inferring the future)
- Smoothing (Inferring the past)

Filtering and smoothing are the processes used in training the HMM (by iteratively applying the EM algorithm), and in determining the likelihood values for datapoints fitting the model-specific parameters. Calculating this likelihood value is achieved by the process of filtering, this process plays an important role in both training and testing. During the training phase of an HMM, likelihood values for the datapoints used to train the model are used to determine the convergence of the model towards a (local) optimum. Thus they are an indicator of the quality of the trained model and are a good predictor for its performance on unseen datapoints. During the testing phase, likelihood values for unseen datapoints are calculated using the filtering algorithm.

Filtering

Filtering is about finding the distribution of a hidden state, using all the information obtained up to that point $p(h_T, v_{1:T})$. Filtering in hidden Markov models corresponds to transmitting information from the left to the right as pictured in figure 2.2 and from the hidden states to the visible states.

$$p(h_t|v_{1:t}) = \sum_{h_{t-1}} p(v_t|h_t)p(h_t|h_{t-1})p(h_{t-1}, v_{1:t-1}) \quad (13)$$

In hidden Markov models the forwards propagation of information by means of this algorithm is called $\alpha - recursion$. It is also the only algorithm required to calculate the likelihood of a new data-point fitting the existing hidden Markov Model.

The α recursion is:

$$\alpha(h_t) = p(v_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1})\alpha(h_{t-1})$$

with $\alpha(h_1) = p(h_1, v_1) = p(v_1|h_1)p(h_1)$

Smoothing

Smoothing is another part of the EM algorithm to train the hidden Markov model. $p(h_t, v_{1:T}) = \underbrace{p(h_t, v_{1:t})}_{past}\underbrace{p(v_{t+1:T}|h_t, v_{1:t})}_{future} = \alpha(h_t)\beta(h_t)$ (13)

The term $\alpha(h_t)$ is obtained from the forward α recursion. The term $\beta(h_t)$ may be obtained by a backwards β-recursion:

$$\beta(h_{t-1}) = \sum_{h_t} p(v_t|h_t)p(h_t|h_{t-1})\beta(h_t).$$

2.1.2 Computing the likelihood

To calculate the likelihood of a single sample fitting the trained parameters of the trained HMM we use the same method that was used during the training phase, though one does not actually want to alter the model its parameters similarly to what happens during the process of training the model. A single iteration of the filtering algorithm is sufficient to determine the likelihood value of that sample being alike the samples the model has been trained with. Since the model is expected to have reached a local optimum after the training phase, the iteration of the $\alpha-recursion$ should yield a high likelihood value for a sample that belongs to the same class as the class the model was trained with. That is assuming that the new sample is not an outlier or has other uncharacteristic properties that do not agree with the samples used in training for that HMM. Conversely, calculating the likelihood value of a sample belonging to a different class is expected to yield a low likelihood value.

Now consider that there exists an HMM for every specific class, this enables us to compare the likelihoods for each of these classes being a good match for the new unseen sample. And thus make a decision on what class is most likely to belong to the unseen data. The likelihood of an observed sequence $(v_{1:T})$ is given by: $p(v_{1:T}) = \sum_{h_T} p(h_T, v_{1:T})$ (13) where $p(h_t, v_{1:t})$ is the forward pass (α-recursion) of transmitting information as used in the problem of filtering in section 2.1:

Experimental setup

Eight individuals participated in the experiment, seven male, and one female. Out of these 8 volunteers, only a single one was experienced with BCI control while the others were completely naïve users of BCI systems. All participants had normal or corrected to normal vision.

The experiment was set up to find if shifts of attention in a number of different spatial directions could produce alpha modulation patterns, and whether or not these patterns could be distinguished from one another. If there are clear distinguishable patterns for attention shifts in different spatial locations this would enable a classifier to detect where a subject focused his or her attention. An offline experiment was conducted where the eight subjects were given the task to shift their covert spatial attention to one out of six different locations, while still fixating on the center. This fixating on the center location was meant to ensure that eye gaze would not affect the EEG signal, and would not produce artifacts in the EEG signal. And most importantly, the experiment aims at analyzing the EEG pattern produced by *covert* directed attention (1).

Electroencephalography (EEG) is a non-invasive BCI method. It is a technique to register and record electrical signals generated inside the brain. A number of electrodes are physically placed on the scalp (outside) of the skull. Invasive surgery is not required as it can be placed on the top layer of the skin. Inside the brain are billions of neurons that generate small electrical impulses, when these are released synchronously they can be detected. One way of detecting this activity is by using EEG recording tools.

The experimental design is shown in figure 3.1 and figure 3.2. It was defined as follows:

- The participant fixates on the center of the screen for 1000 miliseconds receiving no stimuli.
- A cue is placed in the center of the screen indicating where to shift (covert) attention to. This phase lasts 200 miliseconds.
- The covert attention phase lasts anywhere from 500 to 2000 miliseconds.
- A target cue (either an **'x'** or a **'+'**) is displayed for 200 miliseconds, usually (in 80% of the trials) in the same location as the participant has focused on, and 20% of the time in a random

other location. The participant has to press one of two buttons depending on whether the **'x'** or '+' was displayed.

- A **'*'** masker was placed over the target cue.

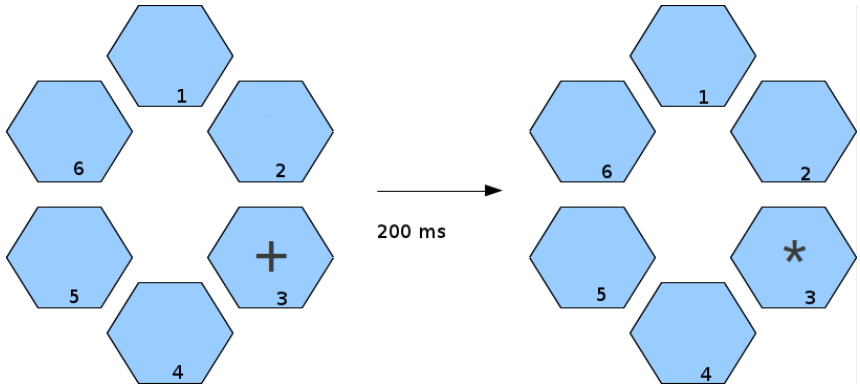

Figure 3.1: A typical experiment epoch, showing the target and masker cue. These appear after the phase of covert directed attention. In the experimental setup, there are six possible locations where the visual stimulus can occur. The original classes are numbered, classes 3 and 5 are the only classes used in the HMM classifiers.

Only the EEG data of the trials that had recorded the full 2000 miliseconds of directed attention from its participants were included in the dataset. Of these 2000 miliseconds of recorded EEG data, the first 500 miliseconds were removed to prevent post-cue activity (1, 16) from leaking into the baseline.

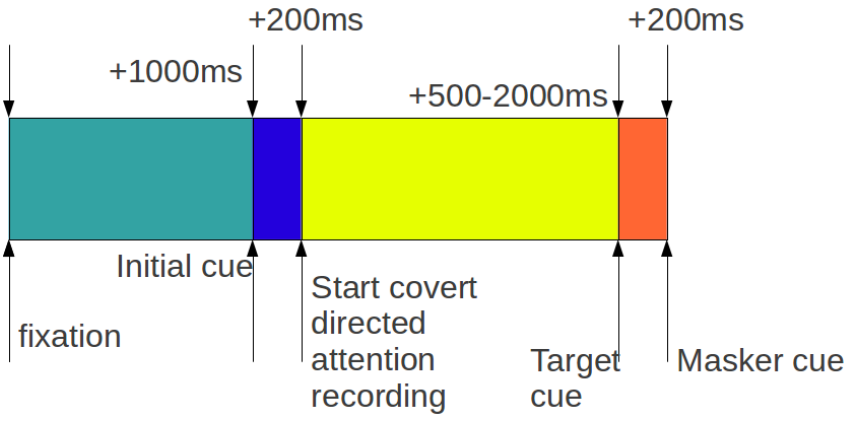

Figure 3.2: Time line of the BCI experiment on recording covert directed attention.

The uncertainty of where the target cue would eventually occur increased the task difficulty, and together with the first task of pressing one of two buttons, it required constant attentiveness from the test subjects. The average percentage of correct button presses was 86.62% ± 8.46%

SEM (1).

In earlier studies on imagined movement in BCI, it has been shown that imagined hand movements produce strong desynchronizations (and in some cases a synchronizations)in μ and β rhythms in the motor- and sensory cortex. An imagined movement of the left hand produces activity in the right hemisphere of the brain, specifically in the motor cortex of the brain its right hemisphere. Conversely, imagined movement in the right hand produces activity in the left hemisphere in the brain (17). In these studies on imagined movement, the signals produced are very well suited to classification algorithms. And can eventually be used in real-time applications. A study by Obermaier *et al.* (6) has shown that hidden Markov models are a feasible candidate as a classification algorithm to use on BCI data. They conclude that the HMM is a better classifier than the LDA (Linear Discriminant Analysis) (18) method.

The fact that a HMM has been used before in a BCI setting using other sets of data, is a good indicator that this particular method may also be useful in this particular case using EEG data on covert directed attention.

Data

The original EEG was recorded using a Brain Products actiCAP using 64 channels, These 64 channels were placed according to the international 10-10 system, the channels used are: *Fp2, AF3,4, Fz, F1-10, FCz, FC1-6, T7,8, Cz, C1-6, TP7,8, CPz, CP1-6, Pz, P1-10, POz, PO3,4,7-10, Oz,1,2 and Iz,1,2.* The last channel (named EOGVu) was placed below the right eye. Out of these 64 channels, 4 were omitted from the eventual dataset leaving 60 channels. The channels that are completely omitted in the dataset are: Fp2, F9-10, EOGVu.

The EOG (electrooculograph) channel is used to be able to classify directed attention based on EOG data. An electrooculograph is used to monitor eye movement. In this experiment, eye movement was completely undesirable. And EOG recording in this experiment makes it possible to prune data that is likely to be polluted by artifacts or muscle-control specific brain activity. Overt attention BCI systems have been shown to perform significantly better than covert attention BCI systems. Visual spellers like the P300 based spellers have a much higher bit rate and are far more reliable (19). The data was processed to remove artifacts and invalid trials.

The resulting dataset is not balanced. On average the dataset contains 48.25 trials belonging to the class 'left', and 53 trials belonging to class 'right'. Because of the method (section 5) used to train and classify the data, this imbalance does not introduce a bias in the classifier. The pair of hidden Markov Models are each trained using data of a different class. The individual hidden Markov models are not trained to separate between two classes, all they can do is to calculate the likelihood of a data point fitting their model. But together they can be used to distinguish between the two classes. Section 5 details the classification method further.

	1	2	3	4	5	6	7	8
Class 1 (Right)	46	46	53	51	51	37	43	51
Class 2 (Left)	56	59	50	58	54	40	57	50

Figure 4.1: Class distribution in the dataset. The number of trials for the eight participants and their class. The experiment trial classes were assigned randomly resulting in an unbalanced dataset.

BCI Data classification

To classify the recorded BCI data, we use a pair of hidden Markov models, each of these trained on a particular direction of focused covert attention (left or right). Using a leave-one-out validation setting, models are trained and the data is classified. The filtering (section 2.1) algorithm enables us to calculate the likelihood of a data point fitting to the existing trained models. Since two models are trained, each of them trained with data belonging to a single class, the two hidden Markov models should have learned the optimal parameters to let their class data fit. Subsequently fitting a new unseen data point to the trained models using the filtering algorithm will return different values for both trained models given that they have different parameters. In the leave-one-out classification method, the model that has the highest likelihood of fitting the unseen data point is chosen and classification is considered successful when the model trained with data belonging to the same class as the unseen data point is selected. Otherwise the wrong model was chosen. Eventually the overall percentage of correctly classified instances is reported.

The BCI data classification was performed several times with different types of hidden Markov models and different parameter initializations. Initially a simple HMM where the data was averaged over time was trained and used to calculate the performance of the classification method. This very simple type of hidden Markov model is essentially a Gaussian Mixture Model. In this baseline comparison (results in section 6.1), the dataset is manipulated to only store data over a single time slice.

The HMM software implementation is Kevin Murphy's HMM Toolbox for Matlab. This toolbox is used to train the hidden Markov models, the classification of the unseen data is done by applying the very same methods in Kevin Murphy's toolbox (20). Calculating the accuracies of the eventual classification results is done in a straightforward fashion. The code for the training and classification phases is included in section 9. The statistical validation technique to ensure the model generalizes to independent data and not only the training data is a leave-one-out cross validation setting. This is computationally expensive because every of the individual data points is used once as validation data, this means the training phase must be repeated for every single datapoint (an epoch in the experiment) where all the samples except for the single validation datapoint are used to train that HMM. Since there is no efficient way to remove or add a single sample to the trained model in a HMM, the training phase is required to start all over at the very first training iteration. This results in a very computationally intensive training and classification method.

5.1 Data preprocessing

The data was pruned to only include information that was suspected or assumed to be interesting and relevant to this experiment. The pre process actions that were applied to the data are as follows:

- All datapoints with a design other than design 3 or design 5 were removed.
- Out of the 60 channels recorded, only the channels associated with an occipital or parietal occipital location on the scalp were preserved. In total 50 out of the 60 channels were removed. (1)
- Remove all datapoints that have recorded frequencies other than 8,9,10,11 or 12 hz.
- Average over the remaining five frequency bands.

Designs 3 and 5 correspond to the lower right and lower left cue locations respectively (figure 3.1). The reason to only look at these two locations is because we only try to distinguish and classify according to two different locations: left and right. Out of the other cue locations only 2 and 6 are supposed to contain information about the directed covert attention on the horizontal axis. However, because we only want to distinguish between two classes, just two cue locations are used. Merging the trials for design 2 and 3 together, and the trials for design 5 and 6 would mean that the training set becomes larger. At the same time, this merging would result in more separated data per class, thus making it harder to classify trials correctly. For the sake of clarity, designs 3 and 5 are referred to in the rest of this paper as class 'right' and class 'left'.

In the next pre process step 50 out of the 60 channels are removed, this results in a great reduction in complexity and a lot of irrelevant data is removed. The 60 channels measure EEG on 60 different locations on the scalp. Figure 5.1 shows the positions of the electrodes on the scalp. Each of these electrodes is represented in the dataset by a channel. Since the type of information we are looking for is based on covert directed attention not nearly all of these positions on the scalp are going to measure brain processes related to the directed covert attention that we are interested in. Several studies have shown that the occipital and parietal-occipital lobes in the brain are concerned with spatial awareness and directed attention (21). The channels located over these lobes are going to be able to get the strongest readings of alpha activity in the underlying parts of the brain, this is also illustrated clearly in figure 6 of Treder *et al.* (1). Directed covert attention is strongly associated with changes in alpha modulation. For that reason only the frequencies in the upper and lower alpha band (8-12 hz) are considered in this experiment.

These pre process steps are instrumental in preparing the data to be used in the classification system utilizing HMMs. Dimensionality and complexity is reduced so that training phase takes less time and data irrelevant to the task is removed as much as possible.

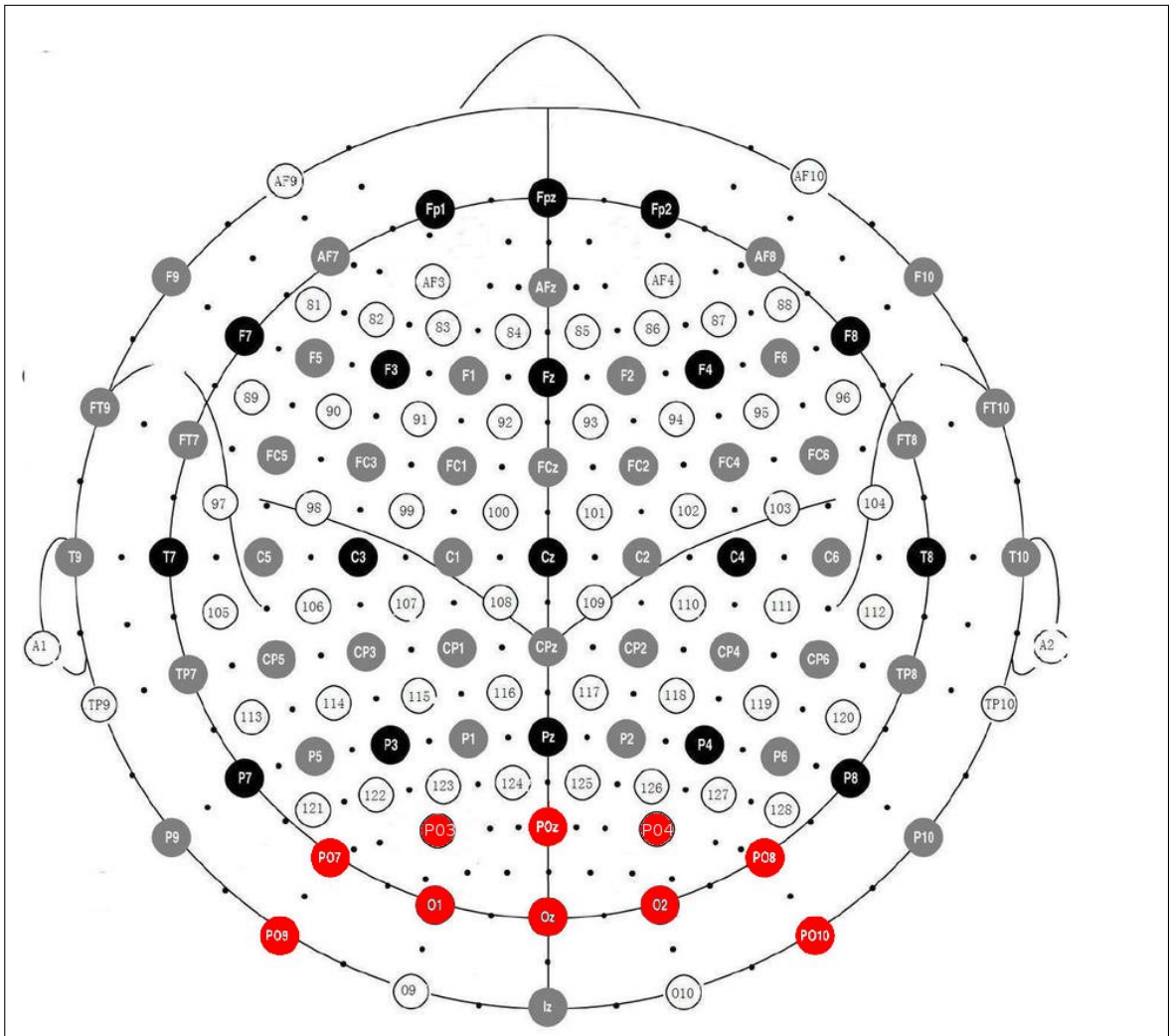

Figure 5.1: 128 EEG channels. The ten channels used are colored red and located over the occipital and occipital-parietal lobes. The most important (1) channels are O1 and O2, located over the centers of the left and right occipital lobes respectively.

Results

The goal of this paper is to give an overview of the suitability of hidden Markov models in solving BCI problems, and to determine if the parameter of time as recorded in the BCI data contains relevant information in distinguishing between separate classes. In order to determine this, several batches of tests to classify the data were done. These batches had different parameters and settings. Note that in this explorative study we restrict ourselves to a qualitiative analysis of the behaviour of the HMM in the context of BCI. Further significance testing is deferred to future work.

In the sections detailing the classification results we differ between the data where ten channels are selected, and those where only two channels are selected. The ten channels included in the final dataset used to classify and train the hidden Markov models are all located over the occipital and parietal-occipital lobes. Their labels are: *POz, PO3,4 PO7,8, PO9,10, Oz, O1, O2.*

Of these ten, *O1* and *O2* are the most important with respect to the signal strength of the underlying alpha activity. They receive the clearest signal from the occipital lobes and are directly located over the left and right occipital lobes, this is shown in the 2011 paper by Treder *et al.* (1) in figure 6. In the tests where the training and test data only contained two channels, *O1* and *O2* were selected, by looking at the grand average (signal strength measured by alpha amplitude) and choosing the first two channels with the highest values.
The other 50 channels not considered as valuable for the purpose of classifying covert directed attention are located over lobes other than the occipital or parietal lobes. Other brain processes interfere with the signals produced involved with covert attention, and due to the nature of EEG recording, the other channels will not obtain the signals generated by the occipital and parietal lobes as clearly as those in closer proximity to the source.

The classification results are separated into five different sets with different settings.

- Baseline comparison. 6.1
- Ten channels, 1-2-4 time windows. 6.2
- Ten channels, 1-2-4 time windows, diagonal covariance matrix. 6.3
- Two channels, 1-2-4 time windows. 6.4

- Two channels, 1-2-4 time windows, diagonal covariance matrix. 6.5

In these five sections containing test results, the difference in settings is in the training data, or the covariance matrix used in training the HMM. The training data differs (ten channels or two channels) to detect if the additional eight channels improve performance, and add relevant information. In training data where only two channels are present, training is faster and less susceptible to overfitting or noisy data.

The difference in the number of time windows was due to the averaging over the available time points in the dataset. While there are a total of fourteen time points available for each and every channel, not all fourteen are directly used. Instead, time windows are created based on the fourteen time points. The dataset contained the values of all the channels from 0.5 seconds after the target cue up until 1.8 seconds after the target cue. This period allows for the test subject to register the location of the cue and mentally focus his or her attention to that location. The delay of 500 milliseconds before recording also negates interference from the brain activity associated with event-related potentials (ERP) caused by the appearance of the initial and target cues. An ERP occurs within the first 700 milliseconds after the appearance of a stimulus cue, with different types of stimuli causing varying ERP signals of differing lengths and/or intensity. The visual type of stimulus used in this particular experiment typically causes an ERP 300 milliseconds after the appearance of the stimulus (16).

- The single time window is simply a single value; the average of all fourteen time points.
- In the tests where the data contained two time windows, the first time window was the average of time points 1-7, and the second was the average of time points 8-14.
- In tests where the data contained four time windows, the first time window was the average value of time points 1-4, the second window was the average of time points 5-8, the third was the average of time points 9-11, and the fourth time window was the average of time points 12-14.

By averaging over multiple datapoints, it is possible to negate noise values to a certain extent (22). But the added value of multiple time points or multiple time windows over a single time slice, is that you end up with more data and more information, this enables the trained model to be more specific.

High-dimensional data, in combination with few training samples presents us with a sparse set of data to work with. Estimates of the covariance matrix are increasingly likely to become singular with sparser data. Where one option is to improve the estimations of the covariance matrix, another option is to constrain the covariance matrix to be spherical or diagonal.

With the additional time windows, complexity of the data increased but the number of training samples remained constant, creating sparse data. Forcing the covariance matrix to be diagonal solved the problem of it becoming strongly singular but did not improve overall classification performance compared to the earlier method.

Curse of dimensionality

The reason to run the training of the classifiers on the data using a slightly different method - by using a covariance matrix that was constrained to be diagonal - was because there appeared to be a numerical instability in the eventual log-likelihoods calculated by the trained HMMs in instances where the combination of number of hidden states, channels and time windows caused a too high dimensionality of the data (23). The generative HMM is trained to find parameters optimal to the fitted data. In the experiments with more than a single time segment, as described in sections 6.3 and 6.5, the implementation of the HMM training and classification code (section 9) concatenates the features of the time segments into a single feature vector. Obviously, this feature vector increases in size with the number of time segments. The number of features is also dependant on the number of channels included in the dataset. Whereas the number of hidden states the model has available in its training process determines the size of the covariance matrix.

Heung-Il Suk and Seong-Whan Lee (23) suggest a number of different solutions in order to deal with the curse of dimensionality in HMM classifiers.

- Use a different type of classifier such as a Support Vector Machine (SVM)
- Apply PCA techniques to reduce the dimensionality of the data
- Use a multi-layer hidden Markov model as described in (23)

Instead the dimensionality was reduced by changing the number of channels in the dataset from ten to eight, and as described above, by constraining the covariance matrix. The former was reasonably effective, it allowed the use of more hidden states in training HMMs. While the latter did not produce HMMs better suited to the task of classifying the data.

6.1 Baseline performance

Determining a baseline performance for the classifier that distinguishes only between left and right is the first step in this research. The baseline classifier is based on the pre-processed data as described in section 5.1, and using that data in a leave-one-out setting to build and test classifiers. An additional step in this preprocessing procedure was to average over all fourteen time points. The time points recorded were all between 0.5 and 1.8 seconds after the target cue, and were all evenly spaced 100 milliseconds after one another. By averaging over the dimension of time, the problem is greatly simplified although a great deal of information is lost. Potentially this lost information

could have helped to distinguish and separate the two different classes.

Two separate hidden Markov models are built, one trained on the trials with class 1 (for covert attention focused to the right) and the other trained on the trials with class 2 (for covert attention focused on the left). These hidden Markov models, essentially one for each class, are then both fitted on a new trial that has not been used to train the initial models. The class belonging to the model that has the highest likelihood of fitting is chosen for that particular trial.

For each of the eight test subjects, the recorded data was classified using hidden Markov models with one through sixteen hidden states. It turns out that the number of hidden states in the hidden Markov model has a strong influence on the degree in which the model can sufficiently learn the data to function as a classifier. At the same time there is a certain maximum number of hidden states for which a model can be best trained to fit the available data. This particular maximum number of hidden states differs for each test subject. But there is a definite trend that suggests that seven or eight hidden states yields the best classification performance for most test subjects.

Figure 6.1 displays the classification results for all eight test subjects. In the plot the average classification performance for all eight test subjects is plotted against the number of hidden states the hidden Markov model was trained with. The error plot indicates the standard error of the mean of the test results. Some test subjects produced far better results than others. These averaged results do, however, show a clear trend in classification performance and the number of hidden states used in training.

The best performance is obtained using seven or eight hidden states, classification performance gradually drops with more or fewer hidden states.

The maximum classification performance plotted against the eight **test subjects** is displayed in section 6.6 together with the accuracies obtained with the other classification methods and settings.The figures in that section give an idea of the quality of the EEG signals produced by the test subjects, as those participants in the experiment that produce consistent EEG signals for the same task they were meant to perform (focus in a certain direction) will score higher.

The *average maximum classification performance over all eight test subjects over all trained HMM* for the baseline classifier was 79.37%. In an online BCI application of this software, it would be impossible to compare the different trained HMMs, seeing how one could not know what direction the test subject had covertly focused his or her attention to. As such it would be impossible to train multiple models, compare them against each other, and pick the very best performing one. However, if an extensive training session precedes the eventual task, the BCI system can be assigned to use the subject-specific settings to optimize the process of recognizing the EEG signals.

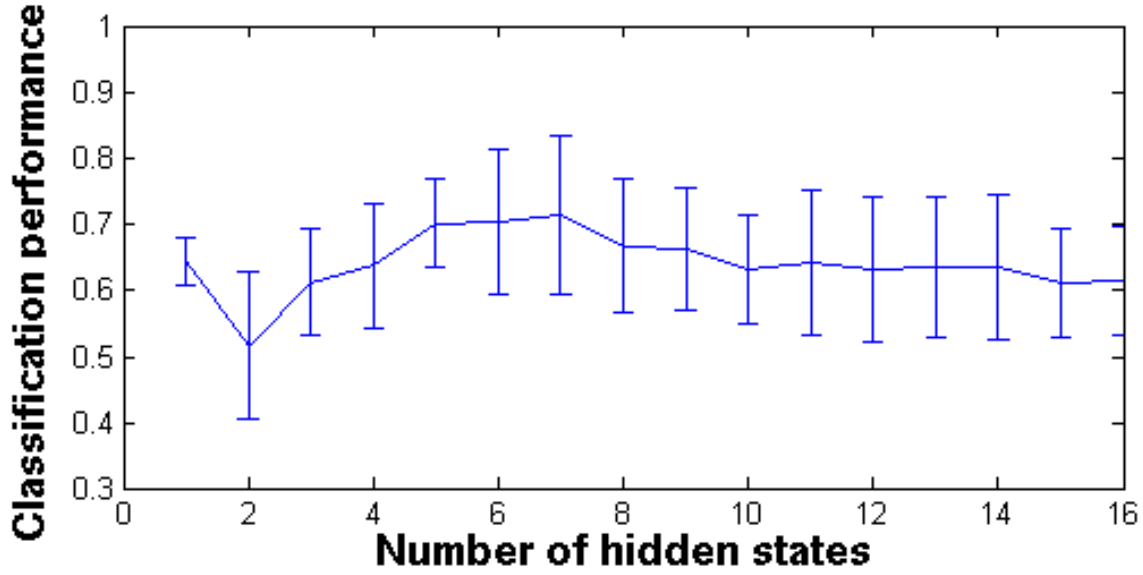

Figure 6.1: Classification results using ten channels and one time window. Average classification performance (X-axis), together with the standard error of the mean, is plotted against the number of hidden states used to train the model (Y-axis).

Nevertheless, the average classification performance over all eight test subjects when the hidden Markov model was trained with 7 hidden states was 71.29%. This is a very slight improvement over the classification performance obtained by Treder *et al.* (1). In that study the average percentage of correctly classified trials for the same two classes (design 3 and design 5 in the original dataset) is 69.569%.

6.2 Ten channels

When using multiple time windows the number of features in the hidden Markov model increases exponentially. The model is able to use more information to fit the data more accurately at the cost of speed and the risk of over fitting. An added advantage to using a single time window is that it is less sensitive to noise and outliers. Whereas a model that uses a single time window has averaged over all the available time points, and as a result can better deal with noisy data, a complex model with more features is vulnerable to noise due to over fitting.

In figure 6.2, classification performance is plotted for models trained with one, two and four time windows. The models with a single time window contained fewer features and was faster to train, while the model with four time windows was more complex and took longer to train.

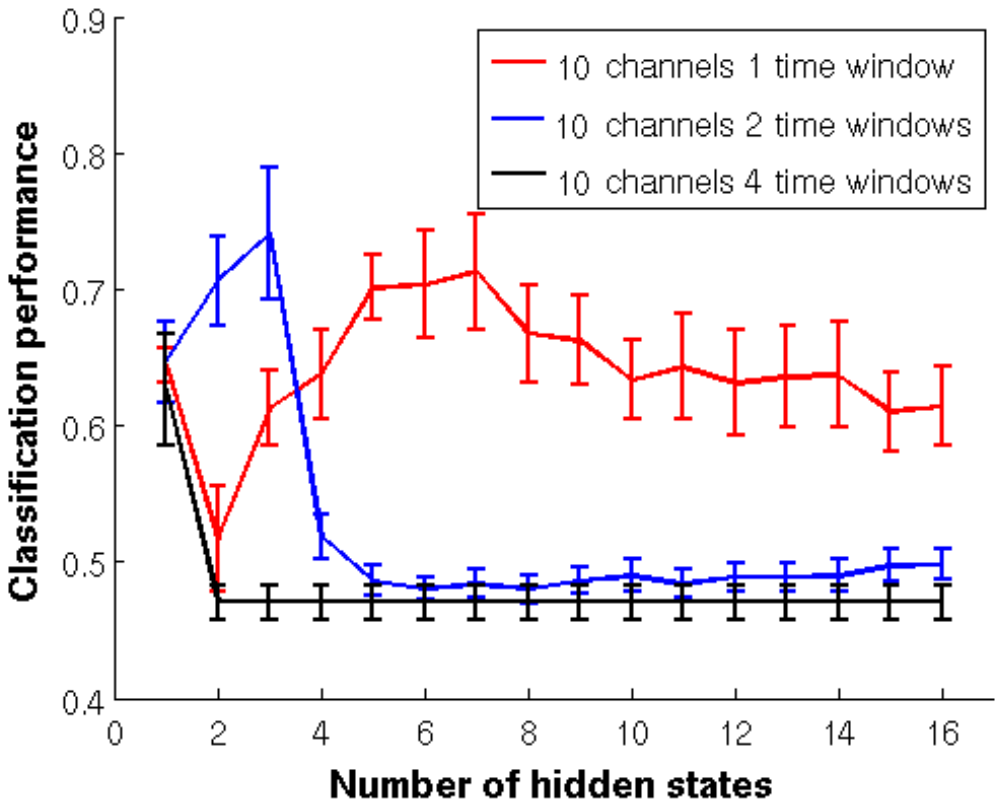

Figure 6.2: Average classification accuracy and standard error of the mean, plotted against the number of hidden states used to train the hidden Markov model with data containing ten channels. Displaying trained model with one, two and four time windows.

6.3 Ten channels - diagonal covariance matrix

Section 6.2 shows that there is a definite problem that occurs when the number of features increases. The model does not seem to be able to cope, and the percentage of correctly classified trials plummets. By forcing the covariance matrix of the hidden Markov model that is being trained to be diagonal, it is less likely to become singular. While this does partially solve the issue, the model is not as well capable at classifying the data as it was with the other method.

In plot 6.3, the model is trained with the additional covariance matrix setting as described above.

6.4 Two channels

In figure 6.2 it is striking how classification performance drops drastically as the model contains more than three hidden states. At this point the model becomes increasingly complex with an exponentially growing number of variables. It becomes computationally intractable to determine

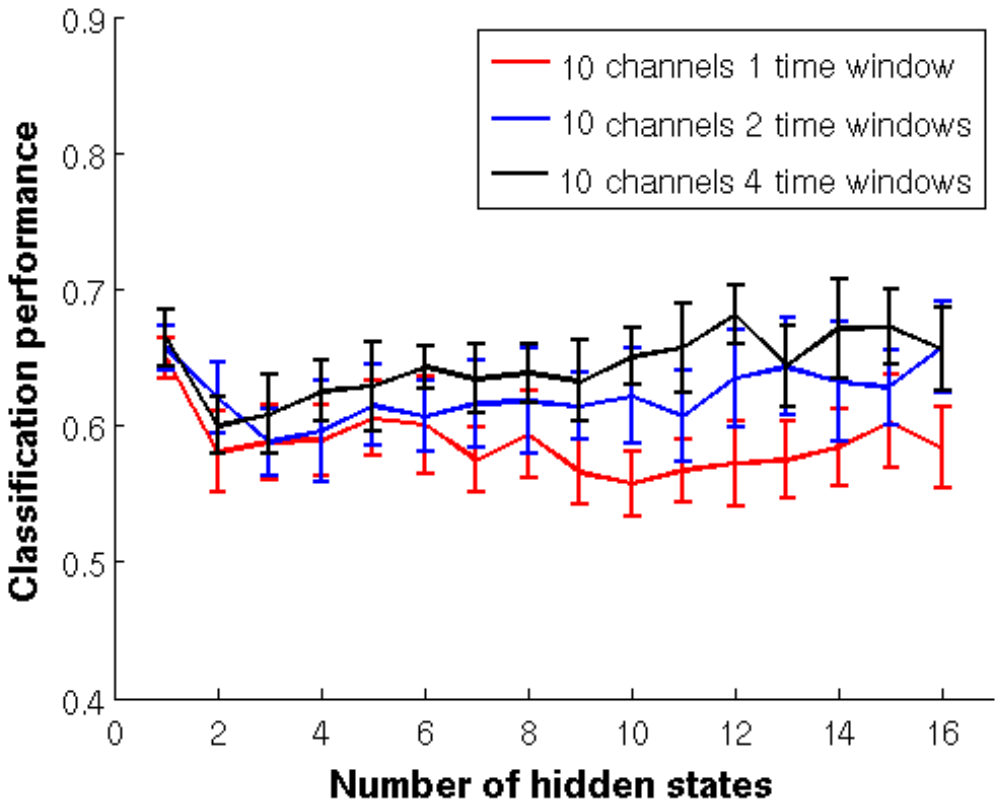

Figure 6.3: Average classification accuracy and standard error of the mean,plotted against the number of hidden states used to train the hidden Markov model with data containing ten channels. Displaying trained model with one, two and four time windows. Diagonal covariance matrix.

the likelihoods of fitting new datapoints. In order to reduce the number of variables, further tests where only two EEG channels (electrodes on the scalp recording data) are included in the training data yield significantly different results. $O1$ and $O2$ are in the very centers of the activated areas of the occipital lobes. Using only these two channels, and a total of four time windows, average *maximum* classification performance for all eight test subject was 82.77%.

Interestingly this is higher than the result obtained in section 6.1 where the models were trained with 10 channels and averaged over a single time window (79.37%). Not only is the maximum average classification performance better, the average classification performance for all models with sixteen hidden states is 76.55%; higher than in the baseline (71.29%).

In plot 6.4, results are plotted for the models trained with only two channels, and one, two or four time windows. Due to the lower number of features, no problems arise as the number of time windows increases or as the number of hidden states increases. On the other hand, classification performance is significantly higher compared to the the baseline 6.1. The strong correlation between

number of time windows and performance is an indicator that the factor of time is possibly of importance and may very well add useful information in this particular pattern recognition problem.

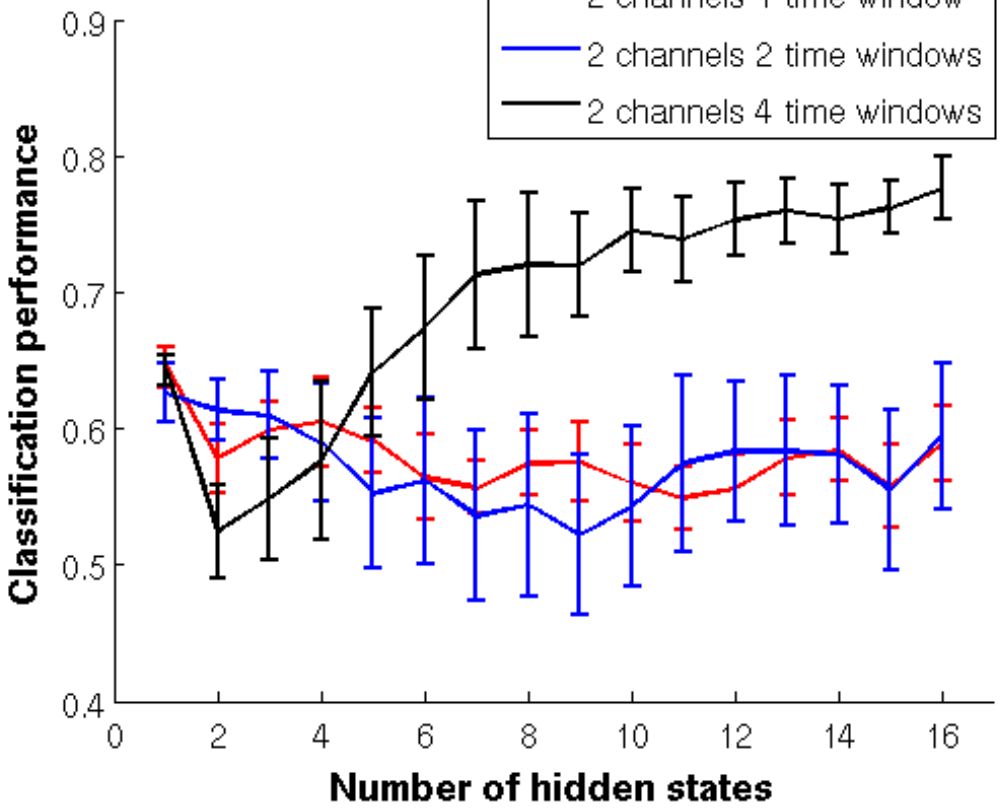

Figure 6.4: Average classification accuracy and standard error of the mean, plotted against the number of hidden states used to train the hidden Markov model with data containing two channels. Displaying trained model with one, two and four time windows.

6.5 Two channels - diagonal covariance matrix

While there is no compelling reason to believe that forcing the covariance matrix to be diagonal is going to have a positive effect on the eventual result, figure 6.5 displays the percentages of correctly classified trials for this set of hidden Markov Models. In his plot the models trained with two channels and a diagonal covariance matrix are shown.

As was the case in section 6.3, the diagonal covariance matrix does not boost the ability of the model to train and learn the data, on the other hand its performance decreases. Since there was no problem in section 6.4 with regards to the complexity or sparsity of the data, this method did not offer an improvement with regards to classification performance as the data became more complex with more time windows or hidden states.

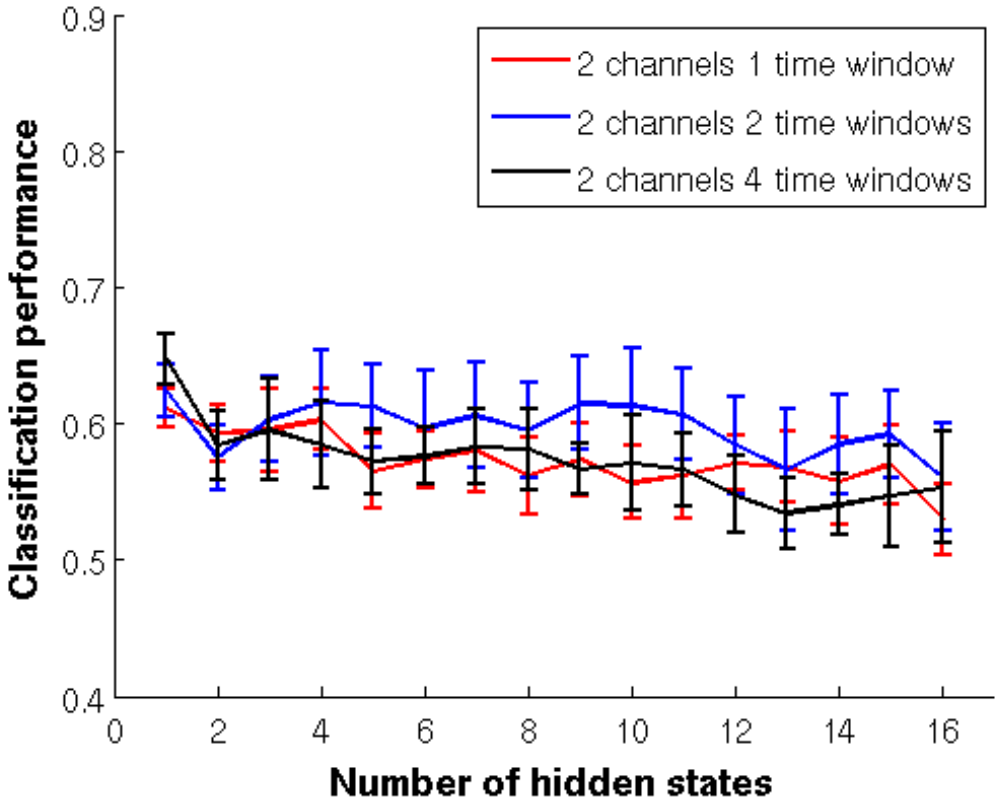

Figure 6.5: Average classification accuracy and standard error of the mean, plotted against the number of hidden states used to train the hidden Markov model with data containing two channels. Displaying trained model with one, two and four time windows. Diagonal covariance matrix.

6.6 Individual subject's results

The participating individuals in the original experiment were not all equally experienced with BCI systems. The EEG signals they produced are very specific to the individuals, BCI systems are still expected to handle multiple different users and maintain a reasonable level of usability. BCI illiteracy is a characteristic recognized in a certain portion of humans, and makes it virtually impossible to create a fully functioning BCI system for these people considered illiterate in this sense. BCI illiteracy is not dependant on the type of system used, but is an inherent property of the patient or subject, it is thought that twenty percent of all people are BCI illiterate (24, 25). The recorded EEG data of the eight participants applied to the HMM classifier software as included in the Appendix yielded very differing results, both for the different parameters used to train the HMM as well as for the individual test subjects.

In sections 6.2 and 6.4 it appears that the highest (average) percentages of correctly classified trials are achieved with HMMs trained with: **10 channels, 1 time slice and 7 hidden states.**

10 channels, 2 time slices and 3 hidden states. And 2 channels, 4 time slices and 16 hidden states. Figures 6.6 and 6.7 display the classification performance plotted against the individuals. In addition to the three HMM classification methods mentioned above, the accuracies achieved in Treder *et al.* (1) are included in the graph.

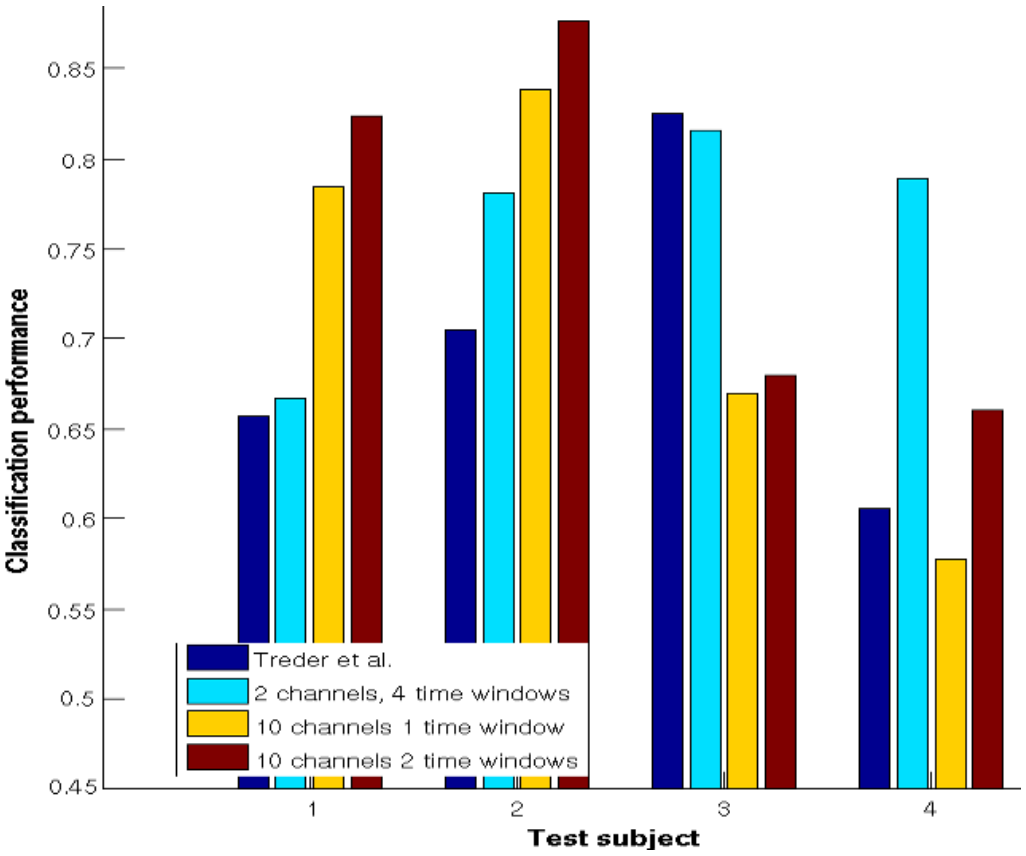

Figure 6.6: Accuracies of four different classification methods applied to the same BCI data for test subjects one through four. Intersubject variability is high with regards to the accuracies achieved with the four methods.

From figures 6.6 and 6.7 it appears that the the different methods produce very different results for the same subjects. Whereas the methods represented by the blue and cyan bars seem to be good choices for subjects 3,4 and 6. Subjects 1 and 2 their recorded EEG signals are easier to classify correctly using the other two methods. Interesting is to see that the results for subject 6 are very disappointing for all methods except for the method that utilizes four time slices, which does surprisingly well with an overall 86.5% accuracy.

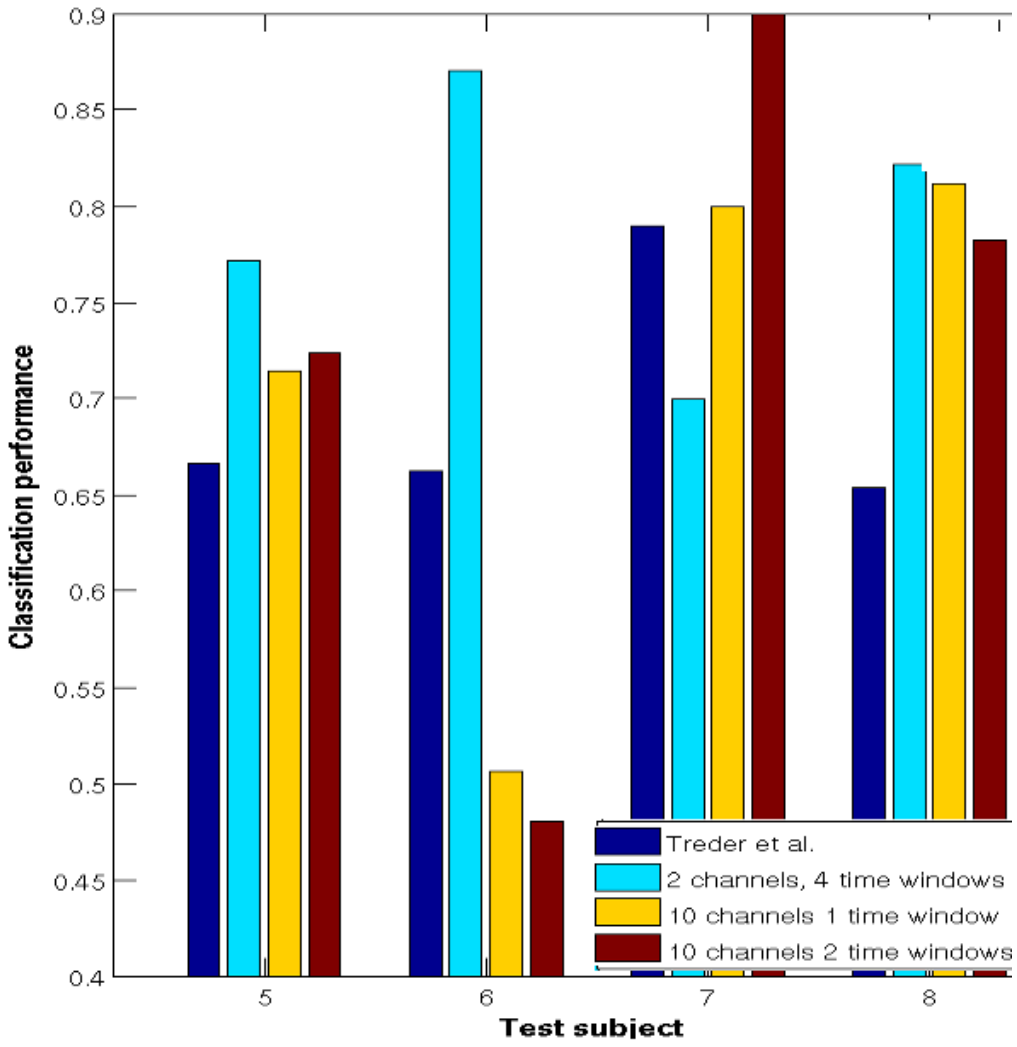

Figure 6.7: Accuracies of four different classification methods applied to the same BCI data for test subjects five through eight. Intersubject variability is high with regards to the accuracies achieved with the four methods.

Discussion

In this paper I have analyzed and compared two different classification techniques on the same BCI data. It was limited to only two directions, whereas the available dataset contained information on six different spatial directions. I only looked at two of those, which should be strongly distinctive because they are laterally opposite. The other pairs of classes are likely to be harder to distinguish from one another, due to the closer proximity of some the other pairs of classes in the dataset, or the fact that other pairs are vertically or diagonally spaced from each other. Treder *et al* (1) showed in figure 3 that the best classification performances of pairs of classes were obtained with contralateral pairs (in seven out of eight subjects) and that pairs of classes on the same vertical axis (design 1 and 4, design 2 and 3, design 5 and 6 as determined in figure 3.1) are harder to separate in a classifier. Mean accuracy for the best pair of directions reported in Treder *et al.* (1) was $74.6\% \pm 2.3\%$. However, the best pair of directions differed between subjects (four separate pairs of classes). So a comparison against the accuracy achieved using the hidden Markov models should be against the mean accuracy for the pair of classes three and five (bottom right and bottom left): 69.569%. In a future study it would be interesting to look at all fifteen different pairs of classes for all subjects. As was found in this paper, performance varied greatly between subjects, confirming the findings in the paper by Treder *et al.* Similar tests using the HMM classifier on all different pairs of directions should be able to further confirm or deny this observation. At the same time it would be possible to have a reliable comparison between the mean accuracies of the two methods when looking at the best pair of directions.

The strong variation between the pairs of directions is an interesting and important aspect to consider when continuing to research this subject of covert directed attention, as it seems that the directions that can be differentiated for people are different for each individual. In a real-time application of a BCI system based on covert directed attention, it is essential to understand more about the reasons for this intersubject variability that seem to result in this variation of pairwise classification. The classification methods compared in this study are hidden Markov models with several different parameters and settings, and the method applied in Treder *et al.*, which is a logistic regression algorithm.

The eye-catching observations that can be drawn from section 6.2 is that there is a certain aspect to the HMM structure or implementation of the HMM software that causes an instability

when the number of hidden states and features grows. This results in a sharp drop-off in accuracy, flatlining around the chance accuracy. Reducing the dimensionality of the data by limiting the number of channels used even further avoids this problem. High-dimensional data in combination with a small sample size can result in strongly singular covariance matrices. Constraining the covariance matrix as described in sections 6.3 and 6.5 is another way to get around this problem of a numerical instability causing this dramatic performance drop. This however results in very poorly trained Markov models. Constraining a covariance matrix to be diagonal assumes that the different dimensions of the data are independent from one another (26). The accuracies obtained from the training and testing on the HMMs trained with the diagonal covariance matrices suggest that this was not the case.

Intuïtively, the channels are not likely to be independent from each other. The brain is an intrinsic network of neurons that is completely interconnected. The recording method of the EEG data is another factor in the apparent dependence between the several channels. A single EEG recording channel can easily pick up activity from a far away, remote part of the brain. This activity may interfere with the recorded signal obtained from the location directly under the electrode. Since the channels used are all located over the parietal and occipital lobes of the brain they are spaced closely together and smearing of the EEG signal pollutes the recordings. This smearing effect is another reason that the dimensions in the data (the channels) are not independent from each other, and is very likely to be the reason for the poor accuracies in classification obtained in sections 6.3 and 6.5.

Further research into the reason of the significant performance drop off for the tests with high-dimensional data and possible solutions would give further insight into the true abilities of the HMM classifier on this particular type of complex data.

Because this paper reports on an observational study and does not prove the statistical significance of the results, a deeper and broader study on the use of HMMs on this BCI data should present results comparable to those in Treder *et al.* (1) and provide a statistical analysis to determine the significant differences. Though it does seem that there is a substantial difference in the accuracies reported for the different methods, suggesting that the HMM classifier for BCI data is an interesting topic for further research.

Conclusion

The research questions to answer are:

- What is the classication performance of directed covert attention in BCIs using the average alpha power in a fixed time window and how does it compare to the method used in Treder *et al.*? (1)
- Does the inclusion of hidden Markov models contribute to the classication performance of directed covert attention in BCIs?
- **What is the classication performance of directed covert attention in BCIs using a hidden Markov model classifier with the data split into multiple time segments?**

Research questions 1 and 3 can be answered by looking at figure 8. The classification performance of directed covert attention in BCIs using the average alpha power in a fixed time window amounts to $71.29\% \pm 4.24\%$. Compared to the logarithmic regression algorithm which reaches an accuracy of $69.57\% \pm 2.64\%$ on the same data, the mean accuracy is only very slightly higher while the intersubject difference is higher. Hidden Markov models using this same BCI data but split into several time segments can achieve better accuracies as evidenced in sections 6.2 and 6.4. Split into two time windows it was possible to reach $74.08\% \pm 4.83\%$, split into four time windows it was possible to build a HMM classifier capable of classifying $77.69\% \pm 2.33\%$ correctly.

The answer to the second research question is harder, while the table below does seem to indicate a certain improvement over the algorithm of logarithmic regression for this particular set of data, section 6 outlines a severe drawback of a HMM in combination with highly dimensional data. BCI data typically consists of many different channels recording data over a certain time window. At the same time it is expensive to record BCI data due to the requirement of expensive equipment, and time required of the test subjects. So BCI data will always be relatively high dimensional and sparse. The results do suggest that an HMM system can positively contribute to a classifier of BCI data, but is still limited by the type and amount of data.

There is no statistical analysis of the results and percentages reported, for that reason I cannot make any claims about the statistical significance of the difference between the classification methods.

Method	Channels	Cov. matrix	Hidden states	Time windows	Accuracy
Log. Regression					69.57% ±2.64%
HMM	10		7	1	71.29% ±4.24%
HMM	10		3	2	74.08% ±4.83%
HMM	10		1	4	62.62% ±4.14%
HMM	10	diagonal	1	1	64.92% ±1.46%
HMM	10	diagonal	1	2	65.64% ±1.63%
HMM	10	diagonal	12	4	66.41% ±2.12%
HMM	2		1	1	64.49% ±1.15%
HMM	2		1	2	62.59% ±2.14%
HMM	2		16	4	77.69% ±2.33%
HMM	2	diagonal	1	1	61.08% ±1.44%
HMM	2	diagonal	1	2	62.36% ±1.91%
HMM	2	diagonal	1	4	64.75% ±1.86%

Figure 8.1: Classification accuracies of the different methods compared against each other. The parameters for *Hidden states* and *Time windows* were chosen because they resulted in the best performing HMM for this particular BCI data and type of covariance matrix. The accuracies are reported together with the standard error of the mean to indicate the level of variation between the experiment participants.

Appendix - Code

```
clear;
rand('seed', 3);
for u=1:8
    % datasets freq1..8 contain the BCI data. Each file contains all the trials for a test subject.
    load(strcat('freq', num2str(u)));

    % Q = nr of hidden states
    minQ = 1;
    maxQ = 16;

    %Mixed gaussians
    M = 1;

% 2 channels, 1 time window
XfRight = squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),:),3));
XfLeft = squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),:),3));
XfRight = mean(XfRight, 3);
XfLeft = mean(XfLeft, 3);

% 2 channels, 2 time windows
%XfRight = horzcat(mean(squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),1:7),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),8:14),3)), 3));
%XfLeft = horzcat(mean(squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),1:7),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),8:14),3)), 3));

% 2 channels, 4 time windows
%XfRight = horzcat(mean(squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),1:4),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),5:8),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),9:11),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[58 60],(freq.freq >= 8) & (freq.freq <= 14),12:14),3)), 3));
%XfLeft = horzcat(mean(squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),1:4),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),5:8),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),9:11),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[58 60],(freq.freq >= 8) & (freq.freq <= 14),12:14),3)), 3));

% 10 channels, 1 time window
%XfRight = squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),:),3));
%XfLeft = squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),:),3));
%XfRight = mean(XfRight, 3);
%XfLeft = mean(XfLeft, 3);

% 10 channels, 2 time windows
%XfRight = horzcat(mean(squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),1:7),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),8:14),3)), 3));
%XfLeft = horzcat(mean(squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),1:7),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),8:14),3)), 3));

% 10 channels, 4 time windows
%XfRight = horzcat(mean(squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),1:4),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),5:8),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),9:11),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==3,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),12:14),3)), 3));
%XfLeft = horzcat(mean(squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),1:4),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),5:8),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),9:11),3)), 3),...
%mean(squeeze(mean(freq.powspctrm(design==5,[15 23 53:60],(freq.freq >= 8) & (freq.freq <= 14),12:14),3)), 3));

    % Permute data so that its in the format channels*trials
    R = permute(XfRight,[2 1]);
    L = permute(XfLeft,[2 1]);

    szfr = size(R);
    szfl = size(L);

    % Error classification using Leave One Out
    %% Loop through all trials
    %% Train R and L classifiers leaving one trial out, then test that trial
    for Q=minQ:maxQ
```

```
        best = 0;
        correct = 0;
        for x=1:(szfl(2)+szfr(2))
            if (x <= szfl(2))
                temp = L(:,1);
                L(:,1) = [];
            else
                temp = R(:,1);
                R(:,1) = [];
            end

            % build Right and Left initialisation matrices
            muR0 = repmat(mean(R,2), [1 Q M]);
            SigmaR0 = repmat(cov(R'), [1 1 Q M]);
            muL0 = repmat(mean(L,2), [1 Q M]);
            SigmaL0 = repmat(cov(L'), [1 1 Q M]);
            bestR.LL(1) = -inf;
            bestL.LL(1) = -inf;
            bestR.LL(end) = -inf;
            bestL.LL(end) = -inf;

            for l=1:10
            % build Right and Left initialisation matrices
            priorR0 = normalise(rand(Q,1));
            transmatR0 = mk_stochastic(rand(Q,Q));
            mixmatR0 = rand(Q,M); %ones(Q,M);

            priorL0 = normalise(rand(Q,1));
            transmatL0 = mk_stochastic(rand(Q,Q));
            mixmatL0 = rand(Q,M); %ones(Q,M);

            %train right Hidden Markov Model.
            [objR.LL, objR.prior, objR.transmat, objR.mu, objR.Sigma, objR.mixmat] =...
            mhmm_em(R, priorR0, transmatR0, muR0, SigmaR0, mixmatR0, 'max_iter', 10, 'verbose', 0);
            if (objR.LL(end) > bestR.LL(end))
            bestR = objR;
            end

            %train left Hidden Markov Model
            [objL.LL, objL.prior, objL.transmat, objL.mu, objL.Sigma, objL.mixmat] =...
            mhmm_em(L, priorL0, transmatL0, muL0, SigmaL0, mixmatL0, 'max_iter', 10, 'verbose', 0);
            if (objL.LL(end) > bestL.LL(end))
            bestL = objL;
            end
            end

            objR = bestR;
            objL = bestL;
            % evaluate data point left out of training phase by the HMM
            loglikR = mhmm_logprob(temp, objR.prior, objR.transmat, objR.mu, objR.Sigma, objR.mixmat);
            loglikL = mhmm_logprob(temp, objL.prior, objL.transmat, objL.mu, objL.Sigma, objL.mixmat);
            if ((loglikL >= loglikR) && x <= szfl(2))
                correct=correct+1;
            elseif ((loglikR > loglikL) && x > szfl(2))
                    correct=correct+1;
            elseif (loglikL == -inf && loglikR ~= -inf && x <= szfl(2))
                    correct=correct+1;
            elseif (loglikR == -inf && loglikL ~= -inf && x > szfl(2))
                    correct=correct+1;
            end

            if (x <= szfl(2))
                L(:,end+1) = temp;
            else
                R(:,end+1) = temp;
            end

            end

            if ((correct)/(szfr(2)+szfl(2)) > best)
               best = (correct)/(szfr(2)+szfl(2));
               name = strcat(num2str(u), ' - ', num2str(Q), ' - ', num2str(best));
               bestL = objL;
               bestR = objR;
            end
        resulttable(u, Q) = best;
        save(name, 'bestL', 'bestR');
    end
end
```

Bibliography

[1] M. S. Treder, A. Bahramisharif, N. M. Schmidt, M. van Gerven, and B. Blankertz, "Brain-computer interfacing using modulations of alpha activity induced by covert shifts of attention," *J Neuroeng Rehabil*, vol. 8, p. 24, 2011. http://www.jneuroengrehab.com/content/8/1/24/abstract. 3, 4, 10, 11, 12, 15, 16, 17, 21, 26, 28, 29, 30

[2] J. R. Wolpaw, "Brain-computer interfaces (BCIs) for communication and control," in *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Assets '07, (New York, NY, USA), pp. 1–2, ACM, 2007. http://doi.acm.org/10.1145/1296843.1296845. 3

[3] G. Bauer, F. Gerstenbrand, and E. Rumpl, "Varieties of the locked-in syndrome.," *J Neurol*, vol. 221, no. 2, pp. 77–91, 1979. http://www.biomedsearch.com/nih/Varieties-locked-in-syndrome/92545.html. 3

[4] A. Kübler, N. Neumann, B. Wilhelm, T. Hinterberger, and N. Birbaumer, "Predictability of brain-computer communication," *Int J Psychophysiol*, no. 18(2-3), p. 121129, 2004. http://www.mendeley.com/research/braincomputer-predictability-braincomputer-communication. 3

[5] A. O. Argunsah and M. Cetin, "AR-PCA-HMM approach for sensorimotor task classification in EEG-based brain-computer interfaces," in *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, (Washington, DC, USA), pp. 113–116, IEEE Computer Society, 2010. http://dx.doi.org/10.1109/ICPR.2010.36. 3, 5

[6] B. Obermaier, C. Guger, C. Neuper, and G. Pfurtscheller, "Hidden Markov models for online classification of single trial EEG data," *Pattern Recogn. Lett.*, vol. 22, pp. 1299–1309, October 2001. http://portal.acm.org/citation.cfm?id=569343.569349. 3, 5, 12

[7] S. Solhjoo, A. Nasrabadi, and M. Golpayegani, "Classification of chaotic signals using HMM classifiers: EEG-based mental task classification," *Proc. the European Signal Processing Conference*, 2005. http://eurasip.org/Proceedings/Eusipco/Eusipco2005/defevent/papers/cr1122.pdf. 3, 5

[8] Y. Shen, Y. Gu, and I. Pe'er, "Poster: A hidden markov model for copy number variant prediction from whole genome resequencing data," *Computational Advances in Bio and Medical Sciences, IEEE International Conference on*, vol. 0, p. 260, 2011. http://doi.ieeecomputersociety.org/10.1109/ICCABS.2011.5729914. 5

[9] I. Patel and Y. S. Rao, "Speech recognition using hidden Markov model with MFCC-subband technique," pp. 168–172, 2010. http://dx.doi.org/10.1109/ITC.2010.45. 5

[10] G. A. Fink, S. Vajda, U. Bhattacharya, S. K. Parui, and B. B. Chaudhuri, "Online bangla word recognition using sub-stroke level features and hidden markov models," in *Proceedings of the 2010 12th International Conference on Frontiers in Handwriting Recognition*, ICFHR '10, (Washington, DC, USA), pp. 393–398, IEEE Computer Society, 2010. http://dx.doi.org/10.1109/ICFHR.2010.68. 5

[11] V. Southavilay, K. Yacef, and R. A. Calvo, "Analysis of collaborative writing processes using hidden markov models and semantic heuristics," *IEEE International Conference on Data Mining Workshops*, pp. 543–548, 2010. http://doi.ieeecomputersociety.org/10.1109/ICDMW.2010.118. 5

[12] H. Jin and X. Sun, "SAR image speckle noise suppression based on DFB hidden markov models using immune clonal selection thresholding," *Image and Video Technology, Pacific-Rim Symposium on*, vol. 0, pp. 288–293, 2010. http://doi.ieeecomputersociety.org/10.1109/PSIVT.2010.55. 5

[13] D. Barber, "Bayesian reasoning and machine learning," 2011. http://www.amazon.com/Bayesian-Reasoning-Machine-Learning-Barber/dp/0521518148/ref=sr_1_1?ie=UTF8&qid=1311434464&sr=8-1. 5, 6, 8, 9

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal statistical society, Series B*, vol. 39, no. 1, pp. 1–38, 1977. http://web.mit.edu/6.435/www/Dempster77.pdf. 7

[15] S. Borman, "The expectation maximization algorithm: A short tutorial." http://www.seanborman.com/publications/EM_algorithm.pdf. 7

[16] S. J. Luck, "An introduction to the event-related potential technique (cognitive neuroscience)," Aug. 2005. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262621967. 11, 18

[17] C. Neuper, A. Schlögl, and G. Pfurtscheller, "Enhancement of left-right sensorimotor EEG differences during feedback-regulated motor imagery.," *J Clin Neurophysiol*, vol. 16, pp. 373–382, July 1999. http://view.ncbi.nlm.nih.gov/pubmed/10478710. 12

[18] C. Guger, H. Ramoser, and G. Pfurtscheller, "Real-time EEG analysis with subject-specific spatial patterns for a brain-computer interface (BCI)," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, pp. 447–456, Dec. 2000. http://dx.doi.org/10.1109/86.895947. 12

[19] M. S. Treder and B. Blankertz, "(C)overt attention and visual speller design in an ERP-based brain-computer interface," *Behav Brain Funct*, vol. 6, p. 28, May 2010. http://www.behavioralandbrainfunctions.com/content/6/1/28. 13

[20] K. Murphy, "HMM toolbox for matlab." http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html. 14

[21] M. van Gerven, A. Bahramisharif, T. Heskes, and O. Jensen, "Selecting features for BCI control based on a covert spatial attention paradigm," *Neural Networks*, vol. 22, no. 9, pp. 1271 – 1277, 2009. http://www.sciencedirect.com/science/article/pii/S0893608009001075. 15

[22] A. Bahramisharif, M. van Gerven, and T. Heskes, "Exploring the impact of alternative feature representations on BCI classification," 2009. http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es2009-52.pdf. 18

[23] H.-I. Suk and S.-W. Lee, "Two-layer hidden markov models for multi-class motor imagery classification," *Brain Decoding: Pattern Recognition Challenges in Neuroimaging, Workshop on*, vol. 0, pp. 5–8, 2010. http://doi.ieeecomputersociety.org/10.1109/WBD.2010.16. 19

[24] C. Guger, G. Edlinger, W. Harkam, I. Niedermayer, and G. Pfurtscheller, "How many people are able to operate an EEG-based brain-computer interface (BCI)?," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on [see also IEEE Trans. on Rehabilitation Engineering]*, vol. 11, no. 2, pp. 145–147, 2003. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1214705. 25

[25] B. Blankertz and F. Popescu, "Challenges for brain-computer interface research for human computer interaction applications," 2008. http://hmi.ewi.utwente.nl/chi2008/chi2008_files/blankertz.pdf. 25

[26] B. Xie, W. Pan, and X. Shen, "Penalized mixtures of factor analyzers with application to clustering high-dimensional microarray data," *Bioinformatics*, vol. 26, no. 4, pp. 501–508, 2010. http://bioinformatics.oxfordjournals.org/content/26/4/501.full.pdf+html. 29