Gaussian Observation Hidden Markov models for EEG analysis

William D. Penny and Stephen J. Roberts

Technical Report,
Neural Systems Research Group,
Department of Electrical and Electronic Engineering,
Imperial College of Science, Technology and Medicine,
London SW7 2BT., U.K.
w.penny@ic.ac.uk, s.j.roberts@ic.ac.uk

October 5, 1998

Abstract

We show, using a number of synthetic data sets, that HMMs can detect the changes in DC levels, correlation, frequency and coherence that are typical of the nonstationary changes in an EEG signal. We also show that the number of hidden states in an HMM can be chosen using a cluster analysis of derived features. Our experiments are based on the use of HMMs with Gaussian observation densities trained on autoregressive (AR) or multivariate autoregressive (MAR) coefficients. The extraction of these coefficients requires that the signal be windowed and the coefficients calculated for each window. The windowing procedure is, however, fundamentally flawed; some windows contain information from more than one state and so can lead to a larger number of inferred states than is necessary and to incorrect estimates of state and state transitions.

1 Introduction

In a review of spatial pattern recognition methods for EEG analysis, Nunez [2] stresses the need to deal with non-stationary and non-localized sources of information. In this brief report we show that Hidden Markov Models (HMMs) fit exactly this requirement; they are the perfect model for EEG analysis. In the HMM, there a number of underlying states each of which is associated with stationary dynamics. The models associated with each state may be multivariate and rely on some local or distributed subset of electrodes. Transitions between states allow for tracking of nonstationary dynamics.

In this report, we focus on HMMs with Gaussian observation densities. The HMMs are trained using the EM algorithm and the most likely state sequences are identified using single-state or Viterbi decoding. Full details of the algorithms are available in the tutorial by Rabiner [5].

The HMM can be applied either to the original electrode recordings or to a time-series of autoregressive (AR) or multivariate autoregressive (MAR) coefficients derived from the original data (see Appendices A and B for details). In this short technical note, we demonstrate a few properties of the HMM (on synthetic data) which make it particularly suitable for EEG analysis.

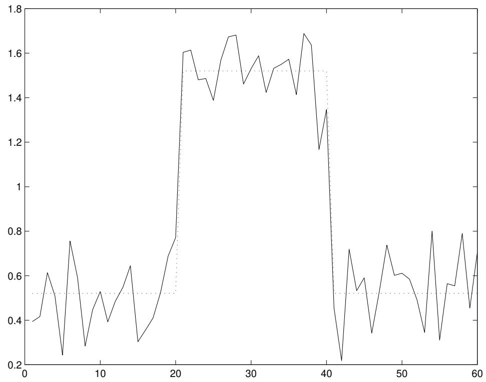


Figure 1: **DC changes**. HMM with two states, one-dimensional observations, different means, common variances. The solid line shows the original time series. The dotted line shows the mean of the Gaussian distribution associated with the HMM state chosen by Viterbi decoding.

2 DC changes

Figure 1 shows a time series generated from an HMM with two hidden states with means $\boldsymbol{\mu} = [0.5, 1.5]$. The variances were $\boldsymbol{Cov} = [0.02, 0.02]$. The hidden state sequence is state 1 for 20 steps, state 2 for 20 steps then state 1 again for the final 20 steps. Given these state durations it is possible to calculate the optimal diagonal elements in the state transition matrix [5] and as we have only a 2 by 2 matrix, the off diagonal elements are also calculable giving $A = [0.95, 0.05; 0.05, 0.95]$.

An HMM was trained on the time series (the observation model was constrained such that the (co)variances were the same for all states). The learned parameters were $\hat{\boldsymbol{\pi}} = [1, 0]$, $\hat{\boldsymbol{A}} = [0.97, 0.03; 0.05, 0.95]$, $\hat{\boldsymbol{\mu}} = [0.52, 1.52]$ and $\hat{\boldsymbol{Cov}} = [0.02, 0.02]$. Viterbi decoding identified the correct hidden state sequence.

3 Correlation changes

Two-dimensional data was generated from a 2-component Gaussian Mixture Model (GMM) as shown in Figure 2a. The first component was spherical (ie. the variables were uncorrelated) and the second component was elliptical. The data was then ordered such that we get 200 points from component (i), 200 from component (ii) and then 200 from component (i) again. This is shown in Figure 2b. The time series are uncorrelated except in the middle portion. The model parameters are $\boldsymbol{\pi} = [1, 0]$, $\boldsymbol{A} = [0.995, 0.005; 0.005, 0.995]$ (ideal transition matrix calculated from state durations), $\boldsymbol{\mu}_1 = [1, 10]$ and $\boldsymbol{\mu}_2 = [1, 10]$ and $\boldsymbol{Cov}_1 = [1, 0; 0, 1]$ and $\boldsymbol{Cov}_2 = [1, 0.9; 0.9, 1]$.

A second GMM was then trained on the data. The learned mean and covariance parameters were $\hat{\boldsymbol{\mu}}_1 = [0.95, 9.80]$ and $\hat{\boldsymbol{\mu}}_2 = [1.07, 10.12]$ and $\hat{\boldsymbol{Cov}}_1 = [0.794, -0.04; -0.04, 0.741]$ and $\hat{\boldsymbol{Cov}}_2 = [1.43, 1.28; 1.28, 1.35]$. The classification error rate was 26.00 per cent. This is'nt great; as 2/3 of the data are from class/state 1 then always guessing class 1 would give an error of 33.00 per cent.

An HMM was then seeded with the mean and covariance values obtained from the GMM and trained with the EM algorithm to fine-tune the means and covariances and to estimate the initial state distribution and state transition probabilities. The learned parameters were $\hat{\boldsymbol{\pi}} = [0.001, 0.999]$, $\hat{\boldsymbol{A}} = [0.995, 0.005; 0.021, 0.979]$, $\hat{\boldsymbol{\mu}}_1 = [0.98, 9.84]$ and $\hat{\boldsymbol{\mu}}_2 = [0.998, 10.02]$

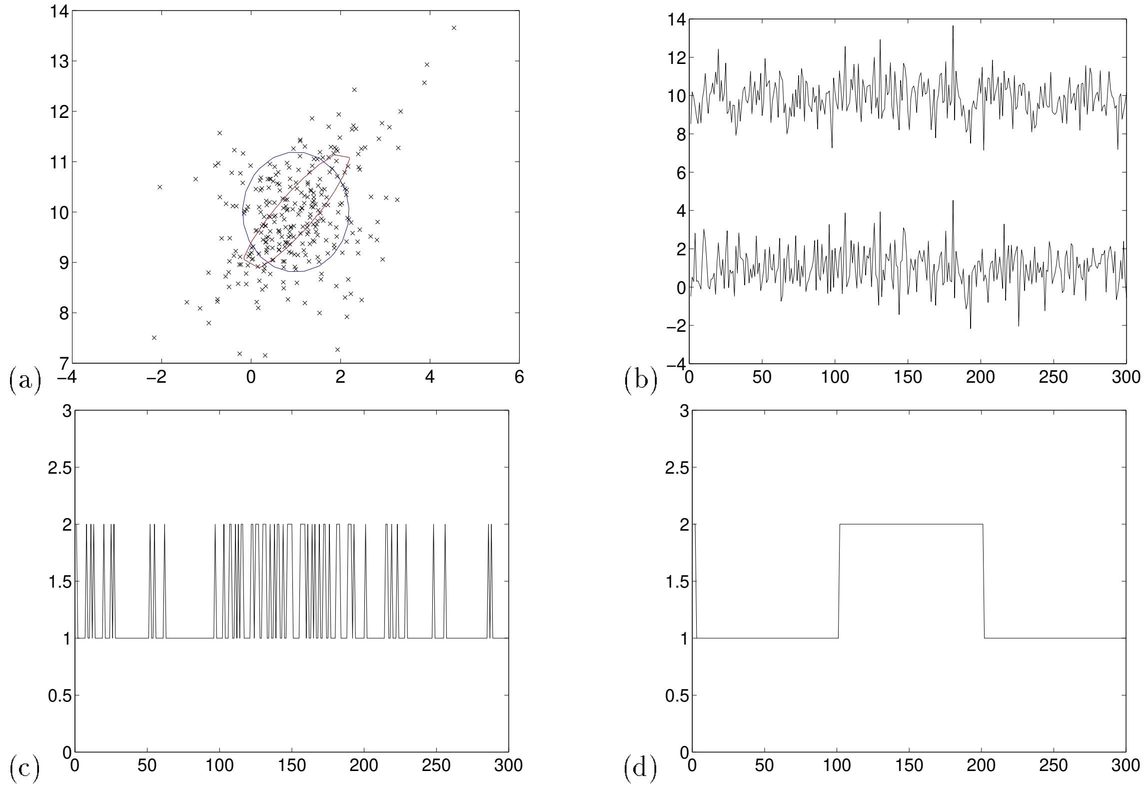


Figure 2: **Correlation changes**. HMM with two states, two-dimensional observations, same means, different covariances: (a) data as generated by a GMM with a mixture of (i) an uncorrelated Gaussian and (ii) a correlated Gaussian (b) the data in plot *a* has been ordered such that we get 100 points from component (i), 100 from component (ii) and then 100 from component (i) again. The time series are uncorrelated at the beginning and end but are correlated in the middle, (c) classifications from a Gaussian Mixture Model, (d) classifications from an HMM.

and $\hat{\boldsymbol{Cov}}_1 = [0.831, -0.013; -0.013, 0.822]$ and $\hat{\boldsymbol{Cov}}_2 = [0.831, -0.013; -0.013, 0.822]$. Single-state decoding then identified the hidden state sequence with an error of 1.00 per cent (3 errors out of 300). Viterbi decoding reduced this to 0.667 per cent (2 out of 300).

The GMM makes a classification at each time step using the current observation only whereas the HMM uses *all* observations to make each classification. Effectively, the GMM tries to discriminate between distributions based on a single data point whereas the HMM is using $\bar{d}_j = 100$ data points, where $\bar{d}_j$ is the average state duration density. This is why the HMM does so much better at recognising the hidden states.

4 Frequency changes

Figure 3 shows nine seconds of a time series generated from an HMM with two hidden states where the observations were generated using autoregressive (AR) models (see Appendix A) of order $p = 4$. There are 128 samples per second. The coefficients of the two models were $a_1 = [-0.3704, -0.1740, 0.2220, 0.3189]$ and $a_2 = [-0.0275, 0.0097, 0.0377, -0.0443]$. These correspond to (i) a 10-Hz sinusoid hidden in Gaussian noise, and (ii) just Gaussian noise. The hidden state sequence is state 2 for 3 seconds, state 1 for 3 seconds then state 2 again for the final 3 seconds.

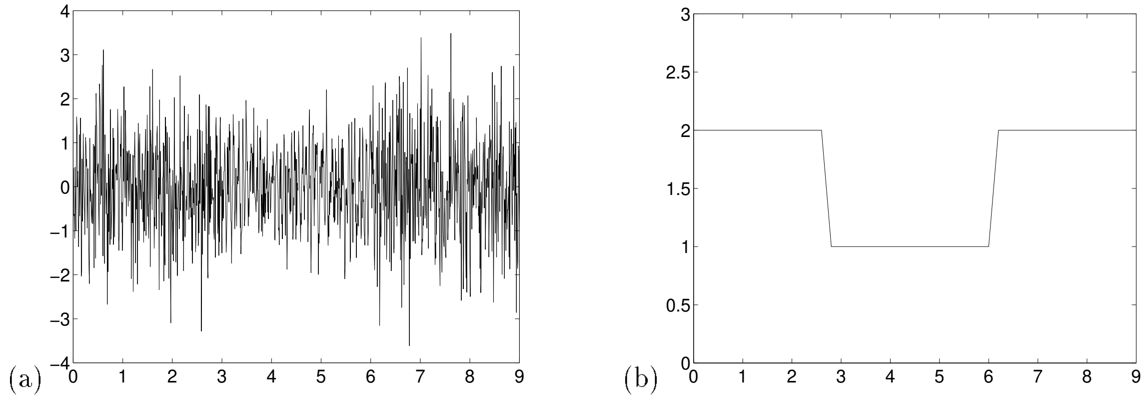


Figure 3: **Frequency changes**. HMM with two states, one-dimensional observations. The two states use different AR models corresponding to (i) a 10-Hz sinusoid hidden in Gaussian noise, and (ii) just Gaussian noise: (a) original time series, (b) classification by HMM.

The observed time series was then split up into overlapping two-second blocks with an offset of 0.2 seconds between blocks. Within each of these blocks an AR(4) model was trained. This generated a series of 46 4-dimensional features on which an HMM was trained.

The learned parameters were $\hat{\boldsymbol{\pi}} = [0, 1]$, $\hat{\boldsymbol{A}} = [0.941, 0.059; 0.036, 0.964]$, $a_1 = [-0.356, -0.035, 0.287, 0.309]$ and $a_2 = [-0.054, 0.013, 0.062, -0.056]$. Viterbi decoding identified the correct hidden state sequence to within the quantisation error introduced by splitting up the data into blocks.

An undesirable aspect of this example is that the time series had to be chopped up into discrete, overlapping blocks. This involved the ad hoc choice of block size and block offset parameters. A desirable aspect of chopping up the signal into discrete blocks and deriving features from those blocks is that we have drastically reduced the dimensionality of the space that the HMM operates in. In this example we have reduced a time series containing (9 seconds times 128 samples per second) 1152 samples into a feature vector containing 46 items. This reduces the computution by a factor of 25.

5 Coherence changes

Figure 4 shows a four second excerpt from a 15-second time series which is sampled at a rate of 128 samples per second. The first and last five seconds were generated from two coherent 10-Hz sinusoids and the middle five seconds were generated by two *incoherent* 10-Hz sinusoids. All data was corrupted by additive Gaussian noise. The cohererence time for the incoherent section was 1 second, that is, after 1 second the phase of one of the signals changed randomly with respect to the other signal. This long coherence time was chosen such that there was no significant change in the spectral density of each signal - thus the only way to tell the signals apart is by looking at the coherency.

The observed time series was then split up into overlapping two-second blocks with an offset of 0.2 seconds between blocks. Within each of these blocks a multivariate autoregressive MAR model (see Appendix B) was trained, with a model order of $p = 4$. This generated a series of 46 16-dimensional (the dimension of the MAR model is p x d x d where $d = 2$ is the dimension of the time series, and $p = 4$ - see Appendix B) features on which a Gaussian observation HMM was trained.

The estimated means of the Gaussians in each of the states correspond to an MAR model for that state. From the MAR coefficients it is possible to reconstruct the spectral densities and the cross-spectral density (see Appendix B). Figure 5 shows the estimated coherency

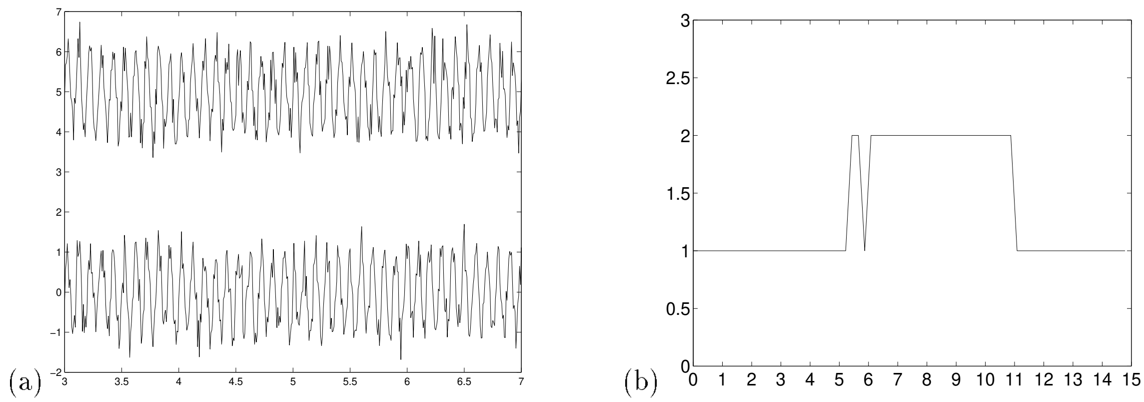


Figure 4: **Coherence changes.** HMM with two states, two-dimensional observations. The two states use different MAR models corresponding to (i) two coherent 10-Hz sinusoids with added Gaussian noise, and (ii) two *incoherent* 10-Hz sinusoids with added Gaussian noise: (a) excerpt of original time series, (b) classification by HMM.

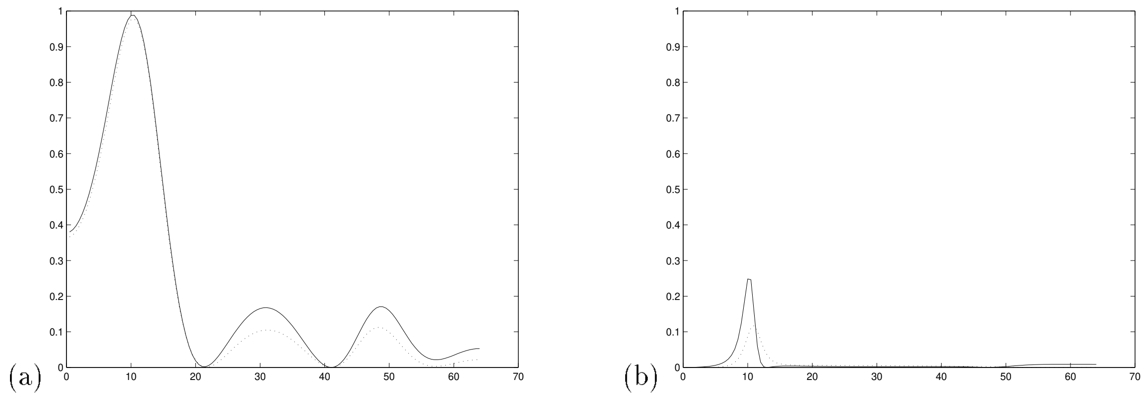


Figure 5: Coherence plots: solid lines for true (sample) coherence values, dotted lines for coherence values as estimated by the HMM for (a) state 1, (b) state 2.

in each state, as calculated from the MAR coefficients associated with each hidden state.

This clearly shows that, in state 1, the signals are recognised as coherent at 10Hz while in state 2 the signals are recognised as incoherent. The spectral densities of each signal can also be accurately estimated in this way. This provides a meaningful interpretation of the otherwise opaque 16-dimensional parameter vector associated with each hidden state.

6 EEG-like data

In this section, we describe a simple model which captures some aspects of the spontaneous EEG signal. We then generate data from the model and infer a Gaussian observation HMM from the observed data alone. Unlike previous examples, we will also infer the number of states in the HMM.

The spontaneous EEG model is itself a three state HMM. For each state, however, data is not generated by a Gaussian but according to

$$y_t = a_1 \sin(2\pi f_1 t) + a_2 \sin(2\pi f_2 t) + e_t \tag{1}$$

where e_t is zero mean IID Gaussian noise with standard deviation 0.3. The two frequencies, f_1 and f_2, were chosen to be 10Hz and 25Hz as they correspond to the frequencies of the EEG alpha (or mu) and beta rhythms. The weighting coefficients were set differently for each state. For state 1 we have $a_1 = 0$, $a_2 = 1$, for state 2 we have $a_1 = 0.5$, $a_2 = 0.5$ and for state 3 we have $a_1 = 1$, $a_2 = 0$. That is, state 1 is pure beta rhythm, state 3 is pure alpha rhythm and state 2 is a mixture of both. The initial state is state 1 and the state transition matrix is

$$\boldsymbol{P} = \begin{pmatrix} 0.8 & 0.2 & 0.0 \\ 0.0 & 0.8 & 0.2 \\ 0.2 & 0.0 & 0.8 \end{pmatrix} \tag{2}$$

The diagonal elements were chosen such that the average state duration density was half a second (this has been observed on real EEG data [3]). The off-diagonal elements were chosen arbitrarily (there is no significance in the fact that we have a cycle of transitions).

Figure 6 shows forty seconds of data generated from the model along with associated hidden state variables. This data was then split up into overlapping half second blocks with an offset of 0.1 seconds between blocks. Within each of these blocks an AR(8) model was trained. This generated a series of 385 8-dimensional features on which a Gaussian observation HMM was trained.

Figure 7a shows a plot of the AR(8) coefficients projected onto their first two principal components. To identify the (assumed unknown) required number of HMM states a Gaussian Mixture Model with full covariance matrices and k components was trained on the AR(8) data. Figure 7b shows of plot of description length versus k. The model with the minimum description length had 3 Gaussians. The spectra of the AR models corresponding to each of these centres is shown in Figure 8. This GMM was then use to initialise the HMM.

The parameters of the HMM were then inferred using the Baum-Welch algorithm. The resulting spectra changed little from the spectra shown in Figure 8. The initial state was inferred, mistakenly, to be state 3. This is a consequence of windowing the data; in the first window, most of the data derives from state 3 (see Figure 6). The inferred state transition matrix was

$$\hat{\boldsymbol{P}} = \begin{pmatrix} 0.79 & 0.21 & 0.00 \\ 0.07 & 0.85 & 0.08 \\ 0.00 & 0.15 & 0.85 \end{pmatrix} \tag{3}$$

We have mistakenly inferred that state 2 (not state 1) is the most likely successor of state 3. This is, again, a consequence of windowing the data. Any transition in the original time series from state 3 (pure alpha) to state 1 (pure beta) will, in the windowed time series, look like a transition from state 3 to state 2 (transitional windows will contain both 10Hz and 25Hz data) and then to state 1. This explains the mistakenly inferred values in $\hat{\boldsymbol{P}}$.

7 Discussion

We have shown, using synthetic data, that HMMs can detect the changes in DC levels, correlation, frequency and coherence that are typical of the nonstationarities in an EEG signal. We have also shown that the number of hidden states in an HMM can be chosen using a cluster analysis of derived features.

All of our experiments have been based on the use of HMMs with Gaussian observation densities trained on AR or MAR coefficients. The extraction of these coefficients requires that the signal be windowed and the coefficients calculated for each window. The windowing procedure is, however, fundamentally flawed; some windows contain information from more

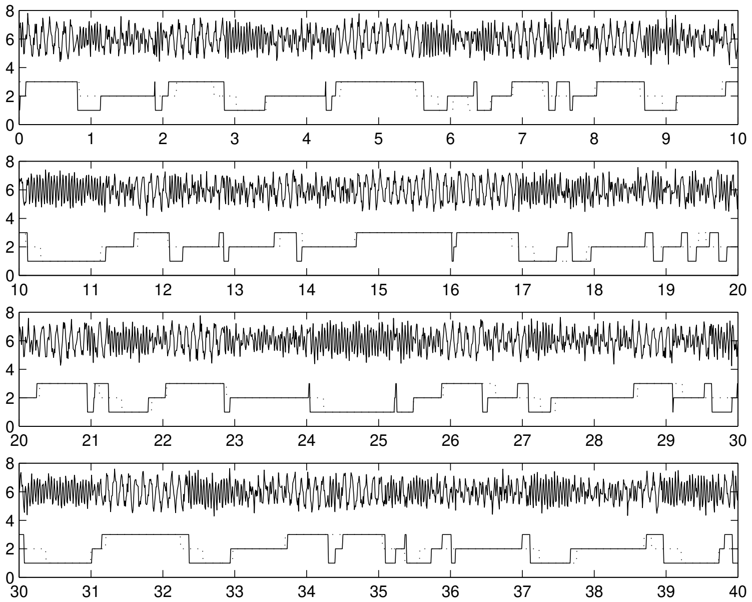
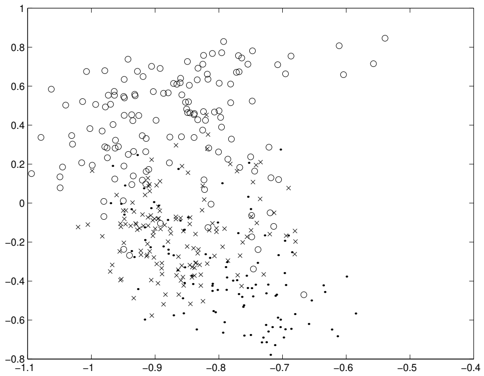


Figure 6: EEG-like data generated from 3-state HMM. The solid staircase plots show the HMM state. The dotted staircase plots show the estimated hidden state as calculated by the inferred HMM. The estimated states are only visible where they differ from the true states. This occurs mainly for (true) transitions from state 3 to state 1 (see text for explanation).

(a) 

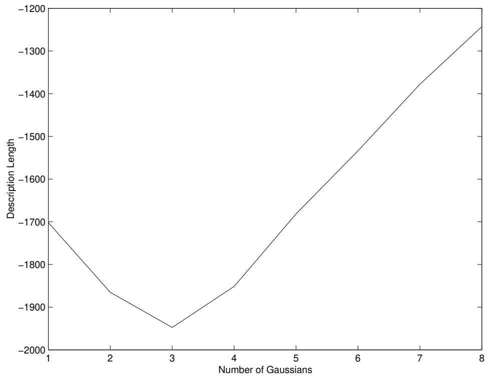
(b) 

Figure 7: (a) AR(8) coefficients projected onto their first two principal components. The data points are identified by their true state labels; state 1 - pure beta (dots), state 2 - alpha and beta (crosses) and state 3 - pure alpha (circles) . These labels are, of course, unknown to the HMM inference algorithm; (b) Description length of Gaussian Mixture Models.

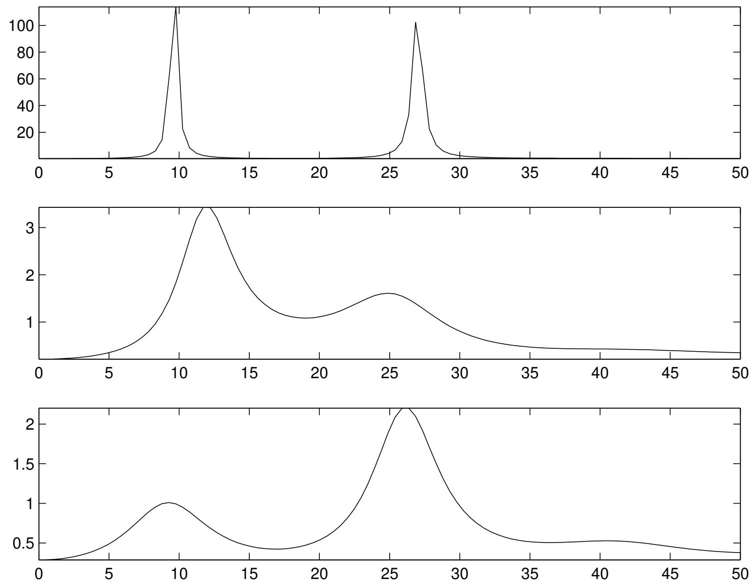


Figure 8: AR spectra associated with the three centres of the Gaussian mixture model. The windowing procedure (used to get *blocks* of AR coefficients) renders the spectral estimates less accurate than they should be.

than one state and so can lead to a larger number of inferred states than is necessary and to incorrect estimates of state and state transitions.

References

[1] S.L. Marple. *Digital spectral analysis with applications.* Prentice-Hall, 1987.

[2] P. L. Nunez. Methods to estimate spatial properties of dynamic cortical source activity. In G. Pfurtscheller and F. H. Lopes da Silva, editors, *Functional Brain Imaging*, pages 3–10. Hans Huber, 1988.

[3] P.L. Nunez. *Neocortical Dynamics and Human EEG Rhythms.* Oxford University Press, 1995.

[4] J. Pardey, S. Roberts, and L. Tarassenko. A Review of Parametric Modelling Techniques for EEG Analysis. *Med. Eng. Phys.*, 18(1):2–11, 1996.

[5] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

A Autoregressive (AR) models

An autoregressive model is a linear predictor used for modelling time series. It is a time-series version of linear regression. An $AR(p)$ model predicts the next value in a time series as a linear combination of the p previous values

$$y_t = -\sum_{i}^{p} a_i y_{t-i} + e_t \tag{4}$$

where a_i are the AR coefficients and e_t is IID Gaussian noise with zero mean and variance σ^2.

The frequency domain characteristics of an AR model may be obtained by taking the z-transform of the above expression and substituting $z = \exp(j2\pi fT)$ to obtain the power at frequency f for sampling period T [4]

$$P(f) = \frac{\sigma^2}{1 + \sum_i^p a_i \exp(-ij2\pi fT)} \tag{5}$$

B Multivariate autoregressive (MAR) models

A multivariate autoregressive (MAR) model is a linear predictor used for modelling multiple time series. An $MAR(p)$ model predicts the next vector value in a d-dimensional time series, $\boldsymbol{y}_t$ as a linear combination of the p previous vector values of the time series

$$\boldsymbol{y}_t = -\sum_i^p \boldsymbol{a}_i \boldsymbol{y}_{t-i} + \boldsymbol{e}_t \tag{6}$$

where each $\boldsymbol{a}_i$ is a $d-$by$-d$ matrix of AR coefficients and $\boldsymbol{e}_t$ is an IID Gaussian noise vector with zero mean and covariance $\boldsymbol{C}$. Thus, the MAR model has the Gaussian observation model as a special case (where only the noise term is included).

The frequency domain characteristics of a multivariate time-series may be summarised by the Hermitian power spectral density (PSD) matrix (Marple [1]; page 387)

$$\boldsymbol{P}(f) = \begin{pmatrix} P_{11}(f) & P_{12}(f) & \cdots & P_{1d}(f) \\ P_{12}(f) & P_{22}(f) & \cdots & P_{2d}(f) \\ \cdots & \cdots & \cdots & \cdots \\ P_{1d}(f) & P_{2d}(f) & \cdots & P_{dd}(f) \end{pmatrix} \tag{7}$$

where the diagonal elements contain the spectra of individual channels and the off-diagonal elements contain the cross-spectra. The complex coherence function is given by (Marple [1]; p. 390)

$$\Phi_{ij}(f) = \frac{P_{ij}(f)}{\sqrt{P_{ii}(f)}\sqrt{P_{jj}(f)}} \tag{8}$$

The coherence spectrum between two channels is given by

$$c_{ij}(f) = \mid \Phi_{ij}(f) \mid^2 \tag{9}$$

The phase spectrum, between two channels is given by

$$\theta_{ij}(f) = tan^{-1}\left[\frac{Im(\Phi_{ij}(f))}{Re(\Phi_{ij}(f))}\right] \tag{10}$$

For an MAR model the PSD is given by (Marple 1987; page 408)

$$\boldsymbol{P}_{MAR}(f) = T\,[\boldsymbol{A}(f)]^{-1}\,\boldsymbol{C}\,[\boldsymbol{A}(f)]^{-H} \tag{11}$$

where T is the sampling period, H denotes the Hermitian transpose and $\boldsymbol{A}(f)$ is given by

$$\boldsymbol{A}(f) = \boldsymbol{I} + \sum_i^p \boldsymbol{a}_i \exp(-ij2\pi fT) \tag{12}$$

where $\boldsymbol{I}$ is the identity matrix.