# Dynamic models for nonstationary signal segmentation

William D. Penny and Stephen J. Roberts

*Submitted to Computers and Biomedical Research*
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 1, 1998

**Abstract**

This paper investigates Hidden Markov Models (HMMs) in which the observations are generated according to an autoregressive (AR) model. The overall model performs nonstationary spectral analysis and automatically segments a time series into discrete dynamic regimes. Because learning in HMMs is sensitive to initial conditions, we initialise the HMM model with parameters derived from a cluster analysis of Kalman filter coefficients. An important aspect of the Kalman filter implementation is that the state noise is estimated on-line. This allows for an initial estimation of AR parameters for the different dynamic regimes. These estimates are then fine-tuned with the HMM model. The method is demonstrated on a number of synthetic problems and on electroencephalogram (EEG) data.

## 1 Introduction

Autoregressive (AR) models are a well known parametric technique for the spectral estimation of stationary signals [1]. By embedding AR models into Kalman filters it is also possible to use them for nonstationary spectral analysis [2].

The Kalman filter uses a state-space representation in which a state vector evolves in time. The evolution of this state vector (eg. vector of AR parameters) allows for the continuous tracking of a nonstationary signal. There is another generic dynamic model, however, which has largely been ignored in the biomedical signal processing community, namely, the Hidden Markov Model (HMM) [3]. The HMM also uses a state-space representation but HMM states are discrete, whereas Kalman filter states are continuous.

The suitability of the dynamic model (Kalman filter or HMM) depends on whether the underlying system dynamics are discrete or continuous. In this paper we concentrate on the case of discrete dynamics. We show how a Kalman filter AR model, even though it is intrinsically suited to continuous dynamics, can be set up to detect and follow discrete dynamic changes. This is achieved by updating the 'state noise' on-line; an increase in state noise allows the Kalman filter coefficients to 'jump' to the next dynamic regime.

The Kalman filter is then used to initialise AR models embedded in an HMM. The resulting HMM-AR model is more suited to finding discrete changes in system dynamics. We show how the HMM-AR models can be trained using an Expectation-Maximisation

(EM) algorithm. The methods are then demonstrated on synthetic problems and on EEG data.

## 2 Kalman filter AR models

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

$$y_t = -\sum_{i}^{p} a_i y_{t-i} + v_t \tag{1}$$

where $a_i$ are the AR coefficients and $v_t$ is independent identically distributed (IID) Gaussian noise with zero mean and variance $\sigma^2$. We define a *time-varying* autoregressive model or Kalman filter AR model by the following state-space equations

$$\boldsymbol{q}_t = \boldsymbol{q}_{t-1} + \boldsymbol{w}_t, \qquad \boldsymbol{w}_t \sim N(0, \boldsymbol{W}_t) \tag{2}$$

$$y_t = \boldsymbol{F}_t \boldsymbol{q}_t + v_t, \qquad v_t \sim N(0, \sigma_t^2) \tag{3}$$

where $\boldsymbol{F}_t = -[y_{t-1}, y_{t-2}, ..., y_{t-p}]$ and $N(\mu, \sigma^2)$ denotes a normal distribution with mean $\mu$ and variance $\sigma^2$. The state variables, $\boldsymbol{q}_t = [a_{1,t}, a_{2,t}, ..., a_{p,t}]^T$, are time-varying AR coefficients. The vector $\boldsymbol{w}_t$ is state noise and the scalar $v_t$ is observation noise. This model is identical to the generic Kalman filter model [4] with the state transition matrix (which acts on $\boldsymbol{q}_{t-1}$) equal to the identity matrix. This simplification means that we have one less parameter matrix to estimate (see end of this section) but, because of the additive state noise, the model is still capable of tracking non-stationarities.

The Kalman filter may be viewed as a Bayesian estimator of the state vector. We can write the probability distribution for the current state vector *before* the observation $y_t$ is made, as the prior distribution

$$p(\boldsymbol{q}_t) = N(\hat{\boldsymbol{q}}_{t-1}, \boldsymbol{\Sigma}_{t-1} + \boldsymbol{W}_t) \tag{4}$$

The mean of the prior distribution is simply the state estimate from the previous time step, $\hat{\boldsymbol{q}}_{t-1}$, and the covariance is the covariance of the state estimate from the previous time step, $\boldsymbol{\Sigma}_{t-1}$, plus the state noise covariance. This second term arises because we are adding on state noise to the previous state estimate (see Eq. 2). The likelihood of making an observation, $y_t$, given state vector $\boldsymbol{q}_t$ can be inferred from an inspection of Eq. 3. It is

$$p(y_t \mid \boldsymbol{q}_t) = N(y_t - \boldsymbol{F}_t \hat{\boldsymbol{q}}_{t-1}, \sigma_t^2) \tag{5}$$

The mean of the likelihood distribution is the difference between the observed value, $y_t$, and the predicted value, $\hat{y}_t = \boldsymbol{F}_t \hat{\boldsymbol{q}}_{t-1}$. This difference is the prediction error, $e_t$. The variance of the likelihood distribution is simply the observation noise variance, $\sigma_t^2$.

The likelihood term acts to reduce the prior distribution to a posterior distribution according to Bayes rule

$$p(\boldsymbol{q}_t \mid y_t) = \frac{p(y_t \mid \boldsymbol{q}_t) p(\boldsymbol{q}_t)}{\int p(y_t \mid \boldsymbol{q}_t) p(\boldsymbol{q}_t) d\boldsymbol{q}_t} \tag{6}$$

Because the prior is Gaussian and the likelihood is Gaussian the posterior is also Gaussian. The mean and covariance of the posterior can be calculated from the above equation.

These calculations, though conceptually simple, are somewhat lengthy. We refer the interested reader to the tutorial by Meinhold and Singpurwalla [4] who show that the posterior is given by

$$p(\boldsymbol{q}_t|y_t) = N(\hat{\boldsymbol{q}}_t, \boldsymbol{\Sigma}_t) \tag{7}$$

where the mean and covariance are

$$\begin{aligned} \hat{\boldsymbol{q}}_t &= \hat{\boldsymbol{q}}_{t-1} + \boldsymbol{K}_t e_t & (8) \\ \boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_{t-1} + \boldsymbol{W}_t - \boldsymbol{K}_t \boldsymbol{F}_t (\boldsymbol{\Sigma}_{t-1} + \boldsymbol{W}_t) & (9) \end{aligned}$$

where $\boldsymbol{K}_t$ is the 'Kalman gain' vector which is calculated as follows

$$\begin{aligned} \boldsymbol{K}_t &= \frac{(\boldsymbol{\Sigma}_{t-1} + \boldsymbol{W}_t)\boldsymbol{F}_t^T}{\sigma_{\hat{y}_t}^2} & (10) \\ \sigma_{\hat{y}_t}^2 &= \sigma_t^2 + \boldsymbol{F}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{F}_t^T + \boldsymbol{F}_t \boldsymbol{W}_t \boldsymbol{F}_t^T & (11) \\ & & (12) \end{aligned}$$

where $\sigma_{\hat{y}_t}^2$ is the estimated prediction variance. This is composed of three terms; the observation noise, $\sigma_t^2$, the component due to the uncertainty in the state variables from the last estimation step and the component due to the injection of state noise.

A useful quantity is the likelihood of an observation given the model parameters before they are updated

$$p(y_t) = N\left(\hat{y}_t, \sigma_{\hat{y}_t}^2\right) \tag{13}$$

In Bayesian terminology this is also known as the evidence for the data point [5]. Data points with low evidence correspond to periods when the statistics of the underlying system are changing (non-stationarity) or, less consistently, to data points having large target noise components.

To apply the model we make initial estimates for the AR parameters, $\hat{\boldsymbol{q}}_0$, and their covariance, $\boldsymbol{\Sigma}_0$, and use the above equations. These can be obtained by passing a static AR model over an initial portion of the time series. We must also plug in estimates for the state noise covariance, $\boldsymbol{W}_t$, and the observation noise variance, $\sigma_t^2$. These quantities can be estimated on-line as described in the next section.

## 2.1 Estimation of state noise

This method, reviewed in [5], is ultimately due to Jazwinski [6]. We assume that the state noise covariance matrix is the isotropic matrix, $\boldsymbol{W}_t = n_t \boldsymbol{I}$. The parameter $n_t$ can be updated according to

$$n_t = h\left(\frac{e_t^2 - \sigma_{n0}^2}{\boldsymbol{F}_t \boldsymbol{F}_t^T}\right) \tag{14}$$

where $h(x)$ is the 'ramp' function

$$h(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

and $\sigma_{n0}^2$ is the estimated prediction variance assuming that $n_t = 0$ (ie. a substitution of $n_t = 0$ into Eq. 12 where $\boldsymbol{W}_t = n_t\boldsymbol{I}$)

$$\sigma_{n0}^2 = \sigma_t^2 + \boldsymbol{F}_t\boldsymbol{\Sigma}_{t-1}\boldsymbol{F}_t^T \tag{16}$$

Thus, if our estimate of prediction error (assuming no state noise) is smaller than our observed error ($e^2$) we should infer that the state noise is non-zero. This will happen when we transit from one stationary regime to another and at these points our estimate of $n_t$ will increase. This estimate can be derived by setting $n_t$ so as to maximise the evidence of a new data point (Eq. 13).

The above re-estimation formula is based on the prediction error from a single prediction and is therefore statistically unreliable. For the state noise to be reliably estimated it is necessary to calculate the prediction error over a window of samples [5]. An alternative heuristic is to smooth our estimates of state noise. One possible smoothed estimate is

$$n_t = \alpha n_{t-1} + (1-\alpha)h\left(\frac{e_t^2 - \sigma_{n0}^2}{\boldsymbol{F}_t\boldsymbol{F}_t^T}\right) \tag{17}$$

where $\alpha$ is a smoothing parameter.

It is also possible to estimate the observation noise on-line using a similar method [5]. However, it is not advisable to estimate both on-line; the discrepancy between the estimated prediction error and the actual prediction error can be explained *either* by a deficiency in state noise *or* as a deficiency in observation noise. To update both on-line (in a single Kalman filter model) would be to account for the discrepancy twice over. For this reason, we set the observation noise to a fixed value estimated by passing a static AR model over an initial portion of the time-series. This is a simplification that we make for all the simulations described in this paper. An alternative procedure is to use multiple Kalman filter models, each with a different observation noise level [7, 8].

The algorithm we have described for updating the state noise relies on a parameter $\alpha$, which smooths the point estimates of the observation noise. Larger values of $\alpha$ correspond to smoother estimates of state noise. In this paper, $\alpha$ is set according to the following argument. The first term in Eq. 17, $n_t = \alpha n_{t-1}$ implies that $n_t = \alpha^t n_0$ where $n_0$ is the initial noise value. Therefore, the point estimate will be reduced to 1 per cent ($n_t/n_0 = 0.01$) of its original value after $t$ samples if $\alpha = 10^{-2/t}$. If we believe the data contains discrete states, with sharp transitions between them, then we would wish $t$ to be 1 or 2 samples. The corresponding value of $\alpha$ is 0.03 or 0.1. In this this paper we use $\alpha = 0.1$.

## 2.2 Effective learning rate

It is interesting to monitor the effective learning rate of a Kalman filter model because it provides a check that the filter is converging to stationary solutions, when given stationary data, and is adapting to new regimes, when given nonstationary data. For a linear predictor, the learning rule for 'on-line' steepest descent is [9]

$$\hat{\boldsymbol{q}}_t = \hat{\boldsymbol{q}}_{t-1} + \rho\boldsymbol{F}_t^T e_t \tag{18}$$

where $\rho$ is the learning rate, which is fixed and chosen arbitrarily beforehand. Comparison of the above rule with the Kalman filter learning rule (Eq. 8) shows that the Kalman filter has a learning rate *matrix* equal to

$$\boldsymbol{\rho} = \frac{\boldsymbol{\Sigma}_{t-1} + n_t\boldsymbol{I}}{\sigma_t^2 + \boldsymbol{F}_t\boldsymbol{\Sigma}_{t-1}\boldsymbol{F}_t^T + n_t\boldsymbol{F}_t\boldsymbol{F}_t^T} \tag{19}$$

The learning rate is a matrix because each state variable (AR coefficient) adapts at a different rate to each different AR input. It is possible to get an idea of the overall learning rate, however, by simply averaging the learning rate over all state variables

$$\rho_{av} = \frac{1}{p} \frac{Tr\left(\boldsymbol{\Sigma}_{t-1} + n_t \boldsymbol{I}\right)}{\left(\sigma_t^2 + \boldsymbol{F}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{F}_t^T + n_t \boldsymbol{F}_t \boldsymbol{F}_t^T\right)} \tag{20}$$

where $Tr()$ denotes the trace of the covariance matrix and $p$ is the number of state variables. The Kalman filter thus uses a learning rate which is directly proportional to the variance of the state variables and is inversely proportional to the estimated prediction variance.

If the prediction variance due to state uncertainty is significantly smaller than the prediction variance due to observation noise ($\boldsymbol{F}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{F}_t^T + n_t \boldsymbol{F}_t \boldsymbol{F}_t^T \ll \sigma_t^2$ ), as it will be once the filter has reached a steady solution, then increasing the state noise parameter, $n_t$, will increase the learning rate. This is the mechanism by which the Kalman filter increases its learning rate when a new dynamic regime is encountered.

# 3 Hidden Markov AR Models

The standard Hidden Markov Model (HMM), as employed for example in speech recognition [3], assumes that each state generates observations according to a Gaussian or Gaussian Mixture model. In this paper we consider HMMs where the observations are generated by AR models. This allows the HMMs to perform non-stationary spectral analysis. We consider HMMs with $N$ discrete hidden states, $[S_1, S_2, ..., S_N]$. The state-space equations defining the HMM are

$$\begin{aligned} P(q_t = S_j | q_{t-1} = S_i) &= r_{ij} && (21)\\ P(\boldsymbol{y}_t | q_t = S_i) &= N(y_t - \boldsymbol{F}_t \hat{\boldsymbol{a}}_i, \sigma_i^2) && (22) \end{aligned}$$

where $q_t$ is the state at time $t$, $r_{ij}$ are the state transition probabilities which are stored in the state transition matrix $\boldsymbol{A}$, $\boldsymbol{F}_t = -[y_{t-1}, y_{t-2}, ..., y_{t-p}]$, $\hat{\boldsymbol{a}}_i$ is the (column) vector of AR coefficients for the $i$th state and $\sigma_i^2$ is the estimated observation noise for the $i$th state. The prediction for the $i$th state is $\hat{y}_t^i = \boldsymbol{F}_t \hat{\boldsymbol{a}}_i$. The above two equations can be compared with the Kalman filter equations 2 and 3. The only difference is that in HMMs the state is discrete whereas in the Kalman filter it is continuous (the Kalman filter state is a continuous vector, $\boldsymbol{q}_t$, containing AR coefficients, whereas the HMM state is a discrete flag, $q_t$, which indexes a vector of AR coefficients, $\hat{\boldsymbol{a}}_{q_t}$. The initial HMM state is set according to

$$P(q_1 = S_j) = \pi_j \tag{23}$$

where we denote the variables $\pi_j$ as components of the vector $\boldsymbol{\pi}$. We now introduce some notation which will allow us to define the learning algorithm. The parameters of the observation model (which we will define later) are concatenated into the parameter set $\boldsymbol{B}$. The parameters of the HMM, as a whole, are concatenated into the parameter set $\boldsymbol{\theta} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$. To train an HMM it is necessary to be able to calculate the probability of being in state $S_i$ at time $t$ given a particular observation sequence

$$P(q_t = S_i | \boldsymbol{y}_1^T) = \gamma_t(i) \tag{24}$$

where $\boldsymbol{y}_1^T$ denotes all observations in a length $T$ sequence. It is also necessary to calculate the probability of state $S_j$ succeeding state $S_i$ given a particular observation sequence

$$P(q_t = S_i, q_{t+1} = S_j | \boldsymbol{y}_1^T) = \xi_t(i,j) \tag{25}$$

The two variables $\gamma_t(i)$ and $\xi_t(i,j)$ can be calculated using the forward-backward algorithm [3]. An Expectation-Maximisation (EM) algorithm may then be used to train the HMM. The EM algorithm is a generic learning method for models which have hidden variables or missing data. It performs maximum likelihood learning via maximisation of an 'auxiliaray function', as desribed in Appendix A. In Appendix B we describe how the EM algorithm can be applied to HMMs, and show that an update formula for the observation model parameters of each hidden state can be found by maximising a component of the EM auxiliary function given by

$$Q = \sum_i \sum_t \gamma_t(i) \log p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) \tag{26}$$

where $p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t)$ is the probability (under the current model parameters) of observing $y_t$ given that the state is $q_t$.

For an AR observation model the probability of observing time-series value $y_t$, given that the HMM is in state $q_t$, is given by

$$p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) = \frac{1}{\hat{\sigma}_{q_t}\sqrt{2\pi}} \exp\left[\frac{-(y_t - \hat{y}_t^{q_t})^2}{2\hat{\sigma}_{q_t}^2}\right] \tag{27}$$

where $\hat{y}_t^i = \boldsymbol{F}_t \hat{\boldsymbol{a}}_i$. By substituting this into Eq. 26 we get

$$Q = \sum_i \sum_t \gamma_t(i) \left[-(y_t - \hat{y}_t^i)^2 / 2\hat{\sigma}_i^2 - \log \hat{\sigma}_i\right] \tag{28}$$

For each state, the AR coefficients can therefore be found by minimising

$$E_i = \sum_t^T \gamma_t(i)(y_t - \hat{y}_t^i)^2 \tag{29}$$

This is a weighted least squares problem. Each 'weight', $\gamma_t(i)$, is the probability that the HMM is in state $i$ at time $t$. If $\boldsymbol{C}_i = diag(\sqrt{\gamma_1(i)}, \sqrt{\gamma_2(i)}, ..., \sqrt{\gamma_T(i)})$, $\boldsymbol{Y}_i$ is the column vector whose $t$th element is $y_t$ and $\boldsymbol{X}_i$ is a matrix whose $t$th row is $[-y_{t-p}, ..., -y_{t-1}]$, then by writing $\tilde{\boldsymbol{X}}_i = \boldsymbol{C}_i \boldsymbol{X}_i$ and $\tilde{\boldsymbol{Y}}_i = \boldsymbol{C}_i \boldsymbol{Y}_i$ we can write the problem in standard least squares form

$$\hat{\boldsymbol{a}}_i = \left(\tilde{\boldsymbol{X}}_i^T \tilde{\boldsymbol{X}}_i\right)^{-1} \tilde{\boldsymbol{X}}_i^T \tilde{\boldsymbol{Y}}_i \tag{30}$$

which can be efficiently computed using singular value decomposition [9]. The noise variance for each state is then calculated as

$$\hat{\sigma}_i^2 = \frac{\sum_t \gamma_t(i)(y_t - \hat{y}_t^i)^2}{\sum_t \gamma_t(i)} \tag{31}$$

To summarise, the HMM model is trained by running the forward backward algorithm to obtain values for $\gamma_t(i)$ and $\xi_t(i,j)$. The initial state probabilities and the state transition matrix are then updated as shown in Appendix B. The AR models are updated using equations 30 and 31. This constitutes one iteration of the EM training algorithm. The computational demands of each training iteration are therefore a single pass of the forward-backward algorithm and the solution of $K$ weighted least squares problems. Typically, training requires tens of iterations.

After the HMM-AR models have been trained, they can be applied to novel data sets and the most likely sequence of hidden states calculated using the Viterbi algorithm [3].

### 3.1 Initialisation

HMMs trained using the EM algorithm are guaranteed to reach only a local maximum likelihood solution (or, less likely, a saddle point of the likelihood function). Because there are many local maxima in the likelihood landscape [3] HMMs are sensitive to how they are initialised. For this reason we initialise the HMM-AR models rather carefully using AR coefficients provided by Kalman filter AR models. The procedure is as follows.

The Kalman filter AR model, as described in section 2, is passed over the data with the state noise smoothing coefficient, $\alpha$, set to a value such that the model is predisposed to detecting sharp state transitions (see section 2.1). The resulting sequence of AR coefficients forms a data set on which we perform a cluster analysis. AR coefficients corresponding to low evidence models are, however, excluded from this data set as these may be at points of state transitions. The resulting set of coefficients are good models of the original time series. A Gaussian Mixture Model (GMM) [9] is then used to perform the cluster analysis. The number of states in the GMM (and therefore the HMM) can be chosen, for example, according to the Minimum Description Length (MDL) [10] principle, or according to prior knowledge. The centre of each Gaussian cluster is then used to initialise the AR coefficients in each state of the HMM-AR model.

An alternative method is to train up a series of HMM models, each with $K$ states, and then to choose $K$ according to the HMM model with the minimum cross-validated likelihood [3]. This procedure is, however, considerably more computationally demanding.

It would also be possible to use a Gaussian observation model in the HMM, instead of an AR model, and to apply it to the Kalman filter coefficients. The problem with this is that it introduces an unnecessary layer of inference and we are not guaranteed to get the same results (tuning up Gaussians on Kalman filter AR coefficients is not the same as tuning up AR coefficients on the original data). Moreover, what are we to do with the low evidence Kalman filter coefficents ? As these are known not to be a good model of the underlying data it seems to pointless to include them, but to exclude them means to accept that there are regions of data which are unclassifiable. For these reasons, the HMM-AR model, which models the time-series directly, is the superior method.

In this section we have so far discussed initialisation of the observation model. It is also important, however, to initialise the state transition matrix. The approach adopted in this paper is to use prior knowledge from the problem domain about average state duration densities, i.e. for how long is the signal stationary ? If $d$ is the average number of samples we wish the HMM to remain in state $i$ for, then the diagonal entry in the state transition matrix is set to $r_{ii} = 1 - \frac{1}{d}$. EEG data for example, is known to be stationary for a period of the order of half a second [11]. For 125Hz data this leads to diagonal entries of 0.984. The off-diagonal entries soak up the remaining probabiltiy.

### 3.2 Window methods

An alternative approach to using Kalman filters for nonstationary spectral estimation is to use Short-Time Fourier Transform or Short-Time AR models. In these methods the time series is split up into a number of discrete overlapping blocks and a Fourier Transform or AR model applied to each.

It is therefore possible to apply an HMM with a Guassian or Gaussian Mixture observation model, not to the original data, but to the sequence of derived feature values. This has the advantage of reducing the dimensionality of the space in which the HMM operates and therefore requires much less computation. This is the approach used in speech recognition [3].

A problem with this approach, however, is that the resulting estimates of state and state transition can be less than accurate. This is because windows which straddle a state transition will contain samples from two different states. The resulting coefficients will be representative of neither state but of a new spurious state which is a combination of the two. It is therefore likely that the optimal number of states will be overestimated and that the state estimates will be somewhat 'smeared'.

# 4 Simulations

In order to validate the dynamic modelling techniques we first analyse, in sections 4.1 and 4.2, time series with known spectral content and segmentation. We then apply the methods to EEG data in sections 4.3 and 4.4. In each of the experiments the order of the AR model was chosen, arbitrarily, to be $p = 8$. Although we made no attempt to optimise the model order, for each experiment and for each state of the HMM, we note that all the usual model order selection methods [12] can be applied.

## 4.1 Spectral Analysis

This example compares spectral estimates from the Kalman filter AR model and the Hidden Markov AR model. It also demonstrates the basic functioning of the Kalman filter model, with a particular focus on the on-line noise estimation algorithm.

A time series was generated consisting of a 10Hz sine wave in the first second, a 20Hz sinewave in the next second and a 30Hz sine wave in the third second. All signals contained additive Gaussian noise with standard deviation 0.1. One hundred samples were generated per second.

Static AR models were derived from several overlapping blocks of data in order to estimate the average observation noise in the time series. This estimate of observation noise was then used in the Kalman filter AR model. The Kalman filter state noise was initially set to zero and was adapted using Jazwinski's algorithm described in section 2.1, using a smoothing value of $\alpha = 0.1$. The model was initialised using linear regression; the first $p$ data points were regressed onto the $p+1$th data point using least squares. The initial state, $q_{p+1}$, was initialised to the resulting least squares solution. The initial state covariance matrix was set to the linear regression covariance matrix, $\Sigma_{p+1} = \sigma_t^2 \boldsymbol{F}_{p+1} \boldsymbol{F}_{p+1}^T$.

Figure 1 shows the original time series and the evidence of each point under the Kalman filter model. Data points occuring at the transitions between different dynamic regimes have low evidence.

Figure 2 shows the evolution of the state noise parameter. The state noise is increased to account for the shortfall between the estimated prediction error and the actual prediction error. The increase in state noise leads to an increase in the learning rate and the filter is seen to quickly adapt to each new dynamic regime.

The Kalman filter model was then used to initialise the HMM-AR model, as described in section 3.1. Figure 3 shows the spectral estimates obtained from the Kalman filter AR model and the HMM-AR model. Although the Kalman filter is able to track rapid state transitions it is not fundamentally suited to doing so. It does, however, provide a useful initialisation for the HMM-AR model which when trained with EM performs the required state and spectral estimates.

## 4.2 State reconstruction

This example demonstrates the ability of the HMM-AR model to recover underlying state dynamics from a nonstationary signal. We generate data from a three state HMM where data in each state is generated according to

$$y_t = a_1 \sin(2\pi 10t) + a_2 \sin(2\pi 25t) + e_t \tag{32}$$

where $e_t$ is zero mean IID Gaussian noise with standard deviation 0.3. 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 $a_1 = 1$, $a_2 = 0$. The initial state is state 1 and the state transition matrix was set to

$$\boldsymbol{P} = \begin{pmatrix} 0.98 & 0.02 & 0.00 \\ 0.00 & 0.98 & 0.02 \\ 0.02 & 0.00 & 0.98 \end{pmatrix} \tag{33}$$

The diagonal elements were chosen such that the average state duration density was half a second. The off-diagonal elements were chosen arbitrarily (there is no significance in the fact that we have a cycle of transitions).

Forty seconds of data were generated at a sampling rate of 125Hz. A Kalman filter AR model was then applied to the data set. Figure 4 shows 10 seconds of data along with the true hidden state and the evidence of the Kalman filter model.

A data set was formed from the AR coefficients corresponding to high evidence predictions. The cut-off was $p(y_t) > 0.5$ (other choices of cut-off values, in the range 0.1 to 0.9, were also tried and resulted in only marginally different initial spectral estimates). Out of 5000 AR vectors only 1807 were used. A three-state GMM was then applied to the data as described in section 3.1. An HMM-AR model was initialised to the centre of each cluster and the model was trained further with the EM algorithm. Other choices of cut-off values, in the range 0.1 to 0.9, were also tried and resulted in only marginally different initial spectral estimates. The resulting HMM-AR models had the state transition matrix

$$\hat{\boldsymbol{P}} = \begin{pmatrix} 0.968 & 0.032 & 0.000 \\ 0.000 & 0.983 & 0.017 \\ 0.023 & 0.000 & 0.977 \end{pmatrix} \tag{34}$$

Application of the Viterbi decoding algorithm resulted in the estimated state sequence shown in Figure 4 (Row 3). Figure 5 shows the estimated spectra from the Kalman filter AR model and HMM-AR model, for a short portion of the signal.

The HMM-AR model thus recovers the correct state transition matrix, state spectra and hidden state sequence. We also applied the windowed HMM method described section 3.2 but the spectral estimates and state transition estimates were considerably worse and are not included here.

### 4.3 EEG hand movement data

In this section we desribe the results of applying an HMM-AR model to a section of EEG data recorded whilst a subject performed cued hand movements [13]. Although recordings were made from 22 electrodes we focus on just one channel as an example. Seconds 30 to 70 of a 90 second EEG recording are shown as the top trace in Figure 7. The dashed vertical lines, preceding seconds 36, 50 and 64 indicate the movement cues which signified to the subject that they should open and close their hand.

A Kalman filter AR model was applied to the data and a two-state Gaussian Mixture model was derived from the resulting AR coefficients. This was used to initialise a two-state HMM-AR model. Two states were used as we wished to build a model in which one state corresponds to 'hand movement' (after the cue) and the other state corresponds to 'no hand movement' (before the cue). We also initialised the state transition matrix such that the average state duration density was half a second. We then trained the HMM-AR model in the usual way. The middle trace in Figure 7 shows the HMM-AR state as calculated by the Viterbi algorithm. The bottom trace in Figure 7 shows the state as estimated by a two-component Gaussian Mixture model trained on the Kalman filter coefficients. The classifications from the GMM, which use full covariance matrices, are equivalent to those obtainable from a quadratic discriminant classifier [9]. The GMM is, however, a static model i.e. the estimate of state at time $t$ is not dependent on the estimate of state at other times. For this reason, the GMM state estimates have many spurious transitions. The HMM-AR model, however, correctly partitions the data into 'hand movement' and 'no hand movement' sections.

Figure 6 shows the spectra associated with each AR model. The 'no hand movement' state is identified with a 10Hz rhythm and the 'hand movement' state with a low-power broad-band frequency spectrum. These two states correspond to before and after the desynchronisation of the 10Hz rhythm, the 'mu' rhythm, that is known to accompany finger and hand movement [13]. Importantly, however, our partitioning of the data was achieved without the knowledge that we should be looking for 10Hz activity.

### 4.4 EEG sleep spindle data

We now describe the results of applying the HMM-AR model to EEG data recorded during a sleep experiment. The aim of the analysis is to identify 'sleep spindles' from background EEG. For this reason we use a two-state model.

The analysis was performed in the usual way. The Kalman filter AR model was applied to a 40-second section of data and high evidence AR-coefficients were used in a two-state Gaussian Mixture Model. This 'unsupervised' method was used to initialise the HMM.

We also tried a 'supervised' initialisation scheme whereby an expert's labelling of the data was used to partition the data into two sets and an AR model was applied to each.

A third initialisation method was also investigated. This consisted of running an AR model over the whole time series and setting the AR model of each state to the global AR solution plus a small noise perturbation vector. This initialisation method is referred to as 'None', as it is rather ad-hoc and we include it only as a baseline measure to quantify the worth of using more sophisticated methods (eg. clustering of Kalman filter AR coefficients).

The state transition matrix was initialised such that the average state duration density was half a second as this is known a priori to be compatible with sleep spindle time scales. We then tried the different initialisation methods in combination with different training schemes whereby the state transition matrices and observation models were either both updated, neither were updated, or only the state transition matrix was updated. The likelihoods of some of the resulting models are summarised in Table 1.

From this table we can conclude that, at least for this data set, (1) unsupervised initialisation is better than no initialisation at all, (2) unsupervised initialisation leads ultimately to the same EM solution as supervised initialisation and (3) the EM solution is a higher likelihood labelling of the data than is the expert's labelling (models 3 or 4).

We also tried varying the initial average state duration densities. Any value above about 0.1 seconds led to the same full EM solution (models 1 or 2). Values below 0.1s led to a sub-optimal solution, with a likelihood of around -540, which involved many spurious state transitions.

Figures 8 (a) and (b) show the spectra associated with the supervised and unsupervised intialisation methods (models 4 and 6). The supervised method identifies the background EEG as being primarily low frequency activity (1-5 Hz) and the sleep spindles as being primarily 12-13Hz activity with a small low frequency component. The unsupervised spectra are similar, but the 12-13Hz peak for the sleep spindle state is much smaller. Figures 9 (a) and (b) show the spectra after training the HMM-AR model with EM and updating both

Table 1: *Likelihood of HMM-AR models on portion of EEG sleep spindle data. Models 1 and 2 (the 'full' EM solutions) are actually the same model but are reached from different initialisations. The likelihood values for the models with no initialisation are median values calculated from 5 trials. For all of the models the state transition matrix was initialised such that the state duration density was half a second.*

| Model | Initialisation | Update transition matrix | Update AR models | Likelihood |
|---|---|---|---|---|
| 1 | Unsupervised | Yes | Yes | -417 |
| 2 | Supervised | Yes | Yes | -417 |
| 3 | Supervised | Yes | No | -455 |
| 4 | Supervised | No | No | -487 |
| 5 | None | Yes | Yes | -545 |
| 6 | Unsupervised | No | No | -811 |
| 7 | None | No | No | -1830 |

the state transition matrix and observation model (the 'full' EM solution; models 1 and 2). Both initialisation methods led to the *same* solution. The background EEG state is relatively unchanged but the spindle state has two equal magnitude peaks; one at 2-3Hz and the other at 12-13Hz.

Figure 10 shows the EEG data, the expert's labelling and the labelling from Viterbi decoding of the full EM solution. These two labellings are largely in agreement, but it could be argued that the expert has missed a number of regions of spindle activity (see seconds 17 and 27 for example).

# 5 Discussion

Both HMMs and Kalman filters can be viewed as variations of unsupervised learning under a single basic generative model [14]. Both models use a state-space representation in which a state vector evolves in time and both models possess the Markov property that the current state depends only on the previous state and not on states before that. For the Kalman filter, the state is continuous and for the HMM it is discrete. In the biomedical signal processing literature, however, it appears that only Kalman filters have been used for state and spectral estimation. In this paper we have shown that their discrete cousins, HMMs, are also a useful tool.

One difference between the HMMs and Kalman filters, as described in this paper, is that the Kalman filters were trained recursively on data up to time $t$ whereas the HMMs were trained in batch mode on the entire data sequence. This is, however, not a fundamental difference as it is possible to train Kalman filters on the entire data sequence either recursively, using a Rauch smoother [19], or in batch mode using an EM algorithm [20].

Our approach consisted of embedding an AR model into an HMM. This approach has also been taken by Fraser *et al.* in the context of time series prediction [15]. The method is also readily extendible to Multivariate Autoregressive (MAR) models which are used for cross-spectral analysis. This can be achieved by plugging in the MAR joint likelihood into the observation model component of the EM auxiliary function in Appendix B and setting the derivatives to zero. The resulting 'HMM-MAR' model performs *non-stationary* cross-spectral analysis, where switching between hidden states causes switching between stationary regimes.

Throughout the paper, we have stressed the need for careful initialisation of the HMM models. This is because there are many sub-optimal solutions in the likelihood landscape that the EM algorithm may be drawn to. The period over which the time series may be considered to be stationary can be used to initialise the state transition matrix. This provides a natural way of incorporating prior knowledge from the problem domain into the statistical model, and can have a crucial influence on the solutions obtained. The observation (AR) models can be initialised via Gaussian Mixture clustering on AR coefficients derived from a Kalman filter model. This 'unsupervised' initialisation method can be crucial to finding good solutions and, surprisingly, can be as good as 'supervised' initialisation methods which use a pre-existing data labelling.

We have incorporated our prior knowledge, from the problem domain and from Kalman filter modelling, by initialising an HMM model. A more rigorous approach is to use a full Bayesian analysis involving Dirichlet priors over the state transition matrix [16] and Gaussian priors over the AR coefficients [17] and to find Maximum Posterior solutions, or full posterior solutions, instead of maximum likelihood solutions. This is an area for future research.

Another key area for future work is the design of algorithms for the automatic selection of the number of states in an HMM, and the application of existing methods [18] to HMM-AR or HMM-MAR models.

## 6 Acknowledgements

The authors would like thank Iead Rezek for many useful discussions regarding AR models. We would also like to thank Nahid Haque who recorded the EEG data for the hand movement experiment and Prof. Rappelsberger, Prof. Trenker, Prof. Zeitlhofer and members of the SIESTA project (funded by the European Community Commision, DG XII, Biomed-2 PL962040) for supplying the sleep spindle data. W. D. Penny was supported by the UK Engineering and Physical Sciences Research Council (EPSRC), grant number GR/K79062, and equipment and technical support for the EEG hand-movement experiments was funded by the Living Again Trust.

## References

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

[2] D. W. Skagen. Estimation of running frequency spectra using a Kalman filter algorithm. *Journal of Biomedical Engineering*, 10:275–279, May 1988.

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

[4] R.J. Meinhold and N.D. Singpurwalla. Understanding the Kalman Filter. *The American Statistician*, 37(2):123–127, May 1983.

[5] J.F.G. DeFreitas, M. Niranjan, and A.H. Gee. Hierarchical Bayesian-Kalman Models for Regularisation and ARD in Sequential Learning. Technical report, Department of Engineering, Cambridge University, 1998.

[6] A.H. Jazwinski. Adaptive filtering. *Automatica*, 5:475–485, 1969.

[7] Y. Bar-Shalom and X.R. Li. *Estimation and Tracking.* Artech House, Boston, MA., 1993.

[8] Z. Ghahramani and G. Hinton. Switching state-space models. Technical report, Department of Computer Science, University of Toronto, 1998.

[9] C. M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, Oxford, 1995.

[10] S. J. Roberts, D. Husmeier, W. Penny, and I. Rezek. Bayesian approaches to Gaussian Mixture Modelling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998. To appear.

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

[12] C. Chatfield. *The analysis of time series: an introduction.* Chapman and Hall, 1996.

[13] S. J. Roberts, W. Penny, and I. Rezek. Temporal and Spatial Complexity measures for EEG-based Brain-Computer Interfacing. *Medical and Biological Engineering and Computing*, 1998. To appear.

[14] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. Technical report, Computation and Neural Systems, California Institute of Technology, 1997.

[15] A.M. Fraser and A. Dimitriadis. Forecasting probability densities by using hidden Markov models with mixed states. In A.S. Weigend and N.A. Gershenfeld, editors, *Time series prediction: forecasting the future and understanding the past*, pages 265–282. Addison-Wesley, 1994.

[16] P. Baldi and S. Brunak. *Bioinformatics: the machine learning approach.* MIT Press, 1998.

[17] D.J.C. Mackay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

[18] T. Ryden. Estimating the order of hidden markov models. *Statistics*, 26(4):345–354, 1995.

[19] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

[20] Z. Ghahramani and G.E. Hinton. Parameter Estimation for Linear Dynamical Systems. Technical Report CRG-TR-96-2, Department of Computer Science, University of Toronto, 1996.

[21] L. E. Baum and T. Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Annals of Mathematical Statistics*, 41:1554–1563, 1966.

[22] 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 B*, 39:1–38, 1977.

# A The EM algorithm

Given a parameter vector $\boldsymbol{\theta}$, a vector of observations $\boldsymbol{y}$ and a vector of hidden (missing or unknown) variables $\boldsymbol{q}$, the EM algorithm [21] [22] will set the parameters to their maximum likelihood values. The hidden variables may be, for example, the label of the Gaussian which generated each data point in a Gaussian mixture model or the label of the state which generated each data point in a Hidden Markov Model. EM operates as follows:

1. Set the parameter vector $\boldsymbol{\theta}$ to an initial value.
2. Choose $\hat{\boldsymbol{\theta}}$ to maximise $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \varepsilon_{p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y})}\left[\log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q}, \boldsymbol{y})\right]$ where $\varepsilon$ denotes the expectation operator.
3. Set $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. If not converged go to 2.

The probability, $p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q}, \boldsymbol{y})$ denotes the joint likelihood of the hidden states and the observations under the new parameter vector. The expectation operator, $\varepsilon_{p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y})}[]$, denotes the expectation with respect to the probability of the hidden states given the observations under the old parameter vector. The algorithm works because an increase in the 'EM auxiliary function', $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, ensures that the likelihood of the data is increased, ie $p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}) > p_{\boldsymbol{\theta}}(\boldsymbol{y})$. This statement can be proved by decomposing the log likelihood using Baye's rule

$$\log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}) = \log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q}, \boldsymbol{y}) - \log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q} \mid \boldsymbol{y}) \tag{35}$$

and taking expectatations with respect to the (old) distribution of hidden variables

$$\log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{y}) = \varepsilon_{p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y})}\left[\log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q}, \boldsymbol{y})\right] - \varepsilon_{p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y})}\left[\log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q} \mid \boldsymbol{y})\right] \tag{36}$$

The term on the left-hand side is unaffected as it does not depend on $\boldsymbol{q}$. Now, the Kullback-Liebler inequality can be written as

$$\varepsilon_{\theta_1}\left[\log p_{\theta_2}(x)\right] \leq \varepsilon_{\theta_1}\left[\log p_{\theta_1}(x)\right] \tag{37}$$

This implies that the second term in Eq. 36 (including the minus sign) is a minimum when when $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}$. Therefore, changing $\boldsymbol{\theta}$ can only increase the second term. Hence, the log-likelihood, and therefore the likelihood, can be maximally increased by setting the first term to its maximum value, that is, by maximising $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$.

# B The EM algorithm for HMMs

The learning algorithm for HMMs is due to Baum and Petrie [21] and may be viewed as a special case of the generic EM algorithm. In an HMM parameterised by some vector $\hat{\boldsymbol{\theta}}$ the observed variable $y_t$ depends *only* on the current state $q_t$ and not previous states and the current state depends *only* on the previous state and not on states before that. This allows the joint likelihood of an observation sequence $\boldsymbol{y} = [y_1, y_t, ..., y_T]$ and hidden variable (state) sequence $\boldsymbol{q} = [q_1, q_2, ..., q_T]$ to be written as

$$p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q}, \boldsymbol{y}) = p_{\hat{\boldsymbol{\theta}}}(q_1) \prod_{t=2}^{T} p_{\hat{\boldsymbol{\theta}}}(q_t \mid q_{t-1}) \prod_{t=1}^{T} p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) \tag{38}$$

The joint log likelihood may be written

$$\log p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{q}, \boldsymbol{y}) = \log p_{\hat{\boldsymbol{\theta}}}(q_1) + \sum_{t=2}^{T} \log p_{\hat{\boldsymbol{\theta}}}(q_t \mid q_{t-1}) + \sum_{t=1}^{T} \log p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) \tag{39}$$

The EM algorithm (see Appendix A) requires us to maximise the EM auxiliary function, $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, the expectation of the joint log likelihood where the expectation is taken relative to the old distribution of hidden (state) variables, $p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y})$. Hence,

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{\boldsymbol{q}} p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y}) \log p_{\hat{\boldsymbol{\theta}}}(q_1) + \sum_{\boldsymbol{q}} p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y}) \sum_{t=2}^{T} \log p_{\hat{\boldsymbol{\theta}}}(q_t \mid q_{t-1}) + \sum_{\boldsymbol{q}} p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y}) \sum_{t=1}^{T} \log p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) \tag{40}$$

where $\sum_{\boldsymbol{q}}$ denotes a sum over all possible hidden state sequences. The EM auxiliary function is seen to consist of three terms; the first for the initial probability parameters, $\pi_i$, the second for the state transition probability paramaters, $\boldsymbol{A}$ and the third for the observation model parameters $\boldsymbol{B}$

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = Q(\pi, \hat{\pi}) + Q(\boldsymbol{A}, \hat{\boldsymbol{A}}) + Q(\boldsymbol{B}, \hat{\boldsymbol{B}}) \tag{41}$$

These terms can be maximised separately giving rise to parameter update equations for each different part of the model. The old distribution of hidden state variables, $p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y})$, can be calculated from the forward-backward algorithm.

An update formula for the initial state distribution parameters, $\pi_i$, can be found by maximising $Q(\pi, \hat{\pi})$. This leads to the estimate [3]

$$\hat{\pi}_i = \gamma_1(i) \tag{42}$$

where $\gamma_t(i)$ is defined in section 3. An update formula for the state transition matrix $\boldsymbol{A}$, having elements $r_{ij}$, can be found by maximising $Q(\boldsymbol{A}, \hat{\boldsymbol{A}})$. This leads to the estimate

$$r_{ij} = \frac{\sum_t \xi_t(i,j)}{\sum_t \gamma_t(i)} \tag{43}$$

where $\xi_t(i,j)$ is defined in section 3. An update formula for the observation model parameters can be found by maximising

$$Q(\boldsymbol{B}, \hat{\boldsymbol{B}}) = \sum_{\boldsymbol{q}} p_{\boldsymbol{\theta}}(\boldsymbol{q}|\boldsymbol{y}) \sum_{t=1}^{T} \log p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) \tag{44}$$

This sum can be re-arranged as

$$\begin{aligned} Q(\boldsymbol{B}, \hat{\boldsymbol{B}}) &= \sum_i \sum_t p_{\boldsymbol{\theta}}(q_t = i \mid \boldsymbol{y}) \log p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) && (45) \\ &= \sum_i \sum_t \gamma_t(i) \log p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t) && (46) \end{aligned}$$

In section 3 we derive a learning rule for the case of an AR observation model by plugging in the appropriate density into $p_{\hat{\boldsymbol{\theta}}}(y_t \mid q_t)$. In principle, however, the density model can be anything. One obvious extension is to Multivariate Autoregressive (MAR) models.

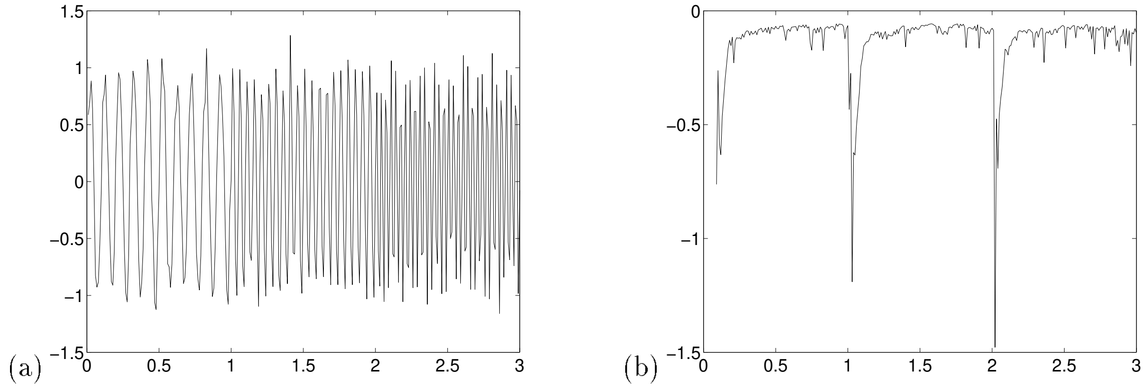



Figure 1: Example 1: (a) Original time series (b) Log evidence of data points under Kalman filter AR model, $\log_{10}(p(y_t))$ (see Eq. 13).

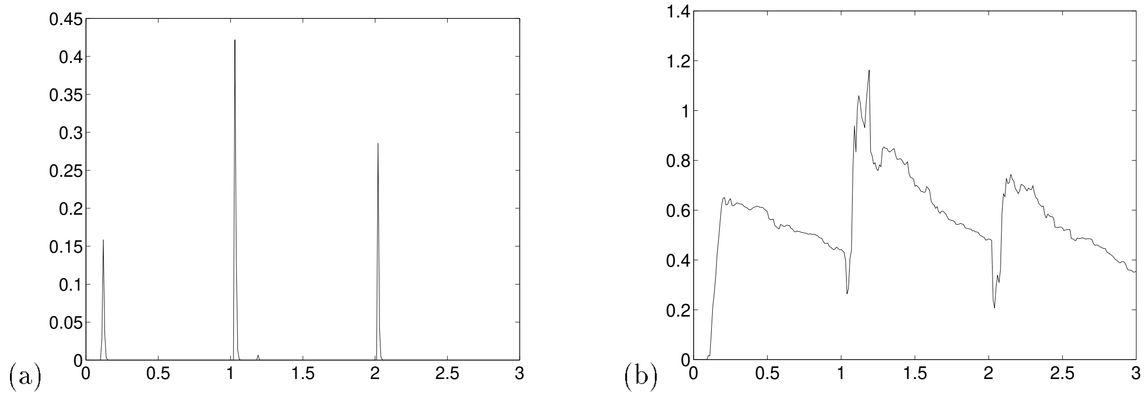



Figure 2: Example 1: (a) Estimate of state noise parameter, $n$, from Eq. 14 (b) effective learning rate, from Eq. 20. The Kalman filter responds to a new dynamic regime by abruptly increasing the estimate of the state noise. This has the effect of increasing the effective learning rate, so that the model adapts to the new data. As the model is then exposed to more data from the same stationary regime, the learning rate reduces in proportion to the reducing uncertainty in model parameters.

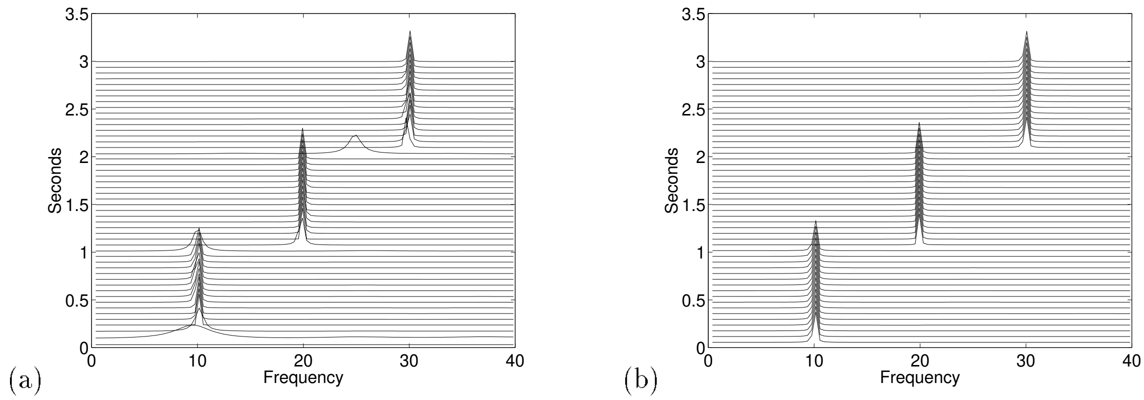



Figure 3: Example 1: Spectrograms from (a) Kalman filter AR model (b) Hidden Markov AR model.

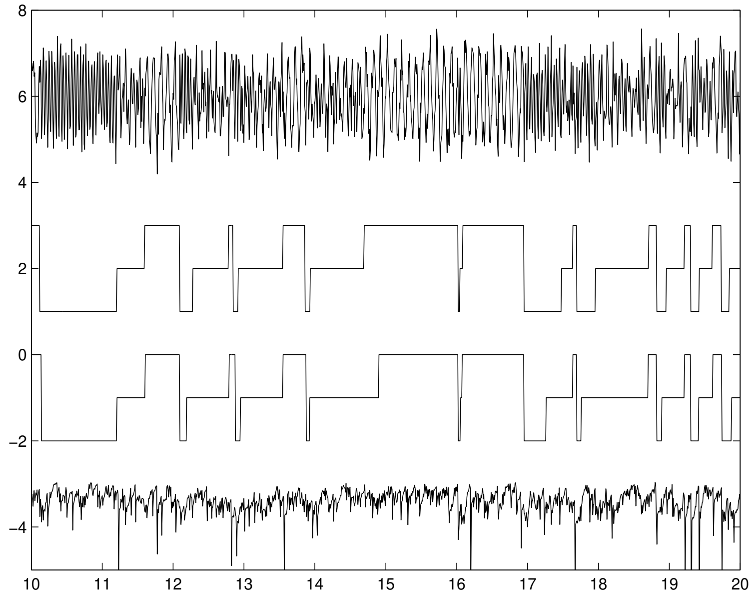
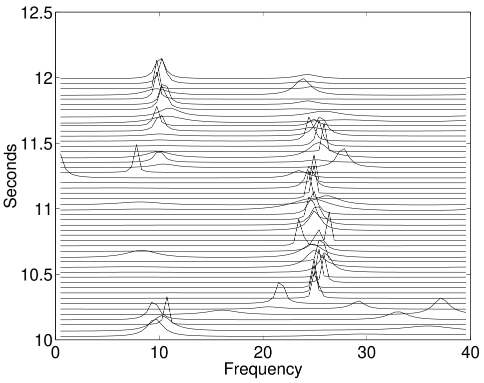



Figure 4: Example 2: (Row 1) Ten seconds of data generated from 3-state sinusoidal HMM, (Row 2) true state, (Row 3) state as estimated by HMM-AR model, (Row 4) log evidence of Kalman filter AR model.

(a) 

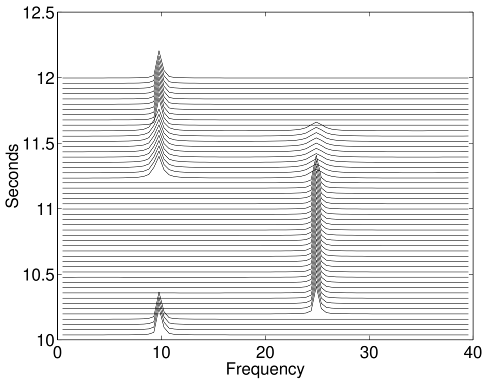

(b) 


Figure 5: Example 2: Spectrograms from (a) Kalman filter AR model (b) Hidden Markov AR model for seconds 10 to 12.

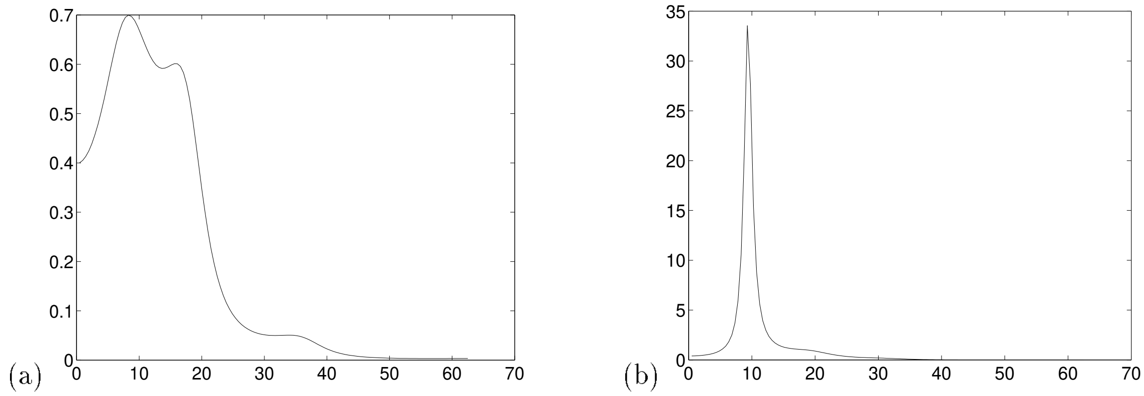



Figure 6: Example 3: Spectra associated with (a) state 1 ('hand movement') and (b) state 2 ('no hand movement') of an HMM-AR model trained on the EEG data.

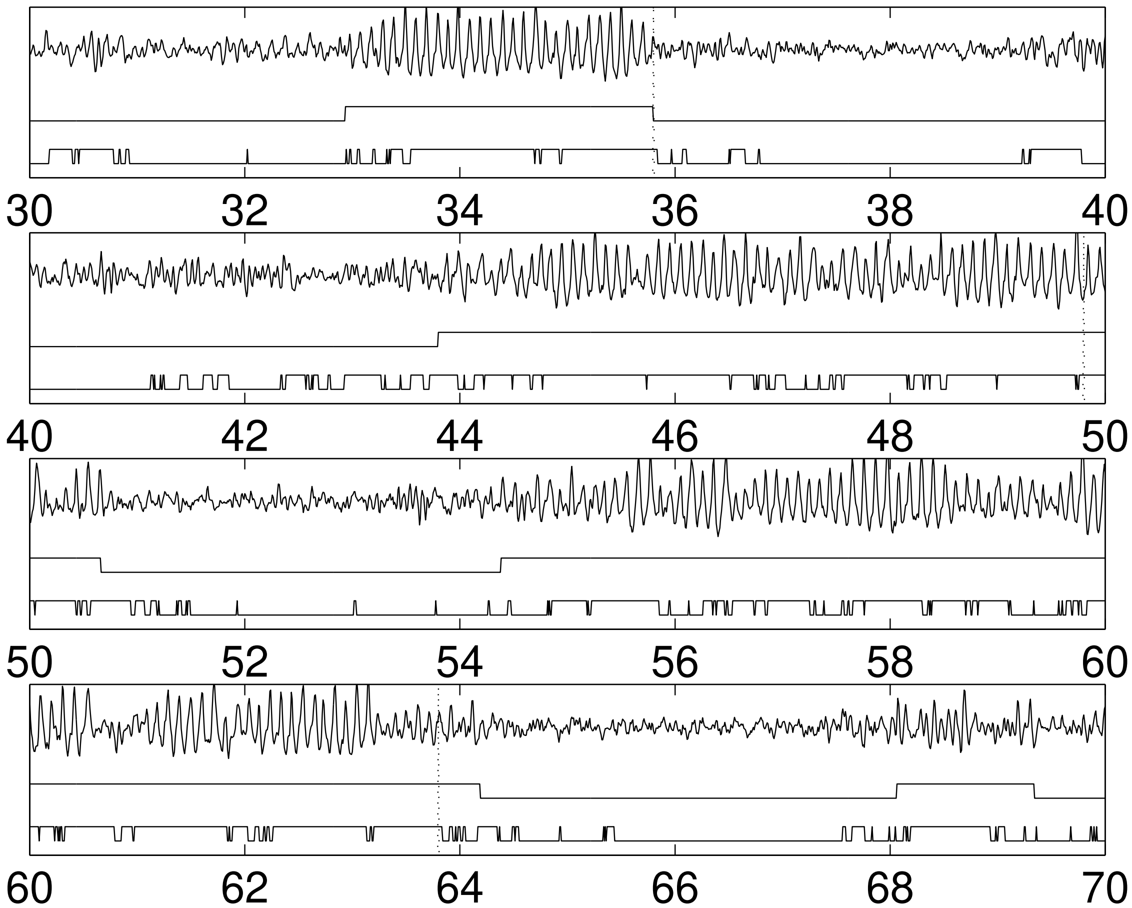



Figure 7: Example 3: (Row 1) Forty seconds of EEG data from a cued hand movement experiment. The mu-rhythm, a 10Hz synchronization in the sensorimotor cortex, can be detected during motor relaxation. When the subject moves their hand (after the cue), the mu-rhythm disappears. The dashed vertical lines, preceding seconds 36, 50 and 64 indicate the movement cues, (Row 2) state as estimated by HMM-AR model, (Row 3) state as estimated by a Gaussian Mixture Model trained on Kalman filter coefficients.

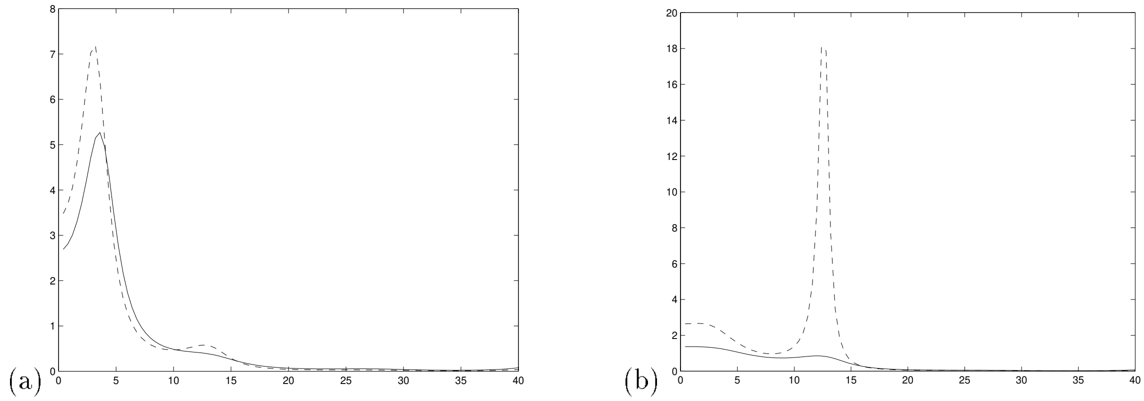



Figure 8: Example 4: (a) state 1 and (b) state 2 spectra used to initialise HMM-AR model. The dashed lines are for supervised initialisation (using the expert's classification) and the solid lines are for unsupervised initialisation. Both initialisation schemes led to the same HMM-AR solution (see Figure 9).

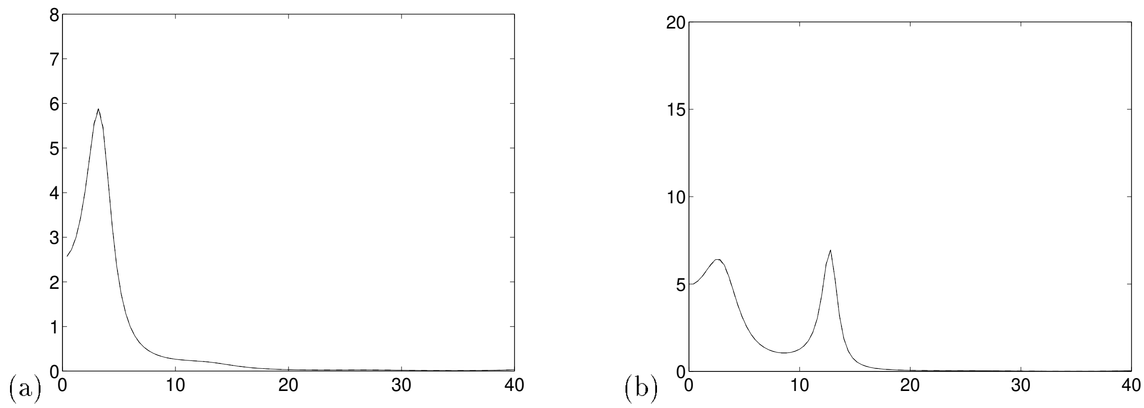



Figure 9: Example 4: (a) state 1 and (b) state 2 spectra after full EM training of the HMM-AR model. Both initialisation schemes led to this solution.

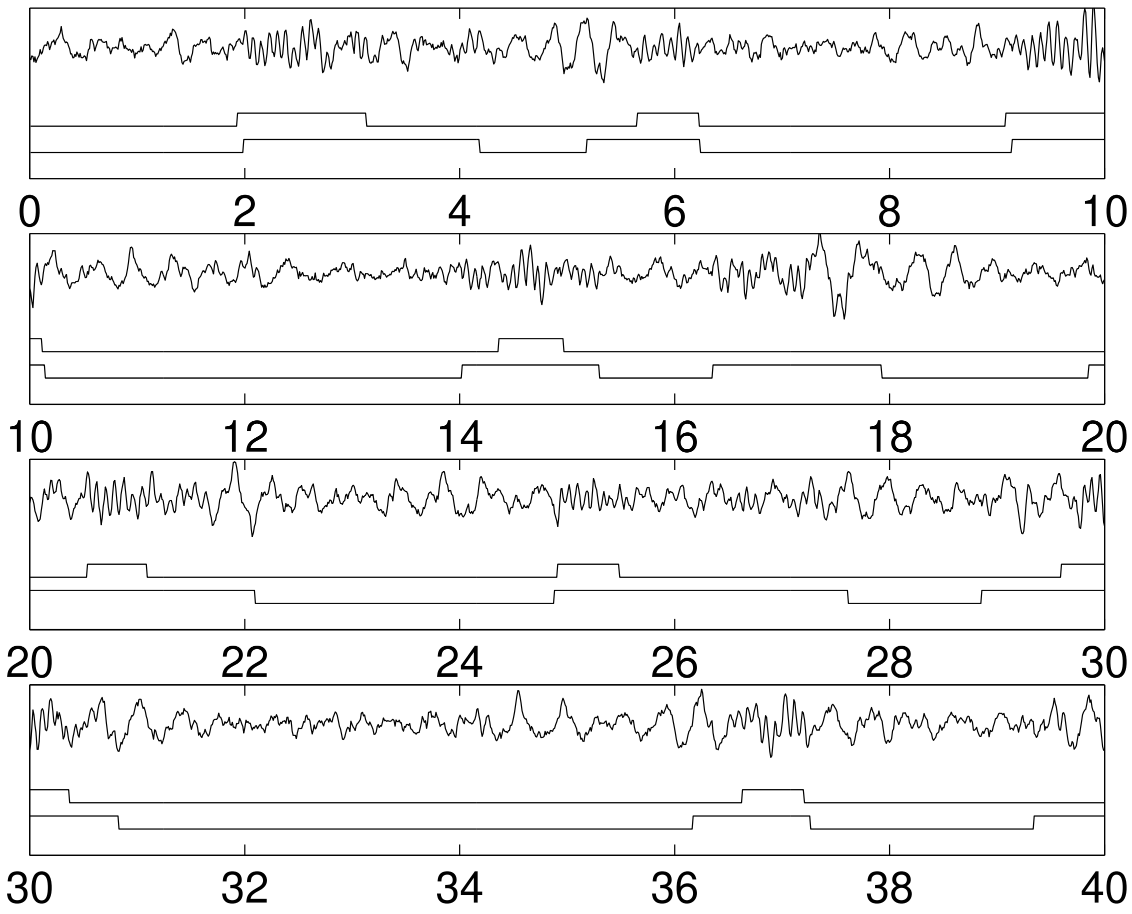



Figure 10: Example 4: (Row 1) Forty seconds of EEG data from a sleep experiment, (Row 2) state estimation by expert, (Row 3) state as estimated by HMM-AR model.