

# Bayes Consistent Classification of EEG Data by Approximate Marginalisation

P. Sykacek, I. Rezek and S. Roberts

University of Oxford, Dept. of Engineering Science, Oxford, U.K.  
{psyk, irezek, sjrob}@robots.ox.ac.uk

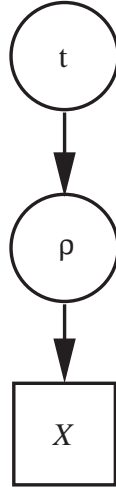
**Summary.** This chapter proposes a generative model and a Bayesian learning scheme for a classifier that takes uncertainty at all levels of inference into account. Classifier inputs will be uncertain if they are estimated, for example using a preprocessing method. Classical approaches would neglect the uncertainties associated with these variables. However, the decisions thus found are not consistent with Bayesian theory. In order to conform with the axioms underlying the Bayesian framework, we must incorporate all uncertainties into the decisions. The model we use for classification treats input variables resulting from preprocessing as latent variables. The proposed algorithms for learning and prediction fuse information from different sensors spatially and across time according to its certainty. In order to get a computationally tractable method, both feature and model uncertainty of the preprocessing stage are obtained in closed form. Classification requires integration over this latent feature space, which in this chapter is done approximately by a Markov chain Monte Carlo (MCMC) method. Our approach is applied to classifying cognitive states from EEG segments and to classification of sleep spindles.

## 1.1 Introduction

Once we opt for decision analysis within a Bayesian framework, we must do this throughout the entire process. The approach studied in this chapter assumes that we are given some segments of time series, each labelled as being in one of  $K$  possible states. Such a setting is typically found in biomedical diagnosis, where successive segments of bio signals have to be classified. Usually, the number of samples within the segments is large. However, the information contained in a segment is often represented by a much smaller number of features. Such problems are typically solved by splitting the whole problem into two parts: a preprocessing method that extracts some features and a classifier.

Using Bayesian inference for each part is common practice. For preprocessing it has been considered among many others by [1] and [2]. A review of recent developments is provided by [3]. The situation in post-processing is similar: see e.g. [4], [5] or [6] for a summary. We could follow these lines and apply Bayesian techniques

to both stages of the decision process separately. However this would mean establishing a non probabilistic link between preprocessing and classification. Such a link does not allow feature or model uncertainty, as found by Bayesian preprocessing, to have an influence on the beliefs reported by the classifier. In a Bayesian sense this is equivalent to approximating the *a-posteriori* probability density over feature variables by a delta peak and ignoring the uncertainty in the stochastic model used for preprocessing.



**Fig. 1.1.** A directed acyclic graph for the hierarchical model. We used  $t$  to denote the unknown state variable of interest,  $\rho$  is a latent (unobserved) variable representing features from preprocessing and  $X$  denotes a segment of a time series. Circles indicate latent variables, whereas the square indicates that  $X$  is an observed quantity.

A model that allows for a probabilistic link between preprocessing and post-processing has to have a structure similar to the directed acyclic graph (DAG) shown in Figure 1.1. The key idea is to treat the features obtained by preprocessing as latent variables. The link between  $\rho$  and  $X$  represents preprocessing, which has to be carried out in a Bayesian setting. The model used in this chapter for preprocessing is a lattice filter representation of an auto regressive (AR) model. The coefficients of this model are the well known reflection coefficients as commonly used in speech processing [7]. We give a short summary of our derivation of “Bayesian reflection coefficients” in section 1.2.

So far approaches using a probabilistic structure, similar to the DAG in Figure 1.1, have focused on marginalising out input uncertainties. The main emphasis in [8] and [9] is to derive a predictive distribution for regression models where input uncertainty is taken into account. In a Bayesian sense this approach is the only way of consistent reasoning if perfect knowledge of inputs is not available. We provide a similar analysis for classification problems. However our approach goes further:

- We allow for different uncertainties at different inputs. This leads to predictions that are dominated by inputs that are more certain. Using generative models to link  $t$  and  $\rho$ , the model provides information about the *true* input values of less certain sensors. Using several sensors will lead to “Bayesian sensor fusion”. The use of a generative model gives us the additional benefit that we can use unlabelled data for parameter inference as well.
- Another extension of the work reported in [8] and [9] is that we take the model uncertainty, inherent to all preprocessing methods, into account. Model uncertainty is represented by the *a-posteriori* probability of the model conditional on the corresponding data segment,  $\mathcal{X}$ .

In the following sections of the chapter, we derive a Bayesian solution for lattice filter AR-models that captures both feature and model uncertainty. We show that marginalising out latent variables in a static DAG indeed performs sensor fusion such that predictions depend more on certain information. Equipped with this theoretical insight, we propose a DAG and inference scheme that is well suited for time series classification. We assume a first order Markov dependency among class labels of interest and derive MCMC updates that sample from the joint probability distribution over latent variables and model coefficients. The experimental section of this chapter provides a synthetic experiment to illustrate the idea and provides then a quantitative evaluation using the proposed method for sleep spindle classification and as a classifier in a brain computer interface experiment.

## 1.2 Bayesian lattice filter

The lattice filter is a representation of an auto regressive (AR) model [7]. Its parameters are the so called reflection coefficients, below denoted as  $r_m$ . Equation (1.1) shows an AR model, where  $x[t]$  is the sample of a time series at time  $t$ ,  $a_\mu$  is the  $\mu$ -th AR coefficient of the  $m$ -th order AR model and  $e[t]$  the forward prediction error, which we assume to be independently identically distributed (i.i.d.) Gaussian with zero mean and precision (inverse variance)  $\beta$ .

$$x[t] = - \sum_{\mu=1}^m x[t-\mu]a_\mu + e[t] \quad (1.1)$$

The autocovariance function, which is the sufficient statistic of a stationary Gaussian noise AR model, is invariant to the direction of time [7]. The corresponding backward prediction model shares thus the same parameters.

$$x[t-m] = - \sum_{\mu=1}^m x[t+1-\mu]a_\mu + b[t] \quad (1.2)$$

To derive Bayesian reflection coefficients, we assume that  $N$  data points,  $\mathcal{X} = \{x[1], \dots, x[N]\}$ , are available to estimate the model. We denote the  $m$ -th order

AR coefficients as  $\varphi_m$ , summarise the forward prediction errors  $e[t]$  as  $\epsilon_m$  and all backward prediction errors  $b[t]$  as  $\mathbf{b}_m$ . Applying the backward time shift operator  $q^{-1}$  on  $\mathbf{b}_m$ , linear regression onto the forward prediction errors,  $\epsilon_m$ , defines the likelihood of the  $m$ -th order AR coefficients and the  $m + 1$ -th reflection coefficient  $r_{m+1}$ .

$$p(\mathcal{X}|r_{m+1}, \beta, \varphi_m) = (2\pi)^{-0.5N} \beta^{0.5N} \times \exp\left(-0.5 (\epsilon_m + r_{m+1}q^{-1}\mathbf{b}_m)^\top (\epsilon_m + r_{m+1}q^{-1}\mathbf{b}_m)\right) \quad (1.3)$$

Since we use this model to represent segments of biological time series, we know with certainty that the underlying AR-process must be stable. As the reflection coefficients of a stable model have to be within the interval  $[-1, 1]$ , we may use a flat prior within this range. Thus the uninformative proper prior over reflection coefficients is:  $p(r_{m+1}) = 0.5$ . Another parameter which appears in the likelihood expression of the lattice filter model is the noise level  $\beta$ . The noise level is a scale parameter and following [10] we use the Jeffreys' prior,  $p(\beta) = 1/\beta$ .

In order to obtain the posterior distribution over the  $m + 1$ -th reflection coefficient  $r_{m+1}$ , the Bayesian paradigm requires us to treat both the  $m$ -th order AR coefficients,  $\varphi_m$ , and the noise level,  $\beta$ , as nuisance parameters and integrate them out. We may simplify calculations considerably by assuming a sharply peaked posterior over  $\varphi_m$ . This results in an order recursive estimation, where we condition on the forward and backward prediction errors that result from the most probable coefficients.

$$p(r_{m+1}|\mathcal{X}) = \frac{1}{\sqrt{2\pi}s} \exp\left(-\frac{1}{2s^2}(r_{m+1} - \hat{r}_{m+1})^2\right) \quad (1.4)$$

as the posterior distribution of the  $m + 1$ -th order reflection coefficient, in which

$$\hat{r}_{m+1} = -\frac{\epsilon_m^\top q^{-1}\mathbf{b}_m}{\mathbf{b}_m^\top \mathbf{b}_m} \quad (1.5)$$

represents the most probable value of the reflection coefficient and,

$$s^2 = \frac{1 - (\hat{r}_{m+1})^2}{(N - 1)}, \quad (1.6)$$

the corresponding variance. We finally need to update the forward and backward prediction errors, to account for the  $m + 1$ -st reflection coefficient.

$$\begin{aligned} \epsilon_{m+1} &= \epsilon_m + \hat{r}_{m+1}q^{-1}\mathbf{r}_m \\ \mathbf{r}_{m+1} &= q^{-1}\mathbf{r}_m + \hat{r}_{m+1}\epsilon_m \end{aligned} \quad (1.7)$$

Our analysis will also consider the uncertainty about the lattice filter model, by allowing for two explanations of the data  $\mathcal{X}$ . We assume that the data is either modelled by an  $M$ -th order lattice filter or that it is white noise. This uncertainty is captured by the posterior probability  $P(I_M|\mathcal{X})$ , where  $I_M \equiv 1$  denotes the lattice filter explanation and  $I_M \equiv 0$  denotes the white noise case. The Bayesian model evidence (marginal likelihood) of the  $M$ -th order lattice filter model,  $I_M \equiv 1$ , is then

$$p(\mathcal{X}|I_M \equiv 1) = 0.5\pi^{-\frac{N}{2}}\Gamma\left(\frac{N}{2}\right)\sqrt{2\pi}s \quad (1.8)$$

$$\times \left((\boldsymbol{\epsilon}_{M-1} + \hat{r}_M \mathbf{b}_{M-1})^\top (\boldsymbol{\epsilon}_{M-1} + \hat{r}_M \mathbf{b}_{M-1})\right)^{-\frac{N}{2}}.$$

Comparing the  $M$ -stage lattice filter model with the Bayesian evidence of explaining the data as white noise gives a measure of model uncertainty if we allow for these two explanations only. We obtain the evidence of a white noise explanation by integrating out the noise level  $\beta$ . We indicate the corresponding model by  $I_0$  and get:

$$p(\mathcal{X}|I_M \equiv 0) = \pi^{-\frac{N}{2}}\Gamma\left(\frac{N}{2}\right) (\boldsymbol{\epsilon}_0^\top \boldsymbol{\epsilon}_0)^{-\frac{N}{2}}, \quad (1.9)$$

where  $\boldsymbol{\epsilon}_0$  refers to the 0-order model residuals (the original time series). Using equal priors for both models, the *a-posteriori* probability of the  $m$ -th reflection coefficient compared to a white noise explanation is:

$$P(I_M \equiv 1|\mathcal{X}) = \frac{p(\mathcal{X}|I_M \equiv 1)}{p(\mathcal{X}|I_M \equiv 1) + p(\mathcal{X}|I_M \equiv 0)}. \quad (1.10)$$

We use  $P(I_M|\mathcal{X})$  to measure the uncertainty of modelling the data by an  $M$ -th order lattice filter, when we allow for a white noise explanation as the other possibility. Our motivation for this two-model hypothesis is our intent to model electroencephalography (EEG) data. EEG is often contaminated by muscle artefacts, which result in measurements that are very similar to white noise.

Equations (1.4), (1.5) and (1.6) imply that the underlying processes are dynamically stable systems since they do not allow reflection coefficients to be outside the interval  $[-1, 1]$ . Since we intend to model the distributions in the latent feature space by a mixture of Gaussian distribution, we have to resolve a mismatch between the domain of the posterior over  $M$ -th order lattice filter coefficients,  $\mathbf{r} \in [-1, 1]^M$  and the domain of the Gaussian mixture model, which requires  $\boldsymbol{\rho} \in \mathbb{R}^M$ . To resolve the problem, we follow [11] and map using Fisher's z-transform

$$\boldsymbol{\rho} = \operatorname{arctanh}(\mathbf{r}), \quad (1.11)$$

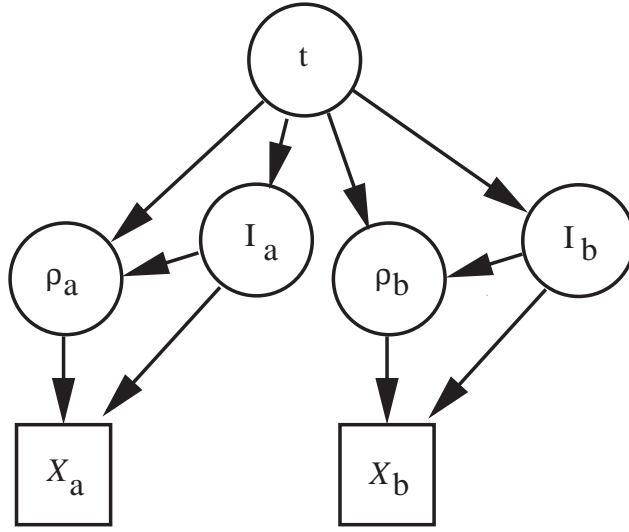
which we use henceforth as representation of the feature space. Using error analysis [12], we can approximate the posterior distribution over the transformed reflection coefficients  $\rho_m$  by

$$p(\rho_m|\mathcal{X}) = \frac{1}{\sqrt{2\pi}}\sqrt{\lambda}\exp\left(-\frac{1}{2}\lambda(\rho_m - \hat{\rho}_m)^2\right) \quad (1.12)$$

where  $\hat{\rho}_m = \operatorname{arctanh}(\hat{r}_m)$   
and  $\lambda = (N-1)(1 - \hat{r}_m^2)$ .

### 1.3 Spatial fusion

In our case the input uncertainty is a result of the limited accuracy of feature estimates as obtained by preprocessing. If we know that the model used in preprocessing



**Fig. 1.2.** A directed acyclic graph that captures both parameter uncertainty as well as model uncertainty in preprocessing. We used  $t$  to denote the unknown state variable of interest,  $\rho_a$  and  $\rho_b$  are two latent variables representing the true values of features estimated by preprocessing. Both  $\mathcal{X}_a$  and  $\mathcal{X}_b$  are the corresponding segments of a time series. The latent binary indicator variables  $I_a$  and  $I_b$  control the dependency of the data segments on the latent variables.

is the *true* model that generated the time series  $\mathcal{X}$ , the approach taken by [8] and [9] would be sufficient. As however was argued in section 1.2, we do not know whether the model used during preprocessing is the *true* one and we have to consider feature as well as model uncertainty. Thus inference has to be carried out with a DAG that allows for both feature *and* model uncertainty. Sensor fusion will only take place if different sensors are conditionally dependent on the state of interest. In order to take this into account, we have to extend the DAG structure as shown in Figure 1.2. We introduce binary indicator variables  $I_a$  and  $I_b$  that control the conditional dependency of the observed data segments  $\mathcal{X}_a$  and  $\mathcal{X}_b$  on the latent variables  $\rho_a$  and  $\rho_b$  respectively.

The DAG in Figure 1.2 illustrates the dependency between a state variable  $t$ , latent variables  $\rho_a$  and  $\rho_b$  and the corresponding segments of a time series  $\mathcal{X}_a$  and  $\mathcal{X}_b$ . The dependency is controlled by two model indicator variables,  $I_a$  and  $I_b$ , one for each sensor. Both models,  $I_a$  and  $I_b$ , representing particular stages of a lattice filter, are *probable* explanations of the corresponding time series  $\mathcal{X}_a$  and  $\mathcal{X}_b$ .

During preprocessing we allow for two possible explanations of each segment of the time series. With probability  $P(I_a|\mathcal{X}_a)$ , the latent variable  $\rho_a$  is conditional on both the time series  $\mathcal{X}_a$  and the state variable  $t$ . However, with probability  $1 - P(I_a|\mathcal{X}_a)$ ,  $\mathcal{X}_a$  is pure white noise and does not require  $\rho_a$ . In this case, we have to condition on  $t$  only. Loosely speaking, we deal with a problem of “probably missing

values". However this interpretation of Figure 1.2 tells us that we need to use the following definition of the conditional probability density of  $\mathcal{X}_a$

$$p(\mathcal{X}_a|\rho_a, I_a) = \begin{cases} p(\mathcal{X}_a|\rho_a, I_a \equiv 1) \\ p(\mathcal{X}_a|I_a \equiv 0) \end{cases}. \quad (1.13)$$

Depending on the value of  $I_a$ , we introduce a conditional independence between the data,  $\mathcal{X}_a$  and the true feature value  $\rho_a$  which is not seen in the DAG in Figure 1.2. When changing indices, we get the same statements for the *latent* variable  $\rho_b$ .

As previously mentioned, we want to predict the belief of state  $t$  conditional on *all* available information. This is the observed time series  $\mathcal{X}_a$  and  $\mathcal{X}_b$  as well as all training data  $\mathcal{D}$  observed so far. Although we will not state this explicitly, *all* beliefs are also conditional on training data  $\mathcal{D}$ . This is a requirement that humans apply intuitively: Whenever a clinical expert wants to monitor a new recording of some biological time series they have prior expectations about the range they should use.

In order to provide deeper insight into the model illustrated in Figure 1.2, we will infer both the *a-posteriori* probability  $P(t|\mathcal{X}_a, \mathcal{X}_b)$  and the *a-posteriori* probability density  $p(\rho_a|\mathcal{X}_a, \mathcal{X}_b)$ . Using Bayes theorem, we express  $P(t)p(\rho_a|t)$  as  $p(\rho_a)P(t|\rho_a)$  and  $P(t)p(\rho_b|t)$  as  $p(\rho_b)P(t|\rho_b)$ . Conditioning on  $\mathcal{X}_a$  and  $\mathcal{X}_b$ , the DAG in Figure 1.2 implies:

$$p(t, \rho_a, I_a, \rho_b, I_b|\mathcal{X}_a, \mathcal{X}_b) = \frac{P(t|\rho_a, I_a)P(t|\rho_b, I_b)p(\rho_a|I_a, \mathcal{X}_a)p(I_a|\mathcal{X}_a)p(\rho_b|I_b, \mathcal{X}_b)p(I_b|\mathcal{X}_b)p(\mathcal{X}_a)p(\mathcal{X}_b)}{P(t)p(\mathcal{X}_a, \mathcal{X}_b)}. \quad (1.14)$$

As none of the variables on the left side of the conditioning bar in (1.14) are observed, we use marginal inference. We obtain  $P(t|\mathcal{X}_a, \mathcal{X}_b)$  by plugging (1.13) into (1.14) and integrating out  $\rho_a, I_a, \rho_b$  and  $I_b$

$$\begin{aligned} P(t|\mathcal{X}_a, \mathcal{X}_b) &= \frac{p(\mathcal{X}_a)p(\mathcal{X}_b)}{p(\mathcal{X}_a, \mathcal{X}_b)P(t)} \\ &\times \sum_{I_a} \int_{\rho_a} P(t|\rho_a, I_a)p(\rho_a|I_a, \mathcal{X}_a)P(I_a|\mathcal{X}_a)d\rho_a \\ &\times \sum_{I_b} \int_{\rho_b} P(t|\rho_b, I_b)p(\rho_b|I_b, \mathcal{X}_b)P(I_b|\mathcal{X}_b)d\rho_b. \end{aligned} \quad (1.15)$$

The naïve Bayes structure of the DAG in Figure 1.2 allows us to expand  $P(t|\rho_a, I_a) = \frac{P(t|\rho_a)P(t|I_a)}{P(t)}$ , which equivalently holds for sensor  $b$ . The influence of feature uncertainty on the probability of the state  $t$  is most easily seen if we assume perfect knowledge of the preprocessing model:

$$P(t|\mathcal{X}_a, \mathcal{X}_b) \propto \frac{1}{P(t)} \int_{\rho_a} P(t|\rho_a)p(\rho_a|\mathcal{X}_a)d\rho_a \int_{\rho_b} P(t|\rho_b)p(\rho_b|\mathcal{X}_b)d\rho_b.$$

This expression shows that that the probability  $P(t|\mathcal{X}_a, \mathcal{X}_b)$  is dominated by  $\mathcal{X}_a$ , if  $\rho_a$  is, under its posterior, more informative about class  $t$ <sup>1</sup>. At a first glance we might

<sup>1</sup> The same argument holds for the other sensor  $\mathcal{X}_b$ .

expect that higher accuracy should dominate and this seems counter intuitive. However, perfect knowledge does not help if the extracted feature is equally likely for different classes. Thus although known with less precision, a variable might dominate if it allows on average for better discrimination. The other extreme is that we know that  $\mathcal{X}_a$  and  $\mathcal{X}_b$  are white noise. In this case we get  $P(I_a \equiv 0|\mathcal{X}_a) = 1$  and  $P(I_b \equiv 0|\mathcal{X}_b) = 1$  and thus

$$P(t|\mathcal{X}_a, \mathcal{X}_b) \propto \frac{P(t|I_a)P(t|I_b)}{P(t)}.$$

This is a very intuitive result, as without parameters  $\rho_a$  and  $\rho_b$  the model indicators  $I_a$  and  $I_b$  are the only information about class  $t$ .

If we are interested in the *a-posteriori* distribution over one of the latent spaces, say  $(I_a, \rho_a)$ , we have to apply similar manipulations. Assuming that the continuous parameter  $\rho_a$  is a point in  $\mathbb{R}^M$ ,  $(I_a, \rho_a)$  lies in  $I_a \equiv 1 \times \rho_a \in \mathbb{R}^M \cup I_a \equiv 0 \times \rho_a \in \mathbb{R}^0$ , where the dimension of  $\rho_a$  in the second case is zero. The posterior is then

$$\begin{aligned} p(\rho_a, I_a|\mathcal{X}_a, \mathcal{X}_b) &= \frac{p(\mathcal{X}_a)p(\mathcal{X}_b)}{p(\mathcal{X}_a, \mathcal{X}_b)} \\ &\times \sum_t \left[ \frac{P(t|\rho_a, I_a)p(\rho_a|I_a, \mathcal{X}_a)P(I_a|\mathcal{X}_a)}{P(t)} \right. \\ &\left. \times \sum_{I_b} \int_{\rho_b} P(t|\rho_b, I_b)p(\rho_b|I_b, \mathcal{X}_b)P(I_b|\mathcal{X}_b)d\rho_b \right]. \end{aligned} \quad (1.16)$$

By exchanging marginalisation over  $(I_b, \rho_b)$  with marginalisation over  $(I_a, \rho_a)$ , we obtain the corresponding *a-posteriori* density over the feature space  $(I_b, \rho_b)$ .

We should now point out that we did not follow an exact Bayesian scheme, since this requires us to use all available information. The derivations presented so far deviate slightly, since the probabilistic dependency between  $t$  and the latent variables  $\rho_a$  and  $\rho_b$  allows, via the conditional distributions  $p(\rho_a|t)$  and  $p(\rho_b|t)$ , for information flow between the two sensors. Hence irrespective whether we know  $t$  or not, we will have information about  $\rho_a$  and  $\rho_b$  that goes beyond the ‘‘stability prior’’ adopted in the last section. However, the problem is that using this information, we can not derive any expressions analytically as was done there. Instead we have to resort to MCMC techniques for the entire analysis including preprocessing [11]. Even with modern computers, this is still a very time consuming procedure. A qualitative argument shows that this approximation will in practical situations not effect the results dramatically. The usual settings in preprocessing will lead to *a-posteriori* densities over coefficients that, compared with the priors, are sharply peaked around the most probable value<sup>2</sup>. In this case using either a uniform prior in the range  $[-1, 1]$ , or the more informative prior provided via  $t$  does not make a big difference.

<sup>2</sup> As can be seen from Equation (1.6), this is just a matter of the number of samples used to estimate the feature values.

As a last step it remains to provide an abstract formulation of an inference procedure of model coefficients for the DAG shown in Figure 1.2. In order to make life easier, we assume for now that we are given only labelled samples. A generalisation to include also unlabelled samples is provided in the next section. Assuming to know the *true* values of features and states, conventional model inference would condition on training data,  $\mathcal{A} = \{\rho_{a,i} \forall i\}$ ,  $\mathcal{B} = \{\rho_{b,i} \forall i\}$  and  $\mathcal{T} = \{t_i \forall i\}$ . Denoting all model coefficients jointly by  $\mathbf{w}$ , inference would lead to  $p(\mathbf{w} | \mathcal{A}, \mathcal{B}, \mathcal{T})$ . In our setting the  $I_{a,i}$ 's,  $I_{b,i}$ 's,  $\rho_{a,i}$ 's and  $\rho_{b,i}$ 's are latent variables and we can not condition on them. Instead we would like to condition on the corresponding  $\mathcal{X}_{a,i}$ 's and  $\mathcal{X}_{b,i}$ 's. Such conditioning can not be done directly. Assuming independence of observations, the DAG in Figure 1.2 implies the likelihood to be

$$\begin{aligned}
p(\mathcal{T}, \mathcal{A}, I_A, \mathcal{X}_A, \mathcal{B}, I_B, \mathcal{X}_B | \mathbf{w}) &= \\
&= \prod_i p(t_i, \rho_{a,i}, I_{a,i}, \mathcal{X}_{a,i}, \rho_{b,i}, I_{b,i}, \mathcal{X}_{b,i} | \mathbf{w}) \\
&= \prod_i \left[ \frac{P(t_i | \rho_{a,i}, I_{a,i}, \mathbf{w}) P(t_i | \rho_{b,i}, I_{b,i}, \mathbf{w})}{P(t_i | \mathbf{w})} \right. \\
&\quad \left. \times p(\rho_{a,i} | I_{a,i}, \mathcal{X}_{a,i}) p(I_{a,i} | \mathcal{X}_{a,i}) p(\rho_{b,i} | I_{b,i}, \mathcal{X}_{b,i}) p(I_{b,i} | \mathcal{X}_{b,i}) \right],
\end{aligned} \tag{1.17}$$

where we use  $I_A = \{I_{a,i} \forall i\}$ ,  $\mathcal{X}_A = \{\mathcal{X}_{a,i} \forall i\}$ ,  $I_B = \{I_{b,i} \forall i\}$  and  $\mathcal{X}_B = \{\mathcal{X}_{b,i} \forall i\}$  and made the dependency on model parameters  $\mathbf{w}$  explicit. We finally get the posterior over model coefficients  $\mathbf{w}$  by multiplication with a prior  $p(\mathbf{w})$  and marginalising out all  $\rho_{a,i}$ ,  $I_{a,i}$ ,  $\rho_{b,i}$  and  $I_{b,i}$ .

$$\begin{aligned}
p(\mathbf{w} | \mathcal{T}, \mathcal{X}_A, \mathcal{X}_B) &\propto p(\mathbf{w}) \prod_i \left[ \frac{1}{P(t_i)} \right. \\
&\quad \times \sum_{I_{a,i}} \int_{\rho_{a,i}} P(t_i | \rho_{a,i}, I_{a,i}, \mathbf{w}) P(I_{a,i} | \mathcal{X}_{a,i}) p(\rho_{a,i} | I_{a,i}, \mathcal{X}_{a,i}) d\rho_{a,i} \\
&\quad \left. \times \sum_{I_{b,i}} \int_{\rho_{b,i}} P(t_i | \rho_{b,i}, I_{b,i}, \mathbf{w}) P(I_{b,i} | \mathcal{X}_{b,i}) p(\rho_{b,i} | I_{b,i}, \mathcal{X}_{b,i}) d\rho_{b,i} \right]
\end{aligned} \tag{1.18}$$

Expression (1.18) is just proportional to the posterior since we still need to normalize. In general neither the integrals nor the normalisation constants in (1.15), (1.16) or (1.18) will be analytically tractable. In all practical problems we will have to apply MCMC methods.

The model in this section illustrates that marginalising out variables and model uncertainty leads to predictions that are dominated by information that allows for reliable discrimination. However the DAG is not optimally suited for classification of biomedical time series. As our interest is classification of adjacent segments of a time series, we can improve on the DAG in Figure 1.2 by allowing for additional temporal dependencies. In the following section we propose a model which allows for a first

order Markov dependency among additional discrete latent variables,  $d_i$ , and thus for information flow across time. We propose Bayesian inference by sampling from the joint probability density over latent variables and model coefficients.

## 1.4 Spatio-temporal fusion

A DAG structure that allows spatio-temporal sensor fusion is obtained by imposing a conditional dependency among adjacent state variables and to spatially different sensors. Sensor fusion is then achieved very naturally by treating this DAG within the Bayesian framework. Bayesian preprocessing will assess both model and coefficient uncertainties. Marginalising out these uncertainties will lead to beliefs about states that are less affected by less reliable information. Such an approach can go even further: it allows one to infer the expected feature values which for unreliable segments will differ from the values obtained from preprocessing alone. This requires that the architecture contains a generative model. Thus, as a by-product, we will be able to use labelled as well as unlabelled data during model inference.

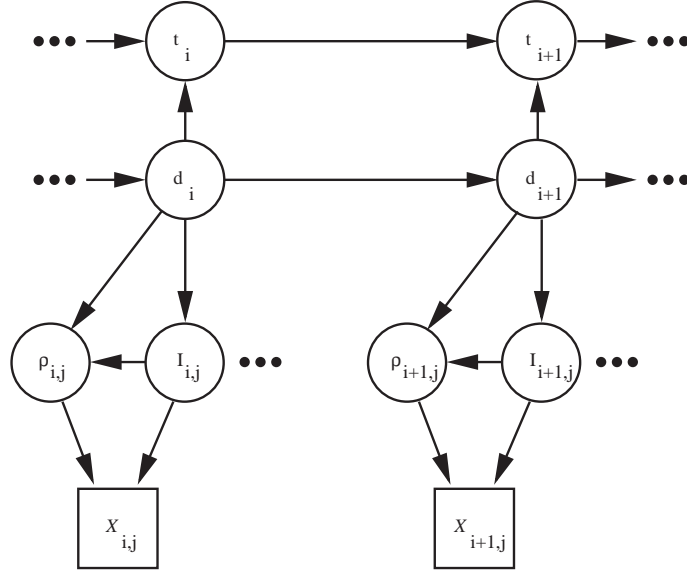
### 1.4.1 A simple DAG structure

This subsection proposes a DAG structure that allows spatio-temporal sensor fusion. Measured in terms of model complexity (number of free parameters), we aim at a simple solution. Assuming that we want to solve a classification task, we regard the class labels as the state of interest. In order to exploit temporal correlations among successive classifications we propose a model that has the latent indicator variables  $d_i$  and the class labels  $t_i$  connected to form a first order Markov chain. The idea behind linking the class labels  $t_i$  is that in cases where one of the state values of  $d_i$  corresponds to a high uncertainty state (i.e. with similar probabilities  $P(t_i|d_i)$  for all  $t_i$ ) this gives us additional information about the class label. Furthermore, we assume conditional independence between all latent variables depending on each state variable. Figure 1.3 shows a DAG structure imposed by these assumptions. Training data consists of labelled as well as unlabelled segments of a time series. Unobserved states are represented by circles and the observed states by squares. We have already decided about the latent variables  $\rho_{i,j}$  and indicator variables  $I_{i,j}$ : they are stages in an AR-lattice filter and  $\mathcal{X}_{i,j}$  are the corresponding segments of a time series. It remains to decide about the generative model between the state variables  $d_i$  and the latent variables  $\rho_{i,j}$  and  $I_{i,j}$ . In our case the  $\rho_{i,j}$ 's are continuous variables and the generative model will be implemented as diagonal Gaussian distributions,

$$p(\rho_{i,j}|d_i) = \mathcal{N}(\rho_{i,j}|\boldsymbol{\mu}_{j,d}, \boldsymbol{\lambda}_{j,d}), \quad (1.19)$$

with mean  $\boldsymbol{\mu}_{j,d}$  and precision  $\boldsymbol{\lambda}_{j,d}$ . The model indicator  $I_{i,j}$  is given a multinomial-1 distribution

$$p(I_{i,j}|d_i) = \mathcal{Mn}(I_{i,j}|\boldsymbol{\Pi}_{j,d}), \quad (1.20)$$



**Fig. 1.3.** A directed acyclic graph for spatio-temporal sensor fusion: The DAG assumes a first order Markov dependency among states  $d_i$  and conditional on these, independence of the latent variables  $\rho_{i,j}$ . We used  $t_i$  to denote the unknown state variables of interest (i.e. the class labels of the segments under consideration), which are linked with a second Markov chain. The idea behind this second chain is that it allows us to resolve ambiguity which might be caused by a high uncertainty state. Both  $\rho_{i,j}$  and  $I_{i,j}$  are latent variables, representing feature and model uncertainty from preprocessing. Finally  $X_{i,j}$  denote the corresponding segments of a time series.

where  $\Pi_{j,d}$  are the binary probabilities observing either one of the two preprocessing models given state  $d_i$ . Model inference will be based on a Markov chain Monte Carlo (MCMC) method. We thus need to consider the likelihood function, design a DAG that shows the relations during inference, specify convenient priors and, as a final step, formulate the MCMC updates.

#### 1.4.2 A likelihood function for sequence models

As already mentioned, we are interested in a fully Bayesian treatment of the model. This can be done as soon as we are able to formulate a normalized likelihood function and priors. We are dealing with a sequence model, where the likelihood is usually (see [13], pp. 150) formulated via paths that are possible sequences of latent states:

$$P(\mathcal{D}, \Pi_d, \Pi_t | \mathbf{w}) = P(d_1) \prod_j p(\rho_{1,j}, I_{1,j} | d_1) P(t_1 | d_1) \quad (1.21)$$

$$\times \prod_{i=2}^N \left( P(d_i | d_{i-1}) P(t_i | d_i, t_{i-1}) \prod_j p(\rho_{i,j}, I_{i,j} | t_i) \right).$$

In (1.21) we used  $\mathcal{D}$  to denote a realisation of all latent variables  $\rho_{i,j}$  and model indicators  $I_{i,j}$ . A sequence of latent variables  $d_i$  is denoted as  $\Pi_d$  and a sequence of labels is denoted by  $\Pi_t$ . Note that for the second sequence not all paths are possible since we have to visit all given labels. We thus obtain the likelihood

$$P(\mathcal{D}, \mathcal{T}|\mathbf{w}) = \sum_{\Pi_d, \Pi_t} P(\mathcal{D}, \Pi_d, \Pi_t|\mathbf{w}), \quad (1.22)$$

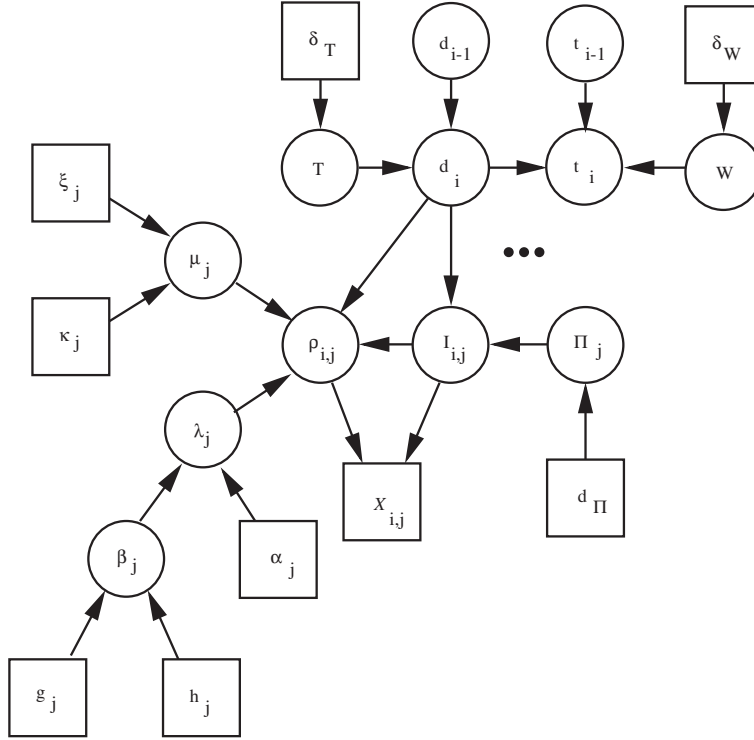
where  $\mathcal{T}$  denotes the observed class labels. If several independent sequences are used to infer model coefficients, the overall likelihood is the product of several expressions like (1.22). In order to obtain a final expression of the likelihood of model coefficients we have to plug Equations (1.19) and (1.20) into Equation (1.22). It is evident that the resulting likelihood is highly nonlinear and parameter inference had to be done by carrying out Metropolis-Hastings updates. The conventional way to maximize such likelihood functions is to apply the expectation maximisation (EM) algorithm [14] which was introduced for HMMs in [15]. The sampling algorithm proposed in the next subsection uses similar ideas. We use both Gibbs updates and Metropolis-Hastings updates to draw samples from the *a-posteriori* distribution over model coefficients and latent variables.

### 1.4.3 An augmented DAG for MCMC sampling

The likelihood function associated with the probabilistic model in Figure 1.3 highly non-linear in the model coefficients and we have to use MCMC methods to infer them. As already indicated, we want to use Gibbs updates wherever possible. Only such variables that do not allow Gibbs moves will be updated with Metropolis-Hastings steps. Following the ideas of the EM procedure, we introduce latent indicator variables,  $d_i$ , which indicate the kernel number of the Gaussian prior over the latent variables  $\rho_{i,j}$ .

Figure 1.4 shows a DAG that results from the DAG in Figure 1.3 when augmented with all coefficients of the probabilistic model and the hyper-parameters of the corresponding priors. In order to keep the graph simple, Figure 1.4 displays only those variables, we need to derive the MCMC updates for inference. In particular we illustrate only one observation model. Other sensors are indicated by dots.

The state variable  $d_i$  is conditionally dependent on its predecessor and on the transition probability  $\mathbf{T}$ . Class labels,  $t_i$ , depend on the state variable and on the preceding label  $t_{i-1}$ . The state conditional transition probabilities are summarized by  $\mathbf{W}$ . For both the transition probabilities,  $\mathbf{T} = P(d_i|d_{i-1}) \forall d_i, d_{i-1}$ , and prior allocation probabilities,  $\mathbf{W} = P(t_i|d_i, t_{i-1}) \forall d_i, t_i, t_{i-1}$ , we use a Dirichlet-prior. Conditional on the latent state  $d_i$ , we have the observation model for the latent variables  $\rho_{i,j}$  and the corresponding model indicator  $I_{i,j}$ . The observation model for  $I_{i,j}$  is a multinomial-one distribution with observation probabilities  $\Pi_j = P(I_{i,j}|d_i) \forall I_{i,j}, d_i$ . These probabilities are given a Dirichlet prior using  $\delta_{\Pi}$  as prior counts. Except that we do not infer the number of kernels, the model for  $\rho_{i,j}$  is largely identical to the model used by [16] for their one dimensional mixture of



**Fig. 1.4.** A DAG for parameter inference. In order to keep it simple, the DAG shows only one of the observation models for  $\rho_{i,j}$ ,  $I_{i,j}$  one pair of latent variables  $d_{i-1}$ ,  $d_i$ , and one pair of class labels,  $t_{i-1}$ ,  $t_i$ . We use three dots to indicate that there could be more sensors. The DAG shows transition probabilities for the states,  $\mathbf{T} = P(d_i|d_{i-1}) \forall d_i, d_{i-1}$  and for the class labels  $\mathbf{W} = P(t_i|d_i, t_{i-1}) \forall d_i, t_i, t_{i-1}$ . The multinomial observation model for the model indicator is specified by  $\mathbf{II}_j = P(I_{i,j}|d_i) \forall I_{i,j}, d_i$ . Finally we have a hierarchical model for specifying the observation model for the lattice filter coefficients specified by  $\mu_j$  and  $\lambda_j$ . As before, square nodes denote observed quantities and circles are latent variables. However there is an exception since during model inference some of the  $t_i$ 's are observed.

Gaussians analysis with varying number of kernels. The means,  $\mu_j$ , are given a normal prior with mean  $\xi_j$  and precision  $\kappa_j$ . Each component has its own precision (inverse variance)  $\lambda_j$ . In order to avoid problems with singular solutions the variances are coupled with hyper-parameters  $\alpha$  and  $\beta_j$ . The latter,  $\beta_j$ , has itself a Gamma prior. This hierarchical prior specification allows for informative priors without introducing large dependencies on the values of the hyper-parameters. The difference between our observation model and the one used in [16] is that  $\rho_{i,j}$  are latent variables, with the time series  $\mathcal{X}_{i,j}$  being conditionally dependent on  $\rho_{i,j}$  and  $I_{i,j}$ .

#### 1.4.4 Specifying priors

In order to be able to derive an MCMC scheme to sample from the *a-posteriori* distributions of model coefficients and latent variables, we need to specify the functional form as well as the parameters of all priors from the DAG in Figure 1.4. Gibbs sampling requires full conditional distributions from which we can draw efficiently. The full conditional distributions are the distributions of model coefficients when conditioning on all other model coefficients, latent variables and data. Apart from the EM-like idea to introduce latent variables, tractable distributions will be obtained by using so-called conjugate priors, as discussed in [17].

In order to allow Gibbs updates for most of the parameters, we use the following prior specification:

- Each component mean,  $\mu_{j,d}$ , is given a Gaussian prior:  $\mu_{j,d} \sim \mathcal{N}_1(\xi_j, \kappa_j^{-1})$ .
- The precision is given a Gamma prior:  $\lambda_{j,d} \sim \Gamma(\alpha, \beta_j)$ .
- The hyper-parameter,  $\beta_j$ , gets a Gamma hyper-prior:  $\beta_j \sim \Gamma(g, h)$ .
- The transition probabilities for the latent kernel indicators,  $\mathbf{T}_d$  get a Dirichlet prior:  $\mathbf{T}_d \sim \mathcal{D}(\delta_T^1, \dots, \delta_T^D)$ , with  $D$  denoting the number of kernels.
- The transition probabilities for the class labels,  $\mathbf{W}_{k,d}$ , get a Dirichlet prior:  $\mathbf{W}_{k,d} \sim \mathcal{D}(\delta_W^1, \dots, \delta_W^K)$ , with  $K$  denoting the number of class labels.
- The observation probabilities of the model indicators,  $\mathbf{\Pi}_d$  get a Dirichlet prior:  $\mathbf{\Pi}_d \sim \mathcal{D}(\delta_\Pi^1, \dots, \delta_\Pi^D)$ , with  $D$  denoting the number of kernels.

The quantitative settings are similar to those used in [16]: Values for  $\alpha$  are between 1 and 2,  $g$  is usually between 0.2 and 1 and  $h$  is typically between  $1/R_{max}^2$  and  $10/R_{max}^2$ , with  $R_{max}$  denoting the largest input range. The mean,  $\mu_j$ , gets a Gaussian prior centred at the midpoint,  $\xi_j$ , with inverse variance  $\kappa_j = 1/R_j^2$ , where  $R_j$  is the range of the  $j$ -th input. The multinomial priors of the prior allocation counts and the prior transition counts are set up with equal probabilities for all counters. We set all prior counts i.e. for  $\delta_T$ ,  $\delta_W$  and  $\delta_\Pi$  to 1, which gives the most uninformative proper prior.

#### 1.4.5 MCMC updates of coefficients and latent variables

The prior specification proposed in the last subsection enables us to use mainly Gibbs updates. The latent variables  $\rho_{i,j}$  however need to be sampled via more general Metropolis-Hastings updates. The main difficulty is that we regard “input” variables as being latent. In other words: conventional approaches condition on the  $\rho_{i,j}$ ’s, whereas our approach regards them as random variables which have to be updated as well.

We will first summarise the Gibbs updates for the model coefficients. The expressions condition on hyper parameters, other model coefficients, hidden states, class labels and the latent representation of preprocessing (i.e.  $\rho_{i,j}$  and  $I_{i,j}$ ). During model inference we need to update all unobserved variables of the DAG, whereas for predictions we update only the variables shown in the DAG in Figure 1.3. The updates for the model coefficients are done using full conditional distributions, which have

the same functional forms as the corresponding priors. These full conditionals follow previous publications [11], with some modifications required by the additional Markov dependency between successive class labels.

### Update of the transition probabilities, $T$

The full conditional of the transition probabilities  $T_{d_i}[d_{i+1}] = P(d_{i+1}|d_i)$  is a Dirichlet distribution. The expression depends on the prior counts,  $\delta_T$ , and on all hidden states,  $d_i$ .

$$T_d \sim \mathcal{D}(\delta_T + n_{d,1}^d, \dots, \delta_T + n_{d,D}^d), \quad (1.23)$$

with  $n_{d,1}^d, \dots, n_{d,D}^d$  denoting the number of transitions from state  $d_i = d$  to  $d_{i+1} = \{1, \dots, D\}$ . The prior probability of the hidden states  $P_T$  is the normalised eigenvector of  $T$  that corresponds to the eigenvalue 1.

### Update of the transition probabilities for class labels, $W$

The full conditional of the transition probabilities,  $W_{d_i, t_{i-1}}[t_i] = P(t_i|d_i, t_{i-1})$  is a Dirichlet distribution. These transition probabilities are conditional on both, the previous class label and the current state. We thus obtain the posterior counts,  $n_{d,t}^\tau$ , by counting transitions from class label  $\tau$  to class label  $t$ , given the current latent state is  $d$ . The transition probability is a Dirichlet distribution.

$$W_d^\tau \sim \mathcal{D}(\delta_W + n_{d,1}^\tau, \dots, \delta_W + n_{d,K}^\tau). \quad (1.24)$$

Conditional on  $d$ , the prior probability of the class labels  $P_{W,d}$  is the normalised eigenvector of  $W_d$  that corresponds to the eigenvalue 1.

### Update of the observation probability of model orders, $\Pi_j$

The full conditional of the observation probabilities of model order  $\Pi_j[d_i, I_{i,j}] = P(I_{i,j}|d_i, j)$  is a Dirichlet distribution. The expression depends on the prior counts,  $\delta_{\Pi}$ , on the model indicators  $I_{i,j}$  and on the state  $d_i$ . Each sensor  $j$  has its own set of probabilities  $P_j$ .

$$\Pi_{j,d} \sim \mathcal{D}(\delta_P + n_{d,0}^{I_j}, \dots, \delta_P + n_{d,I_{\max}}^{I_j}), \quad (1.25)$$

where  $n_{d,I}^{I_j}$  denotes the number of cases, in which  $I_{i,j} \equiv I$  and  $d_i \equiv d$ .

### Updating the Gaussian observation model

We use a separate Gaussian observation model for the latent features  $\rho_{i,j}$  of each sensor  $j$ . Each observation model needs three updates.

The full conditional of the kernel mean,  $\mu_j$ , is a Normal distribution. The expression depends on the hyper parameters  $\kappa_j$  and  $\xi_j$ , on the hidden states  $d_i$ , on the model indicators  $I_{i,j}$ , on the covariance matrix  $\lambda_j$  and on the latent coefficients  $\rho_{i,j}$ .

$$\boldsymbol{\mu}_{j,d}[k] \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_{j,d}[k], \sigma_{j,d}^\mu[k]) \quad (1.26)$$

with

$$\begin{aligned} \hat{\boldsymbol{\mu}}_{j,d}[k] &= (n_{j,d}^\mu[k] \boldsymbol{\lambda}_{j,d}[k, k] + \boldsymbol{\beta}_j[k])^{-1} (n_{j,d}^\mu[k] \boldsymbol{\lambda}_{j,d}[k, k] \bar{\boldsymbol{\rho}}_{d,j}[k] + \kappa_j[k] \boldsymbol{\xi}_j[k]) \\ \sigma_{j,d}^\mu[k] &= (n_{j,d}^\mu[k] \boldsymbol{\lambda}_{j,d}[k, k] + \boldsymbol{\beta}_j[k])^{-1}, \end{aligned}$$

where we use index  $k$  to denote the  $k$ -th dimension of the latent vector  $\boldsymbol{\rho}_{i,j}$ . The dependency of Equation (1.26) on  $I_{i,j}$  is implicit in

$$n_{j,d}^\mu[k] = \sum_{i|I_{i,j} \equiv 1} \delta(d_i \equiv d)$$

and in

$$\bar{\boldsymbol{\rho}}_{d,j}[k] = \frac{1}{n_{j,d}^\mu[k]} \sum_{i|I_{i,j} \equiv 1} \delta(d_i \equiv d) \boldsymbol{\rho}_{i,j}[k].$$

The full conditional of the kernel covariance,  $\boldsymbol{\lambda}_{j,d}[k]$ , is a Gamma distribution. The expression depends on the hyper parameter  $\alpha_j$  and  $\beta_j$ , on the hidden states,  $d_i$ , on the corresponding kernel mean  $\boldsymbol{\mu}_{j,d}[k]$  and on the model indicators  $I_{i,j}$ .

$$\boldsymbol{\lambda}_{j,d}[k] \sim \Gamma(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) \quad (1.27)$$

with

$$\begin{aligned} \hat{\boldsymbol{\alpha}} &= \alpha_j + \frac{n_{j,d}^\mu[k]}{2} \\ \hat{\boldsymbol{\beta}} &= \boldsymbol{\beta}_j[k] + \frac{1}{2} \sum_{i|I_{i,j} \equiv 1} \delta(d_i \equiv d) (\boldsymbol{\rho}_{i,j}[k] - \boldsymbol{\mu}_{j,d}[k])^2 \end{aligned}$$

The full conditional of the hyper parameter  $\boldsymbol{\beta}_j$  depends on the hyper-hyper-parameters  $g_j$  and  $\mathbf{h}_j$ , on  $\alpha_j$  and on  $\boldsymbol{\lambda}_{j,d}$ .

$$w\boldsymbol{\beta}_j[k] \sim \Gamma(g_j + D\alpha_j, \mathbf{h}_j[k] + \sum_d \boldsymbol{\lambda}_{j,d}[k]) \quad (1.28)$$

#### 1.4.6 Gibbs updates for hidden states and class labels

##### Updating $d_i$

The full conditionals for  $d_i$  are multinomial-one distributions. The first state has no preceding state  $d_0$ . We thus use the unconditional prior probability of the state,  $P_T(d_1)$  instead of  $T_{d_0}[d_1]$ .

$$d_i \sim \mathcal{Mn}(1, \{P(d_i|\dots) \forall d_i = 1..D\}) \quad (1.29)$$

with

$$\begin{aligned} P(d_1|\dots) &= \frac{P_T[d_1] T_{d_1}[d_2] P_{W,d_1}[t_1] \prod_j p(\boldsymbol{\rho}_{1,j}, I_{1,j}|d_1)}{\sum_{d_1} (P_T[d_1] T_{d_1}[d_2] P_{W,d_1}[t_1] \prod_j p(\boldsymbol{\rho}_{1,j}, I_{1,j}|d_1))} \\ P(d_{i \neq 1}|\dots) &= \frac{T_{d_{i-1}}[d_i] T_{d_i}[d_{i+1}] W_{d_i, t_{i-1}}[t_i] \prod_j p(\boldsymbol{\rho}_{i,j}, I_{i,j}|d_i)}{\sum_{d_i} (T_{d_{i-1}}[d_i] T_{d_i}[d_{i+1}] W_{d_i, t_{i-1}}[t_i] \prod_j p(\boldsymbol{\rho}_{i,j}, I_{i,j}|d_i))} \end{aligned}$$

For the last state of the Markov chain, the successor  $d_{i+1}$  does not exist and the expression of the probability,  $P(d_{i \neq 1} | \dots)$ , does not include the term  $T_{d_i}[d_{i+1}]$ .

### Updating $t_i$

Unknown class labels,  $t_i$ , are updated by multinomial one distributions.

$$T_i \sim \mathcal{Mn}(1, \{P(t_i | \dots) \forall t_i = 1..K\}) \quad (1.30)$$

with

$$P(t_1 | \dots) = \frac{P_{W, d_1}[t_1] W_{d_2, t_1}[t_2]}{\sum_{t_1} P_{W, d_1}[t_1] W_{d_2, t_1}[t_2]}$$

$$P(t_i \neq 1 | \dots) = \frac{W_{d_i, t_{i-1}}[t_i] W_{d_{i+1}, t_i}[t_{i+1}]}{\sum_{t_i} W_{d_i, t_{i-1}}[t_i] W_{d_{i+1}, t_i}[t_{i+1}]}$$

Since class label  $t_{i+1}$  does not exist for the last segment, we have to remove the term  $W_{d_{i+1}, t_i}[t_{i+1}]$  when expressing the corresponding probability  $P(t_i | \dots)$ .

### 1.4.7 Approximate updates of the latent feature space

We finally have to formulate updates for the latent feature space, i.e. for the variables  $\rho_{i,j}$  and  $I_{i,j}$ . These updates involve the posteriors formulated in Equations (1.10) and (1.12) and are drawn from the ‘‘joint conditional’’ distribution  $p(I_{i,j}, \rho_{i,j} | \dots) \propto p(I_{i,j}, \rho_{i,j} | \mathcal{X}_{i,j}) p(d_i | I_{i,j}, \rho_{i,j})$ . We draw from  $p(I_{i,j}, \rho_{i,j} | \dots)$  by first proposing from  $(I'_{i,j}, \rho'_{i,j}) \sim p(I_{i,j}, \rho_{i,j} | \mathcal{X}_{i,j})$  and then accepting  $(I'_{i,j}, \rho'_{i,j})$  by a Metropolis-Hastings acceptance probability. We draw model indicator  $I'_{i,j}$  from  $p(I_{i,j} | \mathcal{X}_{i,j})$  which is a multinomial-one distribution.

$$I'_{i,j} \sim \mathcal{Mn}(1, \{P(I_{i,j} | \mathcal{X}_{i,j})\}), \quad (1.31)$$

with  $P(I_{i,j} | \mathcal{X}_{i,j})$  defined in Equation (1.10). Using the indicator variable we then propose  $\rho'_{i,j}$ .

$$\rho'_{i,j} \sim \begin{cases} \forall I'_{i,j} \equiv 1 : \rho'_{i,j} \sim p(\rho_{i,j} | \mathcal{X}_{i,j}) \\ \forall I'_{i,j} \equiv 0 : \rho'_{i,j} = \square \end{cases}, \quad (1.32)$$

where  $p(\rho_{i,j} | \mathcal{X}_{i,j})$  is a product of the distributions in Equation (1.12) and the second case denotes the white noise case, in which the latent parameter  $\rho'_{i,j}$  has dimension zero. In order to get a sample from the full conditional distribution, we have to calculate the acceptance probability

$$P_a = \min \left( 1, \frac{P(d_i | \rho'_{i,j}, I'_{i,j})}{P(d_i | \rho_{i,j}, I_{i,j})} \right) \quad (1.33)$$

where

$$P(d_i | \rho'_{i,j}, I'_{i,j}) = \begin{cases} \forall I'_{i,j} \equiv 1 : \frac{p(\rho'_{i,j} | \boldsymbol{\mu}_{j, d_i}, \boldsymbol{\lambda}_{j, d_i}) \Pi_{d_i, j}[I'_{i,j}]}{\sum_{d_i} p(\rho'_{i,j} | \boldsymbol{\mu}_{j, d_i}, \boldsymbol{\lambda}_{j, d_i}) \Pi_{d_i, j}[I'_{i,j}]} \\ \forall I'_{i,j} \equiv 0 : \frac{\Pi_{d_i, j}[I'_{i,j}]}{\sum_{d_i} \Pi_{d_i, j}[I'_{i,j}]} \end{cases}$$

and accept the new values of the latent features  $(I'_{i,j}, \rho'_{i,j})$  according to this probability. We would like to point out that this scheme is an approximation to the exact updates of the latent space since the proposal distributions in Equations (1.10) and (1.12) do not consider the correct prior which would be the mixture distribution  $P(d_i)p(\rho_{i,j}|d_i)p(I_{i,j}|d_i)$ . Instead we use the priors specified in section 1.2. However, we argue that the difference is not large if the number of samples is sufficient since in this case the likelihood by far outweighs the prior.

### 1.4.8 Algorithms

#### Model inference

---

**Algorithm 1** *Pseudo-code for sweeps during model inference.*

---

```

 $\forall d$    initialise( $\mathbf{W}_d, \mathbf{T}_d$ )
 $\forall j$    initialise( $\beta_j$ )
 $\forall j, d$  initialise( $\mu_{j,d}, \lambda_{j,d}, \mathbf{\Pi}_{j,d}$ )
 $\forall i$    initialise( $d_i, t_i$ )           % only missing  $t_i$  are initialised
 $\forall i, j$  initialise( $I_{i,j}, \rho_{i,j}$ )
sweepcounter=0
REPEAT
   $\forall d$    update( $\mathbf{T}_d$ )                 % according to Equation (1.23)
   $\forall d$    update( $\mathbf{W}_d$ )                 % according to Equation (1.24)
   $\forall j, d$  update( $\mathbf{\Pi}_{j,d}$ )             % according to Equation (1.25)
   $\forall j, d$  update( $\mu_{j,d}$ )             % according to Equation (1.26)
   $\forall j, d$  update( $\lambda_{j,d}$ )           % according to Equation (1.27)
   $\forall j$    update( $\beta_j$ )               % according to Equation (1.28)
   $\forall i, j$  ( $\rho'_{i,j}, I'_{i,j}$ )  $\sim p(I_{i,j}, \rho_{i,j} | \mathcal{X}_{i,j})$  % according to Eqns. (1.10) and (1.12)
          accept( $\rho'_{i,j}, I'_{i,j}$ ) with  $P_a$  % according to Equation (1.33)
   $\forall i$    update( $d_i$ )                 % according to Equation (1.29)
   $\forall i$    if  $t_i$  missing
          update( $t_i$ )                 % according to Equation (1.30)
  inc(sweepcounter)
  sample[sweepcounter]= $\{\mathbf{T}_d, \mathbf{W}_d, \mathbf{\Pi}_{j,d}, \mu_{j,d}, \lambda_{j,d} \forall j, d\}$ 
UNTIL (sweepcounter > maxcount)

```

---

For model inference we start the MCMC scheme by drawing initial model parameters from their priors. We initialize all  $(I_{i,j}, \rho_{i,j}) \sim p(I_{i,j}, \rho_{i,j} | \mathcal{X}_{i,j})$  with samples drawn from the posteriors in Equations (1.10) and (1.12) and all unobserved states  $d_i$  are drawn from the prior probabilities  $P(d_i)$ . The class labels of unlabelled segments are drawn from their prior  $P(t_i|d_i)$  as well. After this initialisation we update according to the full conditionals and in the case of feature updates according to the single component Metropolis Hastings step. Pseudo code of these updates is shown in Algorithm 1.

**Predictions**


---

**Algorithm 2** *Pseudo-code for sweeps during predictions.*


---

```

 $\forall i$       initialise( $d_i, t_i$ )
 $\forall i, j$    initialise( $I_{i,j}, \varphi_{i,j}$ )
sweepcounter=0
expcounter=0
REPEAT
  inc(sweepcounter)
   $\{\mathbf{T}_d, \mathbf{W}_d, \mathbf{H}_{j,d}, \boldsymbol{\mu}_{j,d}, \boldsymbol{\lambda}_{j,d} \forall j, d\} = \text{sample}[\text{sweepcounter}]$ 
   $\forall i, j$     $(\boldsymbol{\rho}'_{i,j}, I'_{i,j}) \sim p(I_{i,j}, \boldsymbol{\rho}_{i,j} | \mathcal{X}_{i,j})$  % according to Eqns. (1.10) and (1.12)
           accept( $\boldsymbol{\rho}'_{i,j}, I'_{i,j}$ ) with  $P_\alpha$            % according to Equation (1.33)
   $\forall i$      update( $d_i$ )                               % according to Equation (1.29)
   $\forall i$      update( $t_i$ )                               % according to Equation (1.30)
  if sweepcounter > burnincounter
     $P(t_i) = P(t_i) + 1$ 
    expcounter = expcounter + 1
UNTIL (sweepcounter > maxcount)
 $\forall t_i$      $P(t_i) = P(t_i)/\text{expcounter}$ 

```

---

In order to get consistent estimates of the beliefs of the states, predictions have to be marginalized over  $\rho_{i,j}$  and  $I_{i,j}$ . The integrals need to be solved numerically. We use all samples drawn from the posterior in order to allow the  $\rho_{i,j}$ 's of the test data to converge. All sample expectations are then taken after allowing for a burn in period. Apart from beliefs about states, we can also obtain expectations from all latent variables - most interestingly from the  $I_{i,j}$ 's and  $\rho_{i,j}$ 's.

Predictions are based on an approximation of the *a-posteriori* distribution of class labels,  $t_i$ , latent allocations,  $d_i$  and latent variables,  $\rho_{i,j}$ . We initialize the latent variables  $d_i$  and the class labels  $t_i$  by drawing according to the respective prior probabilities. The initial  $(I'_{i,j}, \boldsymbol{\rho}'_{i,j}) \sim p(I_{i,j}, \boldsymbol{\rho}_{i,j} | \mathcal{X}_{i,j})$  are drawn from the posteriors in Equations (1.10) and (1.12). The coefficients are then updated using full conditional distributions for  $t_i$  and  $d_i$  and the single component Metropolis Hastings step for  $(I_{i,j}, \rho_{i,j})$ . During each round of Algorithm 2, we use the next sample from the Markov chain obtained during parameter inference. We are interested in obtaining the probabilities of class labels  $t_i$  and expected values of the latent variables  $(I_{i,j}, \rho_{i,j})$ . We estimate these quantities by averaging after having allowed for a burn in period.

**Assessing convergence**

Although in theory MCMC algorithms can approximate arbitrary distributions, the difficulty is that there is a random error inherent to the approach. This means that we

need to estimate how many samples from the posterior we need in order to be able to approximate the desired value with sufficient accuracy. Different approaches to obtain such estimates are discussed at length in the MCMC literature (e.g. [18] and [19]). Although all suggested approaches can diagnose non convergence, there is no way that allows to assess convergence with certainty. We thus need to treat quoted numbers with caution. Despite this difficulty, it is important to have at least a rough idea about how many samples to draw. The suggested methods follow two different strategies.

- Along the lines of [20], we can compare the samples obtained from multiple runs.
- As is suggested in [21], we can use one run and apply Markov chain theory to assess how many samples we need to discard at the beginning to get estimates that are independent of the starting value and how many samples we need to converge to a sufficiently accurate result.

Although there are cases where the second case might fail to diagnose slow convergence, [19] point out that the multiple chain approach is less efficient since we have to wait more than once until the Markov chain visits a high probability region. The model proposed in this work has definitely a very complicated structure, such that initial convergence might be slow. We thus apply the method suggested in [21] to the likelihood of the class labels we calculate from the Markov chain. We should point out that we cannot use the model coefficients directly since the mixture model is not identified and label switching might give a wrong sense of the actual mixing. Calculations suggest that we should draw around 15,000 samples which is confirmed by observing virtually no difference in probabilities obtained from repeated runs.

## 1.5 Experiments

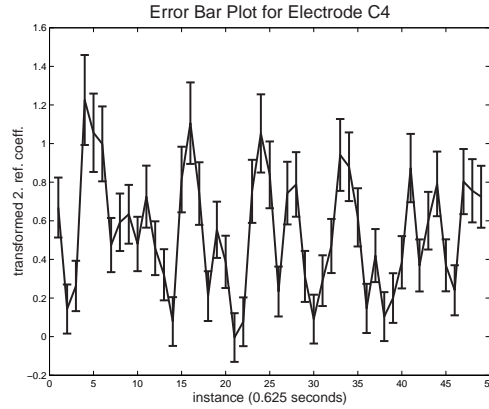
This section evaluates the proposed method on two biomedical classification problems. The first experiment classifies in segments of sleep EEG, whether sleep spindles are present or not. We use this data because it reflects a problem of very unbalanced class labels. For the second experiment we apply the proposed method to classification of cognitive states of segments of EEG recordings. In this case the cognitive experiments have been designed to obtain data with balanced class labels.

### 1.5.1 Data

#### Classification of sleep spindles

Sleep spindles are an important phenomenon in sleep EEG. This experiment uses data recorded from the standard 10-20 electrodes F4, C4 and P4 [22, 23] that were recorded against averaged ear potential. The sampling frequency used here was 102.4 Hz. This data is segmented into segments of 0.625 seconds duration such that we get 16 segments for a 10 second recording. A segment is labelled as containing a spindle, if more than half of the segment shows spindle activity. This setup results in a

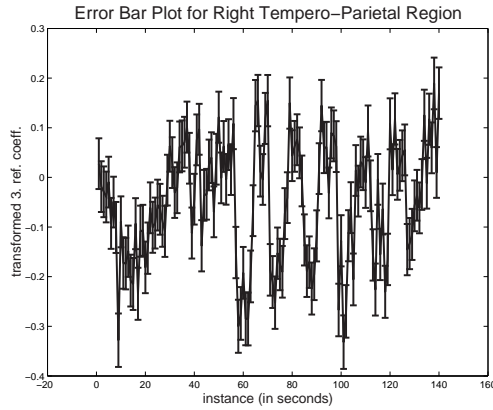
prior probability for “spindle” to be roughly  $P_{\text{spindle}} = 0.15$ . We use data from two different subjects, each containing 7 minutes of sleep EEG that was marked for sleep spindles by a human expert. Figure 1.5 shows the second reflection coefficient and a 1 standard deviation error bar for 30 seconds of data calculated according to Equation (1.12). The classification experiment uses two datasets from different subjects. For every electrode we extract 3 reflection coefficients, including the standard deviation and the probability of the model compared against a white noise explanation. We report within subject results, evaluated on 672 data points using 6 fold cross validation.



**Fig. 1.5.** The transformed reflection coefficient and corresponding 1 standard deviation error bar for 30 seconds of sleep EEG recorded from electrode C4. Positive peaks in the plot correspond to spindle activity.

### Classification of cognitive tasks

The data used in these experiments is EEG recorded from 10 young, healthy and untrained subjects while they perform different cognitive tasks. We classify 2 task pairings: auditory-navigation and left motor-right motor imagination. The recordings were taken from 3 electrode sites: T4, P4 (right temporo-parietal for spatial and auditory tasks), C3' , C3'' (left motor area for right motor imagination) and C4' , C4'' (right motor area for left motor imagination). Note that these electrodes are 3cm anterior and posterior to the electrode positions C3 and C4, as are defined in the classical 10-20 electrode setup [22, 23] The ground electrode was placed just lateral to the left mastoid process. The data were recorded using amplification gain of  $10^4$  and fourth order band pass filter with pass band between 0.1 Hz and 100 Hz. These signals were sampled with 384 Hz and 12 bit resolution. Each cognitive experiment was performed 10 times for 7 seconds. Figure 1.6 shows the second reflection coefficient and a 1 standard deviation error bar for 140 seconds of data from the right



**Fig. 1.6.** Transformed reflection coefficient plus minus one standard deviation for 140 seconds of cognitive EEG recorded in the right tempo-parietal region from electrodes T4 and P4. The rhythmicity in the second half of the experiment corresponds to alternating cognitive states.

tempo-parietal region calculated according to Equation (1.12). We extract for both electrodes 3 reflection coefficients, the standard deviation and the probability of the model compared against a white noise explanation. We report within subject results, evaluated on 140 data points using 5 fold cross validation.

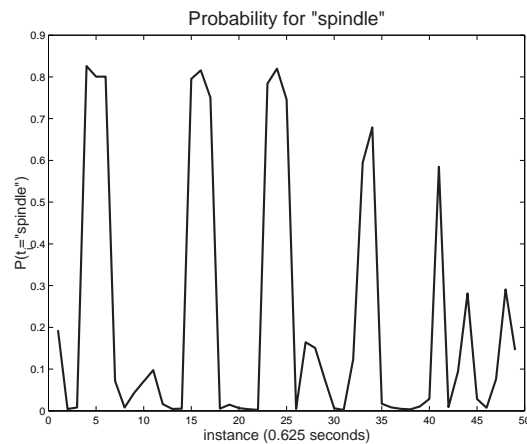
### 1.5.2 Classification Results

The classification results reported in this section have been obtained by drawing 15,000 samples from the posterior distribution of the class labels  $t_i$  in the test data. This allows us to estimate the posterior probability over class labels and hence predict the Bayes optimal class label. Figure 1.8 shows the posterior probabilities for spindle events for the data used above to illustrate preprocessing results. Figure 1.8 shows the probability for cognitive state “auditory” for the data used above to illustrate the preprocessing result. To quantify the results, we obtain independent

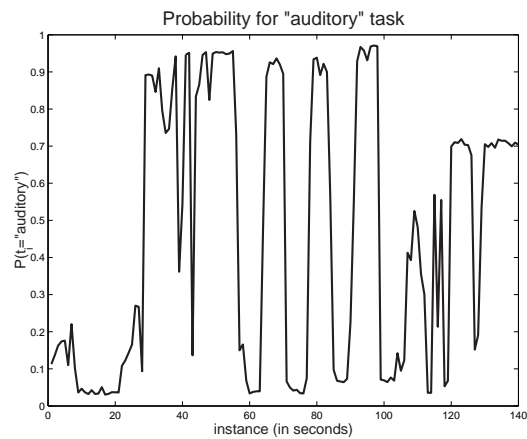
**Table 1.1.** Generalisation accuracies for sleep spindle data and two cognitive experiments

Data	Accuracy
Spindle	93 %
left/right	66 %
aud./navig.	80 %

predictions for all data and calculate the expected generalisation accuracies. For the cognitive experiments this is an average across 10 subjects. The predictions are done on a one second basis to get a situation similar to reality in brain computer interface experiments, where predictions have to be done in real time. The results on spindle



**Fig. 1.7.** Probabilities of “spindle” for the parameters shown in Figure 1.5.

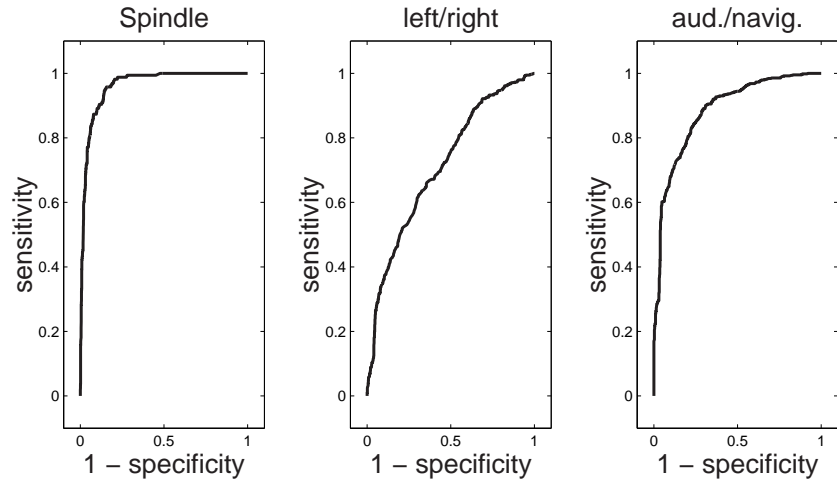


**Fig. 1.8.** Probability for cognitive state “auditory” for the parameters shown in Figure 1.6.

classification are averaged over two subjects. The overall generalisation accuracies are shown in table 1.1. We can also obtain a more general picture by looking at the receiver operator characteristics (ROC) curves shown in Figure 1.9.

## 1.6 Conclusion

Applications of machine learning to biomedical problems often separate feature extraction from modelling the quantity of interest. In a probabilistic sense this must be regarded as approximation to the exact approach, which should model the quantities of interest as *one* joint distribution similar to the DAG in Figure 1.1. To further



**Fig. 1.9.** Receiver operator characteristics (ROC) curve for the three different problems used in this chapter. Plot “spindle” shows the ROC curve for classification of sleep spindles. Plot “left/right” shows the ROC curve for classifying left/right movement imagination. The last plot, “aud./navig.” shows the ROC curve when classifying the auditory versus navigation task.

illustrate this idea, we consider the problem of time series classification which is often found in medical monitoring and diagnosis applications. By time series classification, we mean the problem of classifying subsequent segments of time series data. Examples of that kind are: sleep staging which is usually based on recordings of brain activity (EEG), muscle activity (EMG) and eye movements (EOG); scoring of vigilance, which is again based on EEG; or classifying heart diseases from ECG. The examples used in this chapter to illustrate the proposed approach are classifying of sleep spindles and classifying cognitive activity. The latter is used as part of a brain computer interface. The spatio-temporal nature of these problems is very likely to introduce both spatial and temporal correlations. We thus propose a probabilistic model with dependencies across space and time. For model inference and to obtain probabilities of class labels, we have to solve marginalisation integrals. In this chapter this is done by Markov chain Monte Carlo methods. Algorithms 1 and 2 are approximations of [11], where we use exact inference and allow for different model orders of the latent feature space. The advantage of the approximation proposed here is a significantly reduced computational cost without losing much accuracy.

## Acknowledgements

P. Sykacek is currently funded by the BUPA foundation (Grant No. F46/399). We want to thank our colleagues Prof. Stokes and Dr. Curran, for providing the cognitive data and Prof. Rappelsberger and Dr. Trenker for providing the sleep spindles.

## References

1. G. E. P. Box and G. M. Jenkins. *Time Series Analysis, forecasting and control*. Holden-Day, Oakland, California, 1976.
2. G. L. Bretthorst. Bayesian analysis i: Parameter estimation using quadratur nmr models. *Journal of Magnetic Resonance*, 88:533–551, 1990.
3. J. J. K. Ó Ruanaidh and W. J. Fitzgerald. *Numerical Bayesian Methods Applied to Signal Processing*. Springer-Verlag, New York, 1995.
4. D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.
5. D. J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.
6. R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, New York, 1996.
7. L. Ljung. *System Identification, Theory for the User*. Prentice-Hall, Englewood Cliffs, New Jersey, 1999.
8. W. A. Wright. Bayesian approach to Neural-Network modelling with input uncertainty. *IEEE Trans. Neural Networks*, 10:1261–1270, 1999.
9. P. Dellaportas and S. A. Stephens. Bayesian analysis of errors-in-variables regression models. *Biometrics*, 51:1085–1095, 1995.
10. H. Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, third edition, 1961.
11. P. Sykacek and S. Roberts. Bayesian time series classification. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Processing Systems 14*, pages 937–944. MIT Press, 2002.
12. M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. Dover, New York, 1965.
13. S. Brunak and P. Baldi. *Bioinformatics*. MIT Press, Cambridge, Massachusetts, 1998.
14. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.
15. L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.
16. S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal Royal Stat. Soc. B*, 59:731–792, 1997.
17. J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, Chichester, 1994.
18. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter (ed.). *Markov Chain Monte Carlo in Practice*. Chapman & Hall, London, 1996.
19. C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer, New York, 1999.
20. A. Gelman and D. B. Rubin. Inference from iterative simulations using multiple sequences (with discussion). *Statist. Sci.*, 7:457–511, 1992.
21. A. E. Raftery and S. M. Lewis. Implementing MCMC. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, *Markov Chain Monte Carlo in practice*, pages 115–130. Chapman & Hall, London, Weinheim, New York, 1996.
22. HH. Jasper. Report of the committee on methods of clinical examination in electroencephalography. *Clinical Neurophysiology*, 10:370–1, 1958.
23. HH. Jasper. Appendix to report of the committee on methods of clinical examination in EEG: the ten-twenty electrode system of the International Federation of Electroencephalography. *Clinical Neurophysiology*, 10:371–375, 1958.