

# Towards On-line Hidden Markov Signal Processing

Vikram Krishnamurthy, John B. Moore, Lige Xia

Dept. of Systems Eng., Australian National University, G.P.O. Box 4, Canberra, AUSTRALIA 2601

**ABSTRACT:** A commonly used hidden Markov model signal processing scheme that obtains certain optimal signal statistics and estimates is the forward-backward algorithm. This is a noncausal fixed-interval scheme. Repeated application of this algorithm, along with the Baum Welch re-estimation formulae, allows optimal estimation of the signal model parameters, including signal levels, level transition probabilities, and noise statistics. Here we propose causal schemes with delay that asymptotically achieve signal model identification and optimal signal statistics. The key features of our schemes are sawtooth processing and on-line re-estimation formulae. The intention is to significantly reduce memory requirements, improve computational processing speed, and the adaptive capabilities of hidden Markov model estimation schemes.

## 1 Introduction

An important class of nonlinear signal models is that in which finite-state Markov signals are imbedded in noise. This class is studied in such areas as communication systems, speech processing, biological signal processing and sonar processing. The signal model is characterized in terms of state signal levels, state transition probabilities and noise characteristics. To achieve certain signal statistics necessary for signal model identification, the standard approach is noncausal, involving a forward and backward pass. This constitutes 'fixed-interval smoothing' since the estimates are based on the entire data batch of observations. The so called forward-backward algorithm can be used, along with the Baum Welch [1], [2], [3] re-estimation formulae to achieve improved estimates of the signal model parameters, including signal levels, transition probabilities and noise statistics [5], [4]. Repeated application of the forward-backward algorithm and the Baum Welch re-estimation formulae achieves maximum a posteriori probability (MAP) or conditional mean (CM) estimates [5], [6], [4].

A limitation of the above optimal methods is the 'curse of dimensionality' which arises because the required computational effort, speed requirements are in proportion to the square of the number of states of the finite Markov model. Memory requirements are also proportional to the length of data being processed.

A technique termed 'overlapping' used in [5] can be used to reduce memory requirements. 'Overlapping' is based on the fact that estimates at each time instant  $k$  are influenced at an 'exponentially' decaying rate by past and future data estimates. This 'overlapping' technique has relevance in on-line processing.

In this paper a number of approaches towards on-line processing are explored. In the most attractive form from a computational point of view, the overlapping approach is taken to its extreme with the objective of achieving on-line

estimation. This suboptimal 'overlapping fixed-interval' scheme can also be viewed as an on-line recursive *sawtooth-lag smoothing* scheme with lag  $\Delta$  varying in a sawtooth fashion between  $\Delta_{\min}$  and  $\Delta_{\max}$ .

We also propose an alternative to the Baum Welch re-estimation formulae. The alternative is for on-line re-estimation, and is expressed in terms of sawtooth-lag and fixed-lag smoothed estimates. Whereas Baum Welch re-estimation is done on a block of data (typically a few thousand observations) and uses fixed-interval estimates, our *on-line recursive estimation formulae* are designed to be implemented recursively, thereby achieving on-line learning. When used for processing a fixed batch of data by repeated passes, the number of passes required to learn the signal model, including signal levels and statistics is reduced by up to an order of magnitude over that for the standard scheme.

An important feature of Hidden Markov Model (HMM) learning schemes is their ability to learn the unknown discrete signal levels. Usually, a fine quantization and hence an overparametrized initial signal model is used over the range of possible signal levels. Once learning of this overparametrized model has been achieved, histograms and/or the transition probability matrix can be used to detect the unknown discrete signal levels. We show in a simulation example that the proposed on-line scheme speeds up level-learning by upto an order of magnitude compared to standard HMM processing.

## 2 Hidden Markov Model Signal Processing

In this section we describe the signal model, define learning and estimation objectives and review the standard off-line forward-backwards algorithm.

### 2.1 Signal Model

The underlying dynamics are assumed to be governed by a finite-state, discrete-time first order Markov process. Consequently, the state  $s_k$  at time  $k$  is one of a finite number  $N$  of states  $q_1, q_2, \dots, q_N$  and the probability of being in state  $s_{k+1}$  at time  $k+1$ , given knowledge of states up to time  $k$ , depends only on state  $s_k$  at time  $k$ . That is, defining  $\mathbf{S}_k = s_1, \dots, s_k$ , then  $P(s_{k+1}|\mathbf{S}_k) = P(s_{k+1}|s_k)$ . The state transition probabilities denoted as  $a_{ij} = P(s_{k+1} = q_j | s_k = q_i)$  form a state transition probability matrix  $\mathbf{A} = (a_{ij})$ . These probabilities are assumed to be invariant of  $k$ .

To make a connection with the theory of HMMs, we assume that the Markov process  $s_k$  is hidden, that is indirectly observed by measurements  $y_k$ . We denote the sequence  $y_1, y_2, \dots, y_k$  by  $Y_k$ . The vector of probability functions  $\mathbf{b}(\cdot) = (b_i(\cdot))$  where  $b_i(y_k) = P(y_k | s_k = q_i)$  are assumed invariant of  $k$ , with an independence property  $P(y_k | s_k = q_i, s_{k-1} = q_j, Y_{k-1}) = P(y_k | s_k = q_i)$ . Also we assume that the initial state probability vector  $\underline{\pi} = (\pi_i)$

is defined from  $\pi_i = P(s_1 = q_i)$ . The HMM is denoted  $\lambda = (\mathbf{A}, \mathbf{b}(\cdot), \underline{x})$ .

A special case of interest is when the measurements  $y_k$  consist of a signal corrupted by zero mean, normally distributed white noise as follows:

$$y_k = s_k + w_k, \quad w_k \sim N[0, \sigma_w^2].$$

Then  $b_i(y_k) = (\sqrt{2\pi}\sigma_w)^{-1} \exp(-|y_k - q_i|^2 / (2\sigma_w^2))$ . Notice that the independence assumption on the model reflects itself here as an independence 'whiteness' assumption on  $w_k$ . Our simulations presented subsequently deal with this special case. The theory and algorithms to follow, however, do not rely on this additive normally distributed noise assumption.

## 2.2 Learning and Estimation Objectives

Given the signal model as described above and given observations  $y_1, y_2, \dots, y_k$  denoted  $\mathbf{Y}_k$ , there are four inter-related problems which can be solved [6], [4]. In this paper we provide suboptimal on-line solutions to these problems.

1. Evaluate the likelihood of a given model  $\lambda_i$  generating the given sequence of data, denoted  $P(\mathbf{Y}_k | \lambda_i)$ . This allows comparison of a set of models  $\{\lambda_i\}$  to select the most likely, given the observations.

2. Estimate signal statistics such as a posteriori probabilities for an assumed but not necessarily true model  $\lambda$  and data sequence. For *off-line* processing for a fixed interval  $T$ , estimate and calculate fixed-interval histograms

$$\underline{\gamma}_{k|T} = (\gamma_{k|T}(i)); \quad \gamma_{k|T}(i) \triangleq P(s_k = q_i | \mathbf{Y}_T, \lambda). \quad (2.1)$$

$$\mathbf{h}_{T|T} = (h_{T|T}(i)); \quad h_{T|T}(i) = P(q_i | \mathbf{Y}_T, \lambda) = \frac{1}{T} \sum_{k=1}^T \gamma_{k|T}(i) \quad (2.2)$$

For *on-line* processing with a fixed or sawtooth-lag  $\Delta$ , estimate signal probabilities and calculate histograms

$$\underline{\gamma}_{k|k+\Delta} = (\gamma_{k|k+\Delta}(i)); \quad \gamma_{k|k+\Delta}(i) \triangleq P(s_k = q_i | \mathbf{Y}_{k+\Delta}, \lambda) \quad (2.3)$$

$$\mathbf{h}_{\Delta|T} = (h_{\Delta|T}(i)); \quad h_{\Delta|T}(i) = \frac{1}{T-\Delta} \sum_{k=1}^{T-\Delta} \gamma_{k|k+\Delta}(i) \quad (2.4)$$

From  $\gamma_{k|k+\Delta}(i)$ , state sequence estimates conditioned on  $\lambda$ , denoted  $\{\hat{s}_k | \lambda\}$ , such as maximum a posteriori (MAP) estimate can be generated as

$$\hat{s}_k^{\text{MAP}} = q_j \quad \text{where } j = \arg \max_{1 \leq i \leq N} \gamma_{k|k+\Delta}(i) \quad (2.5)$$

3. Processing of the observations based on a model assumption  $\lambda$ , not necessarily the true model, and adjustment (re-estimation) of the model parameters (functions)  $(\mathbf{A}, \mathbf{b}(\cdot), \underline{x})$ , such that the updated model  $\bar{\lambda} = (\bar{\mathbf{A}}, \bar{\mathbf{b}}(\cdot), \bar{\underline{x}})$  increases the likelihood function, i.e., the probability of the observation sequence given the model. This reestimation is repeated until convergence to the most likely model  $\lambda^{ML}$  amongst the set  $\lambda = (\mathbf{A}, \mathbf{b}(\cdot), \underline{x})$ , given the observations.

4. Find the number of states  $N$ , and learn the actual levels of these states  $q_i$ ,  $i = 1, \dots, N$ , given the observations. Since the finite state levels  $q_i$  are unknown, this *level learning* requires an initial plant model with a very fine quantization over the range of possible state levels, i.e., with

the number of possible states allowed much larger than the actual number.

Optimal schemes for solving the first three problems using off-line processing are presented in [4]. In Sections 3 and 4 we present suboptimal on-line solutions to these problems.

**Review of Standard Forward-backward Procedure**  
Consider an observation sequence  $\mathbf{Y}_T$  of length  $T$  and assumed signal generating model  $\lambda$ . Forward and backward vector variables are defined as probabilities conditioned on the model  $\lambda$ , for  $i = 1, 2, \dots, N$  as follows:

$$\begin{aligned} \underline{\alpha}_k &= (\alpha_k(i)); \quad \alpha_k(i) \triangleq P(\mathbf{Y}_k, s_k = q_i | \lambda) \\ \underline{\beta}_k &= (\beta_k(i)); \quad \beta_k(i) \triangleq P(\bar{\mathbf{Y}}_k | s_k = q_i, \lambda) \end{aligned} \quad (2.6)$$

where  $\bar{\mathbf{Y}}_k$  denotes the future sequence  $y_{k+1}, y_{k+2}, \dots, y_T$  as compared to  $\mathbf{Y}_k$  which denotes the past and present data. Recursive formulae for  $\underline{\alpha}_k$  and  $\underline{\beta}_k$  are readily calculated [5], [4] as

$$\begin{aligned} \alpha_k(j) &= \left( \sum_{i=1}^N \alpha_{k-1}(i) a_{ij} \right) b_j(y_k), \quad \alpha_1(j) = \pi_j b_j(y_1) \\ \beta_k(i) &= \sum_{j=1}^N a_{ij} b_j(y_{k+1}) \beta_{k+1}(j), \quad \beta_T(i) = 1. \end{aligned} \quad (2.7)$$

The optimal *a posteriori* probabilities associated with Problem 2 are given from [5], [4]

$$\underline{\gamma}_k = (\gamma_k(i)); \quad \gamma_k(i) = \frac{\alpha_k(i) \beta_k(i)}{\sum_{i=1}^N \alpha_k(i) \beta_k(i)}, \quad i = 1, 2, \dots, N. \quad (2.8)$$

Notice that calculating  $\underline{\gamma}_k$  involves first computing and storing the  $T$  vectors  $\underline{\beta}_1$  to  $\underline{\beta}_T$ . Then as  $\underline{\alpha}_k$ ,  $1 \leq k \leq T$  is calculated,  $\underline{\gamma}_k$  can also be updated.

## 3 On-line Estimation

We present fixed-lag and sawtooth-lag smoothing schemes based on the standard backward recursions reviewed in the previous section. We seek to significantly decrease the memory requirement of storing  $T$  vectors to compute  $\underline{\gamma}$  in the standard scheme. It turns out that the fixed-lag recursions are less attractive to implement than the sawtooth lag ones. However, the former are included for completeness and to lead into the sawtooth-lag schemes.

### 3.1 Fixed-lag Estimation

For a fixed-lag of  $\Delta$ , let us define the backward vector variable

$$\underline{\beta}_{k|k+\Delta} = (\beta_{k|k+\Delta}(i)); \quad \beta_{k|k+\Delta}(i) = P(\bar{\mathbf{Y}}_{k,k+\Delta} | s_k = q_i, \lambda) \quad (3.1)$$

where  $\bar{\mathbf{Y}}_{k,k+\Delta}$   $y_{k+1}, y_{k+2}, \dots, y_{k+\Delta}$  and  $2 \leq \Delta$  and  $k + \Delta \leq T$ . Thus  $\mathbf{Y}_k$  and  $\bar{\mathbf{Y}}_{k,k+\Delta}$  together form the  $k + \Delta$  length sequence  $\mathbf{Y}_{k+\Delta}$ . Of course  $\underline{\beta}_{k|T} = \underline{\beta}_k$  and  $\bar{\mathbf{Y}}_{k,T} = \bar{\mathbf{Y}}_k$ .

It turns out that the most attractive method for evaluating  $\underline{\beta}_{k|k+\Delta}$  appears to be to compute it directly at each instant  $k$ . This requires of the order of  $\Delta N^2$  multiplications and additions at each  $k$  and is as follows.

**Lemma 3.1:** Let  $\text{diag}(b(y_{k+1}))$  be a diagonal matrix with elements  $\text{diag}(b_i(y_{k+1}))$ ,  $i = 1, \dots, N$ . Then

$$\underline{\beta}_{k|k+\Delta} = \text{rowsum}\{\mathbf{A} \text{diag}(b(y_{k+1})) \cdots \mathbf{A} \text{diag}(b(y_{k+\Delta}))\} \quad (3.2)$$

where  $\mathbf{A}$  is the transition probability matrix.

Lemma 3.1 implies that for the fixed-lag case  $\underline{\beta}_{k|k+\Delta}$  can be computed forwards. This has two main advantages: First, we need not even store  $\Delta$  values of  $\underline{\beta}$  as in the sawtooth-lag smoother but just one to calculate  $\underline{\gamma}_{k|k+\Delta}$ . Second, if we can update estimates of  $\mathbf{A}$  at each iteration (we give algorithms for this in Section 4) then we can use the updated  $\mathbf{A}$  at each time instant  $k$  to calculate  $\underline{\alpha}_k$  and  $\underline{\beta}_{k|k+\Delta}$  leading to faster convergence in the solutions of Problems 2 and 3.

### 3.2 Sawtooth-lag Estimation

Let us take the overlapping approach mentioned in the introduction to its extreme. In the sawtooth-lag scheme we work with a lag varying from a fixed lag  $\Delta_{\min}$  to another of  $\Delta_{\max} \geq 2\Delta_{\min}$  maintaining an average lag of  $(\Delta_{\min} + \Delta_{\max})/2$ . Let us consider a subinterval  $[k_1, k_1 + \Delta_{\min}]$  with  $k_1 + \Delta_{\max} \leq T$  and define a sawtooth lag  $\Delta$  at time  $k$  as

$$\Delta \triangleq k_1 + \Delta_{\max} - k, \quad k_1 \leq k \leq k_1 + \Delta_{\min}. \quad (3.3)$$

Also define the backward vector variable from (3.1) with  $\Delta$ , now  $k$  dependent, as in (3.3)

$$\underline{\beta}_{k|k+\Delta} = (\beta_{k|k+\Delta}(i)); \quad \beta_{k|k+\Delta}(i) = P(\bar{\mathbf{Y}}_{k,k+\Delta} | s_k = q_i, \lambda). \quad (3.4)$$

$\bar{\mathbf{Y}}_{k,k+\Delta}$  denotes the observations  $y_{k+1}, y_{k+2}, \dots, y_{k+\Delta}$  where  $k+\Delta = k_1 + \Delta_{\max}$  is fixed for each interval  $[k_1, k_1 + \Delta_{\max}]$ . Then using the same derivation approach as for (2.7), we have

$$\beta_{k|k+\Delta}(i) = \sum_{j=1}^N a_{ij} b_j(y_{k+1}) \beta_{k+1|k+\Delta}(j), \quad \beta_{k+\Delta|k+\Delta}(i) = 1 \quad (3.5)$$

To calculate  $\underline{\gamma}_{k|k+\Delta}$  we first compute  $\underline{\beta}_{k|k+\Delta}$  for  $k_1 \leq k \leq k_1 + \Delta$  using (3.5) and store for  $k_1 \leq k \leq k_1 + \Delta_{\min}$ . Then for the range  $[k_1, k_1 + \Delta_{\min}]$  of  $k$ , we compute  $\underline{\alpha}_k$  using (2.7) and update  $\underline{\gamma}_{k|k+\Delta}$ .

*Remark:* The sawtooth-lag scheme is computationally attractive because at each time instant  $k$ , the computations involved in calculating  $\underline{\beta}_{k|k+\Delta}$  are of order  $O(N^2)$ . In contrast, the fixed-lag scheme requires  $O(\Delta N^2)$  computations.

## 4 Parameter Re-estimation

We present on-line re-estimation formulae for updating signal models in terms of sawtooth-lag and fixed-lag smoothed estimates. For a rigorous mathematical justification, see [7].

### 4.1 On-line Re-estimation formulae

For a fixed or sawtooth lag  $\Delta$  and given an assumed but not necessarily the true model  $\lambda$ , define the joint conditional probabilities

$$\zeta_{k|k+\Delta}(i, j) = P(s_t = q_i, s_{t+1} = q_j, \mathbf{Y}_{k+\Delta} | \lambda). \quad (4.1)$$

This is calculated to be

$$\zeta_{k|k+\Delta}(i, j) = \alpha_k(i) a_{ij} b_j(y_{k+1}) \beta_{k+1|k+\Delta}(j). \quad (4.2)$$

We introduce the following on-line re-estimation formulae for  $\hat{\mathbf{A}}, \hat{\pi}$  at time  $k$ :

$$\hat{\mathbf{A}}_k = (\hat{a}_{ij}(k)); \quad \hat{a}_{ij}(k) = \frac{\zeta_{k|k+\Delta}(i, j)}{\sum_{j=1}^N \zeta_{k|k+\Delta}(i, j)} \quad (4.3)$$

$$\hat{\pi} = (\hat{\pi}_i); \quad \hat{\pi}_i = \frac{\sum_{j=1}^N \zeta_{i|1+\Delta}(i, j)}{\sum_{i=1}^N \sum_{j=1}^N \zeta_{i|1+\Delta}(i, j)} = \gamma_{i|1+\Delta}(i) \quad (4.4)$$

where  $\hat{a}_{ij}(k) = P(s_{k+1} = q_j | s_k = q_i, \bar{\mathbf{Y}}_{k,k+\Delta})$  and  $\hat{\pi}_i = P(s_1 = q_i | \mathbf{Y}_{\Delta+1})$ . Also, update the vector of probability functions  $\hat{\mathbf{b}}(\cdot)$ , at a finite number of points  $v_1, v_2, \dots, v_M$  in the range of the signals  $y_k$ . Quantizing  $y_k$  to these levels, gives a quantized signal  $\bar{y}_k$  and we propose re-estimating  $b_i(v_j)$  at time  $k$  as

$$\hat{b}(k, v_j) = (\hat{b}_i(k, v_j)); \quad \hat{b}_i(k, v_j) = \frac{\gamma_{k+1|k+\Delta}(i) \delta(v_j - \bar{y}_{k+1})}{\gamma_{k+1|k+\Delta}} \quad (4.5)$$

where  $\hat{b}_i(k, v_j) = P(v_j \text{ at } k+1 | s_k = q_i, \bar{\mathbf{Y}}_{k,k+\Delta})$ . Equations (4.3) to (4.5) give an updated model  $\hat{\lambda}_k = (\hat{\mathbf{A}}_k, \hat{\mathbf{b}}(k, \cdot), \hat{\pi})$  for each  $k$  and constitute the essential ingredients of our on-line re-estimation formulae.

With  $\hat{a}_{ij}(k)$ ,  $\hat{b}_i(k, v_j)$  defined in (4.3) and (4.5) respectively, the re-estimation formulae (4.3) and (4.5) are valid only when  $s_k = q_i$ . In the actual implementation, it makes sense to update (4.3), (4.5) only for  $i : \hat{s}_k^{\text{MAP}} = q_i$ , since from (2.5),  $\hat{s}_k^{\text{MAP}}$  is the best available estimate of the state at time  $k$ . The estimates  $\hat{a}_{ij}$  of states  $i$  not included in the update are set to zero.

We propose two ways of implementing on-line re-estimation.

1. **Batch update:** Here, as in the standard Baum Welch scheme, we propose to update the parameter estimates at the end of each pass as

$$\bar{a}_{ij} = \frac{1}{T_i} \sum_{k=1}^T \hat{a}_{ij}(k) \quad (4.6)$$

where  $T_i$  is the number of  $\hat{a}_{ij}(k) \neq 0$  for  $1 \leq k \leq T$ . Formulae for re-estimating  $\hat{\mathbf{b}}(\cdot)$  follow similarly. The update of  $\pi$  as  $\hat{\pi}$  is done trivially via (4.4). Thus we have an updated model  $\bar{\lambda} = (\bar{\mathbf{A}}, \bar{\mathbf{b}}(\cdot), \bar{\pi})$  which is an average of the models  $\hat{\lambda}_k$ .

2. **Recursive update:** Update the parameters at each time instant  $k$ . For re-estimation with exponential forgetting factor we propose the following:

$$\bar{a}_{ij}(k) = \left(1 - \frac{1}{K}\right) \bar{a}_{ij}(k-1) + \frac{1}{K} \hat{a}_{ij}(k). \quad (4.7)$$

Here the gain  $K$  (typically a few hundredths of the entire data length) is chosen depending on the forgetting factor required. Updates for  $\bar{\mathbf{b}}(k, \cdot)$  are obtained similarly. Observe that  $\bar{\lambda}_k$  is an exponentially weighted average of  $\hat{\lambda}_1$  to  $\hat{\lambda}_k$ .

*Remark:* The on-line re-estimation scheme represents a major departure from the Baum Welch re-estimation formulae. Re-estimation with a forgetting factor exploits the most recent data to upgrade the signal model on line. Thus using an 'out-of-date' model in calculations on subsequent data, which can occur in batch updating, is avoided. The recursive update is especially useful for tracking slowly varying parameters such as transition probabilities.  $\circ$

## 4.2 Simulation studies

To illustrate the advantages of our various on-line schemes we present simulation studies. In particular, we work with a Markov chain of 20000 points with transition probabilities of  $a_{ii} = 0.97$ ,  $i = 1, \dots, 4$  and  $a_{ij} = 0.01$ ,  $i \neq j$ . However, based on the statistics of the generated Markov chain over 20000 points, the actual transition probabilities  $[a_{11}, a_{22}, a_{33}, a_{44}]$  were  $[0.968, 0.972, 0.970, 0.967]$ . To this Markov chain was added unit variance, ( $\sigma_w = 1$ ) zero mean Gaussian white noise. The states  $q_i$  are separated by intervals of  $\sigma_w$ . Initial transition probability estimates are taken as  $a_{ii} = 0.5$ ,  $i = 1, \dots, 4$  and  $a_{ij} = 0.5/3$ ,  $i \neq j$ .

**1. On-line Parameter Re-estimation:** We now illustrate the convergence behaviour of the model estimates  $\bar{\lambda}$  using on-line re-estimation approaches.

*Batch update:* Fig.(4.1) shows the convergence rates of our on-line re-estimation scheme with batch updates for various fixed lags  $\Delta$ . The number of passes for the difference in estimates of  $\bar{a}_{ii}$  to be less than 0.001 between two consecutive passes are plotted versus the fixed lag.

Since the larger  $\Delta$  is, the faster the re-estimation convergence, in the sawtooth-lag scheme it is computationally more efficient to use a larger  $\Delta$  as the computations required are independent of  $\Delta$ . However, there is no improvement in convergence rates for  $\Delta > 20$  in our example.

Fig.(4.2) shows the 'exponential' convergence behaviour of the transition probabilities  $\bar{a}_{ii}$  for  $\Delta = 20$  as the number of passes increase. Here we used  $\eta_i = \hat{s}_k^{\text{MAP}}$  of (2.5), the best available estimate of the state at time  $k$  to compute  $\bar{a}_{ij}(k)$ . The estimates converge to  $[\bar{a}_{11}, \bar{a}_{22}, \bar{a}_{33}, \bar{a}_{44}] = [0.974, 0.982, 0.980, 0.970]$ .

*Recursive update:* The same data was processed by the on-line recursive scheme (4.7) with gain  $K = 50$  and sawtooth lag between  $\Delta_{\min} = 5$  and  $\Delta_{\max} = 10$ . Only about 2500 iterations are required for  $\bar{a}_{ii}(k)$  to be greater than 0.9.

The gain  $K$  affects the learning rate and forgetting factor. Choosing  $K$  large results in slow learning but the resulting estimates yield a better picture of the actual (global) signal model. Simulation studies recommend starting with  $K$  small for fast learning and increasing  $K$  in successive passes.

**2. On-line Fixed-Lag and Sawtooth-Lag Smoothing Signal Extraction:** Our on-line estimation algorithm assumes no knowledge of  $s_k$  or  $a_{ij}$ , but assumes that  $s_k$  belongs to a discrete set separated by  $\sigma_w$ , with  $\sigma_w$  known. Thus for simplicity in the first instance, the signal generating system belongs to the model set.

In Fig.(4.3), we compare the convergence rates of the batch and recursive updates for a sawtooth lag varying from  $\Delta_{\min} = 10$  to  $\Delta_{\max} = 20$ . Notice in Fig.(4.3) that using one recursive pass initially and then using batch updates yields fast convergence and quite acceptable results.

## 5 Level learning and Restricted Search Recursions

Here we address Problem 4 of Section 2 and describe novel implementation techniques that can reduce computations by an order of magnitude.

### 5.1 Level learning

We aim to find the true number of states and learn the actual levels of these states given the observations. Since the finite state levels  $q_i$  are unknown, relatively fine discrete-state level intervals are assumed a priori by the algorithm. That is, we assume an initial overparametrized plant model

with the number of states much larger than the true number of states. Histograms can be estimated from which it may be possible to determine discrete-state levels, at least if these are suitably spaced.

Level learning is implemented by updating the model in successive passes as described in the previous section and computing histograms for each pass using (2.4). Define the set  $S_h \triangleq \{i : h_{\Delta|T}(i) \text{ is a local maximum}\}$ . Then as  $\lambda$  approaches its optimum value, it makes sense to use the elements of  $S_h$ , which are simply the positions of the peaks in the histogram  $\hat{h}_{\Delta|T}$ , to indicate the active state levels. If the active levels are closely spaced, then as discussed below, resolution problems arise with this simple approach. Also the number of elements in  $S_h$  indicates the number of true states.

This process is now illustrated by an example.

### Simulation studies:

Level learning is illustrated for a known first-order finite-state Markov signal in noise  $N[0, \sigma_w]$  with the discrete-state intervals of  $0.8\sigma_w$ ,  $0.6\sigma_w$  and  $0.4\sigma_w$ , and with  $a_{ii} = 0.98$  for all  $i$ , and  $a_{ij} = 0.02/3$  for all  $i \neq j$ .

Our level learning scheme assumes quantization level intervals of  $0.1\sigma_w$ . We used recursive updates and a sawtooth lag varying from  $\Delta_{\min} = 5$  to  $\Delta_{\max} = 10$ . Initial transition probability estimates  $a_{ii}$  were taken as 0.8 for all  $i$ . The evolution of the histograms for 40000 points is illustrated in Fig.(5.1). Studying Fig.(5.1), it is seen that the effective noise variance decreases as the number of passes increases. Of course, the more data processed the greater the potential for improvement of the signal-to-noise ratio due to processing. On a finite data batch, there is clearly a resolution limit in detecting quantization levels. Here we see from Fig.(5.1) that the resolution is better than  $0.4\sigma_w$  for 3 passes. Fig.(5.2) shows the evolution of the transition probability estimates  $\bar{a}_{ii}$  (40000) for the four active states.

We compared our level learning program with that in [5] which uses the standard Baum Welch re-estimation formulae. Fig.(5.3) shows the evolution of the histogram using the standard Baum Welch scheme on the same data. Note that more than 15 passes are required to differentiate between the states separated by  $0.4\sigma_w$ . In comparison our scheme learns the four levels in just 2 passes. This illustrates the attractive convergence speed of our recursive on-line re-estimation formulae.

Once the discrete levels have been estimated, it makes sense to reprocess the data, assuming only these levels, in order to obtain a signal estimate. In this way, there will be less noise in the signal estimate.

To compare resolution of level learning, particularly in processing more realistic data when the assumption of precisely located discrete-state levels does not necessarily hold, a useful technique may be to apply a Gaussian-sum fit to the histogram. Working with reasonable variances, the means in the Gaussian-sum would indicate the discrete-state levels. One would expect that the higher the occupancy rate of a level, the less the variance of uncertainty for the Gaussian term in the sum, representing this level.

### 5.2 Restricted Search Estimation Recursions

The computational effort of the forward-backward scheme and sawtooth-lag versions is proportional to  $N^2 T$ . It makes sense then to consider a variation of the algorithm which avoids calculation of very low probability transitions, given the observations. This can be achieved by considering only significant state transitions. An obvious simplification is to

consider transitions only in the vicinity of the signal, such as within the range  $[y_k - 4\sigma_w, y_k + 4\sigma_w]$ . With say  $N_1$  states in this range, the computational effort is of order  $N_1^2 T$ .

Less obviously, restrictions can be introduced in calculating  $\underline{\alpha}_k$  from (2.7) and  $\underline{\beta}_{k|(\cdot)}$ . Here  $\underline{\beta}_{k|(\cdot)}$  denotes the fixed-interval estimates from (2.7), the sawtooth-lag estimates from (3.2) or the fixed-lag estimates from (3.5). Thus instead of summing over  $i = 1, 2, \dots, N$  as in (2.7) and (3.5), sum over the limited range where  $\alpha_k(i)$  and  $\beta_{k+1|(\cdot)}(i)$  are not negligible as described now.

Restricted search recursions are carried out in a two step procedure.

Step 1: Use a coarse quantization for the possible state levels with the dimension of  $\bar{\mathbf{A}}$  relatively small. Typically, for level learning, it is reasonable to assume the possible state levels are separated by  $0.5\sigma_w$ . The objective of the coarse scheme is to determine the range of significant  $\underline{\alpha}$  and  $\underline{\beta}$  at each instant  $k$ . Thus run successive coarse passes using the on-line re-estimation formulae to update the coarse plant model. On the final coarse pass, store four indices at each instant  $k$ . Two indices,  $\alpha_{min}(k)$  and  $\alpha_{max}(k)$  are used to store the range of indices of elements of  $\underline{\alpha}_k$  when these elements are greater than say 0.1 times the maximum element of  $\underline{\alpha}_k$ . Likewise for  $\underline{\beta}_{k|(\cdot)}$  we store  $\beta_{min}(k)$  and  $\beta_{max}(k)$ . So at the end of the final coarse pass we effectively have an  $\alpha$  'river', which is the set of doublets  $\{\alpha_{min}(k), \alpha_{max}(k)\}$ ,  $1 \leq k \leq T$  and a  $\beta$  'river', which is the set of doublets  $\{\beta_{min}(k), \beta_{max}(k)\}$ ,  $1 \leq k \leq T$ .

Step 2: Use a fine quantization of the possible state levels with the dimension of  $\bar{\mathbf{A}}$  large. Typically for level learning, a quantization resolution of  $0.1\sigma_w$  is used. Then use the  $\alpha$  and  $\beta$  'rivers' to avoid calculating negligible terms. The restricted search recursions corresponding to (2.7) and the sawtooth lag equation (3.5) are

$$\alpha_k(j) = \left( \sum_{i=\alpha_{min}(k)}^{\alpha_{max}(k)} \alpha_{k-1}(i) a_{ij} \right) b_j(y_k),$$

$$\alpha_1(j) = \pi_j b_j(y_1)$$

$$\beta_{k|k+\Delta}(i) = \sum_{j=\beta_{min}(k)}^{\beta_{max}(k)} a_{ij} b_j(y_{k+1}) \beta_{k+1|k+\Delta}(j),$$

$$\beta_{k+\Delta|k+\Delta}(i) = 1 \quad (5.1)$$

## 6 Conclusions

In this paper we have proposed robust, memory efficient, fixed-lag and sawtooth-lag smoothing schemes. Moreover, we have proposed and validated on-line variations of the Baum Welch re-estimation formulae based on fixed-lag and sawtooth-lag smoothed estimates. Simulations confirm that significant improvements in convergence rates for model learning when these on-line variations are used.

## References

- [1] L.E. Baum and T. Petrie, *Statistical inference for probabilistic functions of finite state Markov chains*, Ann.Math.Stat., Vol.37, pp 1554-1563, 1966.
- [2] L.E. Baum, T. Petrie, G. Soules and N. Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Ann.Math.Stat., Vol.41, No.1, pp 164-171, 1970.

- [3] L.E. Baum, *An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes*, Inequalities, Vol.3, pp 1-8, 1972.
- [4] L.R. Rabiner, *A tutorial on Hidden Markov Models and selected applications in speech recognition*, Proc. IEEE, Vol.77, No.2, pp 257-285, 1989.
- [5] S.H. Chung, J.B. Moore, L. Xia, L.S. Premkumar and P.W. Gage, *Hidden Markov Model Techniques for extracting small Ionic currents from noise*, 1990, Phil. Trans. of Royal Society, to appear.
- [6] S.E. Levinson, L.R. Rabiner and M.M. Sondhi, *An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition*, The Bell System Theor. J., No.6, pp 1035-1074, 1983.
- [7] J.B. Moore, V. Krishnamurthy, L. Xia, *Towards On-line Hidden Markov Signal Processing*, 1990, submitted to IEEE Trans. ASSP.

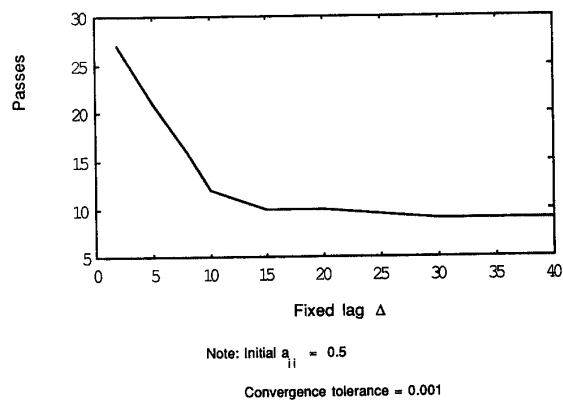


Fig. 4.1 : Re-estimation convergence of  $\mathbf{A}$  using batch update

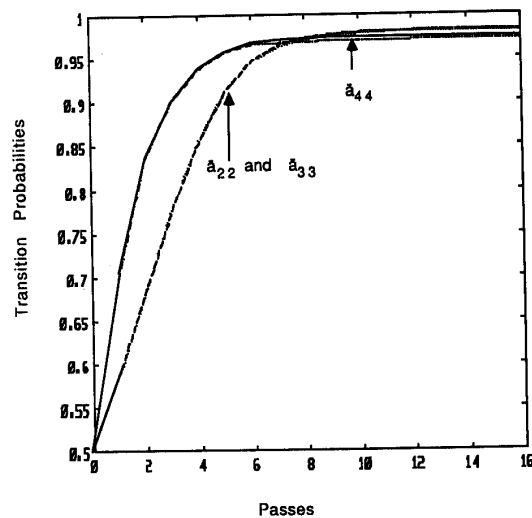
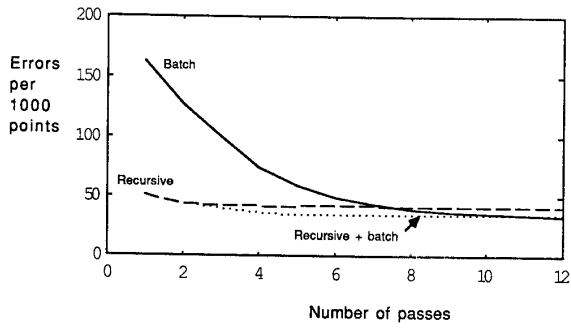


Fig. 4.2 : Re-estimation convergence for  $\Delta = 20$



Note  $\Delta_{\max} = 2 \Delta_{\min} = 20$

Fig. 4.6 : Convergence comparison of re-estimation algorithms used in sawtooth-lag smoothing

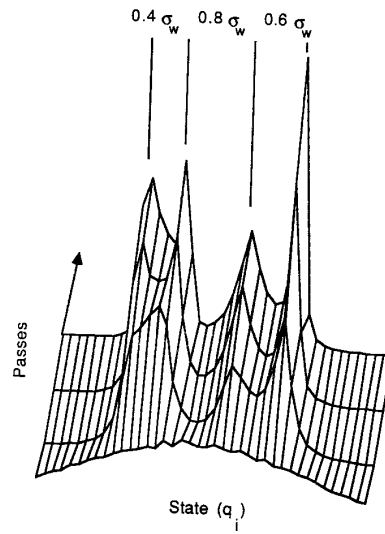
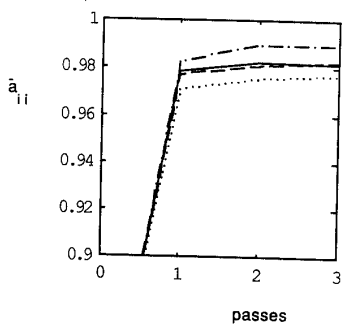


Fig. 5.1: Learning discrete state levels using on-line re-estimation



Note: initial  $\hat{a}_{ii} = 0.8$

Fig. 5.2 : Transition probability estimates

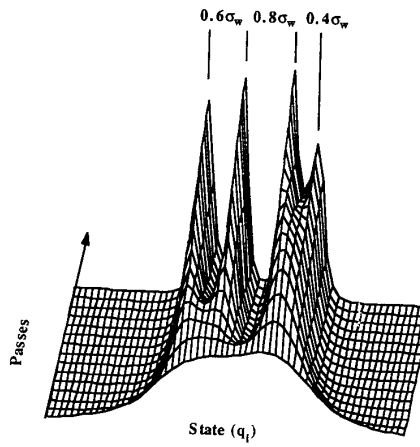


Fig. 5.3 : Learning discrete state levels using Baum Welch re-estimation