Inference in Hidden Markov Models

John MacLaren Walsh, Ph.D.

ECES 632, Winter Quarter, 2010

In this lecture we discuss a theme arising in many of your projects and many formulations of statistical signal processing problems: detection for finite state machines observed through noise. We saw that this was a special case of inference of the state sequence in hidden Markov models, and considered another family of hidden Markov models, the Gaussian Linear state space model.

1 Useful References

Useful references for this material include:

- *Inference in Hidden Markov Models* by O. Cappé, E. Moulines, and T. Royden. Springer Series in Statistics, 2005.
- *Principles of Digital Transmission* by S. Benedetto and E. Biglieri. Kluwer, 1999.
- *Optimal Filtering* by B. D. O. Anderson and J. Moore. Prentice Hall, 1979. Now available in a paperback dover edition.

2 Problem Setup

Many of the statistical processing problem families that we have discussed have observations which can be modeled as the output of finite state machines observed through noise. Let us be more specific about what we mean by finite state machine in this context.

As we shall define it in the context of these notes, a finite state machine is a device which takes a discrete time input signal $x[n]$ which, at each time instant, may only take values in some finite set $\mathcal{X} := \{i_1, \ldots, i_I\}$, to a discrete time output signal $y[n]$ which also may only take values in some finite set $\mathcal{Y} := \{j_1, \ldots, j_J\}$ according to a special structure which we presently identify[1]. In particular, the finite state machine is a causal structure with an internal state $s \in \mathcal{S}$, so that the output $y[n]$ at time n is determined by the current state $s[n]$ and the current input $x[n]$. Similarly, the next state $s[n+1]$ is determined by the current state $s[n]$ and the current input $x[n]$. Thus, on can fully specify the finite state machine with the function $\mathsf{f} : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ which determines the outputs and the function $\mathsf{g} : \mathcal{S} \times \mathcal{X} \to \mathcal{S}$ which determines the next state, so that

$$y[n] := \mathsf{f}(s[n], x[n]), \quad s[n+1] = \mathsf{g}(s[n], x[n]) \quad \forall n \tag{1}$$

We pause now to note that FIR or rational IIR filtering has the structure (1) when the input to the filter is taken from a finite set. To see this for the FIR filter of length P, given its impulse response $h[0], h[1], h[2], \ldots, h[P]$, and identifying the state as $s[n] := [x[n-1], x[n-2], \ldots, x[n-P]]$ we can identify the output function for the corresponding finite state machine as

$$\mathsf{f}(s[n], x[n]) := \sum_{p=0}^{P} h[p] x[n-p]$$

[1] For students who remember their FSMs from digital logic in their undergraduate education, we will be considering Mealy machines, whose output depends on the current state and the current input, as opposed to Moore machines, whose output depends only on the current state.

The state function is then seen to be

$$\mathsf{g}(s[n], x[n]) = \mathsf{g}([x[n-1], x[n-2], \ldots, x[n-P]], x[n]) = [x[n], x[n-2], \ldots, x[n-P+1]]$$

For this reason it is easy to see that convolutional codes of all types, as well as the filtering of a discrete valued signal by a finite impulse response filter, form a finite state machine.

In the situation we are interested in, we observe the output $y[n]$ through noise $w[n]$ which we take to be independent and identically distributed, so that our observations may be written as

$$z[n] := y[n] + w[n]$$

We will consider the finite block length situation so that $n \in \{0, 1, \ldots, N\}$. The goal is to determine which input sequence $\mathbf{x} := [x[0], x[1], \ldots, x[N]]$ gave rise to our observations. To do so, we will consider a Bayesian situation, in which we suppose *a priori* that the samples of $x[n]$ were chosen independently from one another according to a distribution which we know.

Speaking formally, then, there are (at least) two natural criteria we can use to perform the detection. First of all, we could choose the detector $\hat{\mathbf{x}}(z) := [\hat{x}[0], \hat{x}[1], \ldots, \hat{x}[N]]$ which minimizes the probability of selecting the wrong input sequence to the FSM, that is, minimizes the probability $\mathbb{P}[\hat{\mathbf{x}}(\mathbf{r}) \neq \mathbf{x}]$. Such a sequence is the maximum a posteriori (MAP) sequence detector and takes the form

$$\hat{\mathbf{x}}(\mathbf{r}) := \arg \max_{\mathbf{x} \in \mathcal{X}^N} \mathbb{P}[\mathbf{x}|\mathbf{r}] \tag{2}$$

Alternatively, one could choose each $\hat{x}[n]$ to minimize the corresponding probability of symbol error, that is, to minimize the probability $\mathbb{P}[\hat{x}[n] \neq x[n]]$. This is the MAP symbol detector and takes the form

$$\hat{x}[n](z) := \arg \max_{x[n] \in \mathcal{X}} \mathbb{P}[x[n]|z] \tag{3}$$

$$= \arg \max_{x[n] \in \mathcal{X}} \sum_{\mathbf{x}'|x'[n]=x[n]} \mathbb{P}[\mathbf{x} = \mathbf{x}'|z] \tag{4}$$

It should be easy to see from the latter relation that the two detectors (2) and (4) are not the same in general.

3 Minimizing the Sequence Error Rate: Viterbi Algorithm

Were we to try to find the maximum in (2) through exhaustive search of all possible $\mathbf{x} \in \mathcal{X}^N$, we would have to consider $|\mathcal{X}^N| = I^N$ different possibilities. The exponential growth in the sequence length quickly becomes computationally prohibitive. Fortunately, the finite memory structure of the process which generates $\{y[n]\}$ from $\{x[n]\}$ according to (1) can be exploited to find the most probably sequence in a much more computationally efficient way that exhaustive search. The algorithm which does this is often called the Viterbi algorithm. Although an in detail review of the operation of this algorithm is beyond the scope of this class, we provide now a sketch of its operation.

One effective visual way of describing the operation of the Viterbi algorithm is to introduce the trellis description of the structure described by (1). A trellis is a graphical depiction of the recursion (1), which is made of horizontally concatenated stages. A trellis stage is visually two vertical columns of circles. Each circle in the left column represents a state $s[n] \in \mathcal{S}$, and each circle in the right column represents a state $s[n+1] \in \mathcal{S}$. Each state s on the left has $|\mathcal{X}| = I$ lines coming from it, each one connected to a (possibly different) state on the right side $\mathsf{g}(s[n], x[n])$ and is labeled with an input, output pair $(x, \mathsf{f}(s, x))$. Repetitions of such trellis stages may then be connected end to end to form a trellis of length N.

Given the initial state of the trellis $s[0]$, a sequence of inputs and corresponding outputs of the finite state machine corresponds to a path through the trellis. Furthermore, one may observe that the log-probability of a sequence is additive along a given path through the trellis. For this reason, if the goal is to find the most likely sequence, one may proceed from left to right in the trellis, and at each stage, only keep track of the most probable path through the trellis to each state on the left hand side of the trellis stage at that time instant. In this way, at all times one needs only to keep track of $|\mathcal{S}|$ sequences which are the $|\mathcal{S}|$ most likely sequences up until that time instant. We demonstrated this in class.

4 Minimizing the Symbol Error Rate: Forward-Backward Algorithm

Shifting now our attention to (4), we note that to preform the given maximization by calculating each of the shown sums directly, we would have to sum $|\mathcal{X}|^{N-1} = I^{N-1}$ terms for each symbol, which again leads to a computational complexity which grows exponentially in the block length N. Fortunately, as was the case with the Viterbi algorithm, we can exploit the structure (1) to do this with far fewer calculations than direct summation. The algorithm which does this is often called the BCJR algorithm[2] or the forward backward algorithm.

The BCJR algorithm may be considered to be operating on the same trellis as the Viterbi algorithm. The computational savings are effected by noting that the probabilities being calculated may be written as

$$\mathbb{P}\left(x[n], \mathbf{r}\right) = \mathbb{P}\left(x[n]\right) \sum_{s \in \mathcal{S}} \mathbb{P}\left(y[n] = \mathsf{f}(s, x[n]), z[n]\right) \underbrace{\mathbb{P}\left(s[n] = s, z^{-}[n]\right)}_{\alpha_n(s)} \underbrace{\mathbb{P}\left(s[n+1] = \mathsf{g}(s, x[n]), z^{+}[n]\right)}_{\beta_n(\mathsf{g}(s, x[n]))}$$

where

$$z^{-}[n] := [z[0], z[1], \ldots, z[n-1]]$$

and

$$z^{+}[n] := [z[n+1], z[n+2], \ldots, z[N]]$$

Here the αs and βs admit simple recursive calculation via

$$\alpha_{n+1}(s') := \sum_{(s,x)|\mathsf{g}(s,x)=s'} \alpha_n(s) \mathbb{P}\left(y[n] = \mathsf{f}(s, x), z[n]\right) \mathbb{P}\left(x[n] = x\right)$$

and

$$\beta_n(s) := \sum_{x \in \mathcal{X}} \beta_{n+1}(\mathsf{g}(s, x)) \mathbb{P}\left(y[n] = \mathsf{f}(s, x), z[n]\right) \mathbb{P}\left(x[n] = x\right)$$

Since the former recursion operates forward in time, and the latter recursion operates backward in time, the algorithm is often referred to as the forward backward algorithm.

5 Inference in other types of Hidden Markov Models: GLSSMs

Another type of hidden Markov model we discussed in class is the Gaussian linear state space model

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{z}_n \tag{5}$$

$$\mathbf{y}_n = \mathbf{B}\mathbf{x}_n + \mathbf{w}_n \tag{6}$$

Here, $\mathbf{x}_n$ is the state vector of a linear dynamical system being driven by the noise $\mathbf{q}_n$, and which we are only able to indirectly observe via the observations $\mathbf{y}_n$. The state noise $\mathbf{z}_n$ and the observation noise $\mathbf{w}_n$ are independent normally distributed in n with zero mean and covariance matrices $\mathbf{R}_n$ and $\mathbf{Q}_n$ respectively.

5.1 The Kalman Filter

We discussed how the Kalman filter could be used in this model to provide the conditional distribution $p(\mathbf{x}_n|\mathbf{y}_1, \ldots, \mathbf{y}_n)$, while the Rauch Tung Striebel smoother can be used to provide the conditional distribution $p(\mathbf{x}_n|\mathbf{y}_1, \ldots, \mathbf{y}_N)$, i.e. the a posteriori distribution over an entire block of N observations. As the model is linear and the distributions are all Gaussian, these posterior distributions will also be Gaussian, so all that we need to do is specify the mean and covariance of each. Hence, name the mean of the conditional distribution $p(\mathbf{x}_n|\mathbf{y}_1, \ldots, \mathbf{y}_n)$ $\mathbf{m}_{n|n}$ and the covariance $\mathbf{\Sigma}_{n|n}$.

The idea behind the Kalman filtering equations is to determine the mean and covariance $\mathbf{m}_{n+1|n+1}, \mathbf{\Sigma}_{n+1|n+1}$ as a function of $\mathbf{m}_{n|n}$ and $\mathbf{\Sigma}_{n|n}$ the observations $\mathbf{y}_{n+1}$ and the model parameters. This is achieved by

[2]This is after the authors appearing on a paper discussing it: L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inform. Theory, vol. 20, pp. 284–287, Mar. 1974.

first determining the mean and covariance of the prediction distribution $p(\mathbf{x}_{n+1}|\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $\mathbf{m}_{n+1|n}$ and $\mathbf{\Sigma}_{n+1|n}$ via

$$\mathbf{m}_{n+1|n} = \mathbf{A}\mathbf{m}_{n|n} \tag{7}$$

$$\mathbf{\Sigma}_{n+1|n} = \mathbf{A}\mathbf{\Sigma}_{n|n}\mathbf{A}^T + \mathbf{R}_n \tag{8}$$

and then conditioning on $\mathbf{y}_{n+1}$ in the linear model

$$\mathbf{y}_{n+1} = \mathbf{B}\mathbf{x}_{n+1} + \mathbf{w}_{n+1} \tag{9}$$

using $\mathcal{N}(\mathbf{m}_{n+1|n}, \mathbf{\Sigma}_{n+1|n})$ as the distribution on $\mathbf{x}_n$ to get

$$\mathbf{m}_{n+1|n+1} = \mathbf{m}_{n+1|n} + \mathbf{\Sigma}_{n+1|n}\mathbf{B}^T(\mathbf{B}\mathbf{\Sigma}_{n+1|n}\mathbf{B}^T + \mathbf{Q}_{n+1})^{-1}(\mathbf{y}_{n+1} - \mathbf{m}_{n+1|n}) \tag{10}$$

$$\mathbf{\Sigma}_{n+1|n+1} = \mathbf{\Sigma}_{n+1|n} - \mathbf{\Sigma}_{n+1|n}\mathbf{B}^T(\mathbf{B}\mathbf{\Sigma}_{n+1|n}\mathbf{B}^T + \mathbf{Q}_{n+1})^{-1}\mathbf{B}\mathbf{\Sigma}_{n+1|n} \tag{11}$$

The initial distribution on the state determines the initial covariance and mean to use in this recursion.