Discrete Hidden Markov Model
C Source Code

Jerome V. Braun

September 8, 2014

Contents

1 Introduction and Acknowledgments

This is a set of C routines for working with discrete hidden Markov models. I have found writing them useful both as a self-tutorial and in the service of developing several application programs. Therefore, I have decided to make them more widely available.

Several chunks of code from Ernst Stadlober's `Win_rand~1.0` library of random number generators were used. I have provided a set of wrapper functions so that it is feasible to swap new generators into their place if desired. I may implement them myself to make them a little more readable.

The code and this documentation were originally encapsulated in NoWeb. However, for ease of use by others, the code was untangled and this document was used to capture most of the woven text. As a consequence THIS IS NOT A COHERENT DOCUMENT CURRENTLY.

2 Programming conventions

I have used single-letter variable names in many of the routines which implement mathematical functions or algorithms. My hope is that the juxtaposition of the source code with the documentation given here will ease the reading, since the code in many cases is a direct translation of mathematical formulas.

Each program file has a programmer's comment block which gives overall identifying information.

Each function or subroutine has a comment block which provides an example call, the purpose of the function, any notes on how the algorithm works, the expected inputs, the expected outputs, and whether the function returns a value.

3 Hidden Markov model notation

One of the frustrating things about developing code for the hidden Markov model is the wide variety of notations that are used to denote the same or a quite similar underlying statistical model. It is necessary to clearly specify the notation up front, since slightly different models yield slightly different algorithms.

My notation is relatively usual. It follows [3], who provided a nice review of the model and associated algorithms, and [1], who provided an elegant approximation of full Bayesian inference. For programming simplicity, the states and observations are notated by integers – for application it will be necessary to translate real data to this notation and back.

- t: sequence index, $t \in 1...T$
- I: number of hidden states in the model
- M: number of types of observations
- $S = \{s_1, \ldots, s_T\}$: hidden state sequence ($S_t \in 1...I$)
- $X = \{x_1, \ldots, x_T\}$: observed sequence ($X_t \in 1...M$)
- $A = \{a_{ij}\}$, $a_{ij} = P(s_{t+1} = j | s_t = i)$: state transition matrix
- $B = \{b_{im}\}$, $b_{im} = P(x_t = m | s_t = i)$: observation emission matrix
- $\pi = \{\pi_i\}$, $\pi_i = P(s_1 = i)$: initial state distribution
- $\theta = \{A, B, \pi\}$: hidden Markov model parameters
- $U = \{u^{(A)}, u^{(B)}, u^{(\pi)}\}$: hyperparameters which define the prior over θ

4 Statistical theory

With the model defined, we may conduct statistical inference. In the frequentist framework, there are three tasks of inference for hidden Markov models.

1. Find the distribution of the observations given the model, that is, $P(X|\theta)$.
2. Find the most probable state trajectory given the model and the observations, that is, $\arg\max_S P(S|X\theta)$.
3. Find the maximum likelihood estimate of θ, that is, $\arg\max_\theta P(X|\theta)$.

For a given hidden Markov model parameterized by θ, the Markov assumptions give the joint probability of the state sequence and the observations

$$P(XS|\theta) = \pi_{s_1} \left(\prod_{t=1}^{T-1} a_{s_t s_{t+1}} \right) \left(\prod_{t=1}^{T} b_{s_t x_t} \right).$$

By summing over all possible state sequences, we could obtain

$$P(X|\theta) = \sum_S P(XS|\theta),$$

and obtain the conditional distribution of S given X as

$$P(S|X\theta) = \frac{1}{P(X|\theta)} P(XS|\theta).$$

In practice, the number of calculations involved is too many to consider using these analytic approaches to determine either the most likely state or the maximum likelihood estimate of θ. The usual approach is to use an iterative method which converges to the desired quantities (Baum-Welch algorithm, EM algorithm).

In the Bayesian framework we must also consider what (logically) prior information exists for θ. Note that this is a different set of hypotheses than considered in the frequentist version of the problem – thus, although we confusingly use the same notation, we should not be surprised to obtain differing results. There are two more tasks.

4. Assign a prior distribution for θ, that is, $P(\theta|U)$.
5. Determine the posterior distribution of the model parameters given the prior and the data, that is, $P(S\theta|XU)$.

In practice it is too difficult again to conduct the calculation of the posterior distribution. However, it is possible to approximate the posterior, either by Markov chain Monte Carlo techniques or by analytic approximations. We will implement an analytic approximation developed in [1].

Choose prior probabilities for θ to be products of independent Dirichlet distributions with parameters given by U. Then although the posterior distribution $P(S\theta|XU)$ is still intractable, we can develop an approximation $Q(S\theta)$ by constraining it to be separable,

$$Q(S\theta) = Q_S(S)Q_A(A)Q_B(B)Q_\pi(\pi),$$

and minimizing the free energy (Kullback-Leibler distance, cross-entropy) of the approximation to the true posterior distribution. MacKay showed that a variant of the EM algorithm could be used to iteratively obtain the parameters of the approximation.

MacKay noted that initializing the counts of the Baum-Welch algorihtm with the values of the prior distribution given above maximizes the posterior distribution in the case that we are working in the softmax basis [2].

5 More notes on notation

5.1 E-step

The usual recursions:

$$\begin{aligned}
\alpha_i^{(1)} &= \pi_i b_{ix_1} \\
\alpha_j^{(t+1)} &= b_{jx_{t+1}} \sum_i \alpha_i^{(t)} a_{ij} \\
\beta_i^{(T)} &= b_i x_T \\
\beta_i^{(t)} &= b_{ix_t} \sum_j a_{ij} \beta_j^{(t+1)}
\end{aligned}$$

Rabiner & Juang notation:

$$\begin{aligned}
\alpha_i^{(1)} &= \pi_i b_{ix_1} \\
\alpha_j^{(t+1)} &= b_{jx_{t+1}} \sum_i \alpha_i^{(t)} a_{ij} \\
\beta_i^{(T)} &= 1 \\
\beta_i^{(t)} &= b_{ix_t} \sum_j a_{ij} \beta_j^{(t+1)} b_{jx_{t+1}}
\end{aligned}$$

$$\begin{aligned}
\xi_{ij}^{(t)} &= \frac{1}{Z_\xi(t)} \alpha_i^{(t)} a_{ij} \beta_j^{(t+1)} b_{jx_{t+1}} \\
Z_\xi(t) &= \sum_{ij} \alpha_i^{(t)} a_{ij} \beta_j^{(t+1)} b_{jx_{t+1}}
\end{aligned}$$

$$
\begin{aligned}
\gamma_i^{(t)} &= \frac{\alpha_i^{(t)}\beta_i^{(t)}}{Z_\gamma(t)} \\
Z_\gamma(t) &= \sum_i \alpha_i^{(t)}\beta_i^{(t)}
\end{aligned}
$$

5.2 Scaled

With scaling s.

$$
\begin{aligned}
\hat{\alpha}_i^{(1)} &= \pi_i b_{ix_1} \\
s_1 &= \sum_i \hat{\alpha}_i^{(1)} \\
\alpha_i^{(1)} &= \alpha_i^{(1)}/s_1 \\
\hat{\alpha}_j^{(t+1)} &= b_{jx_{t+1}} \sum_i \alpha_i^{(t)} a_{ij} \\
s_t &= \sum_i \hat{\alpha}_i^{(t)} \\
\alpha_i^{(t)} &= \hat{\alpha}_i^{(t)}/s_t \\
\beta_i^{(T)} &= 1/s_T \\
\beta_i^{(t)} &= b_{ix_t} \sum_j a_{ij}\beta_j^{(t+1)} b_{jx_{t+1}}/s_t
\end{aligned}
$$

$$
\begin{aligned}
\xi_{ij}^{(t)} &= \frac{1}{Z_\xi(t)} \alpha_i^{(t)} a_{ij} \beta_j^{(t+1)} b_{jx_{t+1}}/s_{t+1} \\
Z_\xi(t) &= \sum_{ij} \alpha_i^{(t)} a_{ij} \beta_j^{(t+1)} b_{jx_{t+1}} \\
\gamma_i^{(t)} &= \frac{\alpha_i^{(t)}\beta_i^{(t)}}{Z_\gamma(t)} \\
Z_\gamma(t) &= \sum_i \alpha_i^{(t)}\beta_i^{(t)}
\end{aligned}
$$

5.3 M-step

$$
\begin{aligned}
\bar{\pi}_i = \gamma_i^{(1)} &= \frac{\gamma_i^{(1)}}{\sum_j \gamma_j^{(1)}} \\
\bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_{ij}^{(t)}}{\sum_{t=1}^{T-1} \gamma_i^{(t)}}
\end{aligned}
$$

$$\bar{b}_{im} = \frac{\sum_{t=1}^{T} \gamma_i^{(t)} \delta(m, x_t)}{\sum_{t=1}^{T} \gamma_i^{(t)}}$$

References

[1] D. J. C. MacKay. Ensemble learning for hidden Markov models. `http://www.inference.phy.cam.ac.uk/mackay/abstracts/ensemblePaper.html`, 1997.

[2] David J. C. MacKay. Choice of basis for Laplace approximation. *Machine Learning*, 33(1):77–86, October 1998.

[3] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):3–16, 1986.