Notes on Hidden Markov Models

The materials are adopted from Chapter 17 of Kevin's Murphy Book: Machine Learning a Probabilistic Approach

1 The forwards algorithm

We want to compute recursively the posterior density $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$ of the HMM (i.e., the filtered posterior density). We want to find a recursive formula at time t. First we notice that

$$p(z_{tj} = 1|\boldsymbol{x}_{1:t-1}) = \sum_i p(z_{tj} = 1|z_{(t-1)i} = 1)p(z_{(t-1)i} = 1|\boldsymbol{x}_{1:t-1}). \tag{1}$$

Next comes the update step, in which we absorb the observed data from time t using Bayes rule:

$$\begin{aligned}\alpha_t(j) &\triangleq p(z_{tj} = 1|\boldsymbol{x}_{1:t}) = p(z_{tj} = 1|\boldsymbol{x}_t, \boldsymbol{x}_{1:t-1})\\ &= \frac{1}{Z_t}p(\boldsymbol{x}_t|z_{tj} = 1, \boldsymbol{x}_{1:t-1})p(z_{tj} = 1|\boldsymbol{x}_{1:t-1})\\ &= \frac{1}{Z_t}p(\boldsymbol{x}_t|z_{tj} = 1)\sum_i p(z_{tj} = 1|z_{(t-1)i} = 1)p(z_{(t-1)i} = 1|\boldsymbol{x}_{1:t-1})\\ &= \frac{1}{Z_t}p(\boldsymbol{x}_t|z_{tj} = 1)\sum_i p(z_{tj} = 1|z_{(t-1)i} = 1)\alpha_{t-1}(i)\end{aligned}$$

where the normalization constant is given by

$$Z_t \triangleq p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}) = \sum_j p(z_t = j|\boldsymbol{x}_{1:t-1})p(\boldsymbol{x}_t|z_t = j). \tag{2}$$

The above process is known as the **predict-update cycle**. The distribution $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$ is called the (filtered) **belief state** at time t, and is a vector of K numbers, often denoted by $\boldsymbol{\alpha}_t$[1].

[1]In Christopher's Bishop book, as well as, in my first series of notes $\alpha_t(j)$ is defined as the *joint probability* $p(\boldsymbol{z}_t, \boldsymbol{x}_{1:t})$, rather than defining directly the filtered posterior $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$, as we do here. As we have discussed, defining it as $p(\boldsymbol{z}_t, \boldsymbol{x}_{1:t})$, has two problems. First it rapidly suffers from numerical underflow, since the probability of observing any particular sequence of evidence $\boldsymbol{x}_{1:t}$ is very small. Second, the joint probability $p(\boldsymbol{z}_t, \boldsymbol{x}_{1:t})$ is not as meaningful as the posterior distribution over states $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$. Note, however, that these two definitions only differ by a multiplicative constant, so the *algorithmic* difference is just one line of code. (In fact, all good implementations of the forwards algorithm normalize the belief state after each step, to avoid underflow, and hence they are actually computing $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$!).

In matrix-vector notation, we can write the update in the following simple form:

$$\boldsymbol{\alpha}_t \propto \boldsymbol{b}_t \odot (\boldsymbol{A}^T \boldsymbol{\alpha}_{t-1}) \tag{3}$$

where $\boldsymbol{b}_t = p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ is the emission probability vector at time t, $A(i,j) = p(z_{tj} = 1|z_{(t-1)i} = 1)$ is the transition probability matrix, and $\mathbf{u} \odot \mathbf{v}$ is the **Hadamard product**, representing elementwise vector multiplication. See algorithm 1.1 for the pseudo-code, (also hmmFilter of Kevin's Murphy code for some Matlab code).
In addition to computing the hidden states, we can use this algoritm to compute the log probability of the evidence:

$$\log p(\boldsymbol{x}_{1:T}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}) = \sum_{t=1}^{T} \log Z^t \tag{4}$$

(We need to work in the log domain to avoid numerical underflow).

Algorithm 1.1: Forwards algorithm

```
1 Input: Transition matrices $\boldsymbol{A} = [p(z_{tj} = 1|z_{(t-1)i} = 1)]$,
      emission probability vectors $\boldsymbol{b}_t = p(\boldsymbol{x}_t|\boldsymbol{z}_t)$,
      initial state distribution vector $\boldsymbol{\pi} = p(\boldsymbol{z}_1)$;
2 $[\boldsymbol{\alpha}_1, Z_1] = \text{normalize}(\boldsymbol{b}_1 \odot \boldsymbol{\pi})$;
3 for $t = 2 : T$ do
4 $[\boldsymbol{\alpha}_t, Z_t] = \text{normalize}(\boldsymbol{b}_t \odot (\boldsymbol{A}^T \boldsymbol{\alpha}_{t-1}))$;
5 Return $\boldsymbol{\alpha}_{1:T}$ and $\log p(\mathrm{x}_{1:T}) = \sum_t \log Z_t$;
6 Subroutine: $[\mathbf{v}, Z] = \text{normalize}(\mathbf{u}) : Z = \sum_j u_j;\ u_j = u_j/Z$;
```

2 The forwards-backwards algorithm

In the previous section we have described how to compute the filtered posterior $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$. In the following, we will discuss how to compute the smoothed posteriors $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})$.

2.1 Basic idea

The key decomposition relies on the fact that we can break the chain into two parts, the past and the future, by conditioning on $\boldsymbol{z}_t$:

$$p(z_{tj} = 1|\boldsymbol{x}_{1:T}) \propto p(z_{tj} = 1, \boldsymbol{x}_{t+1:T}|\boldsymbol{x}_{1:t}) \propto p(z_{tj} = 1|\boldsymbol{x}_{1:t}) p(\boldsymbol{x}_{t+1:T}|z_{tj} = 1, \boldsymbol{x}_{1:t}). \tag{5}$$

Let $\alpha_t(j) \triangleq p(z_{tj} = 1|\boldsymbol{x}_{1:t})$ be the filtered belief state as before. Also define $\beta_t(j) \triangleq p(\boldsymbol{x}_{t+1:T}|z_{tj} = 1)$ as the conditional likelihood of future evidence given that the hidden state at time t is j. (Note that this is not a probability distribution over states, since it does not need to satisfy $\sum_j \beta_t(j) = 1$.) Finally, define

$$\gamma_t(j) \triangleq p(z_t = j | \boldsymbol{x}_{1:T}) \tag{6}$$

as the desired smoothed posterior. From equation 5, we have

$$\gamma_t(j) \propto \alpha_t(j)\beta_t(j). \tag{7}$$

We have already described how to recursively compute the α's in a left-to-right fashion in the previous section. We now describe how to recursively compute the β's in a right-to-left fashion. If we have already computed β_t, we can compute β_{t-1} as follows:

$$\begin{aligned}
\beta_{t-1}(i) &= p(\boldsymbol{x}_{t:T}|z_{(t-1)i} = 1) \\
&= \sum_j p(z_{tj} = 1, \boldsymbol{x}_{t:T}|z_{(t-1)i} = 1) \\
&= \sum_j p(\boldsymbol{x}_{t:T}|z_{tj} = 1, z_{(t-1)i} = 1)p(z_{tj} = 1|z_{(t-1)i} = 1) \\
&= \sum_j p(\boldsymbol{x}_{t:T}|z_{tj} = 1)p(z_{tj} = 1|z_{(t-1)i} = 1) \\
&= \sum_j p(\boldsymbol{x}_{t+1:T}|z_{tj} = 1)p(\boldsymbol{x}_t|z_{tj} = 1)p(z_{tj} = 1|z_{(t-1)i} = 1) \\
&= \sum_j \beta_t(j)b_t(j)A(i,j).
\end{aligned}$$

We can write the resulting equation in matrix-vector form as

$$\boldsymbol{\beta}_{t-1} = \boldsymbol{A}(\boldsymbol{b}_t \odot \boldsymbol{\beta}_t). \tag{8}$$

The base case is

$$\beta_T(i) = p(\boldsymbol{x}_{T+1:T}|z_{Ti} = 1) = p(0|z_{Ti} = 1) = 1 \tag{9}$$

which is the probability of a non-event (another way to see why $\beta_T(i)$ can be found in the presentation notes).
Having computer the forwards and backwards messages, we can combine them to compute $\gamma_t(j) \propto \alpha_t(j)\beta_t(j)$. The overall algorithm is known as the **forwards-backwards algorithm**. The pseudo code is very similar to the forwards case; (see hmmFwdBack from Kevin's Murphy library for an implementation).
We can think of this algorithm as passing 'messages' from left to right, and then from right to left, and then combining them at each node.

2.2 Two-slice smoothed posterior

Finally we have to compute the joint smoothed posterior of t and $t+1$, i.e., $p(z_{ti} = 1, z_{(t+1)j} = 1|\boldsymbol{x}_{1:T})$ is called a (smoothed) **two-slice posterior**, and can be computed as follows:

$$\begin{aligned}
\xi_{t,t+1} &\triangleq p(\boldsymbol{z}_t, \boldsymbol{z}_{t+1}|\boldsymbol{x}_{1:T}) \\
&\propto p(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t, \boldsymbol{x}_{t+1:T}) \\
&\propto p(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})p(\boldsymbol{x}_{t+1:T}|\boldsymbol{z}_t, \boldsymbol{z}_{t+1})p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t) \\
&\propto p(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{t+1})p(\boldsymbol{x}_{t+2:T}|\boldsymbol{z}_{t+1})p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_t) \\
&= \alpha_t(i)b_{t+1}(j)\beta_{t+1}(j)A(i,j).
\end{aligned}$$

In matrix-vector form, we have

$$\boldsymbol{\xi}_{t,t+1} \propto \boldsymbol{A} \odot (\boldsymbol{a}_t(\mathbf{b}_{t+1} \odot \boldsymbol{\beta}_{t+1})^T).$$