1 Hidden Markov Model

A hidden Markov model (HMM) refers to a set of “hidden” states $X_0, X_1, \ldots, X_t, \ldots, X_T$ and a set of observations, $Y_0, Y_1, \ldots, Y_t, \ldots, Y_T$ with the following joint PMF or PDF:

$$p(x_{0:T}, y_{0:T}) = [p(x_0)[\prod_{t=1}^{T} p(x_t|x_{t-1})]][p(y_0|x_0)[\prod_{t=1}^{T} p(y_t|x_t)]]$$ \hfill (1)

The following conditional independence (c.i.) relationships can be shown either using Theorem 2 of the Graphical Models handout or directly.

1. A state sequence and the corresponding observation sequence form an HMM if and only if
   
   (a) given $X_t, X_{t+1}$ is c.i. of past-X ($X_{0:t-1}$) and of past-Y ($Y_{0:t}$), and
   
   (b) given $X_t, Y_t$ is c.i. of all-X ($X_{0:t-1}, X_{t+1:T}$) and all-Y ($Y_{0:t-1}, Y_{0:t+1:T}$).

2. If $X_{0:T}, Y_{0:T}$ form an HMM, then the following also hold.

   (a) By reversing the Markov chain $\{X_t\}$, we can also claim that given $X_t, X_{t-1}$ is c.i. of all future-X ($X_{t+1:T}$) and all future-Y ($Y_{t:T}$).

   (b) Given $X_t, X_{t+1:T}$ are c.i. of past-X ($X_{0:t-1}$) and of past-Y ($Y_{0:t}$).

   (c) Given $X_{t-k}, Y_t$ is c.i. of $Y_{0:t-k}$ and of $X_{0:t-k-1}$ for $k > 0$. If $k = 0$, replace $Y_{0:t-k}$ by $Y_{0:t-1}$.

   (d) Given $X_{t-k}, Y_{t:T}$ is c.i. of $X_{0:t-k-1}$ and $Y_{0:t-k}$ for $k > 0$. If $k = 0$, replace $Y_{0:t-k}$ by $Y_{0:t-1}$.

   (e) By reversing the Markov chain $\{X_t\}$, the opposite of 2c can also be shown for future.

   (f) Many more

Examples.

1. The state space model used for defining the Kalman filter was an example of an HMM with continuous states, $X_t$ and continuous observations, $Y_t$.

2. $X_t$ refers to today’s weather which can take one of three possible values, \{rainy, cloudy, sunny\}. $Y_t$ is a binary random variable which can take two possible values \{class occurs, no class occurs\}. It is natural to claim that today’s weather depends only on yesterday’s weather, i.e. given yesterday’s weather, today’s weather is c.i. of past weather or of whether class occurred yesterday or in the past or not. Also, the chance that class will occur today or not is governed only by today’s weather (if it is sunny, it is more likely
that the class will not occur!) and given today’s weather, the chance is independent of all past or future weather and also of whether classes occurred in the past or in the future. This, of course models, an irresponsible professor who does not care about whether the material is covered or not!

3. Speech recognition, $X_t$: different phonems, $Y_t$: linear prediction coefficients (LPC’s) of the AR model describing observed speech.

4. Gesture recognition, $X_t$: different gestures out of a set, $Y_t$: outer contour of the observed hand shape (for hand gestures)

5. In last two examples, $X_t$ is discrete, $Y_t$ is continuous, that is allowed too.

Causal Posterior Computation.

1. We refer to $p(x_t|y_{0:t})$ as the causal posterior. In real-time applications, there is a need to compute it recursively, for example, to be able to compute the causal MMSE or causal MAP estimate.

2. “Recursive computation” means use the causal posterior and $t-1$ and the current observation to compute the causal posterior at $t$.

3. Using Bayes’ rule and HMM properties, the causal posterior satisfies

$$p(x_t|y_{0:t}) \propto p(x_t, y_t|y_{0:t-1})$$
$$= p(x_t|y_{0:t-1})p(y_t|x_t, y_{0:t-1})$$
$$= p(x_t|y_{0:t-1})p(y_t|x_t) \quad \text{(using HMM property 1b)}$$
$$= p(y_t|x_t) \int p(x_t, x_{t-1}|y_{0:t-1})dx_{t-1}$$
$$= p(y_t|x_t) \int p(x_{t-1}|y_{0:t-1})p(x_t|x_{t-1}, y_{0:t-1})dx_{t-1}$$
$$= p(y_t|x_t) \int p(x_{t-1}|y_{0:t-1})p(x_t|x_{t-1})dx_{t-1} \quad \text{(using HMM property 1a)(2)}$$

4. The above recursion is another way to derive the Kalman filter recursion: the causal MMSE estimate, $E[X_t|Y_{0:t}]$, is the expectation of $X_t$ under the causal posterior. Since everything there is jointly Gaussian, the posteriors will also be Gaussian and hence completely specified by the mean and covariance. Kay’s book does it this way.

5. The same rules apply for discrete states: just replace $\int$ by $\sum$. 

2
2 Discrete-state HMM

In this handout, we will study the set of techniques developed for discrete-state HMM’s. The material is based on Rabiner’s tutorial (Proc. IEEE, February 1989).

Thus any $X_t$ is a discrete random variable which takes one of $N$ possible values, $i = 1, 2, \ldots N$.

2.1 Notation

A time-homogenous discrete state HMM is completely specified by

\[
\begin{align*}
\pi_i & \triangleq P(X_t = i) \\
\alpha_{i,j} & \triangleq P(X_t = j | X_{t-1} = i) \\
\beta_j(y) & \triangleq P(Y_t = y | X_t = j) \quad \text{(if $Y_t$ is continuous this is replaced by the conditional PDF)}
\end{align*}
\]

The following notation is used in efficient computation of various quantities.

\[
\begin{align*}
\alpha_t(i) & \triangleq p(y_0:t, x_t = i) \\
\beta_t(i) & \triangleq p(y_{t+1:T} | x_t = i) \\
\gamma_t(i) & \triangleq p(x_t = i | y_{0:T}) \quad \text{(note this conditions on all observations)} \\
\xi_t(i, j) & \triangleq p(x_t = i, x_{t+1} = j | y_{0:T}) \quad \text{(note this conditions on all observations)}
\end{align*}
\]

The following notation will be used in computing the posterior MAP estimate using the Viterbi algorithm.

\[
\begin{align*}
\delta_t(x_t) & \triangleq \max_{x_{0:t-1}} p(x_{0:t}, y_{0:t}) \\
\psi_t(x_t) & \triangleq \arg \max_{1 \leq x_{t-1} \leq N} \delta_{t-1}(x_{t-1}) a_{x_{t-1}, x_t}
\end{align*}
\]
2.2 Recursion for $\alpha_t, \beta_t, \gamma_t, \xi_t$

Consider $\alpha_t$

\[ \alpha_t(i) \triangleq p(y_{0:t}, x_t = i) \]
\[ = \sum_{j=1}^{N} p(y_{0:t}, x_t = i, x_{t-1} = j) \]
\[ = \sum_{j=1}^{N} p(y_{0:t-1}, x_{t-1} = j) p(x_t = i | x_{t-1} = j, y_{0:t-1}) p(y_t | x_t = i, x_{t-1} = j, y_{0:t-1}) \]
\[ = \sum_{j=1}^{N} p(y_{0:t-1}, x_{t-1} = j) p(x_t = i | x_{t-1} = j) p(y_t | x_t = i) \quad \text{(using HMM properties 1a 1b)} \]
\[ = \sum_{j=1}^{N} p(y_{0:t-1}, x_{t-1} = j) a_{ji} b_i(y_t) \]
\[ = b_i(y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) a_{j,i} \quad \text{(6)} \]

Consider $\beta_t$

\[ \beta_t(i) \triangleq p(y_{t+1:T} | x_t = i) \]
\[ = \sum_{j=1}^{N} p(y_{t+1:T}, x_{t+1} = j | x_t = i) \]
\[ = \sum_{j=1}^{N} p(x_{t+1} = j | x_t = i) p(y_{t+1} | x_{t+1} = j, x_t = i) p(y_{t+2:T} | x_{t+1} = j) \quad \text{(using HMM properties 1b, 2d)} \]
\[ = \sum_{j=1}^{N} a_{i,j} b_j(y_{t+1}) \beta_{t+1}(j) \quad \text{(7)} \]

Consider $\gamma_t$. Using definitions of $\alpha_t(i)$ and $\beta_t(i)$, it is clear that $\gamma_t(i) \propto \alpha_t(i) \beta_t(i)$. Thus,

\[ \gamma_t(i) = \frac{1}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)} \alpha_t(i) \beta_t(i) \quad \text{(8)} \]
Consider $\xi_t$:

$$\xi_t(i, j) \triangleq \frac{1}{p(y_{0:T})} p(y_{0:T}, x_t = i, x_{t+1} = j)$$

$$= \frac{1}{p(y_{0:T})} p(x_t = i, y_{0:t}) p(x_{t+1} = j | x_t = i, y_{0:t}) p(y_{t+1:T} | x_{t+1} = j, x_t = i, y_{0:t})$$

$$= \frac{1}{p(y_{0:T})} p(x_t = i, y_{0:t}) p(x_{t+1} = j | x_t = i) p(y_{t+1:T} | x_{t+1} = j) \quad \text{(using HMM properties 1a,2d)}$$

$$= \frac{1}{p(y_{0:T})} \alpha_t(i) a_{i,j} \beta_t(j)$$

$$= \frac{1}{\sum_{i'} \sum_{j'} \alpha_t(i') a_{i',j'} \beta_t(j')} \alpha_t(i) a_{i,j} \beta_t(j) \quad (9)$$

### 2.3 Computing $p(y_{0:T})$: Forward algorithm, Backward algorithm

Brute force computation of $p(y_{0:T})$ will require evaluating

$$p(y_{0:T}) = \sum_{x_0:T} p(x_0) \prod_{t=1}^{T} p(x_t | x_{t-1}) p(y_0 | x_0) \prod_{t=1}^{T} p(y_t | x_t) \quad (10)$$

will require $O(N^T)$ computations.

#### 2.3.1 Forward algorithm

A fast and causal way to compute $p(y_{0:T})$ is to go forward in time

$$p(y_{0:T}) = \sum_i \alpha_T(i) \quad (11)$$

The recursion for $\alpha_t(i)$ is given in (6). This takes $O(N^2T)$ computation only.

#### 2.3.2 Backward Algorithm

Another $O(N^2T)$ way to compute $p(y_{0:T})$ is going backwards in time

$$p(y_{0:T}) = \sum_i \beta_0(i) \pi_i \quad (12)$$

The recursion for $\beta_t(i)$ is given in (7).

Typically one would use the forward algorithm to compute this, since that is also causal. There may be situations, e.g. if this is done offline and if observations are stored as last-in-first-out where one may need to use the backward algorithm.
2.4 EM algorithm for discrete-state HMM parameter estimation: Baum Welsch algorithm

Let \( \theta \) denote the set of parameters. In this case, \( \theta \) includes all elements \( \{a_{i,j}\}, \{\pi_i\} \) and the parameters of \( b_i(y) \).

**Assumption 1** Assume for the discussion below that \( Y_t \)'s are also discrete and take \( M \) possible values, \( 1, 2, \ldots M \). Thus, in \( b_i(y) \), \( y \) can be \( 1, 2, \ldots M \).

Then \( \theta = \{\pi_i\}_{i=1,\ldots,N}, \{a_{i,j}\}_{i=1,\ldots,N,j=1,\ldots,N}, \{b_i(y)\}_{i=1,\ldots,N,y=1,\ldots,M} \).

Let \( \theta^k \) denote the parameter estimate at the \( k^{th} \) iteration. Recall that the EM algorithm computes

\[
\theta^{k+1} = \arg \max_{\theta} Q(\theta, \theta^k) \text{ s.t. constraints on } \theta
\]

where \( Q(\theta, \theta^k) = \mathbb{E}[\log \pi_{X_0} + \sum_{t=1}^T \log a_{X_{t-1},X_t} + \sum_{t=0}^T \log b_{X_t}(y_t) | y_0:T; \theta^k] \) (13)

i.e. at each iteration EM maximizes the posterior expectation of the logarithm of the complete data likelihood (the posterior expectation is computed using the parameter estimates from the previous iteration). As discussed earlier (when talking about EM algorithm), under certain assumptions, this leads to maximization of the observed data likelihood, i.e. its solution converges to \( \arg \max_{\theta} p(y_0:T; \theta) \).

Now for our HMM,

\[
\log p(y_0:T, X_0:T; \theta) = \log \pi_{X_0} + \sum_{t=1}^T \log a_{X_{t-1},X_t} + \sum_{t=0}^T \log b_{X_t}(y_t)
\]

Thus the first term is only a function of random variable \( X_0 \), the \( t^{th} \) entry of the second term is only a function of \( X_{t-1}, X_t \) and the \( t^{th} \) entry of the third term is only a function of \( X_t \).

\[
\mathbb{E}[\log p(y_0:T, X_0:T; \theta) | y_0:T; \theta^k] = \mathbb{E}[\log \pi_{X_0} | y_0:T; \theta^k] + \sum_{t=1}^T \mathbb{E}[\log a_{X_{t-1},X_t} | y_0:T; \theta^k] + \sum_{t=0}^T \mathbb{E}[\log b_{X_t}(y_t) | y_0:T; \theta^k] = \sum_{i} p(x_0 = i | y_0:T) \log \pi_i + \sum_{t=1}^T \sum_{i,j} p(x_{t-1} = i, x_t = j | y_0:T) \log a_{i,j} + \sum_{t=0}^T \sum_{i} p(x_t = i | y_0:T) \log b_i(y_t) = \sum_{i} \gamma_0^k(i) \log \pi_i + \sum_{t=1}^T \sum_{i,j} \xi_{t-1}^k(i,j) \log a_{i,j} + \sum_{t=0}^T \sum_{i} \gamma_t^k(i) \log b_i(y_t) = \sum_{i} \gamma_0^k(i) \log \pi_i + \sum_{i,j} T \sum_{t=1}^T \xi_{t-1}^k(i,j) \log a_{i,j} + \sum_{t=0}^T T \sum_{i} \gamma_t^k(i) \log b_i(y_t) (15)
\]
where $\gamma_k^t, \xi_k^t$ are computed using $\theta^k$ in the recursions given in (8) and (9).

We need to maximize the above subject to the constraints

\[
\sum_i \pi_i = 1 \\
\sum_j a_{i,j} = 1, \ \forall i = 1, \ldots N \\
\sum_{y=1}^{M} b_i(y) = 1, \ \forall i = 1, \ldots N
\]  

(16)

Using Lagrange multipliers, differentiating and solving, the final solutions are

\[
\pi_i^{k+1} = \gamma_0^k(i) \\
a_{i,j}^{k+1} = \frac{1}{\sum_{j'=1}^{N} \sum_{t=1}^{T} \xi_{t-1}^k(i, j')} (\sum_{t=1}^{T} \xi_{t-1}^k(i, j)) = \frac{1}{\sum_{t=1}^{T} \gamma_t(i)} (\sum_{t=1}^{T} \xi_{t-1}^k(i, j)) \\
b_i(m)^{k+1} = \frac{1}{\sum_{t=0}^{T} \gamma_t^k(i)} \sum_{t=0}^{T} I(y_t = m) \gamma_t^k(i)
\]  

(17)

where $I(A)$ is 1 if $A$ occurs and 0 otherwise. Here $\gamma_k^t, \xi_k^t$ are computed using the parameter estimates at iteration $k$ in the recursions given earlier.

Thus, the stepwise EM algorithm is as follows. At iteration $k + 1$,

1. Compute $\gamma_k^t(i)$ for all $i$ for all $t$ using (8) and parameter estimates from iteration $k$, $\theta^k$.
2. Compute $\xi_k^t(i, j)$ for all $i, j$ for all $t$ using (9) and parameter estimates from iteration $k$, $\theta^k$.
3. Compute parameter estimates at iteration $k + 1$, $\theta^{k+1}$, using (17).

Now, if $Y_t$’s are not discrete, but are continuous r.v.’s with parameters of their PDF being governed by the current state, e.g. $Y_t$’s can be scalar Gaussians with mean $\mu_i$ and variance $\sigma_i^2$ if the state $X_t = i$. In this case, their estimates can be computed as follows.

We need to maximize the following w.r.t. $\mu_i, \sigma_i^2$.

\[
\sum_i \sum_{t=0}^{T} \gamma_t^k(i) \log b_i(y_t) = \sum_i \sum_{t=0}^{T} \gamma_t^k(i) \left[-\log(\sqrt{2\pi})\sigma_i^2 - \frac{(y_t - \mu_i)^2}{2\sigma_i^2}\right]
\]  

(18)

Thus,

\[
\mu_i^{k+1} = \frac{1}{\sum_{t=0}^{T} \gamma_t^k(i)} \sum_{t=0}^{T} \gamma_t^k(i) y_t \\
\sigma_i^{2(k+1)} = \frac{1}{\sum_{t=0}^{T} \gamma_t^k(i)} \sum_{t=0}^{T} \gamma_t^k(i) (y_t - \mu_i^{k+1})^2
\]  

(19)
2.5 General idea of Viterbi algorithm / dynamic programming

In dynamic programming / Viterbi algorithm, the goal is to find

$$\arg \max_{q_0:T} f_T(q_0:T)$$

(20)

where $f_t(q_0:t)$ at any $t$ (note that $f_t$ is a function only of the first $t + 1$ variables) satisfies

$$f_t(q_0:t) = f_{t-1}(q_0:t-1) + h_t(q_{t-1}, q_t) + g_t(q_t)$$

(21)

Typically path optimization problems are of this type.

Efficient solutions strategy: Let

$$\delta_t(q_t) \triangleq \max_{q_0:t-1} f_t(q_0:t)$$

(22)

Then, using (21),

$$\delta_t(q_t) = \max_{q_0:t-1} [f_{t-1}(q_0:t-1) + h_t(q_{t-1}, q_t) + g_t(q_t)]$$

$$= \max_{q_{t-1}} \max_{q_0:t-2} [f_{t-1}(q_0:t-1) + h_t(q_{t-1}, q_t) + g_t(q_t)]$$

$$= g_t(q_t) + \max_{q_{t-1}} [h_t(q_{t-1}, q_t) + \delta_{t-1}(q_{t-1})]$$

(23)

Also store the optimal path to get to $q_t$ for each value of $q_t$. So if $q_t \in \{1, 2, \ldots N\}$ then store the optimal path to get to $q_t = i$ for each $i$ in the set. For the above problem, this can be done efficiently by only storing the optimal value of $q_{t-1}$ that gets you to $q_t = i$ and doing this for each $i$ at each $t$. Thus, at each $t$, for each $q_t$ value, we store

$$\psi_t(q_t) \triangleq \arg \max_{q_{t-1}} [h_t(q_{t-1}, q_t) + \delta_{t-1}(q_{t-1})]$$

(24)

To summarize the above idea, we do the following.

1. Initialize at $t = 0$ to $\delta_0(i) = f_0(i)$.

2. Starting at $t = 1$, at each $t$, compute the following for all $i = 1, 2 \ldots N$

$$\delta_t(i) = g_t(i) + \max_{q_{t-1}} [h_t(q_{t-1}, i) + \delta_{t-1}(i)]$$

(25)

3. Simultaneously, at each $t$, for each $i = 1, 2 \ldots N$ also store the maximizer of the above, i.e. store

$$\psi_t(i) = \arg \max_{q_{t-1}} [h_t(q_{t-1}, i) + \delta_{t-1}(i)]$$

(26)
4. At $t = T$, find the optimal cost and the optimal value of $q_T$ as

$$
\max_i \delta_T(i),
q_T^* = \arg \max_i \delta_T(i)
$$

(27)

5. Backtrack using $\psi_t$ to find the optimal state sequence, i.e. starting with $t = T - 1$, go backwards,

$$
q_t^* = \psi_{t+1}(q_{t+1}^*)
$$

(28)

### 2.6 Posterior MAP (non-causal) computation: Viterbi algorithm

The non-causal posterior MAP estimate is

$$
x_{0:T}^* \triangleq \arg \max_{x_{0:T}} p(x_{0:T}|y_{0:T})
$$

(29)

Recall that

$$
\delta_t(x_t) \triangleq \max_{x_{0:t-1}} p(x_{0:t}, y_{0:t})
\psi_t(x_t) \triangleq \arg \max_{1 \leq x_{t-1} \leq N} \delta_{t-1}(x_{t-1})a_{x_{t-1}, x_t}
$$

(30)

Recursion for $\delta_t$

$$
\delta_t(i) \triangleq \max_{x_{0:t-1}} p(x_{0:t}, y_{0:t})
= \max_{x_{0:t-1}} p(x_{0:t-1}, y_{0:t-1})p(x_t = i|x_{0:t-1}, y_{0:t-1})p(y_t|x_t, x_{0:t-1}, y_{0:t-1})
= \max_{x_{0:t-1}} p(x_{0:t-1}, y_{0:t-1})p(x_t = i|x_{t-1})p(y_t|x_t = i) \quad \text{(using HMM properties 1a, 1b)}
= \max_{x_{0:t-1}} p(x_{0:t-1}, y_{0:t-1})a_{x_{t-1}, i}b_i(y_t)
= b_i(y_t) \max_{x_{t-1}} \max_{y_{0:t-2}} \delta_{t-1}(x_{t-1})a_{x_{t-1}, i}
= b_i(y_t) \max_j \delta_{t-1}(j)a_{j,i}
$$

(31)

Also,

$$
\psi_t(i) = \arg \max_j \delta_{t-1}(j)a_{j,i}
$$

(32)

Thus,

$$
x_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)
q_T^* = \psi_{t+1}(q_{t+1}^*), \forall t = T - 1, T - 2, \ldots 0
$$

(33)