CSIS8502

7. Hidden Markov Models

Hidden Markov Models

- In Problem with inherent temporality, we may have states at time t that are influenced directly by state at $t-1$.
- There are transition from one state to another (with certain probability) — Markov Model
- The states have a certain probability of generating various output symbols — the observations.
- Human can only see the obervation, but not the underlying Markov Model. (Hence Hidden)
- For example, HMM has been used in Speech Recognition, Handwritten Character Recognition.

First-order Markov Models

- The state at any time t is denoted $\omega(t)$.
- A sequence of states of length T is denoted by

$$\omega^T = \{\omega(1), \omega(2), \cdots, \omega(T)\}$$

- The system can revisit a state at diffferent step, e.g.

$$\omega^6 = \{\omega_1, \omega_4, \omega_2, \omega_2, \omega_1, \omega_4\}$$

- Production of any sequence is described by the *transition probability*:

$$P(\omega_j(t+1)|\omega_i(t)) = a_{ij}$$

- Transition prob. need not be symmetric, i.e. $a_{ij} \neq a_{ji}$ in general.
- A Markov Model, θ: the full set of a_{ij}.
- Given the model θ, the prior prob. of the 1st state $P(\omega(1) = \omega_i)$, we can compute the prob. of a particular sequence ω^T.

Markov Model

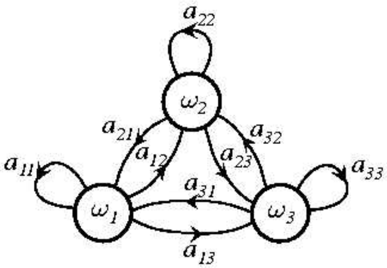

FIGURE 3.8. The discrete states, ω_i, in a basic Markov model are represented by nodes, and the transition probabilities, a_{ij}, are represented by links. In a first-order discrete-time Markov model, at any step t the full system is in a particular state $\omega(t)$. The state at step $t+1$ is a random function that depends solely on the state at step t and the transition probabilities. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Example

For example, in speech:

- The Markov model for the word ”cat”,
  - States for /k/, /a/, and /t/
  - Transition from /k/ to /a/,; transition from /a/ to /t/; and transition from /t/ to silent.
- Other possible applications
  - Online character recognition.
  - Facial Expression Recognition

First Order Hidden Markov Model

- We can observe some visible symbols $v(t)$ at time t.
- However, the underlying state is unknown, i.e. hidden.
- in any state $\omega(t)$, we have a probability of emitting a particular visible state $v_k(t)$, i.e. the same state may emit different symbols, and the same symbol may be emitted by different states.
- We denote this prob.

$$P(v_k(t)|\omega_j(t)) = b_{jk}$$

- Because we can only observe the visible states, while the ω_i are unobservable, it is called *Hidden Markov Model*.

HMM Computation

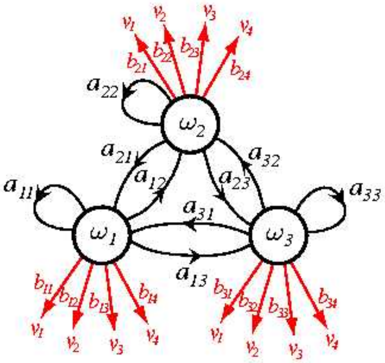

FIGURE 3.9. Three hidden units in an HMM and the transitions between them are shown in black while the visible states and the emission probabilities of visible states are shown in red. This model shows all transitions as being possible; in other HMMs, some such candidate transitions are not allowed. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

HMM Computation

- The underlying network is a finite state machine, and when associated with transition probabilities, they are called *Markov networks*.
- They are strictly causal (the prob. depend only on previous states).
- A Markov model is called *ergodic* if every one of the states has a nonzero prob. of occurring, given some starting states.
- A *final* or *absorbing* state ω_0 is one which if entered, is never left. (i.e. $a_{00} = 1$).
- We require some transition must occur at each step (may be to the same state), and some symbol must be emitted. Thus, we have the normalized conditions:

$$\sum_j a_{ij} = 1 \quad \forall i \text{ and } \sum_k b_{jk} = 1 \quad \forall j$$

Central Issues

There are 3 central issues:

- **Evaluation Problem**
  Given a_{ij} and b_{jk}, Determine the prob. that a particular sequence of visible states V^T was generated by that model.
- **Decoding Problem**
  Given the model and a set of observation V^T, determine the most likely sequence of hidden state ω_T that led to those observation.
- **Learning Problem**
  Given the coarse structure of the model (i.e. number of states, number of visible symbols), and a set of training observation of visible symbols, determine the parameters a_{ij} and b_{jk}.

Evaluation

- The prob. of the model produces a sequence V^T of visible state is
$$P(V^T) = \sum_{r=1}^{r_{\max}} P(V^T|\omega_r^T)P(\omega_r^T)$$
where each r indexes a particular sequence
$$\omega_t = \{\omega(1), \omega(2), \cdots, \omega(T)\}$$
of T hidden states.
- In the general case of c hidden states, there will be $r_{\max} = c^T$ possible terms.
- Since we are working with a 1st order Markov prcoess,
$$P(\omega_r^T) = \prod_{t=1}^{T} P(\omega(t))|\omega(t-1))$$
i.e. a products of a_{ij}'s.

Evaluation

- Note that the output symbol only depends on the hidden states, we can write

$$P(V^T|\omega_r^T) = \prod_{t=1}^{T} P(v(t)|\omega(t))$$

  i.e. a product of b_{jk}'s.
- Hence

$$P(V^T) = \sum_{r=1}^{r_{\max}} \prod_{t=1}^{T} P(v(t)|\omega(t))P(\omega(t)|\omega(t-1))$$

- Interpretation: The prob. that we observe states V^T is equal to the sum over all $r_{\max}$ possible sequence of hidden states of the conditional prob. that the system has made a particular transition, multiplied by the prob. that it then emitted the visible symbol in the target sequence.
- Extremely high computation if computed directly.

Evaluation

- We can compute recursively. Define

$$\alpha_j(t) = \begin{cases} 0 & t = 0 \text{ and } i \neq \text{initial state} \\ 1 & t = 0 \text{ and } i = \text{initial state} \\ \left[\sum_i \alpha_i(t-1)a_{ij}\right] b_j(v(t)) & \text{otherwise} \end{cases}$$

- $\alpha_j(t)$ denotes the probability of observing the sequence up to time t, and ending in state j.

$$\alpha_j(t) = P(v(1)v(2)\cdots v(t), I_t = i)$$

- For the final state ω_0, we return $\alpha_0(T)$ for the final state.
- Computation Complexity $O(c^2T)$.

Evaluation

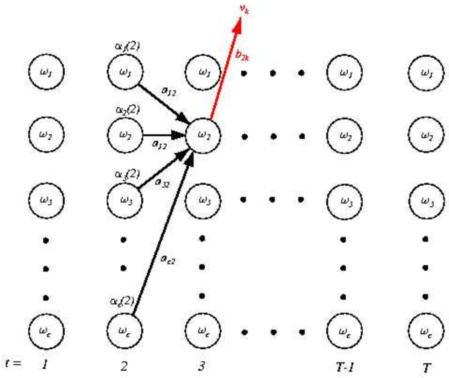

FIGURE 3.10. The computation of probabilities by the Forward algorithm can be visualized by means of a trellis—a sort of "unfolding" of the HMM through time. Suppose we seek the probability that the HMM was in state ω_2 at $t = 3$ and generated the observed visible symbol up through that step (including the observed visible symbol v_k). The probability the HMM was in state $\omega_j(t = 2)$ and generated the observed sequence through $t = 2$ is $\alpha_j(2)$ for $j = 1, 2, \ldots, c$. To find $\alpha_2(3)$ we must sum these and multiply the probability that state ω_2 emitted the observed symbol v_k. Formally, for this particular illustration we have $\alpha_2(3) = b_{2k} \sum_{j=1}^{c} \alpha_j(2)a_{j2}$. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Algorithm — HMM Forward

- Initialize: $t = 0$, a_{ij}, b_{jk}, visible sequence V^T, and $\alpha_j(0)$
- For $t \leftarrow t + 1$

$$\alpha_j(t) \leftarrow \sum_{i=1}^{c} \left[\alpha_i(t-1)a_{ij}\right] b_j(v(t))$$

  Until $t = T$.
- Return $P(V^T) \leftarrow \alpha_0(T)$ for the final state.
- end.

Backward Procedure

- We define backward variable:

$$\beta_j(t) = P(v(t+1), v(t+2), \cdots, v(T) | I_t = j)$$

- $\beta_j(t)$ is the probability of starting from state j at time t, going through the observations and reach the final state.

Algorithm — HMM Backward

- Initialize $t = T$, a_{ij}, b_{jk}, visible sequence V^T, and $\beta_j(T)$
- For $t \leftarrow t - 1$

$$\beta_i(t) \leftarrow \sum_{i=0}^{c} \beta_j(t+1) a_{ij} b_j(v(t+1))$$

  Until $t = 1$.
- Return $P(V^T) \leftarrow \beta_i(0)$ for the known initial state
- end.

Recognition by HMM

- In a recognition task, we have a number of HMMs, usually one for each class. We have to apply the Bayes rule:

$$P(\theta|V^T) = \frac{P(V^T|\theta)P(\theta)}{P(V^T)}$$

  as the evaluation process only returns $P(V^T|\theta)$, where θ is the classification. (the symobl ω has been used)
- Nearly all HMMs for speech recognition are left-to-right models.

Example

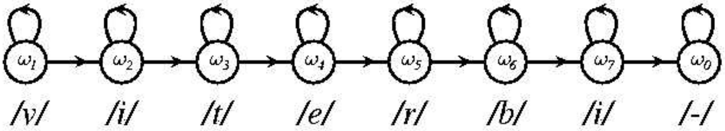

FIGURE 3.11. A left-to-right HMM commonly used in speech recognition. For instance, such a model could describe the utterance "viterbi," where ω_1 represents the phoneme /v/, ω_2 represents /i/,..., and ω_0 a final silent state. Such a left-to-right model is more restrictive than the general HMM in Fig. 3.9 because it precludes transitions "back" in time. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Decoding

- to find the most probable sequence of hidden states.
- Brute force: enumerating every possible path, and calculate the prob.
- complexity: $O(c^T T)$.
- Figure 3.12 shows a simple decoding method.
- can use logarithm of prob, as it involves products of prob. which can cause underflow error.
- time complexity: $O(c^2 T)$.

Example

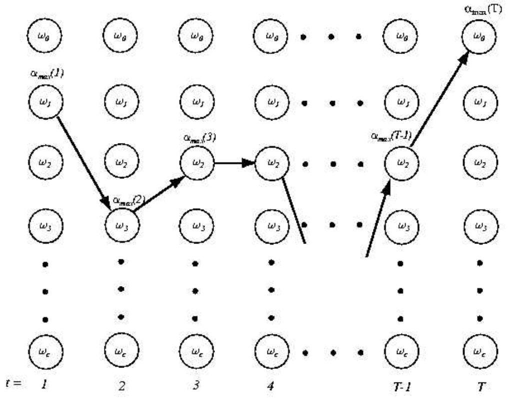

FIGURE 3.12. The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k. The full path is the sequence of such states. Because this is a local optimization (dependent only upon the single previous time step, not the full sequence), the algorithm does not guarantee that the path is indeed allowable. For instance, it might be possible that the maximum at $t = 5$ is ω_1 and at $t = 6$ is ω_2, and thus these would appear in the path. This can even occur if $a_{12} = P(\omega_2(t+1)|\omega_1(t)) = 0$, precluding that transition. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Viterbi Decoding Algorthm

- Essentially a Dynamic Programming algorithm
- Assume the output sequence $O = v(1)v(2)...v(T)$.
- Let the sequence of states $I = \omega(1)\omega(2)\cdots\omega(T)$
- We have

$$\begin{aligned} P(O, I) &= P(O|I)P(I) \\ &= b_{i_1}(v(1))a_{i_1 i_2}b_{i_2}(v(2))\cdots a_{i_{T-1}i_T}b_{i_T}(v(T)) \end{aligned}$$

- Taking logarithm,

$$\log P(O, i) = \log b_{i_1}(v(1)) + \sum_{t=2}^{T} \log(a_{i_{t-1}i_t}b_{i_t}(v(t)))$$

Viterbi Decoding Algorthm

- To find

$$\max_{I} P(O, I) = \max_{i_1 \cdots i_T} P(O, i_1, \cdots, i_T)$$

- Denote

$$U(i_1, i_2, \cdots i_T) = -\left[\log(b_{i_1}(v(1))) + \sum_{t=2}^{T} \log(a_{i_{t-1} i_t} b_{i_t}(v(t)))\right]$$

It becomes

$$\min_{i_1, \cdots, i_T} U(i_1, i_2, \cdots, i_T)$$

- We can use dynamic programming to achieve this.

Viterbi Decoding Algorthm

- Let $W(k, t)$ be the accumulated smallest U value (as in previous slide, i.e. largest prob.) up to time t, generating the observed output and residing in state k.
- Initial weight

$$W(k, 1) = -\log(a_{i_0 k} b_k(v(1)))$$

- Dynamic Programming Steps:

$$W(k.t) = \min_i (W(i, t-1) - \log(a_{ik} b_k(v(t))))$$

- This is repeated until we reach the final state at $t = T$.
- The path can be traced backward, just like other dynamic programming algorithm.
- This algorithm has a complexity of $O(c^2 T)$.

Learning

- to determine the model parameters, a_{ij} and b_{jk} from training data.
- Use Baum-Welch or forward-backward algorithm: an instance of a generalized expectation maximization algorithm
- iteratively update the parameters in order to better explain the observed training sequence.
- We can define $\alpha_i(t)$ as the prob. that the model is in state $\omega_i(t)$ and has generated the target sequence up to step t, which has been defined before.

Baum-Welch Algorithm

- We can define $\beta_i(t)$ as the prob. that the state is in $\omega_i(t)$ and will generate the remainder of the given sequence.

$$\beta_i(t) = \begin{cases} 0 & \omega_i(t) \neq \text{sequence's final state} \\ 1 & \omega_i(t) = \text{sequence's final state} \\ \sum_j \beta_j(t+1) a_{ij} b_j(v(t+1)) & \text{otherwise} \end{cases}$$

- Imagine we know $\beta_i(t)$ up to step T, then the prob. of transition back to $T-1$ will be the prob. of making a transition to $\omega_i(t)$ and the prob. that the final visible symbol is emitted from this state. Hence.

$$\beta_i(T-1) = \sum_j a_{ij} b_j(v(T)) \beta_j(T)$$

We can then go backward from $T-1$ to $T-2 \cdots$

Baum-Welch Algorithm

- Note that these values are calculated based on a_{ij} and b_{jk}, which is just an estimation. We have to update their value based on the new α's and β's.
- We can define the prob. of transition between $\omega_i(t-1)$ and $\omega_j(t)$, given the model generated the entire training sequence V^T by any path:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{jk}\beta_j(t)}{P(V^T|\theta)}$$

  where $b_{jk} = b_j(k)$.
- $P(V^T|\theta)$ is the prob. that the model generated sequence V^T by any path.

Baum-Welch Algorithm

- The expected number of transition between state $\omega_i(t-1)$ and $\omega_j(t)$ at any time in the sequence is $\sum_{t=1}^{T} \gamma_{ij}(t)$, whereas the total expected number of any transition from ω_i is $\sum_{t=1}^{T} \sum_{k} \gamma_{ik}(t)$.
- Hence the improved version of a_{ij} is given by:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{k} \gamma_{ik}(t)}$$

Baum-Welch Algorithm

- Similarly, the improved version of b_{jk} is given by:

$$\hat{b}_{jk} = \frac{\sum_{\substack{t=1 \\ v(t)=v_k}}^{T} \sum_{l} \gamma_{lj}(t)}{\sum_{t=1}^{T} \sum_{l} \gamma_{lj}(t)}$$

- Repeat until convergence.

Baum-Welch Algorithm

- Begin initialize a_{ij}, b_{jk}, training sequence V^T, and convergence criteria θ
  - Do $z \leftarrow z + 1$
  - compute $a(z)$ from $a(z-1)$ and $b(z-1)$
  - compute $b(z)$ from $a(z-1)$ and $b(z-1)$
  - $a_{ij}(z) \leftarrow a_{ij}(z-1)$
  - $b_{ij}(z) \leftarrow b_{ij}(z-1)$
- until

$$\max_{i,j,k} [a_{ij}(z) - a_ij(z-1), b_{jk}(z) - b_{jk}(z-1)] < \theta$$

- Return $a_{ij} \leftarrow a_{ij}(z)$; $b_{jk} \leftarrow b_{jk}(z)$.

HMM Toolkit (HTK)

- HMM Tool Kit developed by Machine Intelligence Laboratory in Cambridge University.
- A portable toolkit for building and manipulating HMMs.
- Current Version 3.4
- Available at:
  `http://htk.eng.cam.ac.uk/download.shtml`