Hidden Markov Model

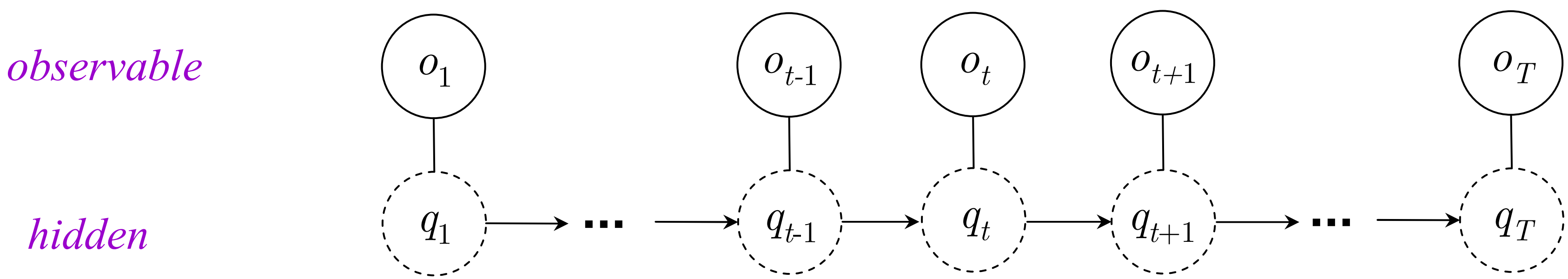
Prof. Yan Wang
Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.
yan.wang@me.gatech.edu

Learning Objectives

- To familiarize the hidden Markov model as a generalization of Markov chain
- To understand the three basic problems (evaluation, decoding, and learning) in HMM model construction and applications

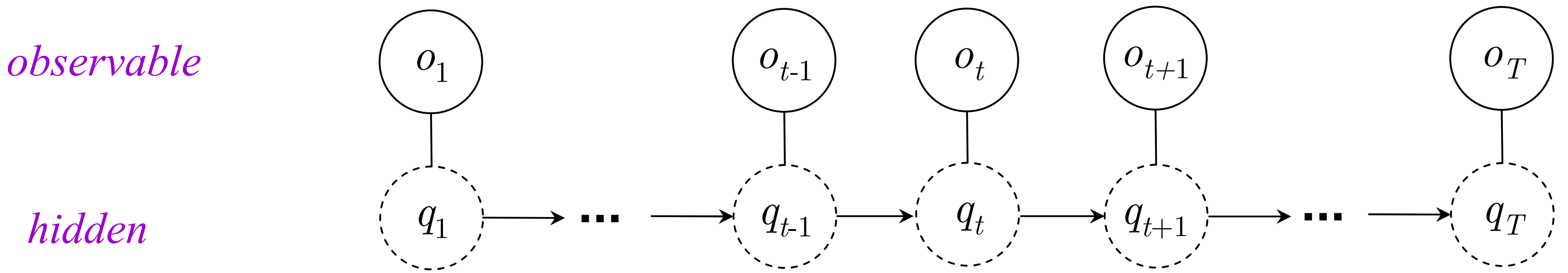
Hidden Markov Model (HMM)

- HMM is an extension of regular Markov chain
- State variables q's are not directly observable
- All statistical inference about the Markov chain itself has to be done in terms of observable o's



HMM

- o's are conditionally independent given $\{q_t\}$.
- However, $\{o_t\}$ is not an independent sequence, nor a Markov chain itself.



HMM Components

- **State sequence:** $Q=\{q_1, q_2, \ldots, q_T\}$ with N possible values
- **Observation sequence:** $O=\{o_1, o_2, \ldots, o_T\}$ with M possible symbols $\{v_1, v_2, \ldots v_M\}$
- **State transition matrix:** $\mathbf{A}=(a_{ij})_{N\times N}$ where $a_{ij}=P(q_{t+1}=j|\,q_t=i)$
- **Observation matrix:** $\mathbf{B}=(b_{ik})_{N\times M}$ where $b_{ik}=b_i(v_k)=P(o_t=v_k|\,q_t=i)$
- **Initial state distribution:** $\boldsymbol{\Delta}=(\delta_i)_{1\times N}$ where $\delta_i=P(q_1=i)$

Model's parameters $\boldsymbol{\lambda}=\{\mathbf{A}, \mathbf{B}, \boldsymbol{\Delta}\}$

Three Basic Problems in HMM

- **Evaluation**: Given a model with parameters λ and a sequence of observations O=$\{o_1, o_2, \ldots, o_T\}$, what is the probability that the model generates those observations P(O|λ)?
- **Decoding**: Given a model with parameters λ and a sequence of observations O=$\{o_1, o_2, \ldots, o_T\}$, what is the most likely state sequence Q=$\{q_1, q_2, \ldots, q_T\}$ in the model that produces the observations?
- **Learning**: Given a set of observations O=$\{o_1, o_2, \ldots, o_T\}$, how can we find a model with the parameters λ with the maximum likelihood P(O|λ)?

Evaluation Problem

- Given an O={$o_1, o_2, \ldots, o_T$}, P(O|λ)=?
- Naïve algorithm
  - because of lemma 3.1

$$P(O \mid \lambda) = \sum_{Q^{(d)}} P(O, Q^{(d)} \mid \lambda) = \sum_{Q^{(d)}} P(O \mid Q^{(d)}, \lambda) P(Q^{(d)} \mid \lambda)$$

where $Q^{(d)}$ is one of all possible combinations of state sequences

  - assume conditional independence between observations

$$P(O \mid Q^{(d)}, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = \prod_{t=1}^{T} b_{q_t}(o_t)$$

$$P(Q^{(d)} \mid \lambda) = \delta_{q_1} \prod_{t=1}^{T-1} P(q_{t+1} \mid q_t, \lambda) = \delta_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}$$

  - However, the number of possible combinations of state sequences is huge!

Evaluation Problem – Forward Algorithm

- Define a forward variable $\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i \mid \lambda)$
- which can be recursively calculated by

$$\alpha_1(i) = P(o_1, q_1 = i \mid \lambda) = P(q_1 = i \mid \lambda) P(o_1 \mid q_1 = i, \lambda) = \delta_i b_i(o_1)$$

$$\begin{aligned}
\alpha_{t+1}(i) &= P(o_1, \ldots, o_{t+1}, q_{t+1} = i \mid \lambda) \\
&= P(q_{t+1} = i \mid \lambda) P(o_1, \ldots, o_{t+1} \mid q_{t+1} = i, \lambda) \\
&= P(q_{t+1} = i \mid \lambda) P(o_{t+1} \mid q_{t+1} = i, \lambda) P(o_1, \ldots, o_t \mid q_{t+1} = i, \lambda) \\
&= P(o_{t+1} \mid q_{t+1} = i, \lambda) P(o_1, \ldots, o_t, q_{t+1} = i \mid \lambda) \\
&= P(o_{t+1} \mid q_{t+1} = i, \lambda) \sum_{q_t} P(o_1, \ldots, o_t, q_t, q_{t+1} = i \mid \lambda) \\
&= P(o_{t+1} \mid q_{t+1} = i, \lambda) \sum_{q_t} P(q_{t+1} = i \mid q_t, \lambda) P(o_1, \ldots, o_t, q_t \mid \lambda) \\
&= b_i(o_{t+1}) \sum_{q_t} a_{q_t, i} \alpha_t(q_t)
\end{aligned}$$

$$\boxed{P(O \mid \lambda) = \sum_{i=1}^{N} P(O, q_T = i \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)}$$

Evaluation Problem – Backward Algorithm

- Define a backward variable $\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = i, \lambda)$
- which can be recursively calculated by

$$\beta_T(i) = 1$$

$$\begin{aligned}
\beta_t(i) &= P(o_{t+1}, \ldots, o_T \mid q_t = i, \lambda) \\
&= P(o_{t+1} \mid q_t = i, \lambda) P(o_{t+2}, \ldots, o_T \mid q_t = i, \lambda) \\
&= P(o_{t+1} \mid q_t = i, \lambda) P(o_{t+2}, \ldots, o_T, q_t = i \mid \lambda) \,/\, P(q_t = i \mid \lambda) \\
&= P(o_{t+1} \mid q_t = i, \lambda) \sum_{q_{t+1}} P(o_{t+2}, \ldots, o_T, q_t = i, q_{t+1} \mid \lambda) \,/\, P(q_t = i \mid \lambda) \\
&= P(o_{t+1} \mid q_t = i, \lambda) \sum_{q_{t+1}} \frac{P(o_{t+2}, \ldots, o_T \mid q_t = i, q_{t+1}, \lambda) P(q_t = i, q_{t+1} \mid \lambda)}{P(q_t = i \mid \lambda)} \\
&= P(o_{t+1} \mid q_t = i, \lambda) \sum_{q_{t+1}} P(o_{t+2}, \ldots, o_T \mid q_{t+1}, \lambda) P(q_{t+1} \mid q_t = i, \lambda) \\
&= b_i(o_{t+1}) \sum_{q_{t+1}} a_{i,q_{t+1}} \beta_{t+1}(q_{t+1})
\end{aligned}$$

Decoding Problem

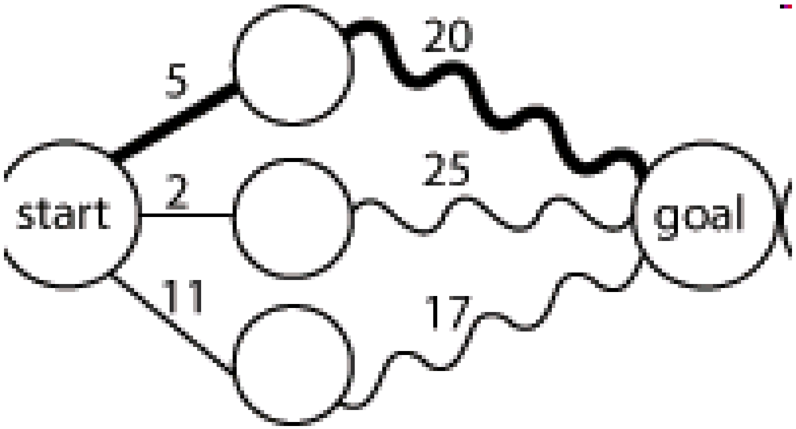
- Given an O=$\{o_1, o_2, \ldots, o_T\}$, find a Q*=$\{q_1^*, q_2^*, \ldots, q_T^*\}$ with the maximum of P(O|Q)
- Viterbi Algorithm is a dynamic programming method to solve the decoding problem

Georgia Tech

Dynamic Programming

- Breaking down complex problems into subproblems in a recursive manner.

- e.g. search the shortest path in the Traveling Salesman Problem



Decoding Problem – Viterbi Algorithm

- Define an auxiliary probability

$$\rho_t(i) := \max_{q_1,\dots,q_{t-1}} P(q_1,\dots,q_{t-1},q_t = i, o_1,\dots,o_t \mid \lambda)$$

which is the highest probability that a single path leads to $q_t = i$ at time t

- Recursively

$$\begin{aligned}\rho_{t+1}(j) &= \max_i \left\{\rho_t(i)P(q_{t+1} = j \mid q_t = i,\lambda)P(o_{t+1} \mid q_{t+1} = j)\right\} \\ &= \max_i \left\{\rho_t(i)P(q_{t+1} = j \mid q_t = i,\lambda)\right\} P(o_{t+1} \mid q_{t+1} = j) \\ &= \max_i \left\{\rho_t(i)a_{ij}\right\} b_j(o_{t+1})\end{aligned}$$

Decoding Problem – Viterbi Algorithm – cont'd

- Define an auxiliary variable

  $$\psi_{t+1}(j) = \arg\max_i \left\{\rho_t(i) a_{ij} b_j(o_{t+1})\right\} = \arg\max_i \left\{\rho_t(i) a_{ij}\right\}$$

  to store the optimal state at time t to reach state j at time $t+1$.

- The algorithm is
  - 1. initialize

    $$\rho_1(j) = \delta_j b_j(o_1) \quad \left(\forall j, j = 1, \ldots, N\right)$$

    $$\psi_1(j) = 0$$

  - 2. recursion

    $$\rho_{t+1}(j) = \max_i \left\{\rho_t(i) a_{ij}\right\} b_j(o_{t+1})$$

    $$\psi_{t+1}(j) = \arg\max_i \left\{\rho_t(i) a_{ij}\right\}$$

Decoding Problem – Viterbi Algorithm – cont'd

- 3. terminate
  - The optimal probability $P^* = \max_j \left\{ \rho_T(j) \right\}$
  - The optimal final state $q_T^* = \arg\max_j \left\{ \rho_T(j) \right\}$
- 4. backtrack state sequence

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

Learning Problem

- Given an $O=\{o_1, o_2, \ldots, o_T\}$, find a $\lambda^*=\{\mathbf{A}^*, \mathbf{B}^*, \mathbf{\Delta}^*\}$ with the maximum likelihood $P(O|\lambda)$

$$\lambda^* = \arg\max_{\lambda} \left\{ P(O \mid \lambda) \right\} = \arg\max_{\lambda} \left\{ \log P(O \mid \lambda) \right\}$$

- Global optimum needs to search all possible state and observation sequences

$$\lambda^* = \arg\max_{\lambda} \left\{ \sum_{O^{(d)}} \sum_{Q^{(d)}} \log P(O^{(d)} \mid Q^{(d)}, \lambda) \right\}$$

- Instead, Baum-Welch Algorithm is usually used to search heuristically

Learning Problem – Baum-Welch Algorithm

- Algorithm
  - 1.Guess some parameters λ
  - 2. Determine some "probable paths" $\{Q^{(1)},\ldots,Q^{(d)}\}$
  - 3. Estimate the number of transitions $\hat{a}_{ij}$, from state i to state j, given the current estimate of λ.
  - 4. Estimate the number of the observation v_k emitted from state i as $\hat{b}_i(v_k)$
  - 5. Estimate the initial distribution $\hat{\delta}_i$
  - 6. Re-estimate λ' from A_{ij}'s and $B_i(v_k)$'s
  - 7. If λ' and λ is close enough, stop; otherwise, assign λ = λ' and go back to step 2.

Learning Problem – Baum-Welch Algorithm – cont'd

- Define an auxiliary likelihood

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid o_1, \ldots, o_T, \lambda)$$

which is the probability that a transition $q_t = i$ and $q_{t+1} = j$ occurs at time t given the complete observations $\{o_1, o_2, \ldots, o_T\}$ and model parameters λ

Learning Problem – Baum-Welch Algorithm – cont'd

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid o_1, \ldots, o_T, \lambda)$$

$$= \frac{P(q_t = i, q_{t+1} = j, o_1, \ldots, o_T \mid \lambda)}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{P(o_1, \ldots, o_T \mid q_t = i, q_{t+1} = j, \lambda) P(q_t = i, q_{t+1} = j \mid \lambda)}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{\begin{bmatrix} P(o_1, \ldots, o_t \mid q_t = i \mid \lambda) P(o_{t+1} \mid q_{t+1} = j, \lambda) \\ P(o_{t+2}, \ldots, o_T \mid q_{t+1} = j \mid \lambda) P(q_{t+1} = j \mid q_t = i, \lambda) P(q_t = i \mid \lambda) \end{bmatrix}}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{\begin{bmatrix} P(o_1, \ldots, o_t, q_t = i \mid \lambda) P(q_{t+1} = j \mid q_t = i, \lambda) \\ P(o_{t+1} \mid q_{t+1} = j, \lambda) P(o_{t+2}, \ldots, o_T \mid q_{t+1} = j \mid \lambda) \end{bmatrix}}{P(o_1, \ldots, o_T \mid \lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_i(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_T(i)}$$

Learning Problem – Baum-Welch Algorithm – cont'd

- Estimate parameters

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} P(q_t = i, q_{t+1} = j \mid o_1, \ldots, o_T, \lambda)}{\sum_{t=1}^{T} P(q_t = i \mid o_1, \ldots, o_T, \lambda)} = \frac{\sum_{t=1}^{T} \xi_t(i, j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \xi_t(i, k)}$$

$$\hat{b}_i(v_k) = \frac{\sum_{t=1}^{T} P(o_t = v_k, q_t = i, q_{t+1} = j \mid o_1, \ldots, o_T, \lambda)}{\sum_{t=1}^{T} P(q_t = i \mid o_1, \ldots, o_T, \lambda)}$$

$$= \frac{\sum_{t=1}^{T} 1_{o_t = v_k} \xi_t(i, j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \xi_t(i, k)}$$

$$\hat{\delta}_i = \sum_{k=1}^{N} P(q_1 = i, q_2 = k \mid o_1, \ldots, o_T, \lambda) = \sum_{k=1}^{N} \xi_1(i, k)$$

Learning Problem – Baum-Welch Algorithm – cont'd

- How to measure two models, λ' and λ, are close enough?

$$p = \sum_{O^{(d)}} P(O^{(d)} \mid \lambda)$$

$$Cross\,Entropy = p_1(x)\log\frac{p_1(x)}{p_2(x)}$$

HMM Applications

- HMM has been applied in many fields that are based on the analysis of discrete-valued time series, such as
  - speech recognition (Rabiner, 1989)
  - Image recognition
  - genetic profile and classification (Eddy, 1998)

Kalman Filter

- Can be regarded as a special case of HMM
- Also known as the *Gaussian linear state-space* model
- State series are linearly dependent on history, subject to process white noises.

$$\mathbf{x}_t = \mathbf{C}\mathbf{x}_t + \mathbf{w}_t$$

- Observations are also linearly dependent on states, subject to measurement noises.

$$\mathbf{y}_t = \mathbf{D}\mathbf{x}_t + \mathbf{v}_t$$

Summary

- Hidden Markov model is a generalization Markov chain

Observable

Hidden

Further Readings

- Rabiner L.R. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2): 267-296
- Eddy S.R. (1998) Profile hidden Markov models. *Bioinformatics Review*, **14**(9): 755-776
- Dempster A.P., Laird N.M., and Rubin D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*. **39**(1): 1-38
- Wu C.F.J. (1983) On the convergence properties of the EM algorithm. *The Annals of Statistics*, **11**(1): 95-103
- HMM Software Packages
  http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html