Hidden Markov Models

Instructor: Jessica Wu -- Harvey Mudd College

Robot Image Credit: Viktoriya Sukhanova © 123RF.com

Based on tutorial by Lawrence Rabiner

Basic Problems for HMMs

Use the compact notation $\lambda = (A, B, \pi)$.

	single path	all paths
scoring / evaluation	scoring O, one path q $P(q, O \mid \lambda)$ probability of a path & observations	scoring O, all paths $P(O) = \Sigma_q P(q, O \mid \lambda)$ probability of observations over all paths [forward-backward algorithm]
decoding	most likely path q^* $q^* = \text{argmax}_q P(q, O \mid \lambda)$ [Viterbi decoding]	path containing most likely state at any time point $\hat{q} = \{q_t \mid q_t = \text{argmax}_{S_i} P(q_t = S_i \mid O, \lambda)\}$ [posterior decoding]
learning	supervised learning of λ^* $\lambda^* = \text{argmax}_\lambda P(q, O \mid \lambda)$ unsupervised learning of λ^* $\lambda^* = \text{argmax}_\lambda \max_q P(q, O \mid \lambda)$ [Viterbi training]	unsupervised learning $\lambda^* = \text{argmax}_\lambda \Sigma_q P(q, O \mid \lambda)$ [Baum-Welch training]

Based on slides by Manolis Kellis

HMM Elements		
states	$S = \{S_1, \ldots, S_N\}$	states $q_t \in S$
observations	$V = \{v_1, \ldots, v_M\}$	observations $O_t \in V$
initial state distribution	$\pi = \{\pi_i\}$ $\pi_i = P(q_1 = S_i)$	$1 \leq i \leq N$
state transition probability distribution	$A = \{a_{ij}\}$ $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i)$	$1 \leq i, j \leq N$
observation symbol probability distribution	$B = \{b_j(k)\}$ $b_j(k) = P(v_k \text{ at } t \mid q_t = S_j\}$	$1 \leq j \leq N, 1 \leq k \leq M$
Forward-Backward Algorithm (Scoring)		
forward variable	$\alpha_t(i) = P(O_1\ O_2 \ldots O_t, q_t = S_i \mid \lambda)$	probability of partial observations $O_1 O_2 \ldots O_t$ (until time t) and state S_i at time t, given model λ
backward variable	$\beta_t(i) = P(O_{t+1}\ O_{t+2} \ldots O_T \mid q_t = S_i, \lambda)$	probability of partial observations $O_{t+1} \ldots O_T$ given state S_i at time t and model λ
Posterior Decoding Algorithm		
	$\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$ $\gamma_t(i) = \alpha_t(i)\beta_t(i) \ / \ \Sigma_i\, \alpha_t(i)\beta_t(i)$	probability of being in state S_i at time t, given observations O and model λ
Viterbi Algorithm (Decoding)		
	$\delta_t(i) = \max_{q_1 \ldots q_{t-1}} P(q_1 \ldots q_t = S_i, O_1 \ldots O_t \mid \lambda)$	best score (highest probability) along single path, at time t, which accounts for first t observations and ends in state S_i

Markov Models and Markov Chains

Learning Goals

- Describe the properties of a Markov chain
- Describe the relationship between a Markov Chain and a Markov Model
- Describe the elements of a Markov Chain

Markov Chains and Markov Models

A **Markov chain** is a stochastic process with the Markov property.

Stochastic process

- probabilistic counterpart to a deterministic process
- a collection of r.v.'s that evolve over time

Markov property

- memoryless: conditional probability distribution of future states depends only on present state

		system state is	
		fully observable	**partially observable**
system is	**autonomous**	Markov chain	Hidden Markov Models
	controlled	Markov decision process	partially observable Markov decision process

Markov Chains

We can model a Markov chain as a triplet (S, π, A), where

- S: finite set of $N = |S|$ states
- π: initial state probabilities $\{\pi_i\}$

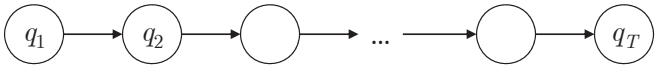
$$\pi_i = P(q_1 = S_i), 1 \le i \le N$$

- A: state transition probabilities $\{a_{ij}\}$

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i), 1 \le i, j \le N$$

What properties must π and A satisfy?

A MC outputs an (observable) state at each (discrete) time step, $t = 1,\ldots,T$.



The probability of observation sequence $O = \{O_1,\ldots,O_T\}$, where $O_t \in S$ is

$$P(O \mid \text{Model}) = P(q_1,\ldots,q_T)$$
$$= P(q_1)\ P(q_2 \mid q_1)\ P(q_3 \mid q_1,q_2)\ \ldots\ P(q_t \mid q_1,\ldots,q_{t-1})\ \ldots\ P(q_T \mid q_1,\ldots,q_{T-1})$$
$$= P(q_1)\ P(q_2 \mid q_1)\ \ldots\ P(q_t \mid q_{t-1})\ \ldots\ P(q_T \mid q_{T-1})$$

Markov Model of Weather

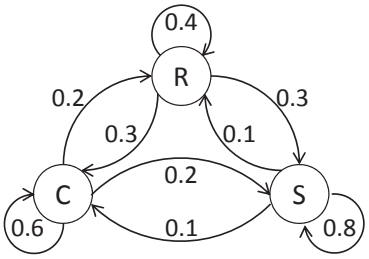
Once a day (e.g. at noon), the weather is observed as one of

state 1 : rainy state 2: cloudy state 3: sunny

The state transition probabilities are

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

(Notice that each row sums to 1.)



Questions:

1. Given that the weather on day 1 ($t = 1$) is sunny (state 3), what is the probability that the weather for the next 7 days will be "sun-sun-rain-rain-sun-cloudy-sun"?
2. Given that the model is state i, what is the probability that it stays in state i for exactly d days? What is the expected duration in state i (also conditioned on starting in state i)?

(This slide intentionally left blank.)

Solution to Q1

$$O = \{S_3,\ S_3,\ S_3,\ S_1,\ S_1,\ S_3,\ S_2,\ S_3\}$$

$$\begin{aligned}
&P(O \mid \text{Model}) \\
&\quad = P(S_3,\ S_3,\ S_3,\ S_1,\ S_1,\ S_3,\ S_2,\ S_3 \mid \text{Model}) \\
&\quad = P(S_3)\ P(S_3|S_3)\ P(S_3|S_3)\ P(S_1|S_3) \\
&\qquad\quad P(S_1|S_1)\ P(S_3|S_1)\ P(S_2|\ S_3)\ P(S_3|S_2) \\
&\quad = \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11}\ \cdot a_{13} \cdot a_{32} \cdot a_{23} \\
&\quad = (1)(0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\
&\quad = 1.536 \times 10^{-4}
\end{aligned}$$

Solution to Q2

$$O = \{S_{i_1},\ S_{i_2},\ S_{i_3},\ \ldots,\ S_{i_d},\ S_{j_{d+1}} \neq S_i\}$$

$P(O \mid \text{Model},\ q_1 = S_i) = (a_{ii})^{d-1}(1 - a_{ii}) = p_i(d)$
where $p_i(d)$ is the (discrete) PDF of duration d in state i.

Notice that $D_i \sim \text{geometric}(p)$, where $p = 1 - a_{ii}$ is the probability of success (exiting state i) and there are $d - 1$ failures before the first success.

Then $\overline{D_i} = \frac{1}{p} = \frac{1}{1-a_{ii}}$

Intuition: Consider a fair die. If the probability of success (a "1") is $p = 1/6$, it will take $1/p = 6$ rolls until a success.

the "math" way: $X \sim \text{geom}(p)$

$$\begin{aligned}
\overline{X} &= \sum_{k=1}^{\infty} k(1-p)^{k-1}p \\
&= p\sum_{k=1}^{\infty} k(1-p)^{k-1} \\
&= p\frac{1}{(1-(1-p))^2} = \frac{p}{p^2} = \frac{1}{p}
\end{aligned}$$

for $x \in \mathbb{R}$, $|x| \leq 1$,
$$\sum_{n=1}^{\infty} nx^{n-1} = \frac{1}{(1-x)^2}$$

For example, the expected number of consecutive days of rainy weather is $1/a_{11} = 1/0.6 = 1.67$; for cloudy, 2.5; for sunny, 5.

Hidden Markov Models

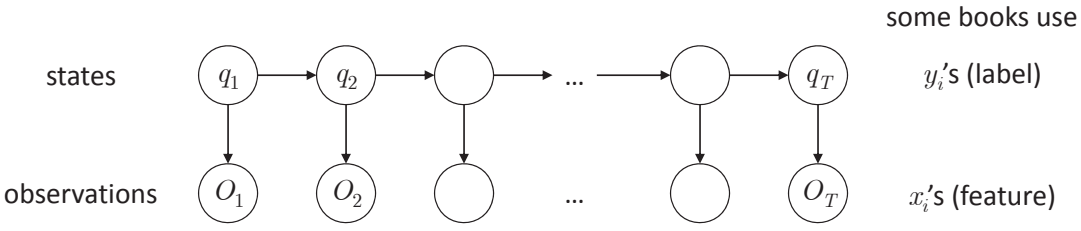
Learning Goals

- Describe the difference between Markov Chains and Hidden Markov Models
- Describe applications of HMMs
- Describe the elements of a HMM
- Describe the basic problems for HMMs

Hidden Markov Models

Now we would like to model pairs of sequences.

- There exists an underlying stochastic process that is **hidden** (not observable directly).
- But it affects observations (that we can collect directly).



HMMs are Everywhere

application	states	observations
weather inference	seasons	
dishonest casino (casino has fair die and loaded die, casino switches between dice on average once every 20 turns)	dice used	
missile tracking	position	
speech recognition	phoneme	
NLP part-of-speech tagging	part of speech	
computational biology	protein structure	
medicine	disease (state of progression)	

Elements of an HMM

A 5-tuple $(S,\ V,\ \pi,\ A,\ B)$, where

- S: finite set of states $\{S_1,\ \ldots,\ S_N\}$
- V: finite set of observations per state $\{v_1,\ \ldots,\ v_M\}$
- π: initial state distribution $\{\pi_i\}$
- A: state transition probability distribution $\{a_{ij}\}$
- B: observation symbol probability distribution $\{b_j(k)\}$

$$b_j(k) = P(v_k \text{ at } t \mid q_t = S_j\}, 1 \leq j \leq N$$
$$1 \leq k \leq M$$

Note that...

transitions...

and emissions...

depend only on current state.

A HMM outputs only emitted symbols $O = \{O_1,\ldots,O_T\}$, where $O_t \in V$.

Both the underlying states and random walk between states are hidden.

HMMs as a Generative Model

Given S, V, π, A, B, the HMM can be used as a generator to give an observation sequence

$$O = O_1 O_2 \ldots O_T.$$

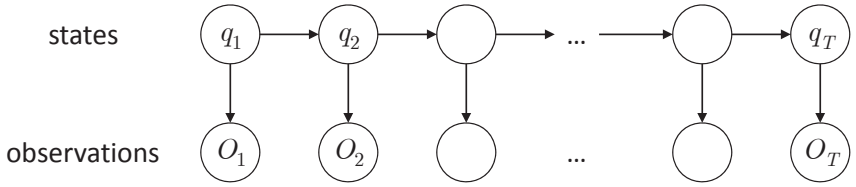
1) Choose initial state $q_1 = S_i$ according to initial state distribution π.
2) Set $t = 1$.
3) Choose $O_t = v_k$ according to symbol probability distribution in state S_i, i.e., $b_i(k)$.
4) Transit to new state $q_{t+1} = S_j$ according to state transition probability distribution for state S_i, i.e., a_{ij}.
5) Set $t = t + 1$. Return to step 3 if $t < T$. Otherwise stop.

Scoring HMMs

Learning Goals

- Describe how to score an observation over a single path and over multiple paths
- Describe the forward algorithm

Scoring a Sequence over a Single Path



Calculate $P(q,\ O \mid \lambda)$.

Scoring a Sequence over All Paths

Calculate $P(O \mid \lambda)$.

Naïve (brute-force) approach

$$\mathrm{P}(O \mid \lambda) = \Sigma_q\, P(q,\ O \mid \lambda)$$

How many calculations are required (big-O)? _____

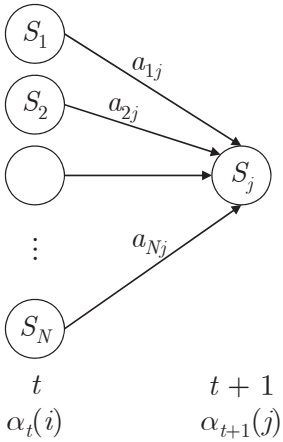
The Forward Algorithm

Define the **forward variable** as

$$\alpha_t(i) = P(O_1\ O_2\ \dots\ O_t,\ q_t = S_i \mid \lambda)$$

i.e. the probability of the partial observation sequence $O_1 O_2 \dots O_t$ (until time t) and state S_i at time t, given the model λ.

Use induction! Assume we know $\alpha_t(i)$ for $1 \le i \le N$.



$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})$$

- $\alpha_{t+1}(j)$: updated sum
- $\alpha_t(i)$: sum ending in state S_i at time t
- a_{ij}: transition from state S_i to state S_j at time t to $t+1$
- $b_j(O_{t+1})$: emission of observation O_{t+1} from state S_j at time $t+1$
- $\sum_{i=1}^{N}$: sum over all possible previous states S_i

The Forward Algorithm

1) Initialization

$$\alpha_1(i) = \pi_i b_i(O_1),\ 1 \le i \le N$$

2) Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \qquad \begin{matrix} 1 \le t \le T-1 \\ 1 \le j \le N \end{matrix}$$

Perform for all states for given t, then advance t.

3) Termination

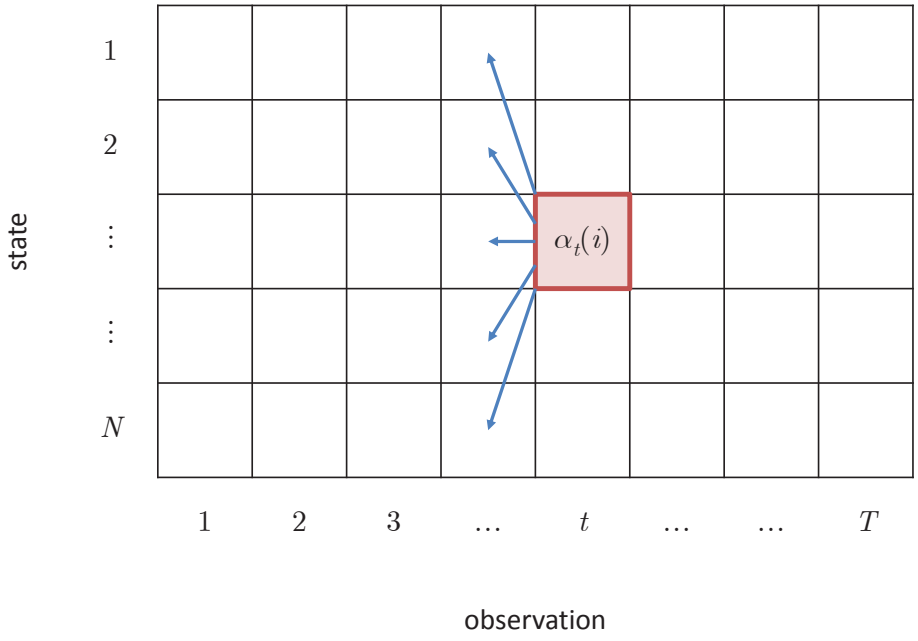
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

[Proofs for Initialization and Termination Steps]

$$\begin{aligned} \alpha_1(i) &= P(O_1, q_1 = S_i|\lambda) \\ &= P(O_1|q_1 = S_i, \lambda) P(q_1 = S_i|\lambda) \\ &= b_i(O_1)\pi_i \end{aligned}$$

$$\begin{aligned} P(O|\lambda) &= P(O_1 \dots O_T|\lambda) \\ &= \sum_{i=1}^{N} P(O_1 \dots O_T, q_T = S_i|\lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned}$$

Dynamic Programming Table



The Forward Variable

We showed the induction step for $\alpha_{t+1}(j)$ through intuition. Can we prove it?

$$\begin{aligned}
\alpha_{t+1}(j) &= P(O_1 \dots O_{t+1}, q_{t+1} = S_j|\lambda) \\
&= \sum_{q_1 \cdots q_t} P(O_1 \dots O_t, O_{t+1}, q_1 \dots q_t, q_{t+1} = S_j|\lambda) \\
&= \left[\sum_{q_1 \cdots q_t} P(O_1 \dots O_t, q_1 \dots q_t, q_{t+1} = S_j|\lambda) \right] P(O_{t+1}|q_{t+1} = S_j, \lambda) \\
&= \left[\sum_{i=1}^{N} \sum_{q_1 \cdots q_{t-1}} P(O_1 \dots O_t, q_1 \dots q_{t-1}, q_t = S_i, q_{t+1} = S_j|\lambda) \right] b_j(O_{t+1}) \\
&= \left[\sum_{i=1}^{N} \sum_{q_1 \cdots q_{t-1}} P(O_1 \dots O_t, q_1 \dots q_{t-1}, q_t = S_i|\lambda) P(q_{t+1} = S_j|q_t = S_i, \lambda) \right] b_j(O_{t+1}) \\
&= \left[\sum_{i=1}^{N} P(O_1 \dots O_t, q_t = S_i|\lambda) a_{ij} \right] b_j(O_{t+1}) \\
&= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1})
\end{aligned}$$

The Forward Algorithm

What is the complexity of the forward algorithm?

- time complexity: _____
  - compare to brute-force $O(N^T \cdot T)$
  - e.g. $N = 5$, $T = 100$, need ~3k computations vs 10^{72}
- space complexity: _____

Practical Issues

- underflow → use log probabilities for model
- for sums of probabilities, use log-sum-exp trick

$$\log \sum_{n=1}^{N} \exp(x_n) = a + \log \sum_{n=1}^{N} \exp(x_n - a) \text{ where } a = \max_n x_n$$

Decoding HMMs

Learning Goals

- Describe how to decode the state sequence
- Describe the Viterbi algorithm

Posterior Decoding

We want to compute

$$q_t = \text{argmax}_{S_i} P(q_t = S_i \mid O, \lambda)$$

Define $\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$, i.e. the probability of being in state S_i at time t, given observation sequence O and model λ.

Then

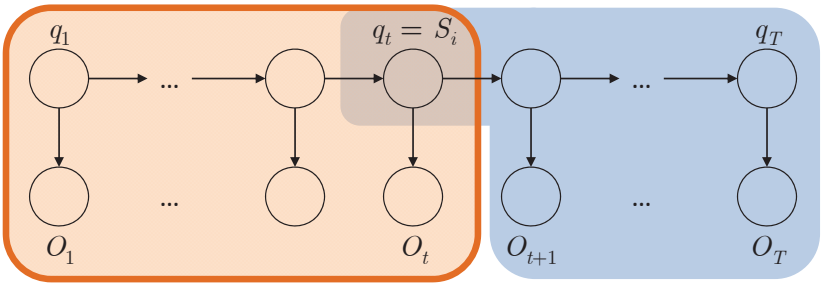
$$\gamma_t(i) = \frac{P(O, q_t = S_i|\lambda)}{P(O|\lambda)}$$

We still need to determine $P(O, q_t = S_i \mid \lambda)$.

We just determined $P(O \mid \lambda)$ using the forward algorithm.

Probabilities for Posterior Decoding

$$P(O, q_t = S_i \mid \lambda) = \underbrace{P(O_1 \ldots O_t, q_t = S_i \mid \lambda)}_{\alpha_t(i)} \underbrace{P(O_{t+1} \ldots O_T \mid q_t = S_i, \lambda)}_{\beta_t(i)}$$



The Backward Algorithm

(an alternative approach, useful later too)

Define the **backward variable** as

$$\beta_t(i) = P(O_{t+1}\ O_{t+2}\ \ldots\ O_T \mid q_t = S_i, \lambda)$$

i.e. the probability of the partial observation sequence $O_{t+1} \ldots O_T$ given state S_i at time t and the model λ.

[Note that the state at time t is now given and on RHS of conditional.]

1) Initialization (arbitrarily define $\beta_T(i)$ to be 1 for all i)

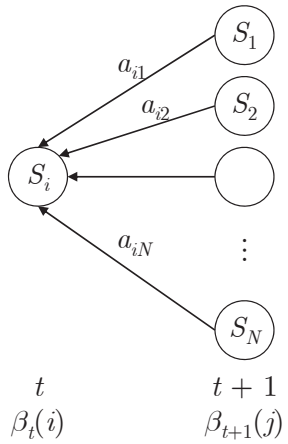
$$\beta_T(i) = 1,\ 1 \le i \le N$$

2) Induction

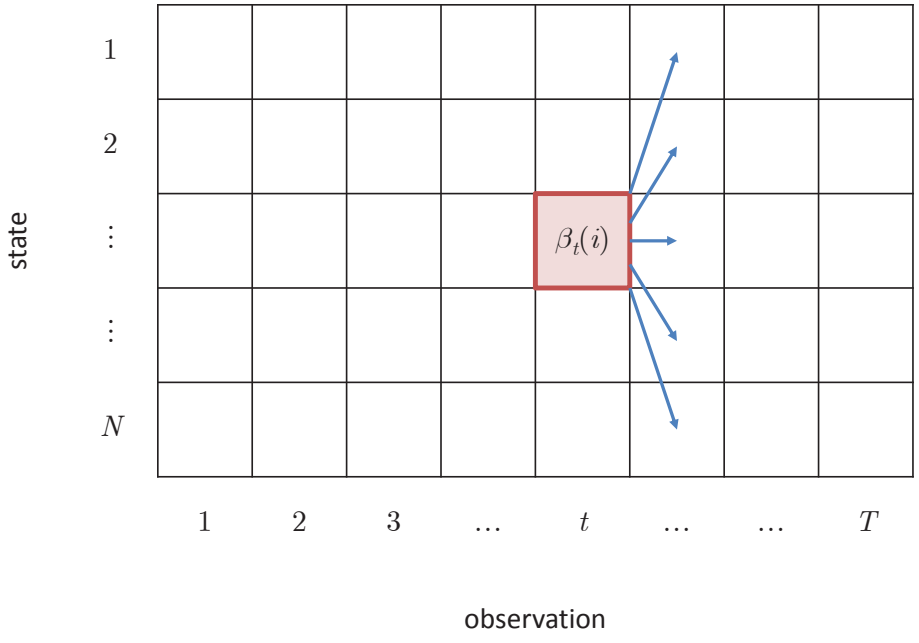
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

$$t = T-1, T-2, \ldots, 1$$

$$1 \le i \le N$$

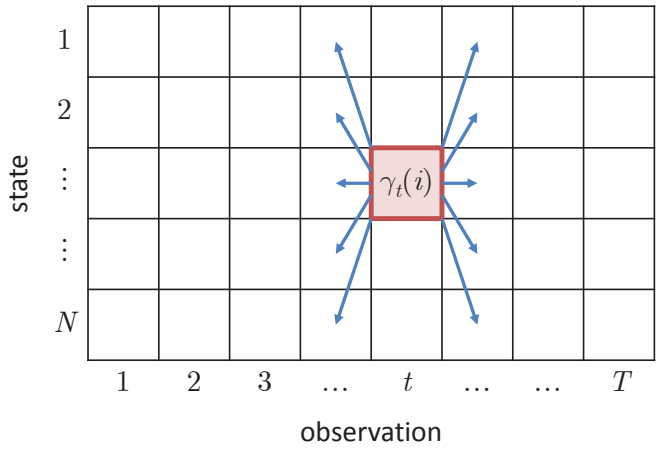


Dynamic Programming Table



Posterior Decoding

Then $$\gamma_t(i) = \frac{P(O, q_t = S_i|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$



Now solve $$q_t^* = \arg\max_{1 \le i \le N} \gamma_t(i), \ 1 \le t \le T$$

Posterior Decoding

We found the individually most likely state q_t at time t.

The Good

- maximizes expected number of correct states

The Bad

- may result in invalid path
  (not all $S_i \rightarrow S_j$ transitions may be possible)

$\Rightarrow$ most probable state is most likely to be correct at any instant, but sequence of individually probable states is not likely to be most probable sequence

Viterbi Decoding

Goal: Find single best **state sequence**.

$$q^* = \text{argmax}_q \, P(q \mid O, \lambda) = \text{argmax}_q \, P(q, O \mid \lambda)$$

Define $\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P(q_1 \ldots q_t = S_i, O_1 \ldots O_t|\lambda)$

i.e. the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state S_i.

Compare to algorithm for $\alpha_t(i) = P(O_1 \ \ldots \ O_t, \ q_t = S_i \mid \lambda)$.

To determine best path to $q_{t+1} = S_j$, compute $\delta_{t+1}(j)$.

- best path to $q_t = S_i$ $\Rightarrow \delta_t(i)$
- transition from $q_t = S_i$ to $q_{t+1} = S_j$ $\Rightarrow a_{ij}$
- emission of O_{t+1} from $q_{t+1} = S_j$ $\Rightarrow b_j(O_{t+1})$
- to retrieve state sequence, also need traceback pointer

$\psi_t(i) =$ state S_i that maximizes $\delta_t(i)$

The Viterbi Algorithm

1) Initialization

2) Induction

3) Termination

4) Path (state sequence) backtracking

The Viterbi Algorithm

1) Initialization

$$\delta_1(i) = \pi_i b_i(O_1), \ 1 \leq i \leq N$$

2) Induction

$$\delta_{t+1}(j) = \left[\max_{1 \leq i \leq N} \delta_t(i) a_{ij} \right] b_j(O_{t+1}) \qquad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix}$$

$$\psi_{t+1}(j) = \arg\max_{1 \leq i \leq N} \delta_t(i) a_{ij} \qquad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix}$$

Perform for all states for given t, then advance t.

3) Termination

$$p^* = \max_{1 \leq i \leq N} \delta_T(i)$$

$$q_T^* = \arg\max_{1 \leq i \leq N} \delta_T(i)$$

4) Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \ t = T-1, T-2, \ldots, 1$$

The Viterbi Algorithm

- similar to forward algorithm (use max instead of sum)
- use DP table to compute
- same complexity as forward algorithm

Practical Issues

- underflow issues → use log probabilities for model
- for logs of products of probabilities, use sum of logs

Learning HMMs

Learning Goals

- Describe how to learn HMM parameters
- Describe the Baum-Welch algorithm

Learning

Goal

Adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O \mid \lambda)$, i.e. the probability of the observation sequence(s) given the model.

Supervised Approach

Assume we have complete data (we know the underlying states). Use MLE.

Supervised Learning Example

state space $S = \{1, 2\}$

observation space $V = \{e, f, g, h\}$

training set $\left\{\begin{bmatrix}1 & 2\\ e & g\end{bmatrix}\right\}$ $\left\{\begin{bmatrix}1 & 2\\ e & h\end{bmatrix}\right\}$ $\left\{\begin{bmatrix}1 & 2\\ f & h\end{bmatrix}\right\}$ $\left\{\begin{bmatrix}1 & 2\\ f & g\end{bmatrix}\right\}$

What are the optimal model parameters?

Pseudocounts

For small training set, the parameters may overfit.

- $P(O \mid \lambda)$ is maximized but λ is unreasonable
- probabilities of 0 are problematic

Add pseudocounts to represent our prior belief.

- large pseudocounts $\rightarrow$ large regularization
- small pseudocounts $\rightarrow$ small regularization
  (just to avoid $P = 0$)

Learning

Unsupervised Approach

- we do not know the underlying states
- no known way to analytically solve for optimal model

Ideas

- use iterative algorithm to locally maximize $P(O \mid \lambda)$
- either gradient descent or EM work
- Baum-Welch algorithm based on EM is most popular

Unsupervised Learning

Goal

$$\hat{\pi}_i = \text{expected frequency (number of times) in state } S_i \text{ at time } (t = 1)$$

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}$$

$$\hat{b}_j(k) = \frac{\text{expected number of times in state } S_j \text{ and observing symbol } v_k}{\text{expected number of times in state } S_j}$$

Recall $\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$, i.e. the probability of being in state S_i at time t, given observation sequence O and model λ. Can we use this to solve for any of the above terms?

Expected Number of Transitions

Define

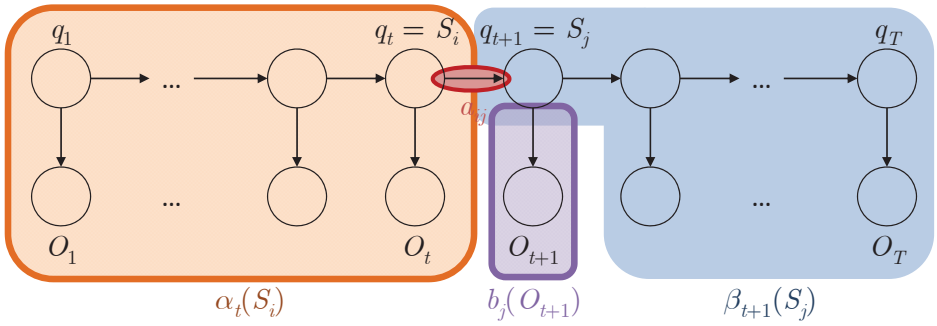
$$\xi_t(i,j) = P(q_t = S_i,\ q_{t+1} = S_j \mid O,\ \lambda)$$

i.e. the probability of being in state S_i at time t, and state S_j at time $t+1$, given the model and the observation sequence.

$$\xi_t(i,j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)}$$

To calculate the numerator, $\alpha_t(S_i)\ a_{ij}\ b_j(O_{t+1})\ \beta_{t+1}(S_j)$

We already know $P(O \mid \lambda)$ using the forward algorithm.



Unsupervised Learning

Goal

$\hat{\pi}_i =$ expected frequency (number of times) in state S_i at time $(t=1)$

$= \gamma_1(i)$

$= \sum_{t=1}^{T-1} \xi_t(i,j)$

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\hat{b}_j(k) = \frac{\text{expected number of times in state } S_j \text{ and observing symbol } v_k}{\text{expected number of times in state } S_j} = \frac{\sum_{t=1 \text{ s.t. } O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Baum-Welch Algorithm

Initialization

- Set $\lambda = (A, B, \pi)$ to random initial conditions (or using prior information)

Iteration (repeat until convergence)

- Compute $\alpha_t(i)$ and $\beta_t(i)$ using forward-backward algo
  $\Rightarrow$ Compute $P(O \mid \lambda)$ [E-step]
- Compute $\gamma_t(i)$ and $\xi_t(i,j)$
  $\Rightarrow$ Update model parameters [M-step]

Baum-Welch Algorithm

- Time complexity: $\mathrm{O}(N^2 T) \cdot (\#\ \mathrm{iterations})$
- Guaranteed to increase likelihood $P(O \mid \lambda)$ via EM but *not* guaranteed to find globally optimal λ^*

Practical Issues

- Use multiple training sequences (sum over them)
- Apply smoothing to avoid zero counts and improve generalization (add pseudocounts)

HMMs and Protein Structure

One biological application of HMMs is to determine the secondary structure (i.e. the general three-dimensional shape) of a protein. This general shape is made up of alpha helices, beta sheets, and other structures. In this problem, we will assume that the amino acid composition of these regions is governed by an HMM.

To keep this problem relatively simple, we do not use actual transition values or emission probabilities. The start state is always "other". We will use the state transition probabilities and emission probabilities below.

	alpha	beta	other
alpha	0.7	0.1	0.2
beta	0.2	0.6	0.2
other	0.3	0.3	0.4

e.g. *P*(Alpha Helix → Beta Sheet) = 0.1

amino acid	alpha	beta	other
M	0.35	0.10	0.05
L	0.30	0.05	0.15
N	0.15	0.30	0.20
E	0.10	0.40	0.15
A	0.05	0.00	0.20
G	0.05	0.15	0.25

Based on exercise by Manolis Kellis

Protein Structure Questions

1) What is the probability $P(q = \text{O}\alpha,\ O = ML)$?

2) How many paths could give rise to the sequence $O = MLN$? What is the total probability $P(O)$?

3) Give the most likely state transition path q^* for the amino acid sequence MLN using the Viterbi algorithm. What is $P(q^*, O)$?

   Compare this to $P(O)$ above. What does this say about the reliability of the Viterbi path?

Based on exercise by Manolis Kellis