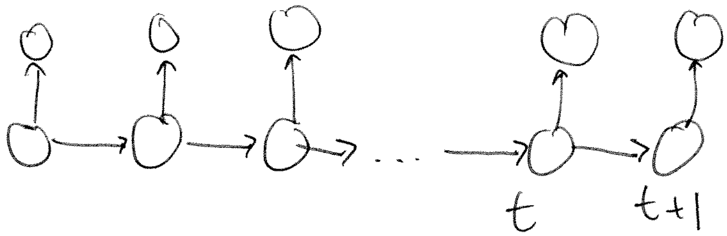
Lecture 12

2/17/15
CSE 150

Review

* Hidden Markov models (HMMs)



observations $O_t \in \{1, 2, \ldots, m\}$

states $S_t \in \{1, 2, \ldots, n\}$

* Joint distribution

$$P(\vec{S}, \vec{O}) = P(s_1)\left[\prod_{t=2}^{T} P(s_t|s_{t-1})\right]\left[\prod_{t=1}^{T} P(o_t|s_t)\right]$$

* Parameters

$\pi_i = P(s_1 = i)$ initial state distribution

$a_{ij} = P(s_{t+1} = j | s_t = i)$ transition matrix

$b_{ik} = P(o_t = k | s_t = i)$ emission matrix

* Key questions

1) How to compute likelihood $P(o_1, o_2, \ldots, o_T)$
2) How to decode hidden states $\vec{S}^* = \underset{\vec{S}}{\mathrm{argmax}}\, P(\vec{S}|\vec{O})$
3) How to update beliefs $P(S_t = i | o_1, o_2, \ldots, o_t)$

} inference

4) How to estimate $\{\pi_i, a_{ij}, b_{ik}\}$ from data } learning

1) Computing likelihood

$$P(o_1, o_2, \dots o_T) = \sum_{\vec{s}} P(\overbrace{s_1, s_2, \dots, s_T}^{\text{hidden}}, o_1, o_2, \dots, o_T)$$

$\vec{s}$ ← sum over n^T sequences of hidden states

$$= \sum_{\vec{s}} P(s_1) \prod_t P(s_t \mid s_{t-1}) \prod_t P(o_t \mid s_t)$$

* Efficient recursion — if know state at time t, can calc t+1
(don't recurse over all t...)

$$P(o_1, o_2, \dots, o_t, o_{t+1}, s_{t+1} = j) = \sum_{i=1}^{n} P(o_1, o_2, \dots, o_t, o_{t+1}, s_t = i, s_{t+1} = j) \quad \text{marginalization}$$

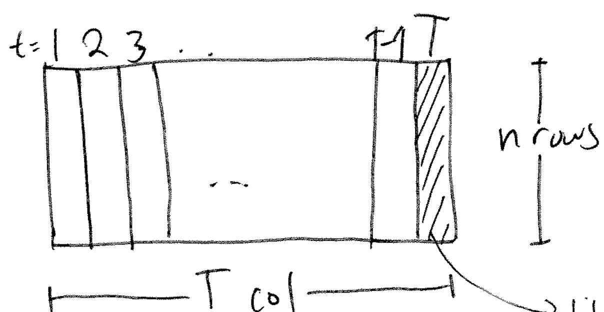
$$= \sum_{i=1}^{n} P(o_1, o_2, \dots, o_t, s_t = i)\, P(s_{t+1} = j \mid o_1, o_2, \dots, o_t, s_t = i) \cdot P(o_{t+1} \mid s_{t+1} = j, s_t = i, o_1, \dots, o_t)$$

$$= \sum_{i=1}^{n} \underbrace{P(o_1, o_2, \dots, o_t, s_t = i)}_{\text{recursive instance}} \cdot \underbrace{\underbrace{P(s_{t+1} = j \mid s_t = i)}_{a_{ij}} \cdot \underbrace{P(o_{t+1} \mid s_{t+1} = j)}_{b_j(o_{t+1})}}_{\text{CPTs of HMM}} \quad \text{cond. indep}$$

* shorthand notation

$$\alpha_{it} = P(o_1, o_2, \dots, o_t, s_t = i)$$

$i = 1 \dots n$ # hidden states; $t = 1 \dots T$ seq. length



→ likelihood is sum of last column $P(o_1, o_2, \dots, o_T)$

* Forward algorithm
  - base case ($t=1$) first column of matrix

$$\alpha_{i1} = P(o_1, S_1=i) \qquad \text{by def}$$
$$= P(S_1=i) \cdot P(o_1 | s_1=i) \qquad \text{product rule}$$
$$= \underbrace{\pi_i}_{\substack{\text{initial}\\\text{state}}} \cdot \underbrace{b_i(o_1)}_{\text{emission matrix}}$$

  - recursive step from time $t-1$ to time t

$$\alpha_{j,t+1} = \sum_{i=1}^{n} \alpha_{it} \cdot a_{ij} \cdot b_j(o_{t+1})$$

* Back to likelihood computation

$$P(o_1, o_2, \ldots, o_T) = \sum_{i=1}^{n} P(o_1, o_2, \ldots, o_T, S_T=i) \qquad \text{marginalization}$$
$$= \sum_{i=1}^{n} \alpha_{iT} \qquad \text{(sum of elements of last column of } \alpha \text{ matrix)}$$

* Scales as $O(Tn^2)$
  - linear, not exponential, in sequence length T
  - quadratic in # of states n

* Warning: naive calculation will underflow for long sequences $T \gg 1$
  because $P(o_1, o_2, \ldots, o_T) \ll 1$
  (fix by rescaling each iteration, eventually taking log)

2) How to compute most likely state sequence?

$$s^* = \{s_1^*, s_2^*, \ldots, s_{T-1}^*, s_T^*\}$$

$$= \underset{\vec{s}}{\operatorname{argmax}}\; P(s_1, s_2, \ldots, s_T \mid o_1, o_2, \ldots, o_T)$$

$$= \underset{\vec{s}}{\operatorname{argmax}} \left[\frac{P(s_1, s_2, \ldots, s_T, o_1, o_2, \ldots, o_T)}{P(o_1, o_2, \ldots, o_T)}\right] \quad \text{product rule}$$

constant w.r.t $\vec{s}$, can drop for argmax

$$s^* = \underset{\vec{s}}{\operatorname{argmax}}\; P(s_1, s_2, \ldots, s_T, o_1, o_2, \ldots, o_T)$$

$\boxed{\text{How to compute } s^*\text{?}}$

Define: $$l_{it}^* = \max_{\substack{s_{1:t-1} \\ \{s_1, s_2, \ldots, s_{t-1}\}}} \log P(s_1, s_2, \ldots, s_{t-1}, \underline{s_t = i}, o_1, o_2, \ldots, o_t)$$

- log-prob. of most likely sequence of t hidden states that ends in <u>state i at time t</u> that explains observations $o_1, \ldots, o_t$

* Recursive algorithm

- base case ($t=1$) leftmost column

$$l_{i1}^* = \log P(s_1 = i, o_1)$$

$$= \log \left[P(s_1 = i) \cdot P(o_1 \mid s_1 = i)\right] \quad \text{Product rule}$$

$$= \log \left[\pi_i \cdot b_i(o_1)\right]$$

$$= \log \pi_i + \log b_i(o_1)$$

• recursive step from $t-1$ to t

$$l^*_{j,t+1} = \max_{s_1,s_2,\ldots,s_t} \log P(s_1,s_2,\ldots,s_t,s_{t+1}=j,o_1,o_2,\ldots,o_t,o_{t+1})$$

$$= \max_{s_1,s_2,\ldots,s_{t-1}} \max_i \log P(s_1,s_2,\ldots,s_{t-1},s_t=i,s_{t+1}=j,o_1,o_2,\ldots,o_t,o_{t+1})$$

$$= \max_{s_1,s_2,\ldots,s_{t-1}} \max_i \log\Big[P(s_1,s_2,\ldots,s_{t-1},s_t=i,o_1,o_2,\ldots,o_t)\cdot$$
$$P(s_{t+1}=j\mid s_1,\ldots,s_t=i,o_1,\ldots,o_t)\cdot P(o_{t+1}\mid s_1,\ldots,s_{t-1},s_t=i,s_{t+1}=j,o_1,\ldots,o_t)\Big]$$

$$= \max_{s_1,s_2,\ldots,s_{t-1}} \max_i \log P(s_1,\ldots,s_{t-1},s_t=i,o_1,\ldots,o_t)\cdot P(s_{t+1}=j\mid s_t=i)\cdot P(o_{t+1}\mid s_{t+1}=$$

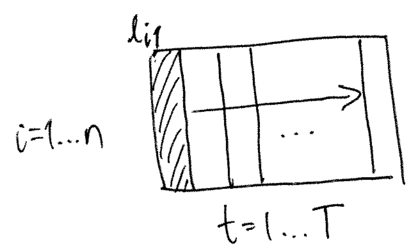
$$= \max_{s_1,\ldots,s_{t-1}} \Big[\max_i \Big\{\log P(s_1,\ldots,s_{t-1},s_t=i,o_1,\ldots,o_t) + \log P(s_{t+1}=j\mid s_t=i)\Big\}$$
$$+ \log P(o_{t+1}\mid s_{t+1}=j)\Big]$$

Reassembling:

$$l^*_{j,t+1} = \max_i \Big[\max_{s_1,\ldots,s_{t-1}} \log P(s_1,\ldots,s_{t-1},s_t=i,o_1,\ldots,o_t) + \log P(s_{t+1}=j\mid s_t=i)\Big]$$
$$+ \log P(o_{t+1}\mid s_{t+1}=j)$$

$$\boxed{l^*_{j,t+1} = \max_i \left[l^*_{it} + \log a_{ij}\right] + \log b_j(o_{t+1})}$$ forward pass

algorithm: fill in l^*_{it} column by column



* How to derive s^* from l^* ?

Record most likely transitions:

$$\Phi_{t+1}(j) = \underset{i}{\operatorname{argmax}} \left[l^*_{it} + \log a_{ij} \right]$$

(if at state j at time $t+1$, what state i at time t got you there, for observations $o_1, o_2, \ldots, o_{t+1}$)

* Compute s^* from backtracking:
  - At time T: $s^*_T = \underset{i}{\operatorname{argmax}} \left[l^*_{iT} \right]$ (what is largest element of last column in l^*)
  - For time $t=T-1$ to $t=1$: backward pass

$$s^*_t = \Phi_{t+1}(s^*_{t+1})$$

* Jargon:
  - s^* is known as "Viterbi" state sequence or "path"
  - Viterbi algorithm is example of dynamic programming