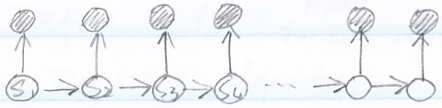
7.19

Review

* Hidden Markov models (HMMs)



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

hidden states $S_t \in \{1, 2, \cdots, n\}$

* Parameters of CPTs

$\pi_i = P(S_1 = i)$ initial state distribution

$a_{ij} = P(S_{t+1} = j \mid S_t = i)$ transition matrix

$b_{ik} = P(O_t = k \mid S_t = i)$ emission matrix

$b_i(k)$

* Key questions

1) How to compute likelihood $P(O_1, O_2, \cdots, O_T)$?

2) How to decode hidden states $\vec{S}^* = \text{argmax}\, P(\vec{S} \mid \vec{O})$?

3) How to update beliefs $P(S_t = i \mid O_1, O_2, \cdots, O_t)$?

} 1-3 are inference

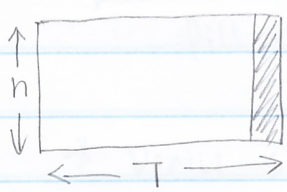
4) How to estimate/learn $\{\pi_i, a_{ij}, b_{jk}\}$ from data? } 4 is learning

1. Computing Likelihood

$\alpha_{it} = P(O_1, O_2, \cdots, O_t, S_t = i)$

$i = 1 \cdots n$

$t = 1 \cdots T$



Forward algorithm:

$(t=1)$ $\alpha_{i1} = \pi_i b_i(O_1)$

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

Likelihood $P(O_1, O_2, \cdots, O_T) = \sum_{i=1}^{n} \alpha_{iT}$

2. Compute most likely state sequence.

$$S^* = \underset{S_1, S_2, \cdots, S_T}{\operatorname{argmax}} \log P(S_1, S_2, \cdots, S_T, O_1, O_2, \cdots, O_T)$$

Define $l_{it}^* = \max_{\{S_1, S_2, \cdots, S_{t-1}\}} \log P(S_1, S_2, \cdots, S_{t-1}, S_t = i, O_1, O_2, \cdots, O_t)$

* Recursive Algorithm

base case ($t=1$) left most column

$$l_{i1}^* = \log P(S_1 = i, O_1)$$
$$= \log [P(S_1 = i)\, P(O_1 | S_1 = i)] \quad \text{product rule}$$
$$= \log [\pi_i\, b_i(O_1)]$$
$$= \log \pi_i + \log b_i(O_1)$$

Recursive step from t to $t+1$

$$l_{j,t+1}^* = \max_{S_1, S_2, \cdots, S_t} [\log P(S_1, S_2, \cdots, S_t, S_{t+1} = j, O_1, O_2, \cdots, O_{t+1})]$$
$$= \max_{S_1, S_2, \cdots S_t} \max_i [\log P(S_1, S_2, \cdots S_t = i, S_{t+1} = j, O_1, O_2, \cdots O_t, O_{t+1})]$$
$$= \max_{S_1, S_2, \cdots S_{t-1}} \max_i [\log \{P(S_1 \cdots S_t = i, O_1 \cdots O_t)$$
$$P(S_{t+1} = j | S_1, \cdots S_t = i, O_1, \cdots O_t)$$
$$P(O_{t+1} | S_1, \cdots S_t = i, S_{t+1} = j, O_1, \cdots O_t)\}]$$

product rule, cond ind.

$$= \max_{S_1, \cdots S_{t-1}} \max_i [\log P(S_1, \cdots, S_t = i, O_1, \cdots, O_t) + \log P(S_{t+1} = j | S_t = i)$$
$$+ \log P(O_{t+1} | S_{t+1} = j)]$$

Note: The last two terms do not depend on $S_1, \cdots S_{t-1}$

$$l_{j,t+1}^* = \max_i [\max_{S_1, \cdots S_{t-1}} \log P(S_1, \cdots, S_{t-1}, S_t = i, O_1, \cdots, O_t)$$
$$+ \log P(S_{t+1} = j | S_t = i) + \log P(O_{t+1} | S_{t+1} = j)]$$

Forward pass

$$\boxed{l_{j,t+1}^* = \max_i [l_{it}^* + \log a_{ij} + \log b_j(O_{t+1})]}$$

fill in l_{it}^* column by column.

* Derive S^* from l^*.

Record most likely transitions:

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

If I must end up in state j at time $t+1$, then what state i preceded at time t?

* Compute S^* by backtracking:

At time T: $S_T^* = \underset{i}{\text{argmax}}\,[l_{iT}^*]$

For times $t = T-1$ to 1: $S_t^* = \Phi_{t+1}(S_{t+1}^*)$

Backward Pass

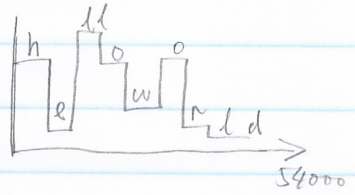
* Jargon:

- S^* is known as "Viterbi" sequence or path
- Viterbi algorithm is example of dynamic programming

HW: $n = 26$ $(S_t \in \{a, b, c, \cdots, z\})$

$m = 2$ $(O_t \in \{0, 1\})$

$T = 54000$



3. Belief updating

How to monitor hidden state in real time based on incoming evidence ?

Key quantity: $q_{jt} = P(S_t = j \mid O_1, O_2, \cdots, O_t)$

$$q_{jt} = P(S_t = j \mid O_1, \cdots, O_{t-1}, O_t)$$

$$= \frac{P(O_t \mid S_t = j, O_1, \cdots, O_{t-1})\, P(S_t = j \mid O_1, \cdots, O_{t-1})}{P(O_t \mid O_1, O_2, \cdots, O_{t-1})}$$

$$b_j(o_t) = \frac{P(O_t \mid S_t = j)\; P(S_t \mid O_1, \cdots, O_{t-1})}{P(O_t \mid O_1, \cdots, O_{t-1})}$$ cond. ind.

Focus on 2nd term in numerator:

$$P(S_t = j \mid O_1, \cdots O_{t-1}) = \sum_{i=1}^{n} P(S_t = j, S_{t-1} = i \mid O_1, O_2, \cdots, O_{t-1})$$ marginalization

$$= \sum_{i=1}^{n} P(S_{t-1} = i \mid O_1, \cdots, O_{t-1})\; P(S_t = j \mid S_{t-1} = i, O_1, \cdots, O_{t-1})$$ product rule

$$= \sum_{i=1}^{n} P(S_{t-1} = i \mid O_1, \cdots, O_{t-1})\, P(S_t = j \mid S_{t-1} = i)$$ cond. ind.

$$= \sum_{i=1}^{n} q_{i,t-1}\, a_{ij}$$

Putting into numerator:

$$q_{jt} = \frac{b_j(o_t) \sum_{i=1}^{n} q_{i,t-1}\, a_{ij}}{P(o_t \mid o_1, \cdots o_{t-1})}$$

normalization

$$q_{jt} = \frac{b_j(o_t) \sum_i q_{i,t-1}\, a_{ij}}{\sum_{j'} b_{j'}(o_t) \sum_{i'} q_{i',t-1}\, a_{i'j'}}$$

How much computation per time step? $O(n)$

Back from break.

Key questions in HMMs

✓ 1) How to compute likelihood $P(O_1, O_2, \cdots, O_T)$?

✓ 2) How to decode $\{s_1^*, s_2^*, \cdots, s_T^*\} = \underset{s_1, \cdots, s_T}{\text{argmax}}\, P(s_1, \cdots s_T \mid O_1, \cdots, O_T)$?

✓ 3) How to update beliefs $P(s_t = i \mid O_1, \cdots O_t)$?

Onto learning ...

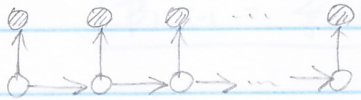
4. Learning in HMMs

Given: sequence of observations $\{O_1, O_2, \cdots, O_T\}$
(just one for simplicity)

Goal: estimate $\{\pi_i, a_{ij}, b_{ik}\}$ to maximize $P(O_1, \cdots, O_T)$

Assume: # hidden state values, n, is known.

* EM algorithm in HMMs:



CPTs to estimate

$\pi_i = P(s_1 = i)$

$a_{ij} = P(s_{t+1} = j \mid s_t = i)$

$b_{ik} = P(O_i = k \mid s_t = i)$

* E-step: compute posteriors

$$P(S_1 = i \mid O_1, \cdots, O_T)$$

$$P(S_t = i, S_{t+1} = j \mid O_1, \cdots, O_T)$$

$$P(S_t = i, O_t = k \mid O_1, \cdots, O_T) = I(O_t, k)\, P(S_t = i \mid O_1 \cdots O_T)$$

Assuming we can compute these posteriors efficiently ...

* M-step: update CPTs

$$\pi_i \leftarrow P(S_1 = i \mid O_1, \cdots O_T)$$

$$a_{ij} \leftarrow \frac{\sum_t P(S_t = i, S_{t+1} = j \mid O_1 \cdots O_T)}{\sum_t P(S_t = i \mid O_1, \cdots, O_T)}$$

$$b_{ik} \leftarrow \frac{\sum_t P(S_t = i \mid O_1, \cdots, O_T)\, I(O_t, k)}{\sum_t P(S_t = i \mid O_1, \cdots O_T)}$$

* How to compute these posterior probabilities?

Recall $\alpha_{it} = P(O_1, \cdots O_t, S_t = i)$

Define $\beta_{it} = P(O_{t+1}, O_{t+2}, \cdots, O_T \mid S_t = i)$

(O_{t+1}: starts at time $t+1$; $S_t = i$: conditioned on $S_t = i$)

* Recursion for filling in β matrix

(1) base case

(a) $\beta_{iT} = P(- \mid S_T = i) = 1$ for all $i = 1 \cdots n$

(b) $\beta_{i,T-1} = P(O_T \mid S_{T-1} = i)$

$$= \sum_{j=1}^{n} P(S_T = j, O_T \mid S_{T-1} = i) \quad \text{marginalization}$$

$$= \sum_j P(S_T = j \mid S_{T-1} = i)\, P(O_T \mid S_T = j, \cancel{S_{T-1} = i}) \quad \text{product rule, cond. ind.}$$

$$\boxed{\beta_{i,T-1} = \sum_j a_{ij}\, b_j(O_T)}$$

(2) backward-step (from time $t+1$ to time t)

$$\beta_{it} = P(O_{t+1}, \cdots, O_T \mid S_t = i)$$

$$= \sum_{j=1}^{n} P(O_{t+1}, \cdots, O_T, S_{t+1} = j \mid S_t = i) \quad \text{marginalization}$$

$$= \sum_{j=1}^{n} P(S_{t+1} = j \mid S_t = i)\, P(O_{t+1} \mid S_{t+1} = j, \cancel{S_t = i}) \quad \text{product rule}$$

$$P(O_{t+2}, \cdots, O_T \mid S_{t+1} = j, \cancel{S_t = i}, \cancel{O_{t+1}}) \quad \text{cond. ind.}$$

$$= \sum_{j=1}^{n} P(S_{t+1}=j \mid S_t=i)\, P(O_{t+1} \mid S_{t+1}=j)\, P(O_{t+2}, \cdots, O_T \mid S_{t+1}=j)$$

$$\boxed{\beta_{it} = \sum_{j=1}^{n} a_{ij}\, b_j(O_{t+1})\, \beta_{j,t+1}}$$

"Forward-backward" algorithm in HMMs computes α_{it} and β_{it} matrices.

Last question:

How do α_{it} and β_{it} give us posteriors for E-step of EM algorithm?

* Return to posteriors:

1) $P(S_1 \mid O_1, \cdots, O_T)$ needed to π_i update

2) $P(S_t=i, S_{t+1}=j \mid O_1, \cdots, O_T)$ needed for a_{ij} update

3) $P(S_t=i \mid O_1, \cdots, O_T)$ needed for b_{ik} update

* Focus on middle probability:

$$P(S_t=i, S_{t+1}=j \mid O_1, \cdots, O_T) = \frac{P(S_t=i, S_{t+1}=j, O_1, \cdots, O_T)}{P(O_1, O_2, \cdots, O_T)}$$

Denominator: $P(O_1, O_2, \cdots, O_T)$ is just the likelihood.

$$P(O_1, O_2, \cdots, O_T) = \sum_{i=1}^{n} \alpha_{iT}$$

Numerator: $P(S_t=i, S_{t+1}=j, O_1, \cdots, O_T)$

$= P(O_1, \cdots, O_t, S_t=i)\, P(S_{t+1}=j \mid S_t=i, O_1, \cdots O_t)$

$P(O_{t+1} \mid S_{t+1}=j, S_t=i, O_1, \cdots, O_t)\, P(O_{t+2} \cdots O_T \mid S_{t+1}=j, S_t=i, O_1, \cdots, O_{t+1})$

product rule, cond. ind.

$= P(O_1, \cdots, O_t, S_t=i)\, P(S_{t+1}=j \mid S_t=i)\, P(O_{t+1} \mid S_{t+1}=j)$

$P(O_{t+2} \cdots O_T \mid S_{t+1}=j)$

$= \alpha_{it} \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{j,t+1}$

Denominator: (by normalization)

$$P(O_1, \cdots, O_T) = \sum_{i,j} P(S_t=i, S_{t+1}=j, O_1, \cdots, O_T)$$

Also correct.

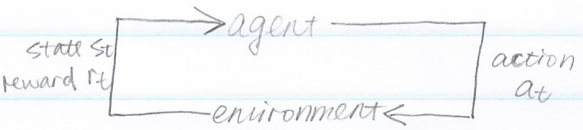
$$= \sum_{i,j} \alpha_{it} \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{j,t+1}$$

$$= \sum_{i} \alpha_{it} \beta_{it} = \sum_{j} \alpha_{j,t+1} \beta_{j,t+1}$$

HW. Compute $P(S_t=i \mid S_{t+1}=j, O_1, \cdots, O_T)$

Reinforcement Learning

Q: How should embodied / embedded / decision-making agents act and learn from experience in the world?



Ex: robot navigation, game playing AI

Textbook: Sutton / Barto

* Challenges
  - handling uncertainty
  - exploration vs. exploitation dilemma
  - delayed vs. immediate rewards — "temporal credit assignment"
  - evaluative vs. instructive feedback
  - complex worlds, computational guarantees