Review 07/14

* Belief network = DAG + CPTs

$$P(X_1, X_2, \dots X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, X_2, \dots, X_{i-1})$$

$$= \prod_{i=1}^{n} \underbrace{P(X_i \mid pa_i)}_{\text{CPTs}}$$

+ Learning from complete data

$$\{(X_1^{(t)}, X_2^{(t)}, \dots X_n^{(t)})\}_{t=1}^{T}$$

$$P_{ML}(X_i = x \mid pa_i = \pi) = \frac{\text{count}(X_i = x, pa_i = \pi)}{\text{count}(pa_i = \pi)}$$

$$= \frac{\sum_t I(X_i^{(t)}, x)\, I(pa_i^{(t)}, \pi)}{\sum_t I(pa_i^{(t)}, \pi)}$$

* Learning from incomplete data

examples $t = 1, 2, \dots, T$

visible nodes $V^{(t)}$

hidden nodes $H^{(t)}$

Maximize likelihood with EM algorithm:

$$P(X_i = x \mid pa_i = \pi) \leftarrow \frac{\sum_{t=1}^{T} P(X_i = x,\ pa_i = \pi \mid V^{(t)})}{\sum_{t=1}^{T} P(pa_i = \pi \mid V^{(t)})}$$

(new CPTs)

(computed from old CPTs)

Ex: $A \to B \to C$ complete data $\{(a_t, b_t, c_t)\}_{t=1}^{T}$

$$P_{ML}(B=b \mid A=a) = \frac{\text{count}(A=a,\ B=b)}{\text{count}(A=a)}$$

$$= \frac{\sum_t I(a^{(t)}, a)\, I(b^{(t)}, b)}{\sum_t I(a^{(t)}, a)}$$

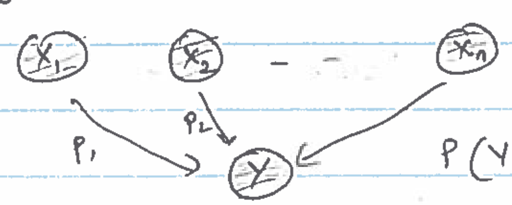
$A \to B \to C$ incomplete data $\{(a_t, c_t)\}_{t=1}^{T}$

$$P(B=b \mid A=a) \leftarrow \frac{\sum_t I(a^{(t)}, a)\, \boxed{P(b \mid a_t\, c_t)}}{\sum_t I(a^{(t)}, a)}$$

(new CPTs)

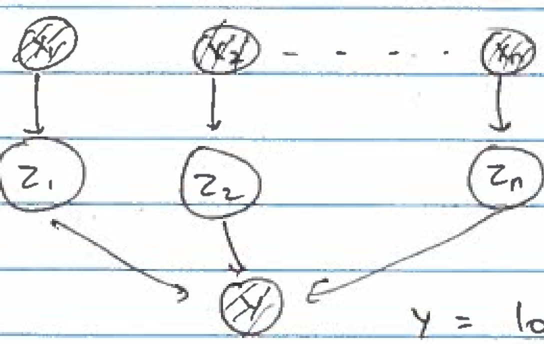
computed via Bayes rule using current CPTs

* EM algorithm for noisy-OR model




$$P(Y=1 \mid X) = 1 - \prod_{i=1}^{n} (1-p_i)^{x_i}$$

Equivalent to :




$P(Z_i = 0 | X_i) = (1-p_i)^{x_i}$

$Y = \text{logical-OR}(Z_1, Z_2 \dots Z_n)$

$P(Y=1 | \vec{X})$ same form as noisy-OR

---

$\boxed{\text{Markov models of language}}$

Let $w_\ell = \ell^{th}$ word in sentence. How to model $P(w_1\ w_2 \dots w_L)$ ? Shorthand $\vec{w} = (w_1\ w_2 \dots w_L)$

| Model | $P(\vec{w})$ | ML estimate | DAG |
|---|---|---|---|
| unigram | $\prod_\ell P_1(w_\ell)$ | $P_1(w) = \dfrac{\text{count}(w)}{\sum_{w'} \text{count}(w')}$ | $w_1 \quad w_2 \dots w_L$ |
| bigram | $P_1(w_1) \prod_{\ell=1} P(w_\ell \| w_{\ell-1})$ | $P_2(w' \| w) = \dfrac{\text{count}(w \to w')}{\text{count}(w)}$ | $w_1 \to w_2 \to \dots w_L$ |
| trigram | | | |

* Evaluating n-grams

- Train on corpus A, $P_1(\vec{w}) \leq P_2(\vec{w})$ on corpus A.
- Test on corpus B, $P_1(\vec{w}) \geq P_2(\vec{w})$ on corpus B

especially if $P_2(\vec{w}) = 0$ (unseen bigrams)

## Linear interpolation

* Also known as mixture model

$$P_M(w_l \mid w_{l-1}) = \lambda P_1(w_l) + (1-\lambda) P_2(w_l \mid w_{l-1})$$

How to estimate $\lambda$ ?

* Methodology

- Train $P_1, P_2$ on corpus A
- fix $P_1$ and $P_2$
- estimate $\lambda$ on corpus C

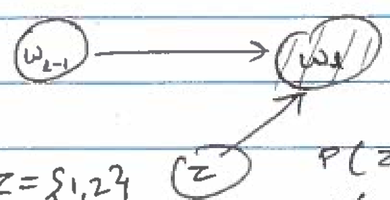
A = 'training set'
B = 'test set'
C = 'development set'

Choose $\lambda$ to maximize likelihood $\prod_{l=1}^{L} P_M(w_l \mid w_{l-1})$ on corpus C.

Why not estimate $\lambda$ on other corpora?
- Don't estimate $\lambda$ on B (test) - cheating
- Don't estimate $\lambda$ on A (train) - this would always yield $\lambda = 0$

* Hidden variable model

$$P(w_l \mid w_{l-1}, z) = \begin{cases} P_1(w_l) & \text{if } z=1 \\ P_2(w_l \mid w_{l-1}) & \text{if } z=2 \end{cases}$$




$z = \{1,2\}$

$P(z=1) = \lambda$
$P(z=2) = 1-\lambda$

How to estimate $\lambda$ on corpus $C$?

$\{(w_{\ell-1}, w_\ell)\}_{\ell=1}^{L}$ incomplete data

In this model:

$$P(w_\ell | w_{\ell-1}) = \sum_{z=1}^{2} P(w_\ell, z | w_{\ell-1}) \quad \text{marginalization}$$

$$= \sum_z P(z | w_{\ell-1}) P(w_\ell | z, w_{\ell-1}) \quad \text{product rule}$$

$$= \sum_z P(z) P(w_\ell | z, w_{\ell-1}) \quad \text{independence}$$

$$= P(z=1) P(w_\ell | w_{\ell-1}, z=1) + P(z=2) P(w_\ell | w_{\ell-1}, z=2)$$

$$= \lambda P_1(w_\ell) + (1-\lambda) P_2(w' | w) \quad \text{matches mixture model}$$

* E-step: compute posterior prob data

$$P(z | w_{\ell-1}, w_\ell) = \frac{P(w_\ell | z, w_{\ell-1}) \, P(z | \cancel{w_{\ell-1}})}{P(w_\ell | w_{\ell-1})} \quad \text{Bayes, marginal ind.}$$

$$P(z=1 | w_{\ell-1}, w_\ell) = \frac{\lambda P_1(w_\ell)}{\lambda P_1(w_\ell) + (1-\lambda) P_2(w_\ell | w_{\ell-1})}$$

$$P(z=2 | w_{\ell-1}, w_\ell) = 1 - P(z=1 | w_\ell, w_{\ell-1})$$

* M-step of EM algorithm

General rule:

$$P(\text{child}=c \mid pa=\pi) \leftarrow \frac{\sum_t P(\text{child}=c, pa_i=\pi \mid V^{(t)})}{\sum_t P(pa=\pi \mid V^{(t)})}$$

In this model:

$$P(z=1) \leftarrow \frac{\sum_{l=1}^{L} P(z=1 \mid w_{l-1}, w_l)}{L}$$

$$\boxed{\lambda \leftarrow \frac{1}{L} \sum_l P(z=1 \mid w_{l-1}, w_l)}$$

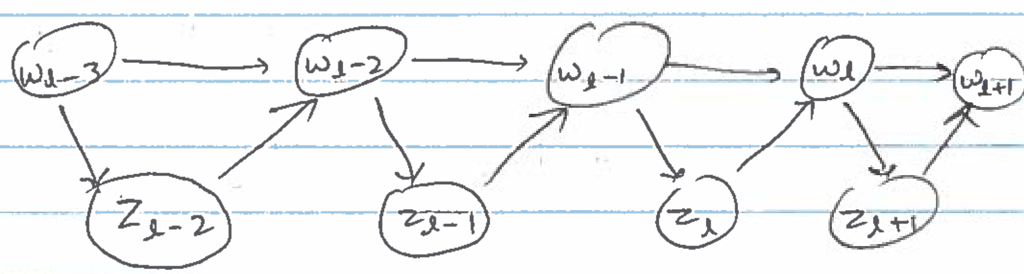
* Iterate EM:

$$\lambda \leftarrow \frac{1}{L} \sum P(z=1 \mid w_l, w_{l-1})$$

guarantees improvement on corpus C of log-likelihood

$$\mathcal{L}(\lambda) = \sum_l \log P_M(w_l \mid w_{l-1})$$

* In real-world application, mixing parameter would depend on previous word.




old $P(z_l = 1) = \lambda$

new $P(z_l = 1 \mid w_{l-1}) = \lambda_{w_{l-1}}$

* EM used to estimate as many params as words in vocab.

## Hidden Markov models (HMMs)

* Variables

$s_t \in \{1, 2, \ldots n\}$ hidden state at time $t$

$o_t \in \{1, 2, \ldots m\}$ observation at time $t$

(partial, noisy reflection of underlying hidden state)

Ex: house-training puppy

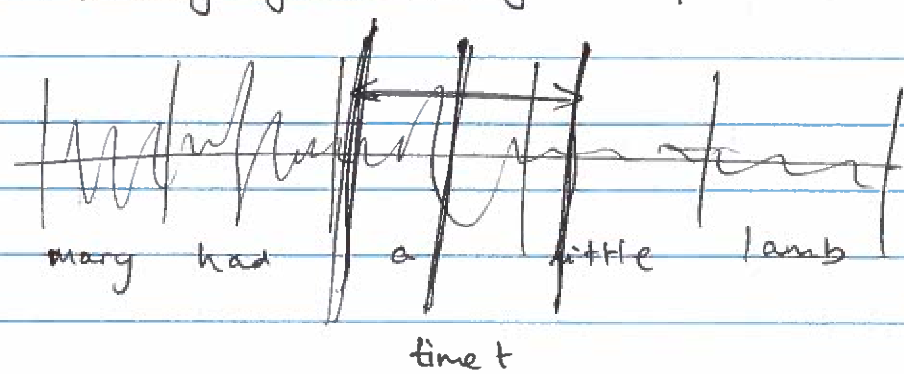
$s_t \in$ { have to go, doesn't need to, went }

$O_t \in$ { wagging tail, barking, whimpering, running ... }

Ex: speech recognition

$S_t$ = units of language (words, syllables, phonemes ...)

waveform
↓
English sentence




$O_t$ = acoustic measurements in sliding window centered at time $t$.

## Ex: robotics

$S_t$ = location, orientation, etc of robot at time $t$.

$O_t$ = sensor readings at time $t$
(camera, radar, heat map, ...)

### Markov assumptions

- finite context

$$P(S_t \mid S_1, S_2 \ldots S_{t-1}) = P(S_t \mid S_{t-1})$$

$$P(O_t \mid \underbrace{S_1, S_2 \ldots S_t, S_{t+1}, \ldots S_T}_{\text{all of time}}) = P(O_t \mid S_t)$$

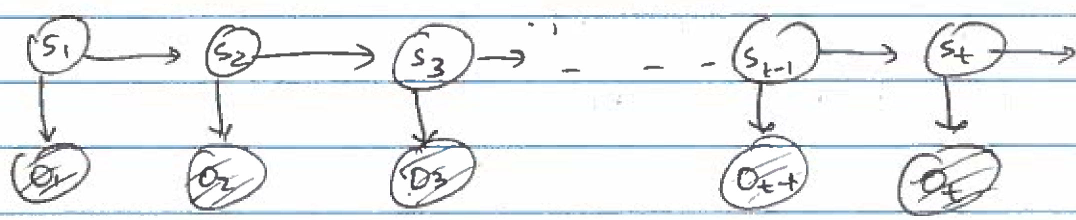
- Shared CPTs

$$P(S_{t+1} = s' \mid S_t = s) = P(S_{t+k+1} = s' \mid S_{t+k} = s)$$

$$P(O_t = o \mid S_t = s) = O(t+k = o \mid S_{t+k} = s)$$

### Belief network

$S_t \in \{1, 2, \ldots n\}$ hidden states
$O_t \in \{1, 2, \ldots m\}$ observations




Is it a polytree? Yes!

## Joint distribution

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

$\vec{s} = (s_1 s_2 \ldots s_T)$
$\vec{o} = (o_1 o_2 \ldots o_T)$ shorthand

## Parameters :

$\pi_i = P(s_1 = i)$ initial state distribution
$a_{ij} = P(s_{t+1} = j | s_t = i)$ transition matrix $(n \times n)$
$b_{ik} = P(o_t = k | s_t = i)$ emission matrix $(n \times m)$

## Key questions

1) How to compute likelihood $P(o_1 o_2 \ldots o_{T-1} o_T)$?

2) How to compute most likely hidden state sequence?

$$\underset{\vec{s}}{\operatorname{argmax}} \left\{ P(\underbrace{s_1 s_2 \ldots s_T}_{n^T \text{ possible state sequences of length } T} | o_1 o_2 \ldots o_T) \right\} = (s_1^*, s_2^* \ldots s_T^*)$$

$\vec{s} = (s_1, s_2 \ldots s_T)$

3) How to update beliefs for real-time monitoring?
How to compute $P(s_t = i | o_1, o_2 \ldots o_{t-1} o_t)$?

4) How to learn an HMM from data? How to estimate $\{\pi_i, a_{ij}, b_{ik}\}$ to maximize $P(o_1 o_2 \ldots o_T)$?

(Use EM algorithm)

Questions 1-3 are inference for fixed HMM with given $\{\pi_i, a_{ij}, b_{ik}\}$

Question 4 - learning

i) How to compute likelihood?

$$P(o_1 o_2 \ldots o_T) = \underbrace{\sum_{\vec{S}}}_{\text{sum of } n^T \text{ sequences of hidden states}} P(s_1 s_2 \ldots s_T, o_1 o_2 \ldots o_T) \quad \text{marginalization}$$

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

* Efficient recursion

$$P(o_1, o_2 \ldots o_{t-1}, o_t, o_{t+1}, s_{t+1} = j)$$

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

$$= \sum_{i=1}^{n} P\underbrace{(o_1 o_2 \ldots o_t, s_t = i)}_{\text{up to time } t} P(s_{t+1} = j | s_t = i, o_1 \ldots o_t) \times P(o_{t+1} | s_{t+1} = j, s_t = i, o_1 \ldots o_t) \quad \text{(product rule)}$$

$$= \sum_{i=1}^{n} \underbrace{P(o_1 \ldots o_t, s_t = i)}_{\text{recursive instance}} \underbrace{P(s_{t+1} = j | s_t = i)}_{a_{ij}} \underbrace{P(o_{t+1} | s_{t+1} = j)}_{b_{j, o_{t+1}}} \quad \text{(CI)}$$

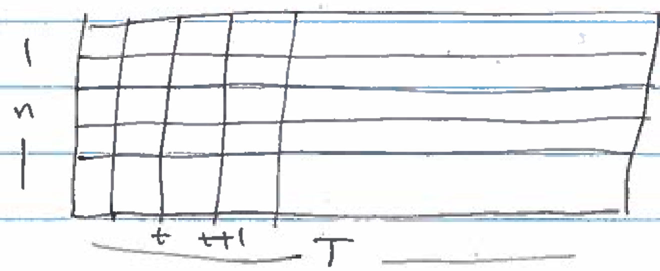
CPTs

* shorthand

$$\alpha_{it} \doteq P(O_1 \ldots O_t, S_t = i)$$

↳ $i = 1 \ldots n$ # hidden states

$t = 1 \ldots T$ sequence length.



* Forward algorithm in HMMs

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

$$\alpha_{i1} = P(O_1, S_1 = i) \quad \text{by definition}$$
$$= P(S_1 = i)\, P(O_1 | S_1 = i)$$
$$= \pi_i \underbrace{b_i(O_1)}_{\text{emission matrix element } b_{iO_1}}$$

- recursion step from time $t$ to $t+1$

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

* Back to likelihood :

$$P(O_1 O_2 \ldots O_T) \overset{\text{marginalization}}{=} \sum_{i=1}^{n} P(O_1 O_2 \ldots O_T, S_T = i)$$

$$= \sum_{i=1}^{n} \alpha_{iT}$$

* Scales as $O(Tn^2)$

- linear, not exponential, in sequence length.
- quadratic in # hidden states $n$

* Warning: naive calculation with underflow for long sequences $T >> 1$ because $P(O_1 O_2 \ldots O_T) \ll 1$
(fix by rescaling at each iteration computing log-likelihood)

2) How to compute most likely state sequence?

$$\vec{s}^* = \{s_1^*, s_2^* \ldots s_T^*\}$$

$$= \underset{\vec{s}}{\mathrm{argmax}} \left[ P(s_1 s_2 \ldots s_T \mid O_1 O_2 \ldots O_T) \right]$$

$$= \underset{\vec{s}}{\mathrm{argmax}} \left[ \frac{P(s_1 s_2 \ldots s_T, O_1 \ldots O_T)}{\underbrace{P(O_1 O_2 \ldots O_T)}_{\text{ind of } \vec{s}}} \right] \quad \text{product rule}$$

$$= \underset{\vec{s}}{\mathrm{argmax}} \left[ P(s_1 s_2 \ldots s_T, O_1 \ldots O_T) \right]$$

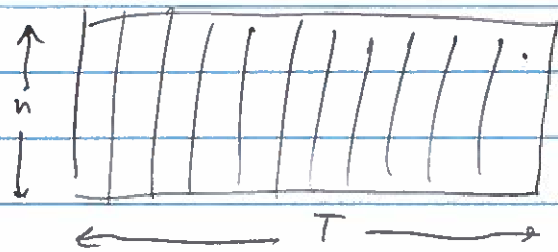
How to compute $s^*$?

Define:

$$l^*_{it} = \max_{s_1 s_2 \dots s_{t-1}} \left\{ \log P(s_1\ s_2 \dots s_{t-1}, s_t = i, o_1, \dots o_t) \right\}$$

$i = 1 \dots n$ # hidden states

$t = 1 \dots T$ sequence length




= log-probability of most likely sub-sequence of hidden states $s_1\ s_2 \dots s_t$ that ends in state $s_t = i$ and explains observations $o_1, o_2 \dots o_t$