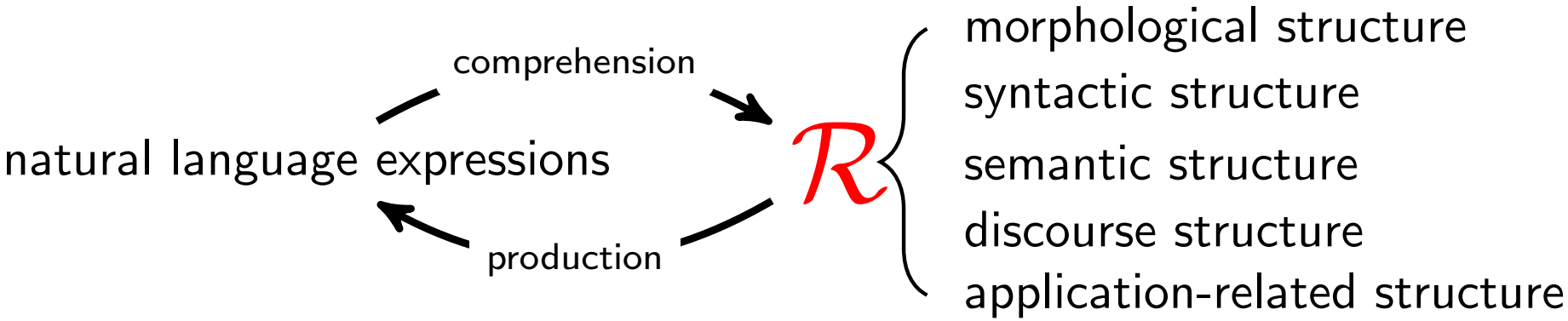
Overview of Natural Language Processing
Part II & ACS L390

Lecture 7: Grammar Induction

Weiwei Sun

Department of Computer Science and Technology
University of Cambridge

Michaelmas 2024/25



How can we get proper $\mathcal{R}$?

Learning from incomplete data

- Observed data $D = \{x^{(1)}, x^{(2)}, \ldots, x^{(u)}\}$ ▷e.g. word sequences
- Hidden structure $\mathcal{H}$ ▷e.g. POS tag sequences
- $p(D;\theta) = \sum_{\mathcal{H}} p(D, \mathcal{H};\theta)$
- **Goal**: find θ to maximize $p(D;\theta)$

Expectation Maximisation

EM in general

Challenge

$$\log p(D;\theta) = \log \sum_{\mathcal{H}} p(D,\mathcal{H};\theta)$$

How

- An EM iteration starts from $\theta^{(t)}$
- The E-step:

$$q(\mathcal{H}) = p(\mathcal{H}|D;\theta^{(t)})$$

- The M-step:

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_{\mathcal{H}} q(\mathcal{H}) \log \frac{p(D,\mathcal{H};\theta)}{q(\mathcal{H})}$$

EM in general

Challenge

$$\log p(D;\theta) = \log \sum_{\mathcal{H}} p(D,\mathcal{H};\theta)$$

How

- An EM iteration starts from $\theta^{(t)}$
- The E-step: (E=Expectation)

$$q(\mathcal{H}) = p(\mathcal{H}|D;\theta^{(t)})$$

- The M-step: (M=Maximization)

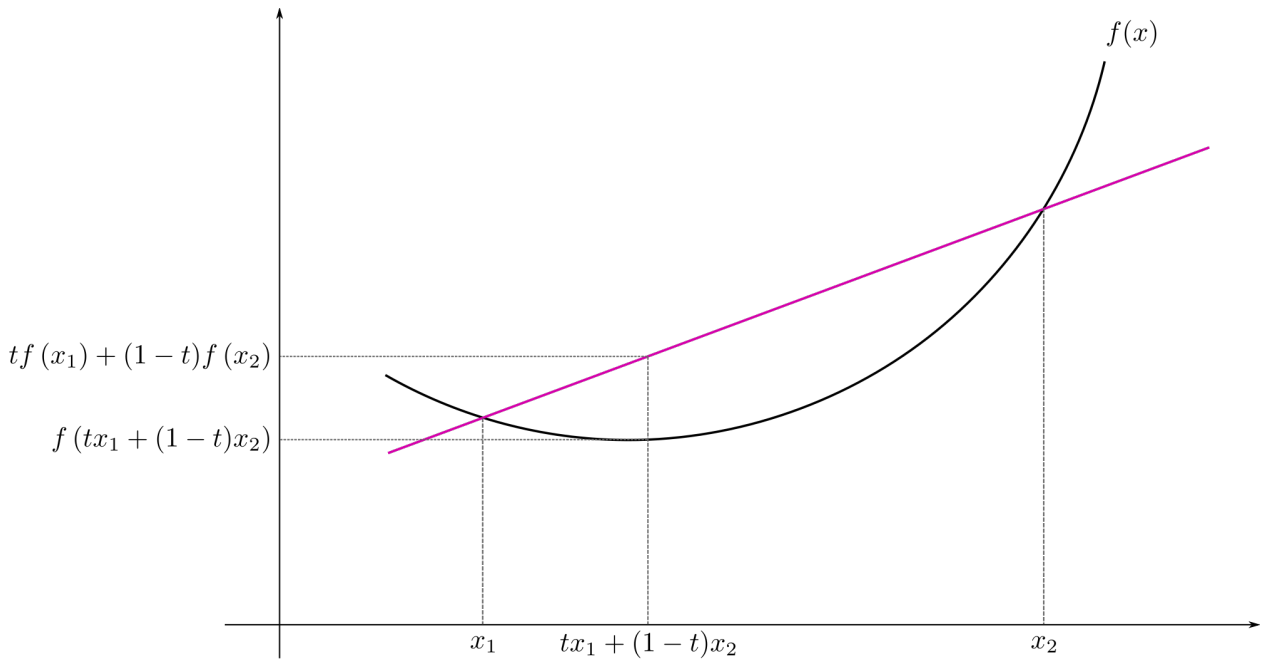
$$\theta^{(t+1)} = \arg\max_{\theta} \sum_{\mathcal{H}} q(\mathcal{H}) \log \frac{p(D,\mathcal{H};\theta)}{q(\mathcal{H})}$$

Soundness (1)

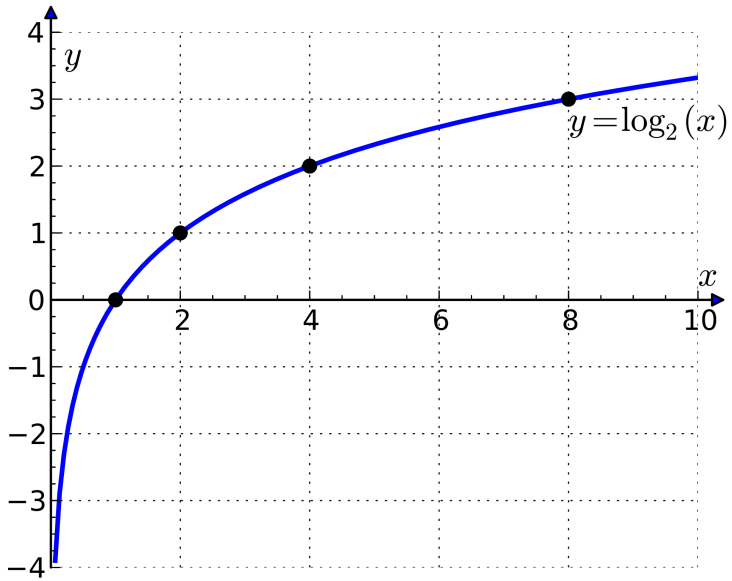
Jensen's inequality

Let f be a convex function, and let X be a random variable. Then:

$$E[f(X)] \geq f(E[X])$$



Soundness (2)



$$
\begin{aligned}
L(\theta) &= \sum_i \log p(x^{(i)}; \theta) \\
&= \sum_i \log \sum_y p(x^{(i)}, y; \theta) \\
&= \sum_i \log \sum_y q_i(y) \frac{p(x^{(i)}, y; \theta)}{q_i(y)} \\
&\geq \sum_i \sum_y q_i(y) \log \frac{p(x^{(i)}, y; \theta)}{q_i(y)}
\end{aligned}
$$

Key idea

- $\sum_i \sum_y q_i(y) \log \frac{p(x^{(i)}, y; \theta)}{q_i(y)}$ is a lower bound of $L(\theta)$.
- If this bound is tight, we can instead maximize it.

Soundness (3)

Key idea

$\sum_i \sum_y q_i(y) \log \frac{p(x^{(i)},y;\theta)}{q_i(y)}$ is a lower bound of $L(\theta)$.

Make the bound tight for a particular value of θ

$$\log \sum_y q_i(y) \frac{p(x^{(i)}, y; \theta)}{q_i(y)} ? = \sum_y q_i(y) \log \frac{p(x^{(i)}, y; \theta)}{q_i(y)}$$

Posterior Distribution

$$\begin{aligned} q_i(y) &\propto p(x^{(i)}, y; \theta) \\ q_i(y) &= \frac{p(x^{(i)}, y; \theta)}{\sum_y p(x^{(i)}, y; \theta)} \\ &= p(y|x^{(i)}; \theta) \end{aligned}$$

Soundness (4)

The EM algorithm

E-step:

$$q_i(y) = p(y|x^{(i)}; \theta^{(t)})$$

M-step:

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_i \sum_y q_i(y) \log \frac{p(x^{(i)}, y; \theta)}{q_i(y)}$$

Soundness (5)

$$L_b(\theta^{(t)}) = \sum_i \sum_y q_i^{(t)}(y) \log \frac{p(x^{(i)}, y; \theta^{(t)})}{q_i^{(t)}(y)}$$

Convergence

$$\begin{aligned} L_b(\theta^{(t+1)}) &\geq \sum_i \sum_y q_i^{(t)}(y) \log \frac{p(x^{(i)}, y; \theta^{(t+1)})}{q_i^{(t)}(y)} \\ &\geq \sum_i \sum_y q_i^{(t)}(y) \log \frac{p(x^{(i)}, y; \theta^{(t)})}{q_i^{(t)}(y)} \\ &= L_b(\theta^{(t)}) \end{aligned}$$

Hidden Markov Model and EM

Probabilistic models for sequence pairs

- We have two sequences of random variables: $X_1, X_2, ..., X_n$ and $S_1, S_2, ..., S_n$
- Intuitively, each X_i corresponds to an observation and each S_i corresponds to an underlying state that generated the observation. Assume that each S_i is in $\{1, 2, ..., k\}$, and each X_i is in $\{1, 2, ..., o\}$.
- How do we model the joint distribution

$$P(X_1 = x_1, ..., X_n = x_n, S_1 = s_1, ..., S_n = s_n)$$

Hidden Markov Models

An HMM takes the following form

$$p(x_1...x_n, s_1...s_n; \theta) = t(s_1) \prod_{j=2}^{n} t(s_j|s_{j-1}) \prod_{j=1}^{n} e(x_j|s_j)$$

Parameters in the model

1. Initial state parameters ϕ_s for $s \in \{1, 2, ..., k\}$
2. Transition parameters $\phi_{s'|s}$ for $s, s' \in \{1, 2, ..., k\}$
3. Emission parameters $\phi_{e|s}$ for $s \in \{1, 2, ..., k\}$ and $e \in \{1, 2, ..., o\}$

If we use a specific symbol to denote *stop of a sequence*: $s_0 = *$

- Initial state parameters $\phi_{s|*}$
- Just look like transition parameters

Parameter estimation with fully observed data

- $\underline{\text{count}}(i, s \to s')$: the number of times state s' follows state s in the i'th training example.

$$\underline{\text{count}}(i, s \to s') = \sum_{j=1}^{|s^{(i)}|-1} \mathbb{1}_{\{s_j^{(i)}=s \wedge s_{j+1}^{(i)}=s'\}}$$

- $\underline{\text{count}}(i, s \to e)$: the number of times state s is paired with emission x in the i'th training example.

$$\underline{\text{count}}(i, s \to e) = \sum_{j=1}^{|s^{(i)}|} \mathbb{1}_{\{s_j^{(i)}=s \wedge e_j^{(i)}=e\}}$$

- $\underline{\text{count}}(i, s)$: to be 1 if state s is the initial state, and 0 otherwise.

$$\underline{\text{count}}(i, s) = \mathbb{1}_{\{s_1^{(i)}=s\}}$$

$\mathbb{1}_A$ is an indicator function: $\mathbb{1}_A(x) = 1$ if $x \in A$, and $\mathbb{1}_A(x) = 0$ otherwise.

Maximum-likelihood estimation

1. Transition probabilities:

$$\phi_{s'|s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s \to s')}{\sum_{i=1}^{m} \sum_{s'} \underline{\text{count}}(i, s \to s')}$$

2. Emission probabilities:

$$\phi_{e|s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s \to e)}{\sum_{i=1}^{m} \sum_{e} \underline{\text{count}}(i, s \to e)}$$

3. Initial state probabilities:

$$\phi_{s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s)}{m}$$

The Viterbi algorithm (1)

Task 1: get the most likely state sequence for x is

$$\arg \max_{s_1,\ldots,s_n} p(x_1 \ldots x_n, s_1 \ldots s_n; \theta)$$

Define a dynamic programming table

$\pi_t(s)$ = maximum probability of a tag sequence ending in tag s at position t

that is,

$$\pi_t(s) = \max_{s_1,\ldots,s_{t-1}} \left(\prod_{j=1}^{t-1} \phi_{s_j|s_{j-1}} \phi_{x_j|s_j} \right) \times \phi_{s_t|s_{t-1}} \times \phi_{x_t|s_t}$$

The Viterbi algorithm (2)

$$\pi_t(s) = \max_{s_1,\ldots,s_{t-1}} \left(\prod_{j=1}^{t-1} \phi_{s_j|s_{j-1}} \phi_{x_j|s_j} \right) \times \phi_{s_t|s_{t-1}} \times \phi_{x_t|s_t}$$

Recursive definition

- Base case:

$$\pi_1(s) = \phi_{x_1|s} \times \phi_{s|*}$$

- For any $t \in \{2, \ldots, n\}$, for any $s \in \{1, \ldots, k\}$:

$$\pi_t(s) = \max_{s' \in \{1,\ldots,k\}} (\pi_{t-1}(s') \times \phi_{s|s'} \times \phi_{x_t|s})$$

The Viterbi algorithm (3)

- **Input:** a sentence $x_1, ...x_n$, parameters $\phi_{s'|s}$ and $\phi_{e|s}$
- **Initialization**: Set $\pi_0 * = 1$, and $\pi_0(s) = 0$ for all $s \in \{1, ..., k\}$
- **Algorithm**:
  - For $t = 1, ..., n$,
    - For $s \in \{1, ..., k\}$,

$$\pi_t(s) = \max_{s' \in \{1,...,k\}} (\pi_{t-1}(s') \times \phi_{s|s'} \times \phi_{x_t|s})$$

$$bp_t(s) = \arg \max_{s' \in \{1,...,k\}} (\pi_{t-1}(s') \times \phi_{s|s'} \times \phi_{x_t|s})$$

  - Set

$$s_n = \arg \max_{s' \in \{1,...,k\}} (\pi_n(s') \times \phi_{s'|s} \times \phi_{*|s})$$

  - For $t = (n-1), ..., 1$,

$$s_t = bp_{t+1}(s_{t+1})$$

- **Return** the tag sequence $s_1, ..., s_n$.

The forward algorithm (1)

Task 2: for a given input sequence $x_1, ..., x_n$, find

$$p(x_1 ... x_n; \theta) = \sum_{s_1, ..., s_n} p(x_1 ... x_n, s_1 ... s_n; \theta)$$

Define a dynamic programming table

$\alpha_t(s)$ = sum of probabilities of all tag sequences ending in tag s at position t

that is,

$$\alpha_t(s) = \sum_{s_1, ..., s_{t-1}} \left(\prod_{j=1}^{t-1} \phi_{s_j|s_{j-1}} \phi_{x_j|s_j} \right) \times \phi_{s_t|s_{t-1}} \times \phi_{x_t|s_t}$$

The forward algorithm (2)

$$\alpha_t(s) = \sum_{s_1,\ldots,s_{t-1}} \left(\prod_{j=1}^{t-1} \phi_{s_j|s_{j-1}} \phi_{x_j|s_j} \right) \times \phi_{s_t|s_{t-1}} \times \phi_{x_t|s_t}$$

Recursive definition

- For $s \in \mathcal{S}$,

$$\alpha_1(s) = \phi_{x_1|s} \times \phi_{s|*}$$

- For any $t \in \{2, \ldots, n\}$, for any $s \in \mathcal{S}$:

$$\alpha_t(s) = \sum_{s' \in \mathcal{S}} \alpha_{t-1}(s') \times \phi_{s|s'} \times \phi_{x_t|s}$$

The forward algorithm (3)

- **Input:** a sentence $x_1, ...x_n$, parameters $\phi_{s'|s}$ and $\phi_{e|s}$
- **Initialization**: Set $\alpha_0* = 1$, and $\alpha_0(s) = 0$ for all $s \in \{1, ..., k\}$
- **Algorithm**:
  - For $t = 1, ..., n$,
    - For $s \in \{1, ..., k\}$,

$$\alpha_t(s) = \sum_{s' \in \{1,...,k\}} (\alpha_{t-1}(s') \times \phi_{s|s'} \times \phi_{x_t|s})$$

- **Return**:

$$\sum_{s \in \{1,...,k\}} \alpha_n(s)$$

Generalization: Semirings (1)

The Viterbi algorithm and the forward algorithm are almost the same.

- Viterbi and Forward algorithms correspond to exactly the same calculation, except one maximizes and the other sums.

Only some mathematical properties about operations are relevant $\Rightarrow$
Thinking about abstract algebra

- The same abstract algorithm instantiated in two different semirings.

Semirings

A semiring is a set $\mathcal{A}$ equipped with two binary operations $\oplus$ and $\otimes$, such that:

- $\oplus$ is associative and commutative
  - $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
  - $a \oplus b = b \oplus a$
- $\otimes$ is associative and distributes over $\oplus$
  - $(a \otimes b) \otimes c = a \otimes (b \otimes c)$
  - $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
  - $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
- Identity elements
  - $a \oplus \mathbf{0} = a$
  - $a \otimes \mathbf{1} = a$
  - $a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0}$

	Forward	Viterbi
$\mathcal{A}$	$\mathbb{R}_{\geq 0}$	$\mathbb{R}_{\geq 0}$
$\oplus$	$a + b$	$\max(a, b)$
$\otimes$	$a \times b$	$a \times b$
$\mathbf{0}$	0	0
$\mathbf{1}$	1	1

Generalization: Semirings (2)

Generalization

To calculate:

$$\bigoplus_{\mathbf{s}\in\mathcal{S}^n} \left(\bigotimes_{t=2}^{n} \phi(s_t, s_{t-1}) \right)$$

A dynamic programming solution:

$$\gamma_t(s) \quad = \quad \bigoplus_{s'\in\mathcal{S}} \left(\gamma_{t-1}(s') \otimes \phi(S_t = s, S_{t-1} = s') \right)$$

Generalization: Semirings (3)

If

$$\gamma_t(s) = \bigoplus_{s_1 \ldots s_{t-1} \in \mathcal{S}, S_t = s} \left(\bigotimes_{j=2}^{t} \phi(s_j, s_{j-1}) \right)$$

then,

$$\begin{aligned}
\gamma_{t+1}(s) &= \bigoplus_{s' \in \mathcal{S}} \left(\gamma_t(s') \otimes \phi(S_{t+1} = s, S_t = s') \right) \\
&= \bigoplus_{s' \in \mathcal{S}} \left\{ \left[\bigoplus_{s_1 \ldots s_{t-1} \in \mathcal{S}, S_t = s'} \left(\bigotimes_{j=2}^{t} \phi(s_j, s_{j-1}) \right) \right] \otimes \phi(s, s') \right\} \\
&= \bigoplus_{s' \in \mathcal{S}} \left\{ \bigoplus_{s_1 \ldots s_{t-1} \in \mathcal{S}, S_t = s'} \left[\left(\bigotimes_{j=2}^{t} \phi(s_j, s_{j-1}) \right) \otimes \phi(s, s') \right] \right\}
\end{aligned}$$

Generalization: Semirings (4)

$$
\begin{aligned}
\gamma_{t+1}(s) &= \bigoplus_{s' \in \mathcal{S}} \left\{ \bigoplus_{s_1 \ldots s_{t-1} \in \mathcal{S}, S_t = s'} \left[\left(\bigotimes_{j=2}^{t} \phi(s_j, s_{j-1}) \right) \otimes \phi(s, s') \right] \right\} \\
&= \bigoplus_{s' \in \mathcal{S}} \left(\bigoplus_{s_1 \ldots s_{t-1} \in \mathcal{S}, S_t = s', S_{t+1} = s} \left(\bigotimes_{j=2}^{t+1} \phi(s_j, s_{j-1}) \right) \right) \\
&= \bigoplus_{s_1 \ldots s_t \in \mathcal{S}, S_{t+1} = s} \left(\bigotimes_{j=2}^{t+1} \phi(s_j, s_{j-1}) \right)
\end{aligned}
$$

By induction, we can prove,

$$
\bigoplus_{\mathbf{s} \in \mathcal{S}^n} \left(\bigotimes_{t=2}^{n} \phi(s_t, s_{t-1}) \right) = \bigoplus_{s \in \mathcal{S}} \gamma_n(s)
$$

Probability of each state/transition

- Transition probabilities:

$$\phi_{s'|s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s \to s')}{\sum_{i=1}^{m} \sum_{s'} \underline{\text{count}}(i, s \to s')}$$

- Emission probabilities:

$$\phi_{e|s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s \to e)}{\sum_{i=1}^{m} \sum_{e} \underline{\text{count}}(i, s \to e)}$$

Probability of each state/transition

- Transition probabilities:

$$\phi_{s'|s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s \to s')}{\sum_{i=1}^{m} \sum_{s'} \underline{\text{count}}(i, s \to s')}$$

- Emission probabilities:

$$\phi_{e|s} = \frac{\sum_{i=1}^{m} \underline{\text{count}}(i, s \to e)}{\sum_{i=1}^{m} \sum_{e} \underline{\text{count}}(i, s \to e)}$$

- For $j = \{1, ..., n\}$ and $s = \mathcal{S}$, find

$$p(x_1 ... x_n, S_j = s; \theta) = \sum_{s_j = s} p(x_1, ..., x_n, s_1, ..., s_n; \theta)$$

- For $j = \{1, ..., n-1\}$ and $s, s' = \mathcal{S}$, find

$$p(x_1 ..., S_j = s, S_{j+1} = s'; \theta) = \sum_{s_j = s, s_{j+1} = s'} p(x_1, ..., x_n, s_1, ..., s_n; \theta)$$

The forward-backward algorithm (1)

- ...: Something independent with $S_j, S_{j+1}, ...$
- ...: Something independent with $..., S_{j-1}, S_j.$
- Consider: $aA + aB + bA + bB = (a + b)(A + B)$

Decompostion

$$\sum_{S_j=s} ...p(S_j = s|s_{j-1})p(x_j|S_j = s)p(s_{j+1}|S_j = s)p(x_{j+1}|s_{j+1})...$$

$$= \underbrace{\sum_{S_j=s} ...p(S_j = s|s_{j-1})p(x_j|S_j = s)}_{forward} \underbrace{\sum_{S_j=s} p(s_{j+1}|S_j = s)p(x_{j+1}|s_{j+1})...}_{backward}$$

Forward and backward probabilities

Forward probabilities $\alpha_t(s)$

Sum of probabilities of all tag sequences ending in tag s at position t:

$$\alpha_t(s) = \sum_{s_0=*, s_1 \ldots s_{t-1}} \left[\left(\prod_{j=1}^{t-1} \phi_{s_j|s_{j-1}} \phi_{x_j|_j s_j} \right) \phi_{s|s_{t-1}} \phi_{x_t|s} \right]$$

Backward probabilities $\beta_t(s)$

Sum of probabilities of all tag sequences starting with state s at position j and going to the end of the sequence:

$$\beta_t(s) = \sum_{s_{t+1} \ldots s_n} \left[\phi_{x_{t+1}|s_{t+1}} \phi_{s_{t+1}|s} \left(\prod_{j=t+2}^{n} \phi_{s_j|s_{j-1}} \phi_{x_j|_j s_j} \right) \right]$$

Recursive definitions of the backward probabilities

Recursive definition

- For $s \in \mathcal{S}$,

$$\beta_n(s) = 1$$

- For any $t \in \{n-1, ..., 1\}$, for any $s \in \mathcal{S}$:

$$\beta_t(s) = \sum_{s' \in \mathcal{S}} \beta_{t+1}(s') \times \phi_{s'|s} \times \phi_{x_{t+1}|s'}$$

The forward-backward algorithm (2)

$$p(x_1 ... x_n, S_j = s) = \sum_{S_j=s} \left(\prod_{i=1}^{n} p(s_i|s_{i-1}) p(x_i|s_i) \right)$$

$$= \sum_{S_j=s} \left(\prod_{i=1}^{j-1} p(s_i|s_{i-1}) p(x_i|s_i) \right) \times p(s_j|s_{j-1}) p(x_j|s_j)$$

$$\times p(s_{j+1}|s_j) p(x_{j+1}|s_{j+1}) \times \left(\prod_{i=j+2}^{n} p(s_i|s_{i-1}) p(x_i|s_i) \right)$$

$$= \alpha_j(s) \times \beta_j(s)$$

Consider: $aA + aB + bA + bB = (a + b)(A + B)$

Per-word posterior decoding

- Once you have all forward/backward probabilites, you can calculate the posteriors for every word's label.

$$\hat{s}_i = \arg\max_{s \in \mathcal{S}} p(S_i = s | \mathbf{x}; \theta)$$

- This is sometimes called posterior decoding.
- Compare with MAP decoding (sometimes called Viterbi decoding):

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s} \in \mathcal{S}^n} p(\mathbf{s} | \mathbf{x}; \theta)$$

Calculate $p(x_1 ... x_n, S_{j-1} = s', S_j = s)$ is similar to $p(x_1 ... x_n, S_j = s)$.

Unsupervised training for an HMM

Now we are able to compute *expected counts*:

$$\underline{\text{count}}(s \to s') = E_\theta(s \to s' | \mathbf{x}^{(i)})$$

and

$$\underline{\text{count}}(s \to x) = E_\theta(s \to x | \mathbf{x}^{(i)})$$

Then we are able to learn parameters using EM!

Improving supervised HMM tagger

Treat additional information as latent variables

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N

Bigram	92.25
Bigram+LA	94.53

Improving supervised HMM tagger

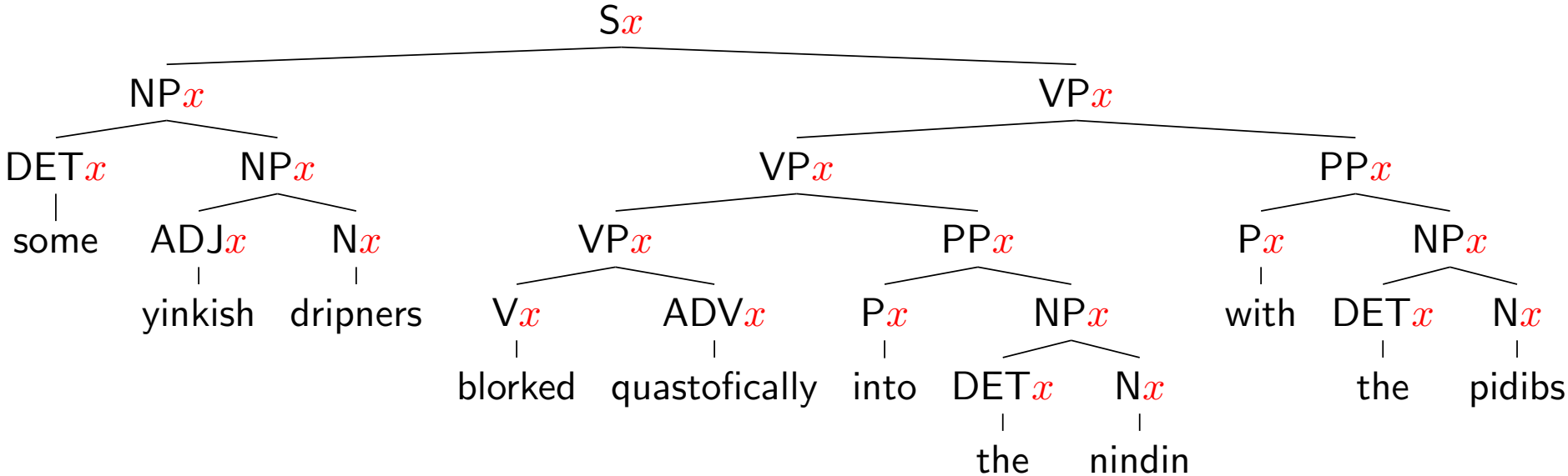
Treat additional information as latent variables

Profits/Nx soared/Vx at/Px Boeing/Nx Co./Nx ,/,x easily/ADVx topping/Vx forecasts/Nx on/Px Wall/Nx Street/Nx ,/,x as/Px their/POSSx CEO/Nx Alan/Nx Mulally/Nx announced/Vx first/ADJx quarter/Nx results/Nx

Bigram	92.25
Bigram+LA	94.53

Improving supervised PCFG parsing

Latent annotation or symbol refinement



Improvement: roughly 72.6 $\rightarrow$ 91.1.

Compound PCFG for grammar induction

Kim et al. (2019) https://aclanthology.org/P19-1228.pdf

$$q(T \to w; \boldsymbol{z}) = \frac{\exp(\boldsymbol{u}_w^\top f_2([\boldsymbol{w}_T; \boldsymbol{z}]))}{\sum_{w' \in \Sigma} \exp(\boldsymbol{u}_{w'}^\top f_2([\boldsymbol{w}_T; \boldsymbol{z}]))}$$

$$q(A \to BC; \boldsymbol{z}) = \frac{\exp(\boldsymbol{u}_{BC}^\top [\boldsymbol{w}_A; \boldsymbol{z}])}{\sum_{B'C'} \exp(\boldsymbol{u}_{B'C'}^\top [\boldsymbol{w}_A; \boldsymbol{z}])}$$

Where

$$f_i(\boldsymbol{x}) = g_{i,1}(g_{i,2}(W_i, \boldsymbol{x}))$$
$$g_{i,j}(\boldsymbol{y}) = \mathsf{ReLU}(V_{i,j}\mathsf{ReLU}(U_{i,j}\boldsymbol{y})) + \boldsymbol{y}$$

Readings

- Appendix A: Hidden Markov Models. *Speech and Language Processing*. D Jurafsky and J Martin.
  `https://web.stanford.edu/~jurafsky/slp3/A.pdf`