Language and Statistics II

Lecture 19: EM for Models of Structure

Noah Smith

Expectation-Maximization

- E step:

$$\forall i, y, q(y|x_i) \leftarrow p_{\vec{\theta}^{(t)}}(y|x_i) = \frac{p_{\vec{\theta}^{(t)}}(x_i, y)}{\sum_{y'} p_{\vec{\theta}^{(t)}}(x_i, y')}$$

soft assignment or voting

- M step:

$$\vec{\theta}^{(t+1)} \leftarrow \arg\max_{\vec{\theta}} \sum_{x,y} \underbrace{\tilde{p}(x) q(y|x)}_{\text{"pretend" } \tilde{p}(x,y)} \log p_{\vec{\theta}}(x, y)$$

fully-observed data MLE

Proof that EM = Partial-Data MLE

- Claim: EM iterations improve likelihood, converging to a local optimum.

maximizing likelihood

$$\prod_{i=1}^{n} p_{\vec{\theta}}(x_i) = \prod_{i=1}^{n} \sum_{y} p_{\vec{\theta}}(x_i, y) \cong \sum_{i=1}^{n} \log \sum_{y} p_{\vec{\theta}}(x_i, y) \cong \sum_{x} \tilde{p}(x) \log \sum_{y} p_{\vec{\theta}}(x, y)$$

the M step

$$\sum_{x,y} \tilde{p}(x) q(y|x) \log p_{\vec{\theta}}(x, y) = \sum_{x} \tilde{p}(x) \sum_{y} q(y|x) \log p_{\vec{\theta}}(x, y)$$

MLE-Objectve - M-Step-Objective

$$\overbrace{\sum_x \tilde{p}(x)\log\sum_y p_{\vec{\theta}}(x,y)}^{\text{what MLE wants maximized}} - \overbrace{\sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(x,y)}^{\text{what the M step maximizes}}$$

$$= \sum_x \tilde{p}(x)\sum_y q(y|x)\log\sum_{y'} p_{\vec{\theta}}(x,y') - \sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(x,y)$$

$$= -\sum_x \tilde{p}(x)\sum_y q(y|x)\log\frac{p_{\vec{\theta}}(x,y)}{\sum_{y'} p_{\vec{\theta}}(x,y')}$$

$$= -\sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(y|x)$$

$$\overbrace{\sum_x \tilde{p}(x)\log\sum_y p_{\vec{\theta}}(x,y)}^{\text{what MLE wants maximized: }\Lambda} = \overbrace{\sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(x,y)}^{\text{what the M step maximizes: }\Phi} - \overbrace{\sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(y|x)}^{\Delta}$$

Central Claim

$$\Lambda\left(\vec{\theta}^{(t+1)}\right) \geq \Lambda\left(\vec{\theta}^{(t)}\right)$$

$$\Phi\left(\vec{\theta}^{(t+1)}, q\right) - \Delta\left(\vec{\theta}^{(t+1)}, q\right) \geq \Phi\left(\vec{\theta}^{(t)}, q\right) - \Delta\left(\vec{\theta}^{(t)}, q\right)$$

part 1: M step

$$\Phi\left(\vec{\theta}^{(t+1)}, q\right) = \max_{\vec{\theta}} \Phi\left(\vec{\theta}, q\right) \geq \Phi\left(\vec{\theta}^{(t)}, q\right)$$

part 2: E step

$$\Delta\left(\vec{\theta}^{(t)}, q\right) - \Delta\left(\vec{\theta}^{(t+1)}, q\right) = \sum_x \tilde{p}(x) \sum_y q(y|x) \log \frac{p_{\vec{\theta}^{(t)}}(y|x)}{p_{\vec{\theta}^{(t+1)}}(y|x)}$$

$$= \sum_x \tilde{p}(x) \sum_y p_{\vec{\theta}^{(t)}}(y|x) \log \frac{p_{\vec{\theta}^{(t)}}(y|x)}{p_{\vec{\theta}^{(t+1)}}(y|x)} = \mathbf{E}_{\tilde{p}}\left[D\left(p_{\vec{\theta}^{(t)}} \middle\| p_{\vec{\theta}^{(t+1)}}\right)\right] \geq 0$$

$$\overbrace{\sum_x \tilde{p}(x) \log \sum_y p_{\vec{\theta}}(x,y)}^{\text{what MLE wants maximized: } \Lambda} = \overbrace{\sum_x \tilde{p}(x) \sum_y q(y|x) \log p_{\vec{\theta}}(x,y)}^{\text{what the M step maximizes: } \Phi} - \overbrace{\sum_x \tilde{p}(x) \sum_y q(y|x) \log p_{\vec{\theta}}(y|x)}^{\Delta}$$

Central Claim

$$\Lambda\left(\vec{\theta}^{(t+1)}\right) \geq \Lambda\left(\vec{\theta}^{(t)}\right)$$

$$\Phi\left(\vec{\theta}^{(t+1)},q\right) - \Delta\left(\vec{\theta}^{(t+1)},q\right) \geq \Phi\left(\vec{\theta}^{(t)},q\right) - \Delta\left(\vec{\theta}^{(t)},q\right)$$

M step guarantees an increase in Φ

E step guarantees a decrease in Δ

$$\overbrace{\sum_x \tilde{p}(x)\log\sum_y p_{\vec{\theta}}(x,y)}^{\text{what MLE wants maximized: }\Lambda} = \overbrace{\sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(x,y)}^{\text{what the M step maximizes: }\Phi} - \overbrace{\sum_x \tilde{p}(x)\sum_y q(y|x)\log p_{\vec{\theta}}(y|x)}^{\Delta}$$

Convergence

- EM iterations will never decrease likelihood.
- Under some conditions, EM converges to a **saddle point**; generally it is assumed that EM will converge to a local maximum.
- Linear convergence (i.e., slow); depends on how much information is missing.

Expectation-Maximization

- E step:

$$\forall i, y, q(y|x_i) \leftarrow p_{\vec{\theta}^{(t)}}(y|x_i) = \frac{p_{\vec{\theta}^{(t)}}(x_i, y)}{\sum_{y'} p_{\vec{\theta}^{(t)}}(x_i, y')}$$

soft assignment or voting

- M step:

$$\vec{\theta}^{(t+1)} \leftarrow \arg\max_{\vec{\theta}} \sum_{x,y} \underbrace{\tilde{p}(x) q(y|x)}_{\text{"pretend" } \tilde{p}(x,y)} \log p_{\vec{\theta}}(x, y)$$

fully-observed data MLE

Toward Structure Models

- There are way too many values of Y to sum over!
- Two key points:
  - Never need to sum over Y by enumeration.
  - Never need q to be computed explicitly.

Consider the M Step

$$\vec{\theta}^{(t+1)} \leftarrow \arg\max_{\vec{\theta}} \sum_{x,y} \underbrace{\tilde{p}(x)q(y|x)}_{\text{"pretend" } \tilde{p}(x,y)} \log p_{\vec{\theta}}(x,y)$$

To maximize likelihood, what do we need?

- For multinomial-based models (HMMs, PCFGs, etc.), we need **counts**.
- For log-linear models in general, we need **counts**.

Simplifying the M Step (multinomials)

$$\vec{\theta}^{(t+1)} \leftarrow \arg\max_{\vec{\theta}} \sum_{x,y} \underbrace{\tilde{p}(x) q(y|x)}_{\text{"pretend" } \tilde{p}(x,y)} \log p_{\vec{\theta}}(x,y)$$

$$\sum_{x,y} \tilde{p}(x) q(y|x) \log p_{\vec{\theta}}(x,y) = \sum_{x,y} \tilde{p}(x) q(y|x) \log \prod_{e} p_e^{\text{count}(e;x,y)}$$

$$= \sum_{x,y} \tilde{p}(x) q(y|x) \sum_{e} \text{count}(e;x,y) \log p_e$$

$$= \sum_{e} \log(p_e) \sum_{x,y} \tilde{p}(x) q(y|x) \text{count}(e;x,y)$$

$$= \sum_{e} \log(p_e) \sum_{x} \tilde{p}(x) \sum_{y} q(y|x) \text{count}(e;x,y)$$

$$= \frac{1}{n} \sum_{e} \log(p_e) \sum_{i=1}^{n} \underbrace{\sum_{y} q(y|x_i) \text{count}(e;x_i,y)}_{\text{expected count of } e \text{ given } x}$$

Simplifying the M Step (log-linear models)

$$\vec{\theta}^{(t+1)} \leftarrow \arg\max_{\vec{\theta}} \sum_{x,y} \underbrace{\tilde{p}(x) q(y|x)}_{\text{"pretend" } \tilde{p}(x,y)} \log p_{\vec{\theta}}(x,y)$$

$$\sum_{x,y} \tilde{p}(x) q(y|x) \log p_{\vec{\theta}}(x,y) = \sum_{x,y} \tilde{p}(x) q(y|x) \left[\vec{\theta} \cdot \vec{f}(x,y) - \log \overbrace{\sum_{x',y'} \exp\left(\vec{\theta} \cdot \vec{f}(x',y')\right)}^{Z(\vec{\theta})} \right]$$

$$= \left(\vec{\theta} \cdot \sum_{x,y} \tilde{p}(x) q(y|x) \vec{f}(x,y) \right) - \log Z(\vec{\theta})$$

$$= \frac{1}{n} \left(\vec{\theta} \cdot \sum_{i=1}^{n} \sum_{y} q(y|x_i) \vec{f}(x_i,y) \right) - \boxed{\log Z(\vec{\theta})}$$

next time!

Sufficient Statisics

- A statistic is **sufficient** for a parameter when

$$p\left(\text{data}\middle|\vec{\theta}\right) = p\left(\text{data}\middle|S\left(\vec{\theta}\right)\right)$$

- The M step only requires sufficient statistics under q.
- For NLP models, this usually means **expected counts**.

HMM Forward and Backward Probabilities

$$\alpha(i,c) = p\left(s_{i+1}^{n} \mid C_i = c\right)$$

"backward" probability

$$\beta(i,c) = p\left(s_1^{i}, C_i = c\right)$$

"forward" probability

$$\alpha(i,c) \cdot \beta(i,c) = p\left(s_1^{n}, C_i = c\right)$$

$$\frac{\alpha(i,c) \cdot \beta(i,c)}{\beta(n+1, \mathrm{stop})} = p\left(C_i = c \mid s_1^{n}\right)$$

posterior probability that s_i is labeled with class c

$$\sum_{i=1}^{n} \frac{\alpha(i,c) \cdot \beta(i,c)}{\beta(n+1, \mathrm{stop})} = \mathbf{E}\left[\left|\{i : C_i = c\}\right| \mid s_1^{n}\right]$$

expected count of class c

CKY Inside and Outside Probabilities

$$\alpha(i,j,N) = p\left(s_1^{i-1}\ N_{ij}\ s_{j+1}^{n} \mid S_{1n}\right)$$

“outside” probability

$$\beta(i,j,N) = p\left(s_i^j \middle| N_{ij}\right)$$

“inside” probability

$$\alpha(i,j,N)\cdot\beta(i,j,N) = p\left(s_1^n, N_{ij}\right)$$

$$\frac{\alpha(i,j,N)\cdot\beta(i,j,N)}{\beta(1,n,S)} = p\left(N_{ij} \mid s_1^n\right)$$

$$\frac{\alpha(i,k,N)\cdot\beta(i,j,N')\cdot\beta(j+1,k,N'')\cdot p(N \to N'N'')}{\beta(1,n,S)} = p\left(N_{ik} \to N'_{ij}N''_{(j+1)k} \mid s_1^n\right)$$

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j+1}^{n} p\left(N_{ik} \to N'_{ij}N''_{(j+1)k} \middle| s_1^n\right) = \mathbf{E}\left[\left|\left\{i,j,k : N_{ik} \to N'_{ij}N''_{(j+1)k}\right\}\right| \mid s_1^n\right]$$

expected count of rule

In General

- Don't compute q directly in the E step.
  - Just get the sufficient statistics.
  - Inside and Outside algorithms can help for some models!
  - Other alternatives (less common in NLP):
    - Sample from q(y | x) to get sufficient statistics.
    - Use a variational approximation to q(y | x).
- This should remind you of the **factored dual** in structured maximum margin training!
  - Use statistics on structure pieces instead of whole structures.

Pereira and Schabes (1992)

- Suppose you have a **partially bracketed** corpus.
- Want to constrain re-estimation to respect the known bracketings. Everything else is hidden.

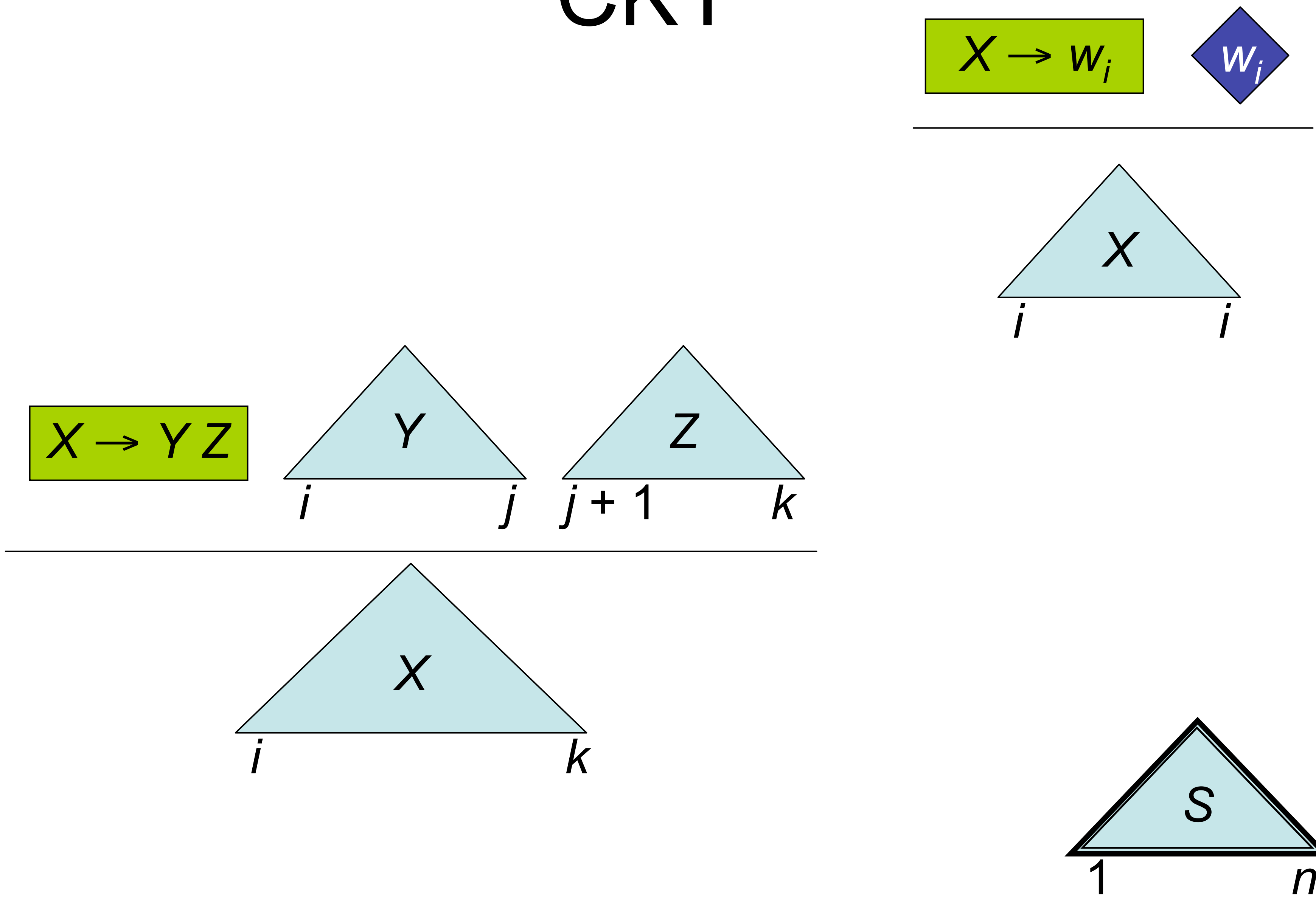
(Democrats took control of both houses) no information

((Democrats) took control of (both houses)) base NPs

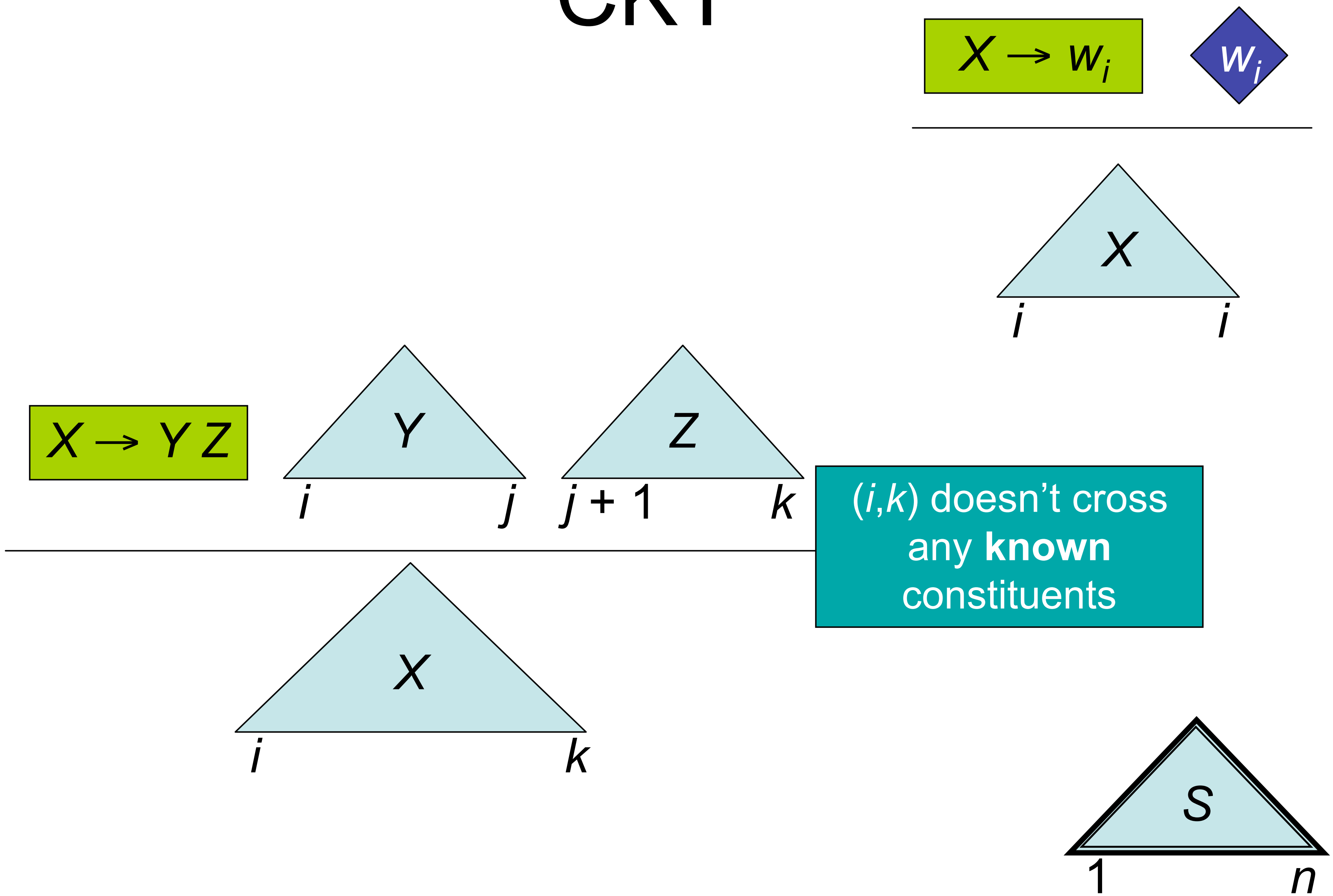
((Democrats) took (control of (both houses))) all NPs

(Democrats (took control of both houses)) VP

CKY



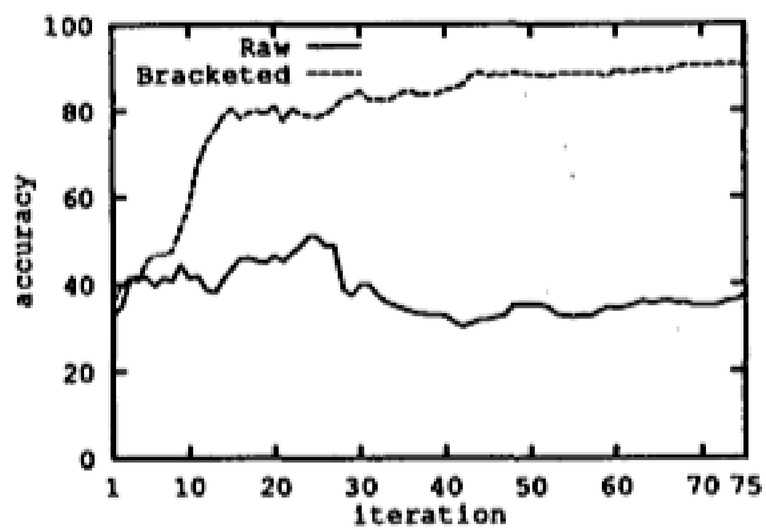
CKY



X → w_i
w_i
X
i
i
X → Y Z
Y
Z
i
j
j + 1
k
(i,k) doesn't cross any **known** constituents
X
i
k
S
1
n

Pereira and Schabes (1992)

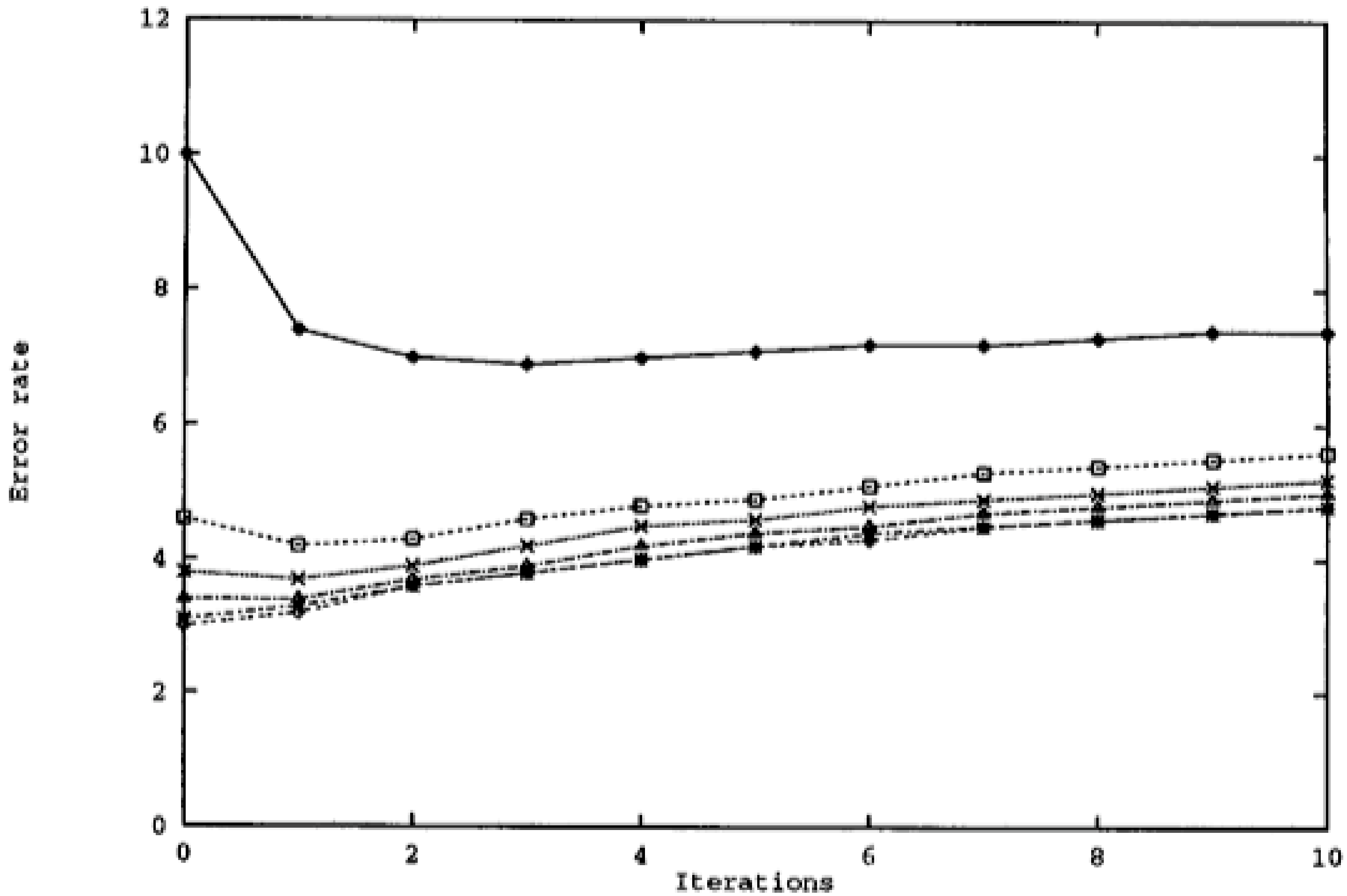
- When compared with unconstrained EM:
  - Better fit to the data (cross-entropy)
  - Better accuracy
  - Faster convergence
- Later result (Hwa, 1999): high-level structure helps more than low-level structure.



Merialdo (1994)

- Suppose you have some tagged text and some untagged text.
- You could train a tagger on the tagged text.
- Can you use the untagged text to help?
- Merialdo:
  - Vary the amount of tagged text
  - Use the tagged text to **initialize**
  - Run EM.

Merialdo (1994)



Merialdo (1994)

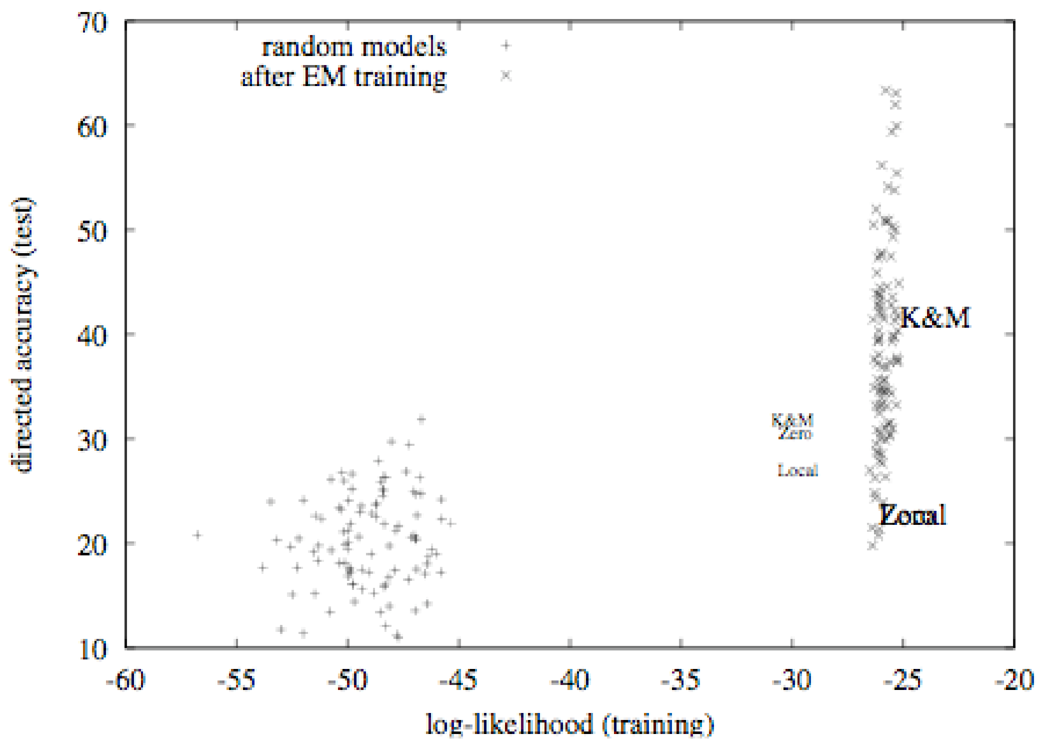
- Similar results by Elworthy (1994).
- Another way to combine the data:

$$\max_{\vec{\theta}} \sum_{i \in L} \log p(x_i, y_i) + \sum_{i \in U} \log p(x_i)$$

- Equivalently, augment E step counts with observed counts before each M step.
- (Same effect, anecdotally.)

The Two Main Problems With EM

- Marginal likelihood ≠ Accuracy
- Local optima



Plot from Smith (2006);
similar results in Charniak (1993)

Variants

- MAP instead of MLE: add a prior (smoothing)
- For speed:
  - Viterbi approximation (mode instead of expectation in E step)
  - Incremental EM (M step after every example)
- To improve search quality:
  - Deterministic annealing: gradually relax an entropy constraint on q (affects the E step only)
  - Random restarts
  - Random reweighting of examples
- Really good initialization
- Alternative objective (next week)

Klein & Manning (2002)

- A highly deficient grammatical model that predicts POS tag sequences.
- Constituent-context model (CCM).
- Best published unsupervised parsing results on WSJ-10 (in 2002)
- Trained using EM … with an interesting initializer: $\Pr_{split}$

Constituent-Context Model

$t = (t_1 \dots t_n)$ is the tag sequence.

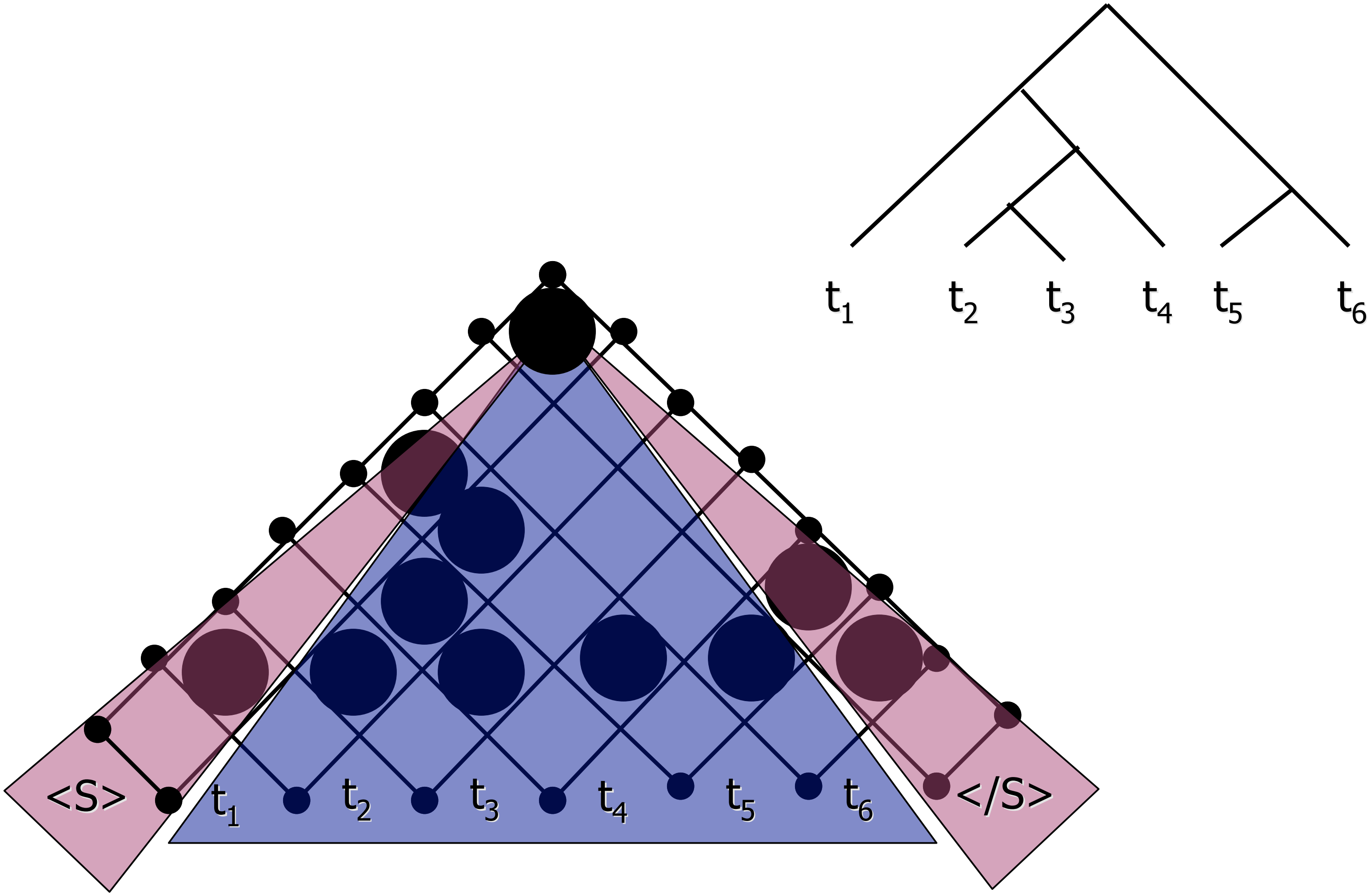
Let $C_{i,j}$ be a r.v., equal to 1 if $t_i \dots t_j$ is a constituent, 0 if not.

C is the set of all $C_{i,j}$ (an upper matrix). The valid values of C are the ones that are binary trees.

$$\Pr(t, C) = \Pr(C) \cdot \Pr(t \mid C)$$

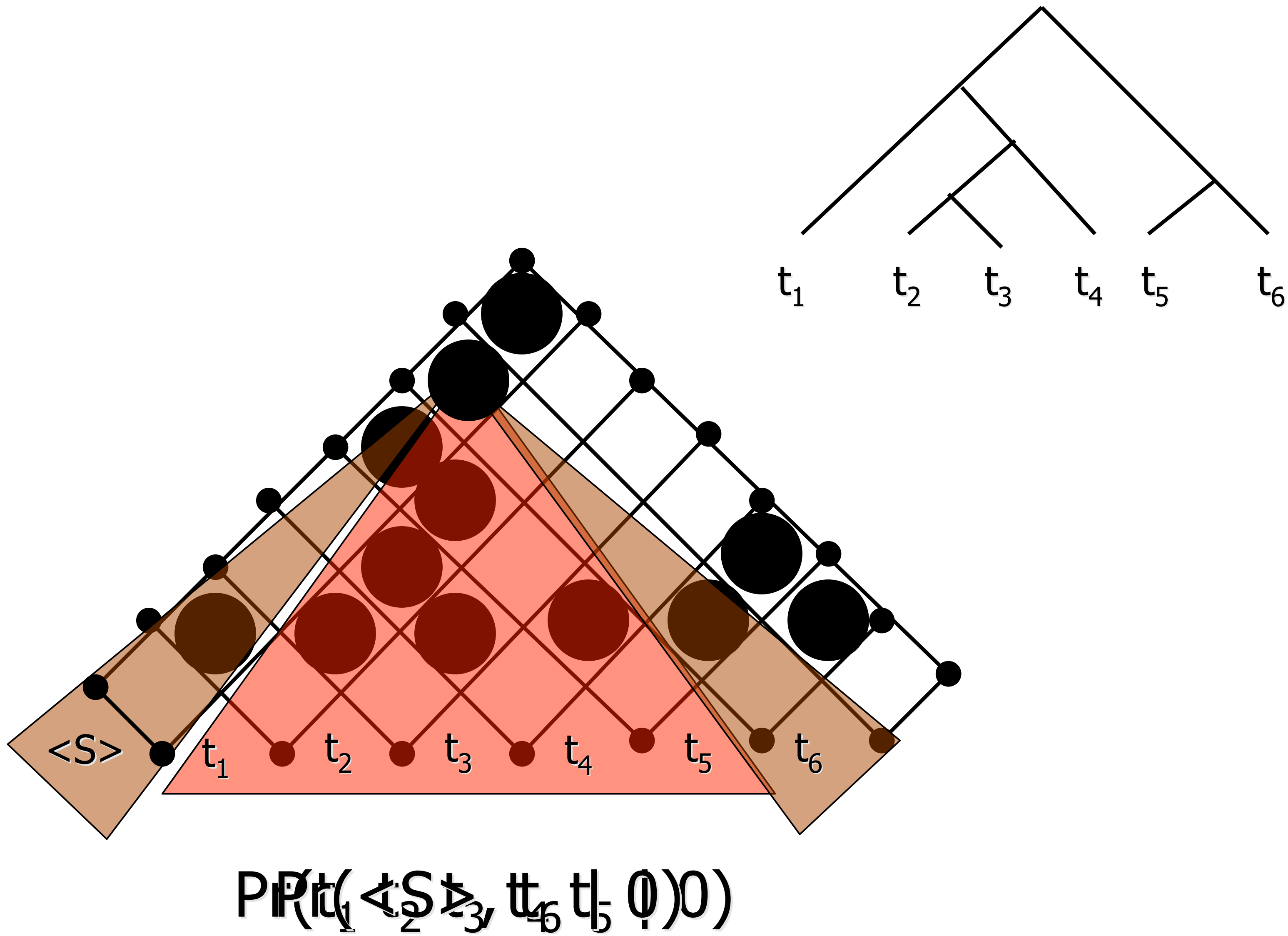
$$= \Pr(C) \cdot \prod_{i,j:\ 1 \le i \le j \le n} \Pr(t_i \dots t_j \mid C_{i,j}) \cdot \Pr(t_{i-1}, t_{j+1} \mid C_{i,j})$$

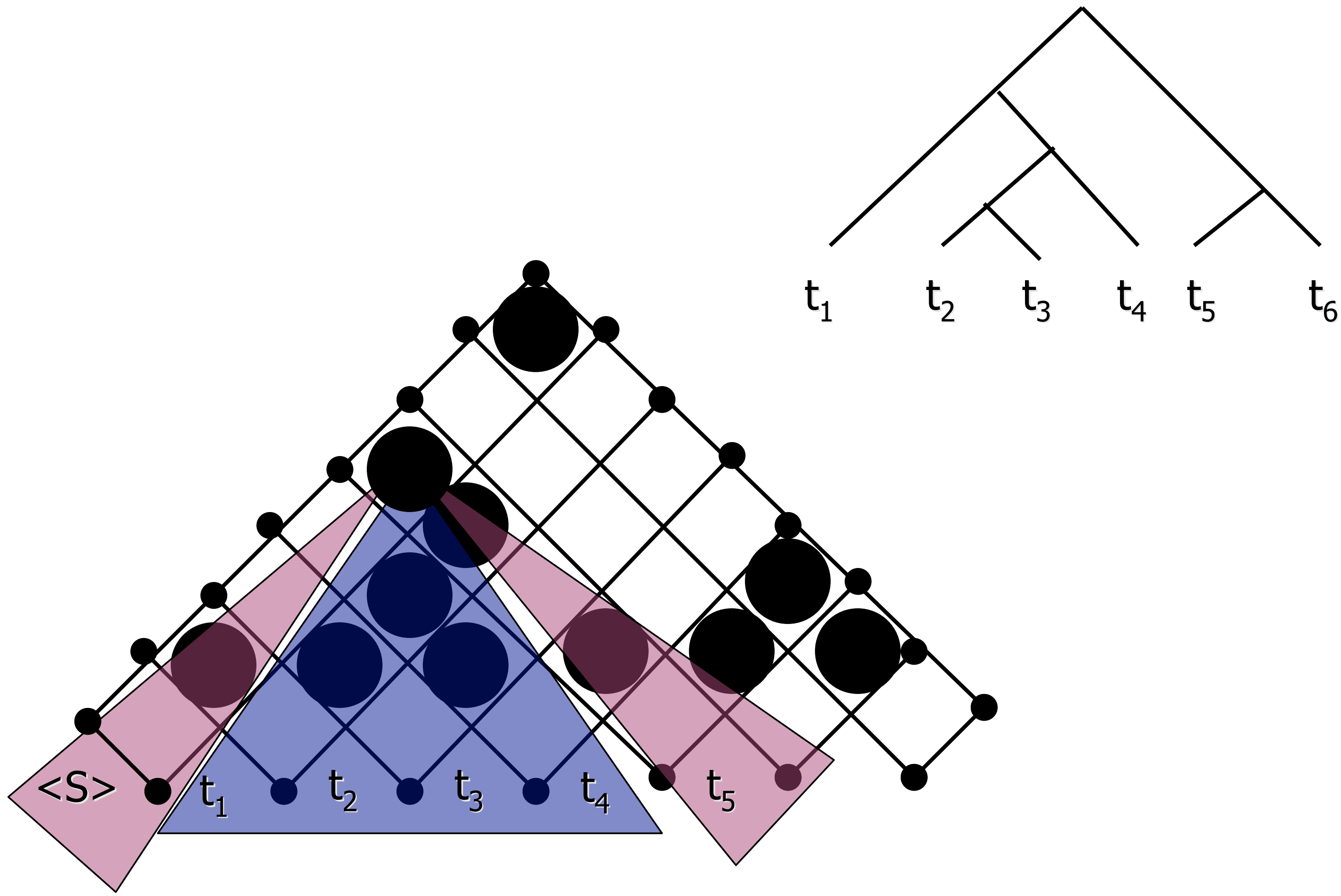
$$= \underbrace{(1/\tau(n))}_{\text{uniform distribution over trees}} \cdot \prod_{i,j:\ 1 \le i \le j \le n} \underbrace{\Pr(t_i \dots t_j \mid C_{i,j})}_{\text{“constituent”}} \cdot \underbrace{\Pr(t_{i-1}, t_{j+1} \mid C_{i,j})}_{\text{“context”}}$$



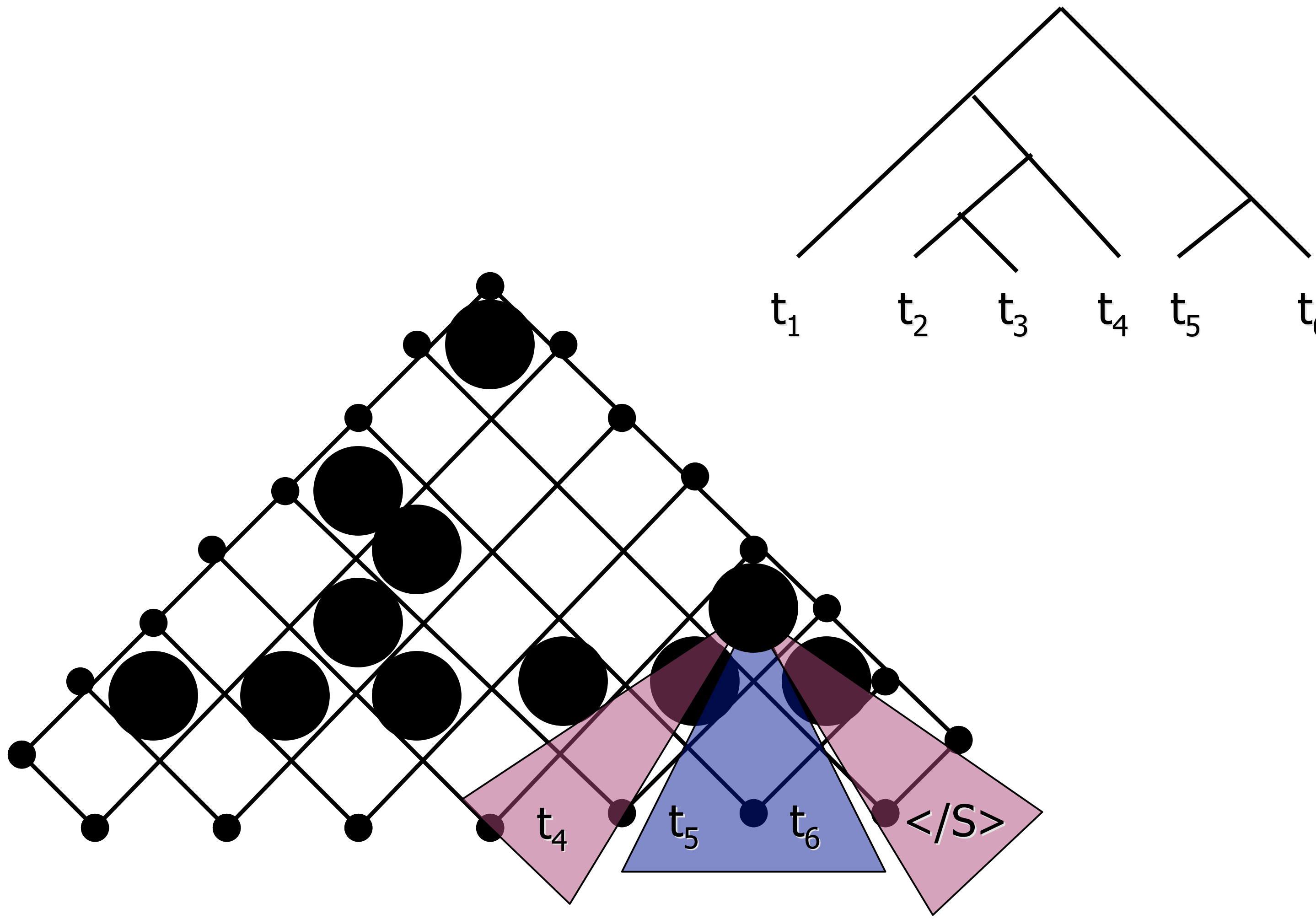
$\Pr(t_1\ t_2\ t_3\ t_4\ t_5\ t_6 \mid 1)$

$\Pr(\langle S \rangle, \langle /S \rangle \mid 1)$


<S>
t1
t2
t3
t4
t5
t6
t1
t2
t3
t4
t5
t6
Pr(<S>, t6 | 0)
Pr(t1 t2 t3 t4 t5 | 0)



$\Pr(\langle S \rangle, t_5 \mid 1)$



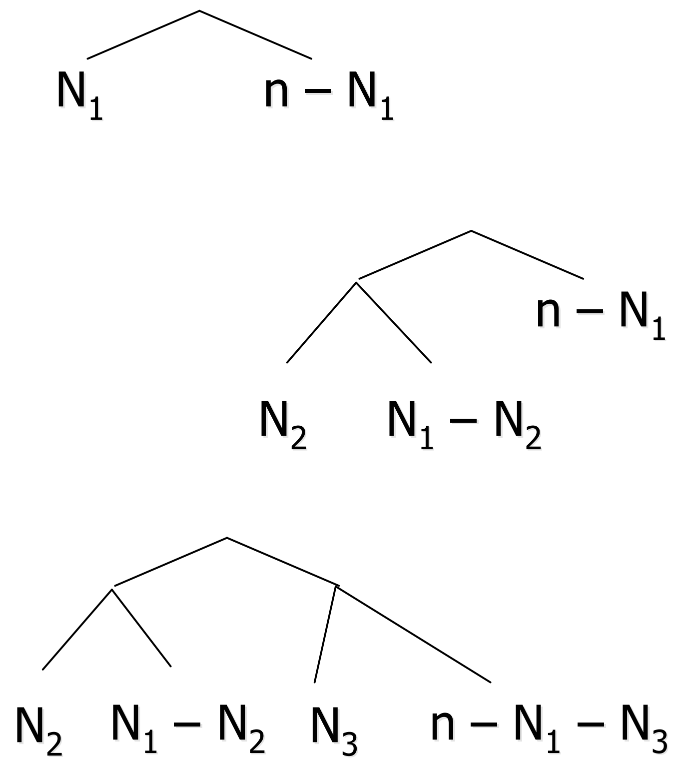
Pr(t5 t6 | 1)

Pr(t4, </S> | 1)

About CCM

- Only four multinomials to estimate.
- Need two features for every substring of tags! (This is why they used WSJ-10.)
- Highly deficient!

Pr_{split}



- Given length of sentence, n.
- Choose N_1 u.a.r. from {1, 2, …, n}
- Choose N_2 u.a.r. from {1, 2, …, N_1 – 1}
- Choose N_3 u.a.r. from {1, 2, …, n – N_1 – 1}
- Continue until no further splits can be made.

This model gives a closed form for the expectations; K&M use it to generate initial expectations, then start with an M step.

*See also Cover & Thomas problem 4.3 (p. 72).

Importance of Pr_{split}

Alg'm	Model	UP (%)	UR (%)	Ave. CB	Perf'ct (%)	0CB	≤2CB	It.'s	Cross-E (bits)
Right-branching trees		*46.62*	*62.54*	*1.78*	*13.54*	*28.13*	*71.42*	--	--
EM (Pr_{split})	CCM	58.24	78.14	**0.98**	16.86	50.39	**87.48**	123	725.17
EM (unif.)	CCM	45.62	61.20	1.69	11.28	26.53	71.79	145	**724.96**
Upper bound (binary trees)		*74.54*	*100.00*	*0.00*	*25.93*	*100.00*	*100.00*	--	--

Next Time

- Contrastive estimation as an alternative to EM
  - Application to unsupervised tagging, parsing