Lecture 3: Inside-Outside

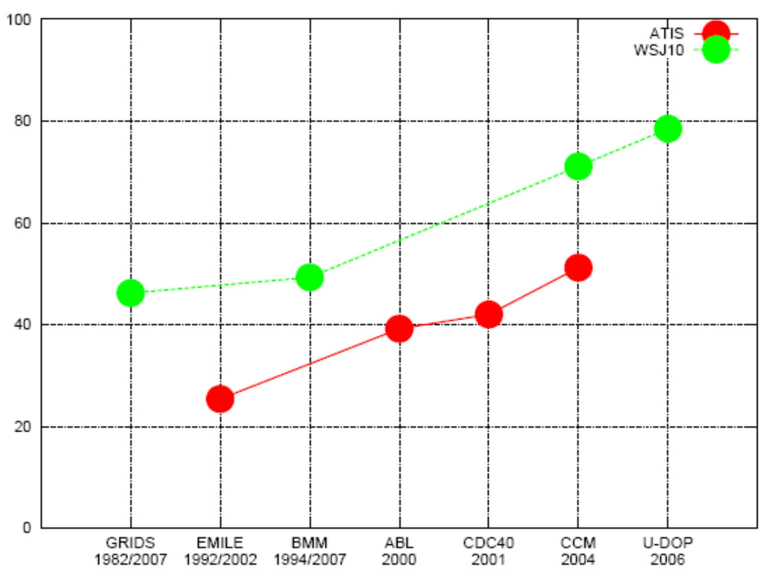
Jelle Zuidema
ILLC, Universiteit van Amsterdam

Unsupervised Language Learning, 2013

The Challenge (lecture 1)

Developing algorithms for learning about the syntax (semantics, pragmatics, phonology) of natural language from unlabeled data.

- Classic, hard problem in artificial intelligence
- Many unsuccessful attempts to develop heuristic algorithms for grammar induction.


100
80
60
40
20
0
ATIS
WSJ10
GRIDS 1982/2007
EMILE 1992/2002
BMM 1994/2007
ABL 2000
CDC40 2001
CCM 2004
U-DOP 2006

Statistical Inference (lecture 1, assignment 1, readings)

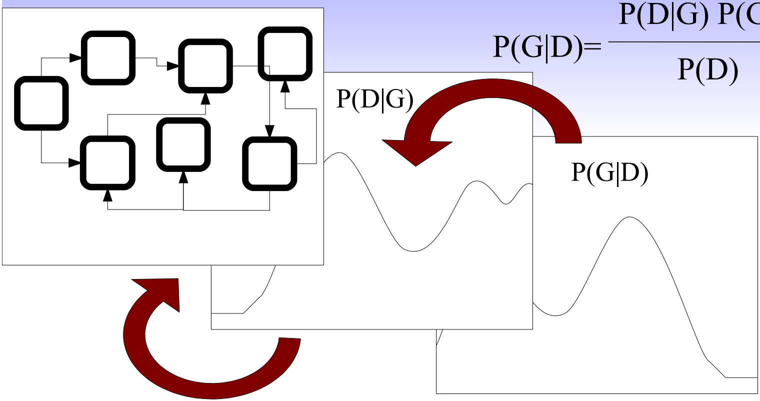
- Statistical Models can deal with the noise and uncertainties (hidden information, “latent variables”) inherent in real world data;
- Statistical Inference offers a flexible toolbox of techniques and concepts:
  - Probabilities: likelihood, prior, posterior, data prior;
  - Criteria: maximum likelihood, MAP, minimum risk;
  - Optimization techniques: grid search, stochastic hillclimbing, EM, MCMC;
- Conceptual separation of objective function and optimization technique.

Generative models

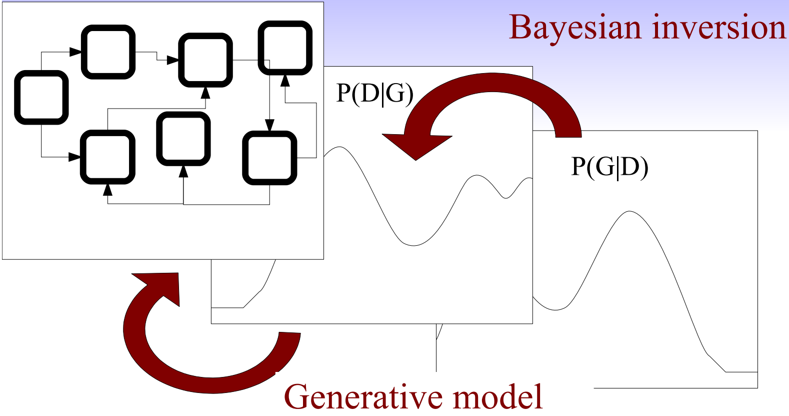
- "Generative story": we define probability distributions by decribing the mechanism by which the data could have been generated.
- "Graphical models": often graphs are used to define the statistical dependencies in the generative story.

Statistical inference

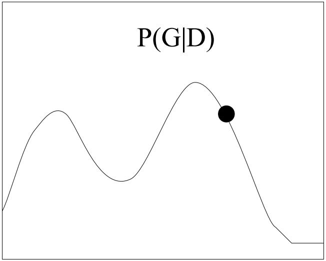
$$P(G|D) = \frac{P(D|G)\,P(G)}{P(D)}$$



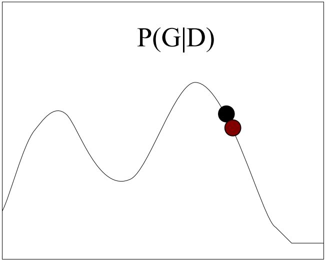
Statistical inference



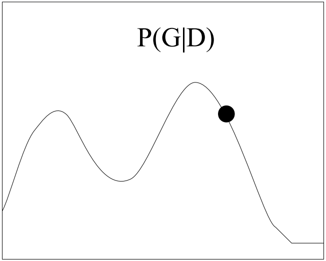
Stochastic hillclimbing



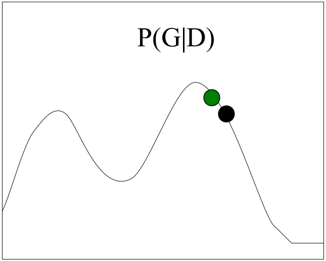
Stochastic hillclimbing



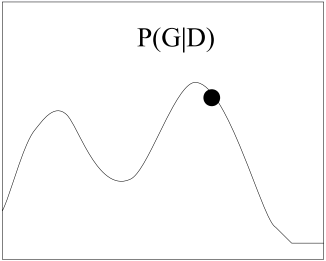
Stochastic hillclimbing



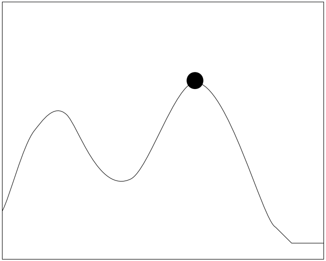
Stochastic hillclimbing



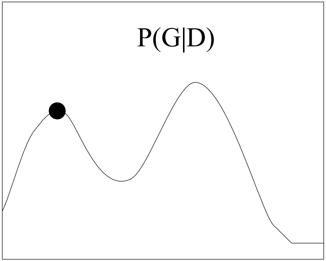
Stochastic hillclimbing



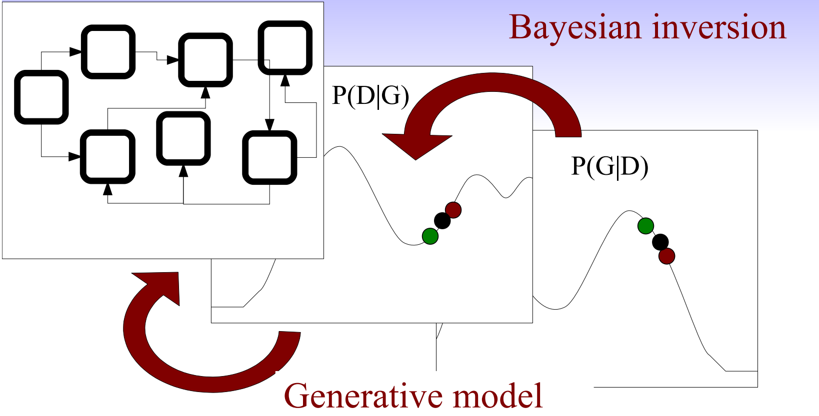
Stochastic hillclimbing



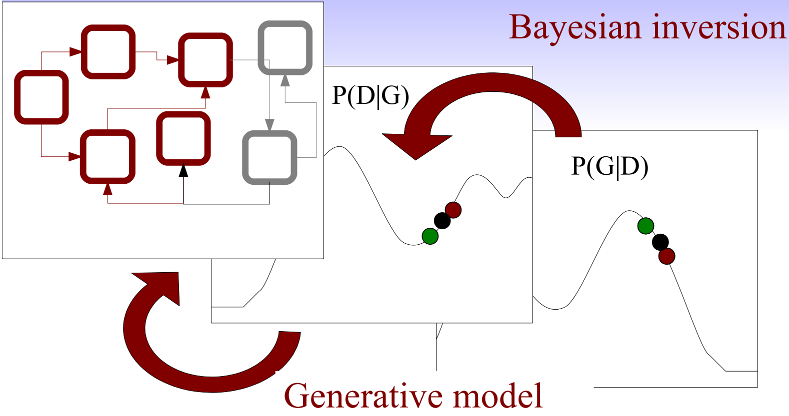
Local optimum



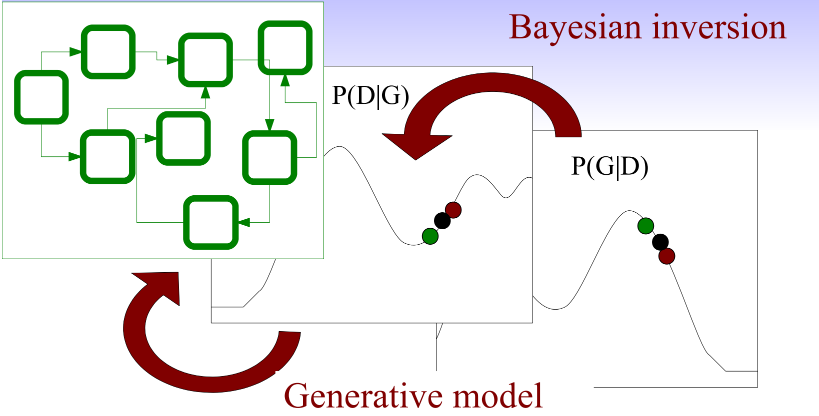
Statistical inference



Statistical inference



Statistical inference



Probabilistic Grammars

- For natural language grammar, generative models are instantiatied with probabilistic grammars
- The extended Chomsky hierarchy for symbolic grammars, is mirrored by a hierarchy of probabilistic grammars.
- Classes on the hierarchy are proper subsets of eachother;
  - Corrolary: everything lower in the hierarchy can in principled be modelled by formalisms for probabilistic grammars higher in the hierarchy;
  - E.g., PCFGs can model ngrams and HMMs (and probabilistic bilexical dependency grammars).

Probabilistic Context Free Grammars

- Add probabilities to the rules of a context-free grammar;
- The PCFG now defines a probability distribution (G_S) over trees (with S as root, and words as leaves);
- It also defines probability distributions over sentences;
- It also defines probability distributions (G_A) over trees rooted in any other nonterminal A;
- G_S can be defined using the probabilities of all rules with S as left-hand side and $G_A \ldots G_Z$.

PCFGs as recursive mixtures

The distributions over strings induced by a PCFG in *Chomsky-normal form* (i.e., all productions are of the form $A \to B\,C$ or $A \to x$, where $A, B, C \in N$ and $x \in T$) is G_S where:

$$G_A = \sum_{A \to B\,C \in R_A} pA \to B\,C G_B \bullet G_C + \sum_{A \to w \in R_A} pA \to x \delta_x$$

$$\begin{aligned}(P \bullet Q)(z) &= \sum_{xy=z} P(x)Q(y) \\ \delta_x(w) &= 1 \text{ if } w = x \text{ and } 0 \text{ otherwise}\end{aligned}$$

In fact, $G_A(w) = \mathrm{P}(A \Rightarrow^{\star} w|\theta)$, the sum of the probability of all trees with root node A and yield w

Things we want to compute with PCFGs

Given a PCFG G and a string $w \in T^\star$,

- (parsing): the most likely tree for w,

$$\mathrm{argmax}_{\psi \in \Psi_G(w)} \mathrm{P}_G(\psi)$$

- (language modeling): the probability of w,

$$\mathrm{P}_G(w) \; = \sum_{\psi \in \Psi_G(w)} \mathrm{P}_G(\psi)$$

Learning rule probabilities from data:

- (maximum likelihood estimation from visible data): given a corpus of trees $\boldsymbol{d} = (\psi_1, \ldots, \psi_n)$, which rule probabilities p makes $\boldsymbol{d}$ as likely as possible?
- (maximum likelihood estimation from hidden data): given a corpus of strings $\boldsymbol{w} = (w_1, \ldots, w_n)$, which rule probabilities p makes $\boldsymbol{w}$ as likely as possible?

Parsing and language modeling

The probability $\mathrm{P}_G(\psi)$ of a tree $\psi \in \Psi_G(w)$ is:

$$\mathrm{P}_G(\psi) = \prod_{r \in R} p(r)^{f_r(\psi)}$$

Suppose the set of parse trees $\Psi_G(w)$ is finite, and we can enumerate it.

Naive parsing/language modeling algorithms for PCFG G and string $w \in T^\star$:

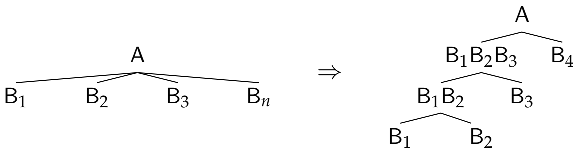
1. Enumerate the set of parse trees $\Psi_G(w)$
2. Compute the probability of each $\psi \in \Psi_G(w)$
3. Argmax/sum as appropriate

Chomsky normal form

A CFG is in *Chomsky Normal Form* (CNF) iff all productions are of the form $A \rightarrow B\ C$ or $A \rightarrow x$, where $A, B, C \in N$ and $x \in T$.

PCFGs *without epsilon productions* $A \rightarrow \epsilon$ can always be put into CNF.

Key step: *binarize* productions with more than two children by introducing new nonterminals



Substrings and string positions

Let $w = w_1 w_2 \dots w_n$ be a string of length n

A *string position* for w is an integer $i \in 0, \dots, n$ (informally, it identifies the position between words w_{i-1} and w_i)

$$\begin{array}{ccccccccc} \bullet & \text{the} & \bullet & \text{dog} & \bullet & \text{chases} & \bullet & \text{cats} & \bullet \\ 0 & & 1 & & 2 & & 3 & & 4 \end{array}$$

A *substring* of w can be specified by beginning and ending string positions

$w_{i,j}$ is the substring starting at word $i+1$ and ending at word j.

$$\begin{aligned} w_{0,4} &= \text{the dog chases cats} \\ w_{1,2} &= \text{dog} \\ w_{2,4} &= \text{chases cats} \end{aligned}$$

Language modeling using dynamic programming

- *Goal:* To compute $\mathrm{P}_G(w) = \sum_{\psi \in \Psi_G(w)} \mathrm{P}_G(\psi) = \mathrm{P}_G(S \Rightarrow^* w)$
- *Data structure:* A table called a *chart* recording $\mathrm{P}_G(A \Rightarrow^* w_{i,k})$ for all $A \in N$ and $0 \leq i < k \leq |w|$
- *Base case:* For all $i = 1, \ldots, n$ and $A \to w_i$, compute:

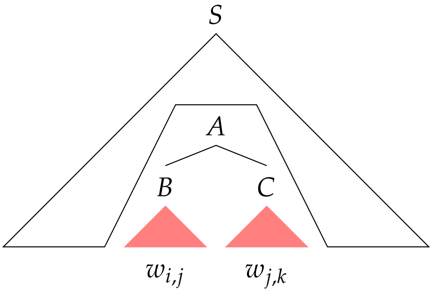
$$\mathrm{P}_G(A \Rightarrow^* w_{i-1,i}) \;=\; p(A \to w_i)$$

- *Recursion:* For all $k - i = 2, \ldots, n$ and $A \in N$, compute:

$$\mathrm{P}_G(A \Rightarrow^* w_{i,k}) \\ = \sum_{j=i+1}^{k-1} \sum_{A \to B\,C \in R(A)} p(A \to B\,C)\mathrm{P}_G(B \Rightarrow^* w_{i,j})\mathrm{P}_G(C \Rightarrow^* w_{j,k})$$

Dynamic programming recursion

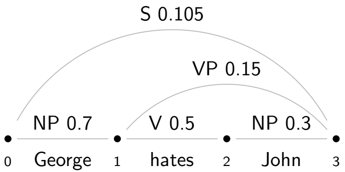
$$\begin{aligned}
&\mathrm{P}_G(A \Rightarrow^* w_{i,k}) \\
&\quad = \sum_{j=i+1}^{k-1} \sum_{A \to B\,C \in R(A)} p(A \to B\,C)\mathrm{P}_G(B \Rightarrow^* w_{i,j})\mathrm{P}_G(C \Rightarrow^* w_{j,k})
\end{aligned}$$



$\mathrm{P}_G(A \Rightarrow^* w_{i,k})$ is called the *inside probability* of A spanning $w_{i,k}$.

Example PCFG string probability calculation

$$
\begin{aligned}
w &= \textsf{George hates John} \\
R &= \left\{ \begin{array}{llll}
1.0 & \textsf{S} \rightarrow \textsf{NP VP} & 1.0 & \textsf{VP} \rightarrow \textsf{V NP} \\
0.7 & \textsf{NP} \rightarrow \textsf{George} & 0.3 & \textsf{NP} \rightarrow \textsf{John} \\
0.5 & \textsf{V} \rightarrow \textsf{likes} & 0.5 & \textsf{V} \rightarrow \textsf{hates}
\end{array} \right\}
\end{aligned}
$$



Right string position

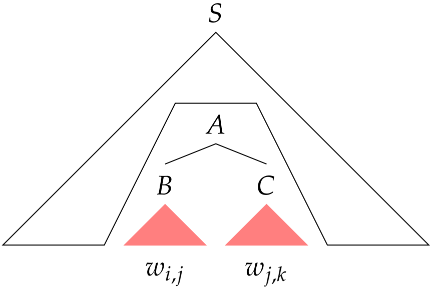
Left string position	1	2	3
0	NP 0.7		S 0.105
1		V 0.5	VP 0.15
2			NP 0.3

Intermediate Summary

- PCFGs define probability distributions over trees, subtrees and strings. These distributions can be viewed as *recursive mixtures*.
- The probability of a (sub)string can be calculated the naive way by summing the probabilities of all the trees that contain it;
- We can make use of the recursive nature of PCFG distributions to calculate string probabilities more efficiently: the inside algorithm.

Computational complexity of PCFG parsing

$$\mathrm{P}_G(A \Rightarrow^* w_{i,k}) = \sum_{j=i+1}^{k-1} \sum_{A \to B\,C \in R(A)} p(A \to B\,C)\mathrm{P}_G(B \Rightarrow^* w_{i,j})\mathrm{P}_G(C \Rightarrow^* w_{j,k})$$



For each production $r \in R$ and each i, k, we must sum over all intermediate positions $j \Rightarrow O(n^3|R|)$ time

Estimating (learning) PCFGs from data

Estimating productions and production probabilities from *visible data* (corpus of parse trees) is straight-forward:

- the productions are identified by the local trees in the data
- *Maximum likelihood principle:* select production probabilities in order to make corpus as likely as possible
- Bayesian estimators often produce more useful estimates

Estimating production probabilities from *hidden data* (corpus of terminal strings) is much more difficult:

- The *Expectation-Maximization* (EM) algorithm finds probabilities that *locally maximize* likelihood of corpus
- The *Inside-Outside* algorithm runs in time polynomial in length of corpus
- Bayesian estimators have recently been developed

Estimating the productions from hidden data is an open problem.

Estimating PCFGs from visible data

Data: A *treebank* of parse trees $\Psi = \psi_1, \ldots, \psi_n$.

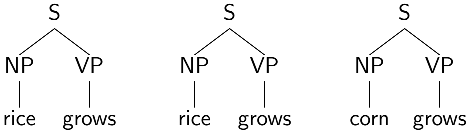
$$L(p) \;=\; \prod_{i=1}^{n} \mathrm{P}_G(\psi_i) \;=\; \prod_{A \to \alpha \in R} p(A \to \alpha)^{f_{A\to\alpha}(\Psi)}$$

Introduce $|N|$ Lagrange multipliers $c_B, B \in N$ for the constraints $\sum_{B \to \beta \in R(B)} p(B \to \beta) = 1$:

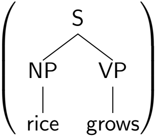
$$\frac{\partial \left(L(p) - \sum_{B \in N} c_B \left(\sum_{B \to \beta \in R(B)} p(B \to \beta) - 1 \right) \right)}{\partial p(A \to \alpha)} \;=\; \frac{L(p) f_r(\Psi)}{p(A \to \alpha)} - c_A$$

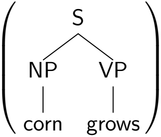
Setting this to 0, $\quad p(A \to \alpha) \;=\; \dfrac{f_{A\to\alpha}(\Psi)}{\sum_{A \to \alpha' \in R(A)} f_{A\to\alpha'}(\Psi)}$

Visible PCFG estimation example

$\Psi =$ 

Rule	**Count**	**Rel Freq**
S $\rightarrow$ NP VP	3	1
NP $\rightarrow$ rice	2	2/3
NP $\rightarrow$ corn	1	1/3
VP $\rightarrow$ grows	3	1

P  $= 2/3$

P  $= 1/3$

Estimating production probabilities from hidden data

Data: A corpus of sentences $\boldsymbol{w} = w_1, \ldots, w_n$.

$$L(\boldsymbol{w}) \;=\; \prod_{i=1}^{n} \mathrm{P}_G(w_i). \qquad \mathrm{P}_G(w) \;=\; \sum_{\psi \in \Psi_G(w)} \mathrm{P}_G(\psi).$$

$$\frac{\partial L(\boldsymbol{w})}{\partial p(A \rightarrow \alpha)} \;=\; \frac{L(\boldsymbol{w}) \sum_{i=1}^{n} \mathrm{E}_G(f_{A\rightarrow\alpha}|w_i)}{p(A \rightarrow \alpha)}$$

Setting this equal to the Lagrange multiplier c_A and imposing the constraint $\sum_{B\rightarrow\beta\in R(B)} p(B \rightarrow \beta) = 1$:

$$p(A \rightarrow \alpha) \;=\; \frac{\sum_{i=1}^{n} \mathrm{E}_G(f_{A\rightarrow\alpha}|w_i)}{\sum_{A\rightarrow\alpha'\in R(A)} \sum_{i=1}^{n} \mathrm{E}_G(f_{A\rightarrow\alpha'}|w_i)}$$

This is an iteration of the *expectation maximization* algorithm!

The EM algorithm for PCFGs

Input: a corpus of strings $\boldsymbol{w} = w_1, \ldots, w_n$

Guess initial production probabilities $p^{(0)}$

For $t = 1, 2, \ldots$ do:

1. Calculate *expected frequency* $\sum_{i=1}^{n} \mathrm{E}_{p^{(t-1)}}(f_{A\to\alpha}|w_i)$ of each production:

$$\mathrm{E}_p(f_{A\to\alpha}|w) = \sum_{\psi\in\Psi_G(w)} f_{A\to\alpha}(\psi)\mathrm{P}_p(\psi)$$

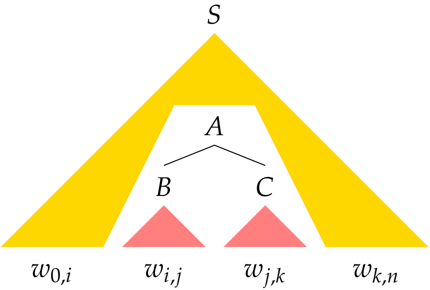
2. Set $p^{(t)}$ to the *relative expected frequency* of each production

$$p^{(t)}(A \to \alpha) = \frac{\sum_{i=1}^{n} \mathrm{E}_{p^{(t-1)}}(f_{A\to\alpha}|w_i)}{\sum_{A\to\alpha'} \sum_{i=1}^{n} \mathrm{E}_{p^{(t-1)}}(f_{A\to\alpha'}|w_i)}$$

It is as if $p^{(t)}$ were estimated from a visible corpus Ψ_G in which each tree ψ occurs $\sum_{i=1}^{n} \mathrm{P}_{p^{(t-1)}}(\psi|w_i)$ times.

Dynamic programming for $E_p(f_{A\to B\,C}|w)$

$$E_p(f_{A\to B\,C}|w) = \frac{\displaystyle\sum_{0\le i<j<k\le n} \mathrm{P}(S \Rightarrow^* w_{1,i}\,A\,w_{k,n})\,p(A \to B\,C)\,\mathrm{P}(B \Rightarrow^* w_{i,j})\,\mathrm{P}(C \Rightarrow^* w_{j,k})}{\mathrm{P}_G(w)}$$



Calculating "outside probabilities"

Construct a table of "outside probabilities"
$\mathrm{P}_G(S \Rightarrow^* w_{0,i}\, A\, w_{k,n})$ for all $0 \leq i < k \leq n$ and $A \in N$

Recursion from *larger to smaller* substrings in w.

Base case: $\mathrm{P}(S \Rightarrow^* w_{0,0}\, S\, w_{n,n}) = 1$

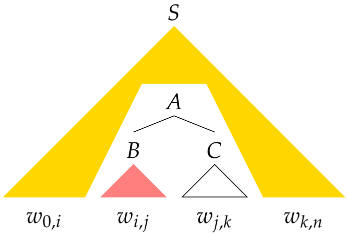
Recursion: $\mathrm{P}(S \Rightarrow^* w_{0,j}\, C\, w_{k,n}) =$

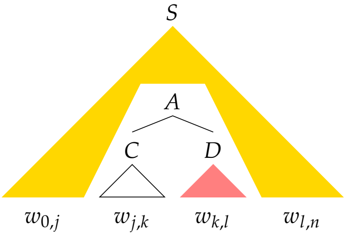
$$\sum_{i=0}^{j-1} \sum_{\substack{A,B \in N \\ A \to B\,C \in R}} \mathrm{P}(S \Rightarrow^* w_{0,i}\, A\, w_{k,n}) p(A \to B\, C) \mathrm{P}(B \Rightarrow^* w_{i,j})$$

$$+ \sum_{l=k+1}^{n} \sum_{\substack{A,D \in N \\ A \to C\,D \in R}} \mathrm{P}(S \Rightarrow^* w_{0,j}\, A\, w_{l,n}) p(A \to C\, D) \mathrm{P}(D \Rightarrow^* w_{k,l})$$

Recursion in $\mathrm{P}_G(S \Rightarrow^* w_{0,i}\, A\, w_{k,n})$

$$\mathrm{P}(S \Rightarrow^* w_{0,j}\, C\, w_{k,n}) = \sum_{i=0}^{j-1} \sum_{\substack{A,B \in N \\ A \to B\,C \in R}} \mathrm{P}(S \Rightarrow^* w_{0,i}\, A\, w_{k,n})\, p(A \to B\,C)\, \mathrm{P}(B \Rightarrow^* w_{i,j}) + \sum_{l=k+1}^{n} \sum_{\substack{A,D \in N \\ A \to C\,D \in R}} \mathrm{P}(S \Rightarrow^* w_{0,j}\, A\, w_{l,n})\, p(A \to C\,D)\, \mathrm{P}(D \Rightarrow^* w_{k,l})$$





Example: The EM algorithm with a toy PCFG

Initial rule probs

rule	**prob**
...	...
VP $\rightarrow$ V	0.2
VP $\rightarrow$ V NP	0.2
VP $\rightarrow$ NP V	0.2
VP $\rightarrow$ V NP NP	0.2
VP $\rightarrow$ NP NP V	0.2
...	...
Det $\rightarrow$ the	0.1
N $\rightarrow$ the	0.1
V $\rightarrow$ the	0.1

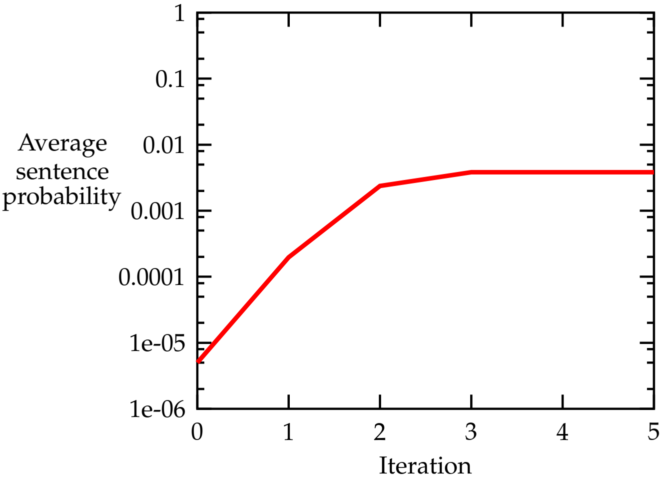
"English" input

the dog bites
the dog bites a man
a man gives the dog a bone
...

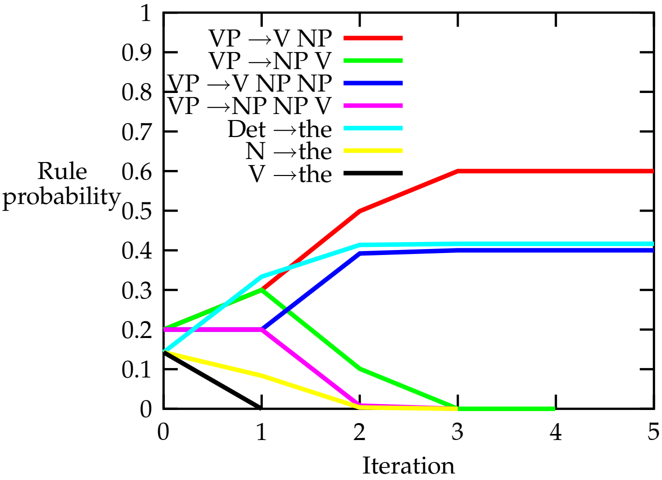
"pseudo-Japanese" input

the dog bites
the dog a man bites
a man the dog a bone gives
...

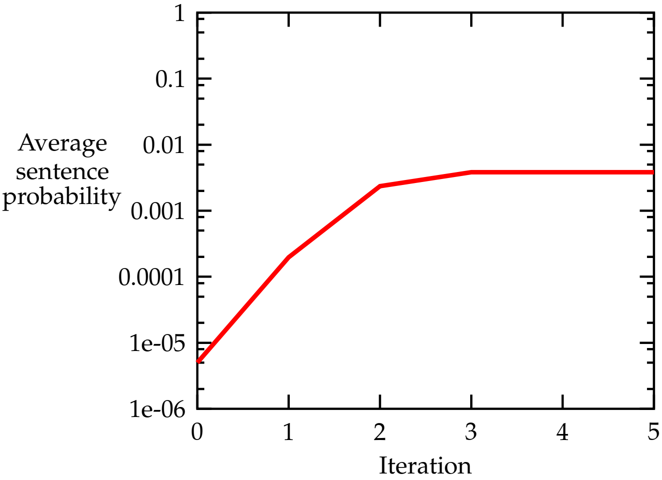
Probability of "English"



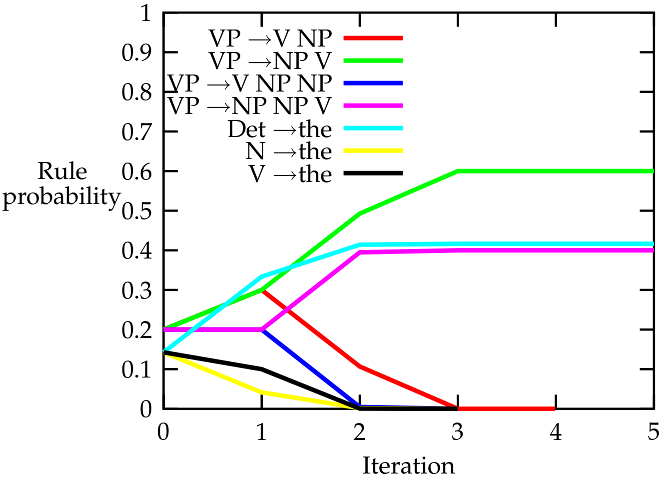
Rule probabilities from "English"



Probability of “Japanese”



Rule probabilities from “Japanese”



Learning in statistical paradigm

- The likelihood is a differentiable function of rule probabilities
  $\Rightarrow$ learning can involve small, incremental updates
- Learning structure (rules) is hard, but . . .
- Parameter estimation can approximate rule learning
  - start with “superset” grammar
  - estimate rule probabilities
  - discard low probability rules
- Non-parametric Bayesian estimators combine parameter and rule estimation