Stochastic Context-Free-Grammars

Challenges in Computational Biology

Genome Assembly

Regulatory motif discovery

Gene Finding

DNA

Sequence alignment

Comparative Genomics

Evolutionary Theory

TCATGCTAT
TCGTGATAA
TGAGGATAT
TTATCATAT
TTATGATTT

Database lookup

RNA folding

Gene expression analysis

RNA transcript

Cluster discovery

Gibbs sampling

Protein network analysis

Regulatory network inference

Emerging network properties

The world before DNA or Protein

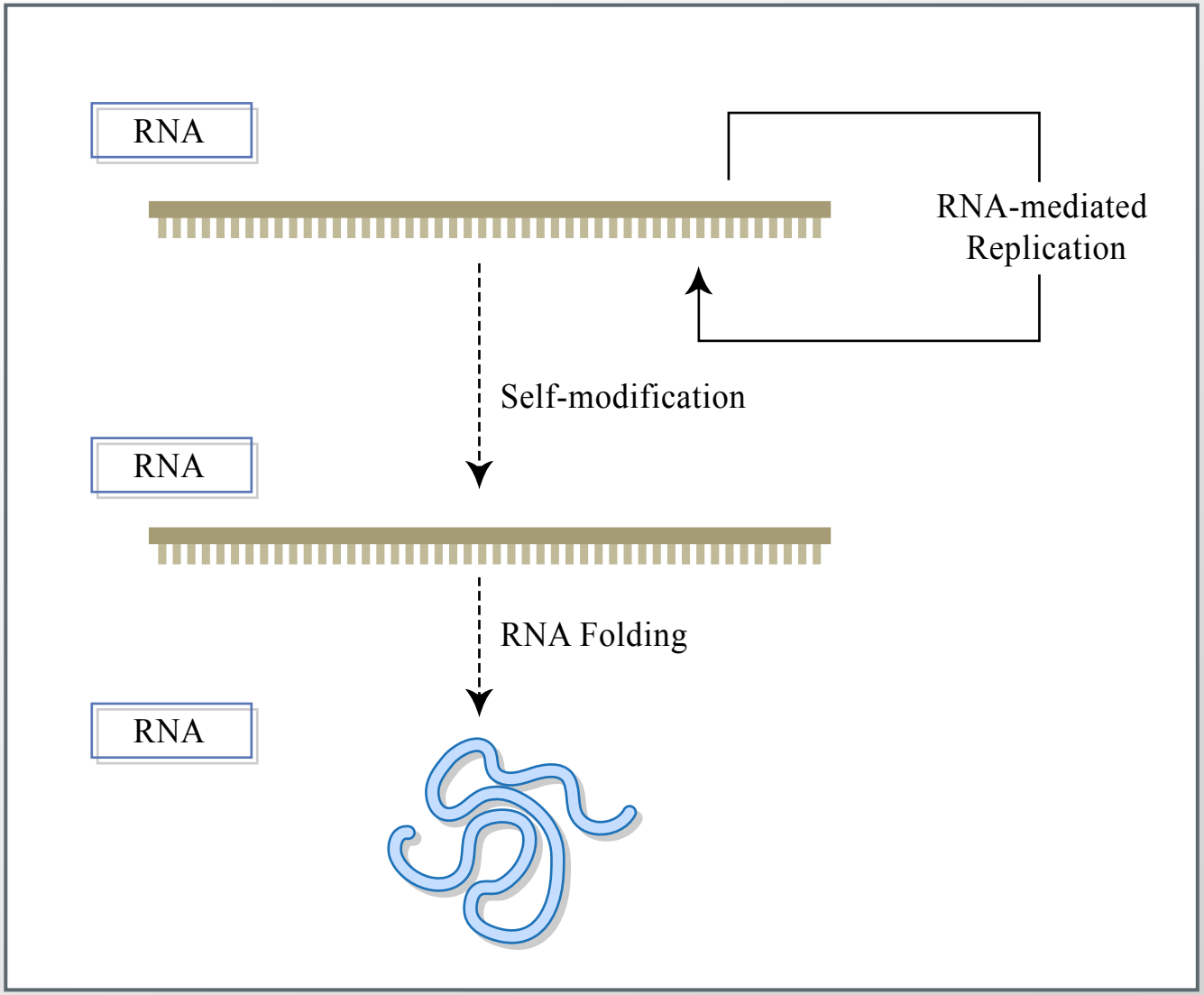


Figure by MIT OCW.

RNA World

- RNA can be protein-like
  - Ribozymes can catalyze enzymatic reactions by RNA secondary fold
  - Small RNAs can play structural roles within the cell
  - Small RNAs play versatile roles in gene regulatory
- RNA can be DNA-like
  - Made of digital information, can transfer to progeny by complementarity
  - Viruses with RNA genomes (single/double stranded)
  - RNA can catalyze RNA replication
- RNA world is possible
  - Proteins are more efficient (larger alphabet)
  - DNA is more stable (double helix, less flexible)

RNA structure

GCGGAUUUAGCUCAGUUGG
GAGAGCGCCAGACUGAAGA
UCUGGAGGUCCUGUGUUCG
AUCCACAGAAUUCGCACCA

Images removed due to copyright restrictions.

Please see: http://www.designeduniverse.com/articles/Nobel_Prize/trna.jpg

Primary Structure

Tertiary Structure

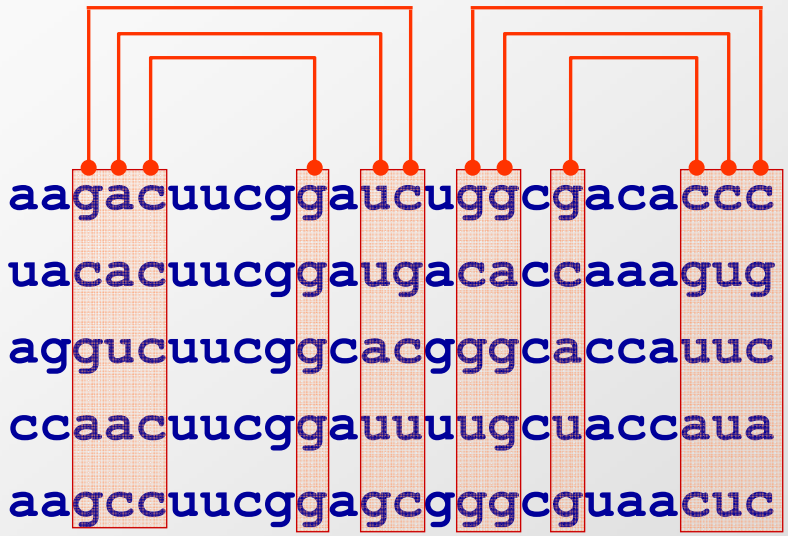
Secondary Structure

Adaptor molecule between DNA and protein

Comparative methods for RNA structure prediction

Multiple alignment and RNA folding

Given K homologous aligned RNA sequences:



If i^{th} and j^{th} positions are always base paired and covary, then they are likely to be paired

Mutual information

$$M_{ij} = \sum_{a,b \in \{a,c,g,u\}} f_{ab}(i,j) \log_2 \frac{f_{ab}(i,j)}{f_a(i)\, f_b(j)}$$

Where $f_{ab}(i,j)$ is the # of times the pair a, b are in positions i, j

Given a multiple alignment, can infer structure that maximizes the sum of mutual information, by DP

<u>In practice:</u>

1. Get multiple alignment
2. Find covarying bases – deduce structure
3. Improve multiple alignment (by hand)
4. Go to 2

A manual EM process!!

Results for tRNA

Image removed due to copyright restrictions.

Image removed due to copyright restrictions.

- Matrix of co-variations in tRNA molecule

Context Free Grammars
(review)

A Context Free Grammar

$S \rightarrow AB$

$A \rightarrow aAc \mid a$

$B \rightarrow bBd \mid b$

Nonterminals: S, A, B

Terminals: a, b, c, d

<u>Derivation:</u>

$S \rightarrow AB \rightarrow aAcB \rightarrow \ldots \rightarrow aaaacccB \rightarrow aaaacccbBd \rightarrow \ldots \rightarrow$
$aaaacccbbbbbddd$

Produces all strings $a^{i+1}c^{i}b^{j+1}d^{j}$, for $i, j \geq 0$

Example: modeling a stem loop

$S \rightarrow a\ W_1\ u$

$W_1 \rightarrow c\ W_2\ g$

$W_2 \rightarrow g\ W_3\ c$

$W_3 \rightarrow g\ \ L\ \ c$

$L \rightarrow agucg$

```
ACGG AG
     U
UGCC CG
```

What if the stem loop can have other letters in place of the ones shown?

Example: modeling a stem loop

$S \rightarrow a\, W_1\, u \quad | \quad g\, W_1\, u$

$W_1 \rightarrow c\, W_2\, g$

$W_2 \rightarrow g\, W_3\, c \quad | \quad g\, W_3\, u$

$W_3 \rightarrow g\ \ L\ \ c \quad | \quad a\ \ L\ \ u$

$L \rightarrow agucg \quad | \quad agccg \quad | \quad cugugc$

```
      AG
ACGG      U
UGCC
      CG
```

More general: Any 4-long stem, 3-5-long loop:

$S \rightarrow aW_1u \mid gW_1u \mid gW_1c \mid cW_1g \mid uW_1g \mid uW_1a$

$W_1 \rightarrow aW_2u \mid gW_2u \mid gW_2c \mid cW_2g \mid uW_2g \mid uW_2a$

$W_2 \rightarrow aW_3u \mid gW_3u \mid gW_3c \mid cW_3g \mid uW_3g \mid uW_3a$

$W_3 \rightarrow aLu \mid gLu \mid gLc \mid cLg \mid uLg \mid uLa$

```
      AG
GCGA      C
UGCU
      CG
```

$L \rightarrow aL_1 \mid cL_1 \mid gL_1 \mid uL_1$

$L_1 \rightarrow aL_2 \mid cL_2 \mid gL_2 \mid uL_2$

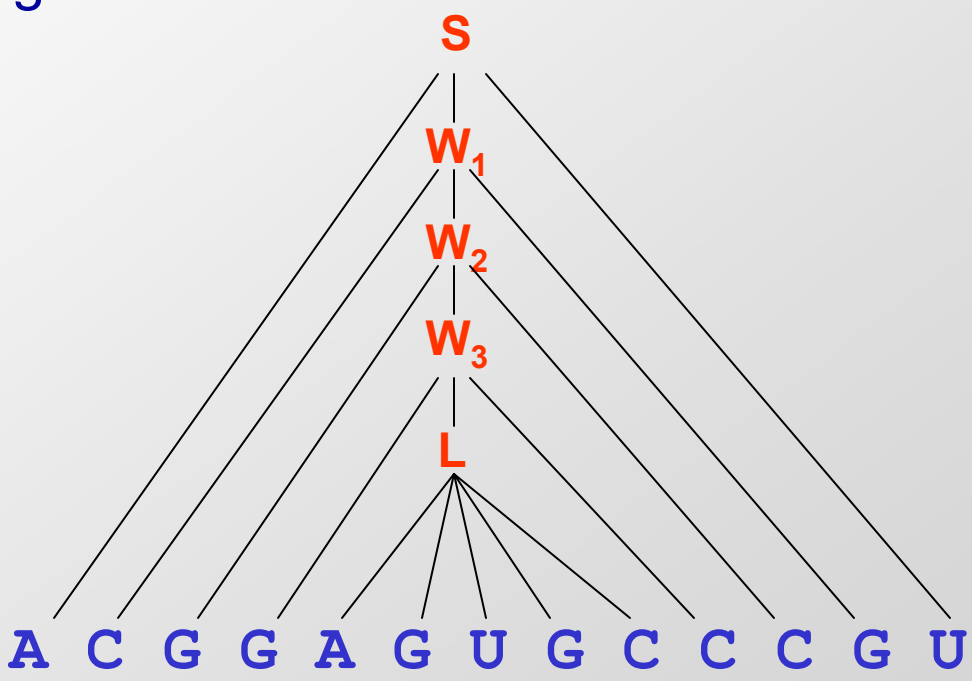
$L_2 \rightarrow a \mid c \mid g \mid u \mid aa \mid \ldots \mid uu \mid aaa \mid \ldots \mid uuu$

```
      CUG
GCGA      U
UGUU
      CG
```

A parse tree: alignment of CFG to sequence

- S → a W1 u
- W1 → c W2 g
- W2 → g W3 c
- W3 → g L c
- L → agucg

ACGG AG U
UGCC CG



Alignment scores for parses

We can define each rule X → s, where s is a string, to have a score.

Example:

W → a W' u:	3	(forms 3 hydrogen bonds)
W → g W' c:	2	(forms 2 hydrogen bonds)
W → g W' u:	1	(forms 1 hydrogen bond)
W → x W' z	-1, when (x, z) is not an a/u, g/c, g/u pair	

Questions:

- How do we best align a CFG to a sequence? (DP)
- How do we set the parameters? (Stochastic CFGs)

The Nussinov Algorithm and CFGs

Define the following grammar, with scores:

$$
\begin{aligned}
S \rightarrow\ & a\,S\,u : 3 \quad | \quad u\,S\,a : 3 \\
& g\,S\,c : 2 \quad | \quad c\,S\,g : 2 \\
& g\,S\,u : 1 \quad | \quad u\,S\,g : 1 \\
& \\
& S\,S : 0 \quad | \\
& \\
& a\,S : 0 \quad | \quad c\,S : 0 \quad | \quad g\,S : 0 \quad | \quad u\,S : 0 \quad | \quad \varepsilon : 0
\end{aligned}
$$

<u>Note:</u> ε is the “” string

Then, the Nussinov algorithm finds the optimal parse of a string with this grammar

Reformulating the Nussinov Algorithm

Initialization:

$F(i, i-1) = 0;$ for $i = 2$ to N

$F(i, i) = 0;$ for $i = 1$ to N

$S \rightarrow a \mid c \mid g \mid u$

Iteration:

For $i = 2$ to N:

For $i = 1$ to $N - l$

$j = i + l - 1$

$$F(i, j) = \max \begin{cases} F(i+1, j-1) + s(x_i, x_j) & S \rightarrow a\ S\ u \mid \ldots \\ \max\{ i \leq k < j \} \quad F(i, k) + F(k+1, j) & S \rightarrow S\ S \end{cases}$$

Termination:

Best structure is given by $F(1, N)$

Stochastic Context Free Grammars

Stochastic Context Free Grammars

- In an analogy to HMMs, we can assign probabilities to transitions:

- Given grammar

$X_1 \rightarrow s_{11} \mid \ldots \mid s_{in}$

…

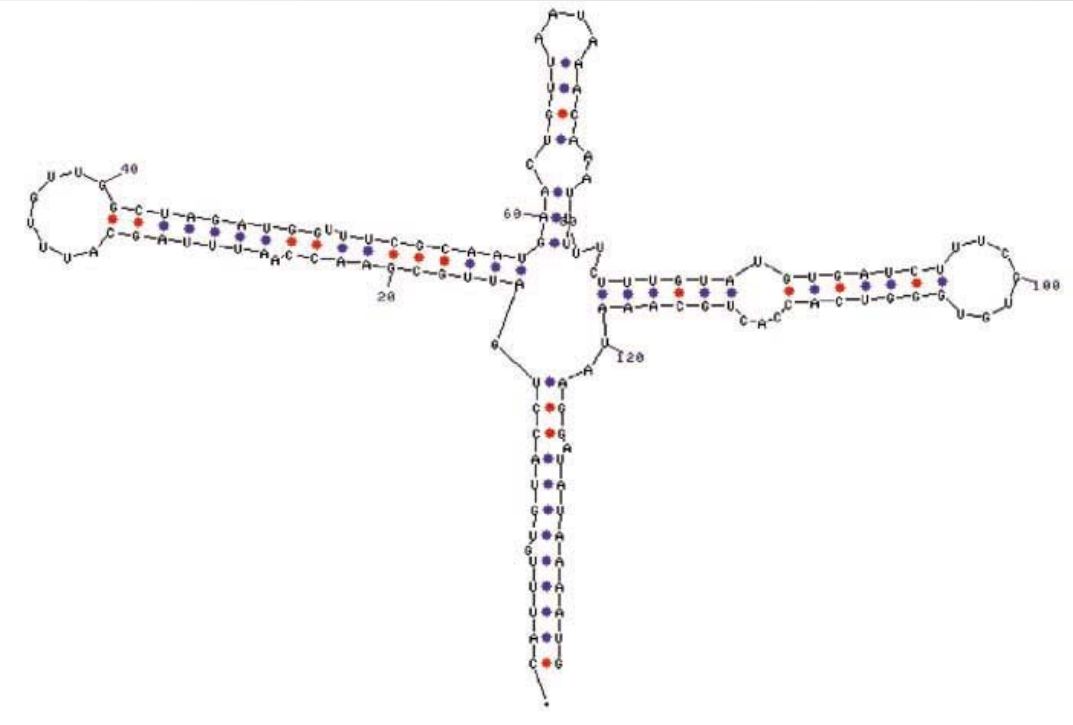
$X_m \rightarrow s_{m1} \mid \ldots \mid s_{mn}$

- Can assign probability to each rule, s.t.

$P(X_i \rightarrow s_{i1}) + \ldots + P(X_i \rightarrow s_{in}) = 1$

Scoring

- We can find best folding
- E.g.,
  - P(S $\rightarrow$ u S a)=0.2
  - P(S $\rightarrow$ c S a)=0.02
  - P(S $\rightarrow$ S a)=0.01
  - …



Courtesy of MFold Program, Michael Zuker.

Decoding: the CYK algorithm

- Given: $x = x_1 \ldots x_N$, and a SCFG G,

- Goal: find the most likely parsing of x according to G

- Dynamic programming variable:

  $\gamma(i, j, V)$ = probability that $x_i \ldots x_j$ can be generated from nonterminal V

The CYK algorithm (Cocke-Younger-Kasami)

Initialization:

For $i = 1$ to N, any nonterminal V,

$\gamma(i, i, V) = P(V \rightarrow x_i)$

Iteration:

For $i = 1$ to $N-1$

For $j = i+1$ to N

For any nonterminal V,

$$\gamma(i, j, V) = \max_{X,Y} \max_{i \le k < j} \gamma(i,k,X) * \gamma(k+1,j,Y) * P(V \rightarrow XY)$$

Before: $\gamma(i, j, V) = \max_{V \rightarrow XY} \max_{i \le k < j} \gamma(i,k,X) * \gamma(k+1,j,Y)$

Or: $\log \gamma(i, j, V) = \max_{X,Y} \max_{i \le k < j} \log \gamma(i,k,X) + \log \gamma(k+1,j,Y) + \log P(V \rightarrow XY)$

As in Nussinov's algorithm!

Computational Problems

- Calculate an optimal alignment of a sequence and a SCFG

  (DECODING)

- Calculate Prob[sequence | grammar]

  (EVALUATION)

- Given a set of sequences, estimate parameters of a SCFG

  (LEARNING)

Normal Forms for CFGs

Chomsky Normal Form:

$X \rightarrow YZ$
$X \rightarrow a$

All productions are either to 2 nonterminals, or to 1 terminal

<u>Theorem (technical)</u>

Every CFG has an equivalent one in Chomsky Normal Form

(That is, the grammar in normal form produces exactly the same set of strings)

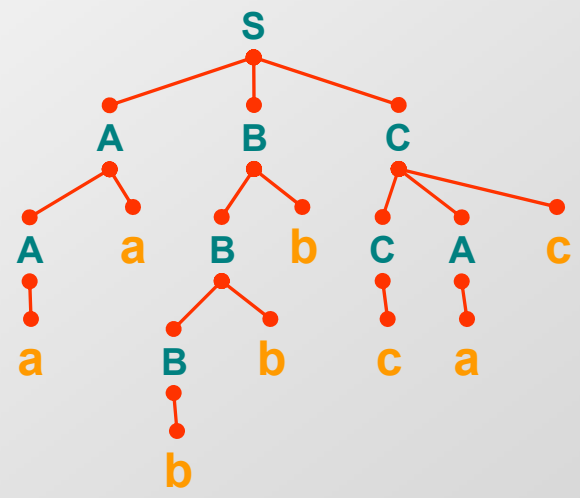
Example of converting a CFG to C.N.F.

$S \rightarrow ABC$

$A \rightarrow Aa \mid a$

$B \rightarrow Bb \mid b$

$C \rightarrow CAc \mid c$



Converting:

$S \rightarrow AS'$

$S' \rightarrow BC$

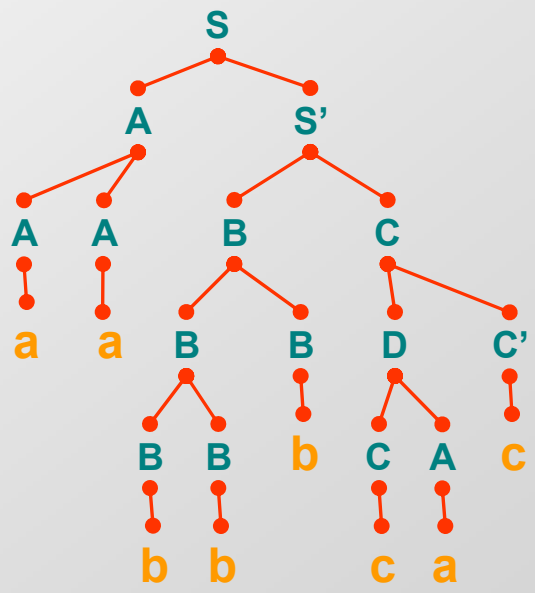
$A \rightarrow AA \mid a$

$B \rightarrow BB \mid b$

$C \rightarrow DC' \mid c$

$C' \rightarrow c$

$D \rightarrow CA$



Another example

$S \rightarrow ABC$
$A \rightarrow C \mid aA$
$B \rightarrow bB \mid b$
$C \rightarrow cCd \mid c$

Converting:
$S \rightarrow AS'$
$S' \rightarrow BC$
$A \rightarrow C'C'' \mid c \mid A'A$
$A' \rightarrow a$
$B \rightarrow B'B \mid b$
$B' \rightarrow b$
$C \rightarrow C'C'' \mid c$
$C' \rightarrow c$
$C'' \rightarrow CD$
$D \rightarrow d$

Algorithms for learning Grammars

Decoding: the CYK algorithm

Given $x = x_1....x_N$, and a SCFG G,

Find the most likely parse of x
(the most likely alignment of G to x)

Dynamic programming variable:

$\gamma(i, j, V)$: likelihood of the most likely parse of $x_i \ldots x_j$, rooted at nonterminal V

Then,

$\gamma(1, N, S)$: likelihood of the most likely parse of x by the grammar

The CYK algorithm (Cocke-Younger-Kasami)

Initialization:

For i = 1 to N, any nonterminal V,

$\gamma(i, i, V) = \log P(V \rightarrow x_i)$

Iteration:

For i = 1 to N-1

For j = i+1 to N

For any nonterminal V,

$$\gamma(i, j, V) = \max_X \max_Y \max_{i \le k < j} \gamma(i,k,X) + \gamma(k+1,j,Y) + \log P(V \rightarrow XY)$$

Termination:

$\log P(x \mid \theta, \pi^*) = \gamma(1, N, S)$

Where π^* is the optimal parse tree (if traced back appropriately from above)

A SCFG for predicting RNA structure

$$
\begin{aligned}
S &\rightarrow aS \mid cS \mid gS \mid uS \mid \varepsilon \\
&\rightarrow Sa \mid Sc \mid Sg \mid Su \\
&\rightarrow aSu \mid cSg \mid gSu \mid uSg \mid gSc \mid uSa \\
&\rightarrow SS
\end{aligned}
$$

- Adjust the probability parameters to reflect bond strength etc

- No distinction between non-paired bases, bulges, loops
- Can modify to model these events
  - L: loop nonterminal
  - H: hairpin nonterminal
  - B: bulge nonterminal
  - etc

CYK for RNA folding

Initialization:

$\gamma(i, i\text{-}1) = \log P(\varepsilon)$

Iteration:

For i = 1 to N

For j = i to N

$$\gamma(i, j) = \max \begin{cases} \gamma(i+1, j-1) + \log P(x_i\, S\, x_j) \\ \gamma(i, j-1) + \log P(S\, x_i) \\ \gamma(i+1, j) + \log P(x_i\, S) \\ \max_{i < k < j} \gamma(i, k) + \gamma(k+1, j) + \log P(S\, S) \end{cases}$$

Evaluation

Recall HMMs:

Forward: $f_l(i) = P(x_1 \ldots x_i, \pi_i = l)$

Backward: $b_k(i) = P(x_{i+1} \ldots x_N \mid \pi_i = k)$

Then,

$P(x) = \Sigma_k f_k(N) a_{k0} = \Sigma_l a_{0l} e_l(x_1) b_l(1)$

Analogue in SCFGs:

Inside: $a(i, j, V)$ = $P(x_i \ldots x_j$ is generated by nonterminal V)

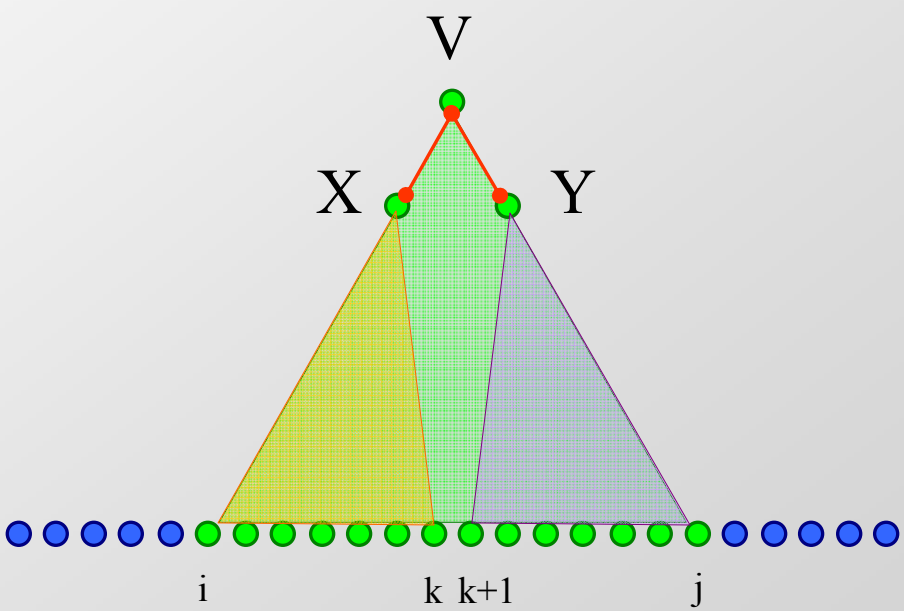
Outside: $b(i, j, V)$ = P(x, excluding $x_i \ldots x_j$ is generated by S and the excluded part is rooted at V)

The Inside Algorithm

To compute

$a(i, j, V) = P(x_i \ldots x_j, \text{ produced by } V)$

$a(i, j, v) = \Sigma_X \Sigma_Y \Sigma_k \, a(i, k, X) \, a(k+1, j, Y) \, P(V \rightarrow XY)$



Algorithm: Inside

Initialization:

For i = 1 to N, V a nonterminal,

$$a(i, i, V) = P(V \rightarrow x_i)$$

Iteration:

For i = 1 to N-1

For j = i+1 to N

For V a nonterminal

$$a(i, j, V) = \Sigma_X \Sigma_Y \Sigma_k a(i, k, X)\, a(k+1, j, X)\, P(V \rightarrow XY)$$

Termination:

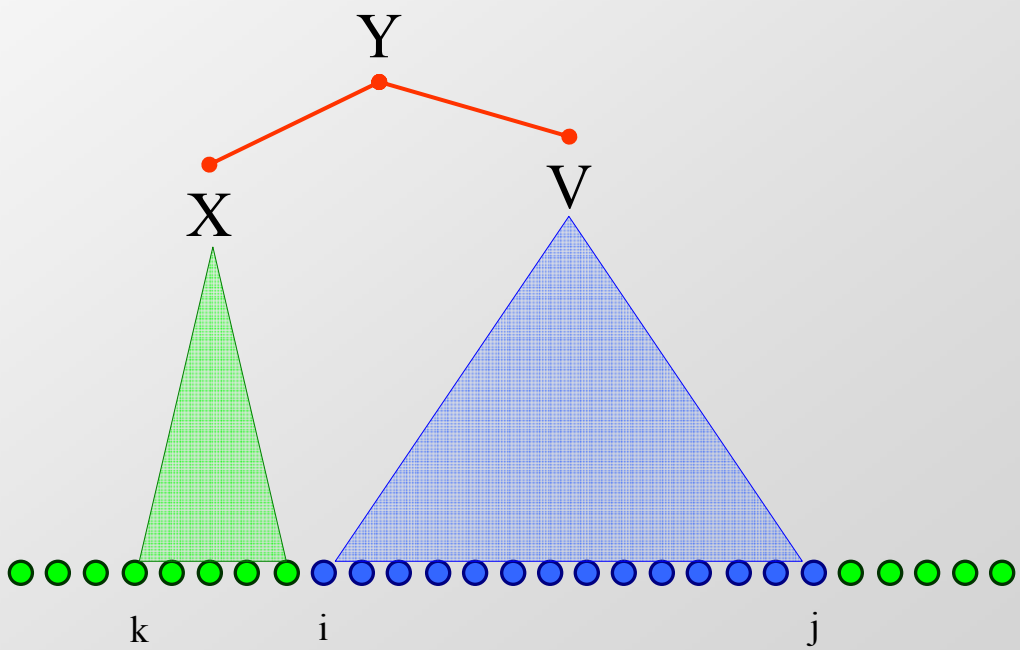
$$P(x \mid \theta) = a(1, N, S)$$

The Outside Algorithm

$b(i, j, V) = \text{Prob}(x_1 \ldots x_{i-1}, x_{j+1} \ldots x_N$, where the "gap" is rooted at V)

Given that V is the right-hand-side nonterminal of a production,

$$b(i, j, V) = \Sigma_X \, \Sigma_Y \, \Sigma_{k<i} \, a(k, i-1, X) \, b(k, j, Y) \, P(Y \rightarrow XV)$$



Algorithm: Outside

Initialization:

$b(1, N, S) = 1$

For any other V, $b(1, N, V) = 0$

Iteration:

For i = 1 to N-1

For j = N down to i

For V a nonterminal

$$b(i, j, V) = \Sigma_X \Sigma_Y \Sigma_{k<i} a(k, i-1, X)\, b(k, j, Y)\, P(Y \rightarrow XV) + \Sigma_X \Sigma_Y \Sigma_{k<i} a(j+1, k, X)\, b(i, k, Y)\, P(Y \rightarrow VX)$$

Termination:

It is true for any i, that:

$$P(x \mid \theta) = \Sigma_X b(i, i, X)\, P(X \rightarrow x_i)$$

Learning for SCFGs

We can now estimate

c(V) = expected number of times V is used in the parse of $x_1 \ldots x_N$

$$c(V) = \frac{1}{P(x \mid \theta)} \sum_{1 \le i \le N} \sum_{i \le j \le N} a(i, j, V)\, b(i, j, v)$$

$$c(V \rightarrow XY) = \frac{1}{P(x \mid \theta)} \sum_{1 \le i \le N} \sum_{i < j \le N} \sum_{i \le k < j} b(i,j,V)\, a(i,k,X)\, a(k+1,j,Y)\, P(V \rightarrow XY)$$

Learning for SCFGs

Then, we can re-estimate the parameters with EM, by:

$$P^{new}(V \rightarrow XY) = \frac{c(V \rightarrow XY)}{c(V)}$$

$$P^{new}(V \rightarrow a) = \frac{c(V \rightarrow a)}{c(V)} = \frac{\sum_{i:\, x_i = a} b(i, i, V)\, P(V \rightarrow a)}{\sum_{1 \le i \le N} \sum_{i < j \le N} a(i, j, V)\, b(i, j, V)}$$

Summary: SCFG and HMM algorithms

GOAL	HMM algorithm	SCFG algorithm
Optimal parse	Viterbi	CYK
Estimation	Forward Backward	Inside Outside
Learning	EM: Fw/Bck	EM: Ins/Outs
Memory Complexity	$O(N\ K)$	$O(N^2\ K)$
Time Complexity	$O(N\ K^2)$	$O(N^3\ K^3)$

Where K: # of states in the HMM
# of nonterminals in the SCFG