Stochastic Context Free Grammars for RNA Modeling

CS 838

www.cs.wisc.edu/~craven/cs838.html

Mark Craven

craven@biostat.wisc.edu

May 2001

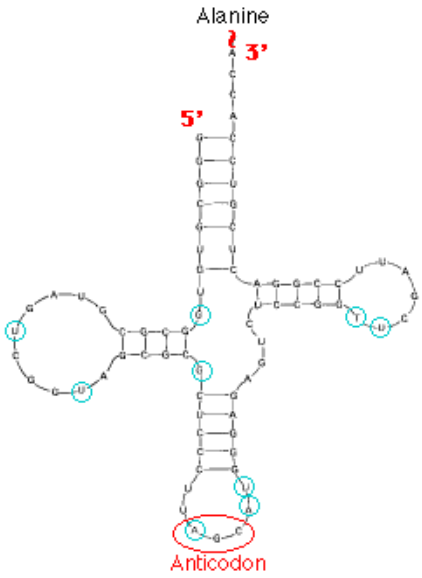
Why RNA Is Interesting

- in addition to messenger RNA (mRNA), there are other RNA molecules that play key roles in biology
  - ribosomal RNA (rRNA)
    - ribosomes are complexes that incorporate several RNA subunits in addition to numerous protein units
  - transfer RNA (tRNA)
    - transport amino acids to the ribosome during translation
  - the spliceosome, which performs intron splicing, is a complex with several RNA units
  - the genomes for many viruses (e.g. HIV) are encoded in RNA
  - etc.

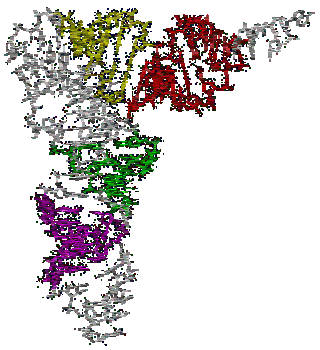
RNA Secondary Structure

- RNA is typically single stranded
- folding, in large part is determined by base-pairing
  A-U and C-G are the canonical base pairs
  other bases will sometimes pair, especially G-U
- the base-paired structure is referred to as the *secondary structure* of RNA
- related RNAs often have homologous secondary structure without significant sequence similarity

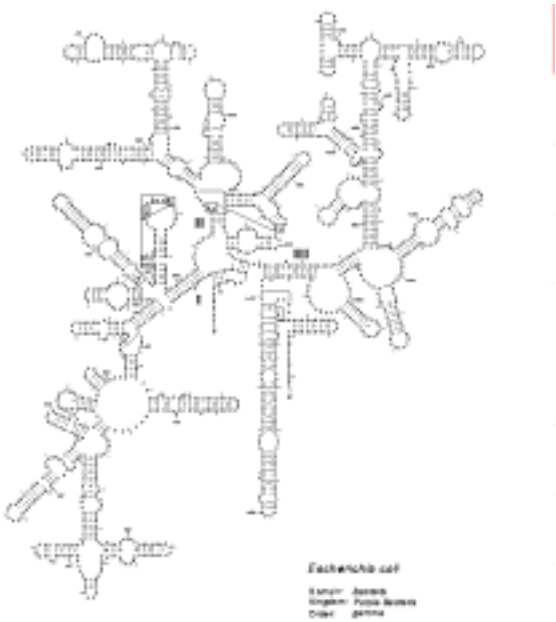
tRNA Secondary Structure



tertiary structure



Small Subunit Ribosomal RNA



Modeling RNA with Stochastic Context Free Grammars

- consider tRNA genes
  - 274 in yeast genome, ~1500 in human genome
  - get transcribed, like protein-coding genes
  - don't get translated, therefore base statistics much different than protein-coding genes
  - but secondary structure is conserved
- to recognize new tRNA genes, model known ones using stochastic context free grammars [Eddy & Durbin, 1994; Sakakibara et al. 1994]
- but what is a grammar?

Transformational Grammars

- a transformational grammar characterizes a set of legal strings
- the grammar consists of
  - a set of abstract *nonterminal* symbols
    $\{s,\ c_1,\ c_2,\ c_3,\ c_4\}$
  - a set of *terminal* symbols (those that actually appear in strings)
    $\{A,\ C,\ G,\ U\}$
  - a set of *productions*

$$c_1 \rightarrow \mathrm{U}c_2 \qquad c_2 \rightarrow \mathrm{A}c_3 \qquad c_3 \rightarrow \mathrm{A}$$
$$c_2 \rightarrow \mathrm{G}c_4 \qquad c_3 \rightarrow \mathrm{G}$$
$$c_4 \rightarrow \mathrm{A}$$

A Grammar for Stop Codons

$$s \rightarrow c_1 \quad c_1 \rightarrow \mathrm{U}c_2 \quad c_2 \rightarrow \mathrm{A}c_3 \quad c_3 \rightarrow \mathrm{A}$$
$$c_2 \rightarrow \mathrm{G}c_4 \quad c_3 \rightarrow \mathrm{G}$$
$$c_4 \rightarrow \mathrm{A}$$

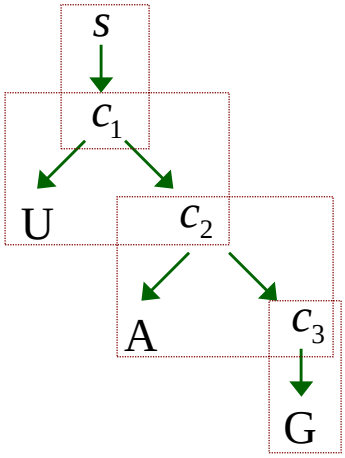
- this grammar can generate the 3 stop codons: UAA, UAG, UGA
- with a grammar we can ask questions like
  - what strings are derivable from the grammar?
  - can a particular string be derived from the grammar?

The Parse Tree for UAG

$s \rightarrow c_1$ $c_1 \rightarrow \mathrm{U}c_2$ $c_2 \rightarrow \mathrm{A}c_3$ $c_3 \rightarrow \mathrm{A}$

$c_2 \rightarrow \mathrm{G}c_4$ $c_3 \rightarrow \mathrm{G}$

$c_4 \rightarrow \mathrm{A}$



A Probabilistic Version of the Grammar

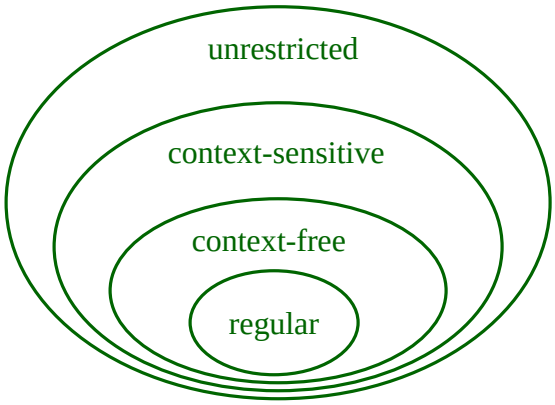
$s \xrightarrow{1.0} c_1$ $c_1 \xrightarrow{1.0} \mathrm{U}c_2$ $c_2 \xrightarrow{0.7} \mathrm{A}c_3$ $c_3 \xrightarrow{0.2} \mathrm{A}$

$c_2 \xrightarrow{0.3} \mathrm{G}c_4$ $c_3 \xrightarrow{0.8} \mathrm{G}$

$c_4 \xrightarrow{1.0} \mathrm{A}$

- each production has an associated probability
- the probabilities for productions with the same left-hand side sum to 1
- *this* grammar has a corresponding Markov chain model

The Chomsky Hierarchy



- a hierarchy of grammars defined by restrictions on productions

The Chomsky Hierarchy

- regular grammars

  $u \rightarrow \mathrm{X}v \qquad u \rightarrow \mathrm{X}$
- context-free grammars

  $u \rightarrow \beta$
- context-sensitive grammars

  $\alpha_1 u \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$
- unrestricted grammars

  $\alpha_1 u \alpha_2 \rightarrow \gamma$

- where u is a nonterminal, X a terminal, α, γ any sequence of terminals/nonterminals except the null string, and β any sequence of terminals/nonterminals

CFGs and RNA

- context free grammars are well suited to modeling RNA secondary structure because they can represent base pairing preferences
- a grammar for a 3-base stem with and a loop of either GAAA or GCAA

$$s \rightarrow \mathrm{A}w_1\mathrm{U} \mid \mathrm{C}w_1\mathrm{G} \mid \mathrm{G}w_1\mathrm{C} \mid \mathrm{U}w_1\mathrm{A}$$
$$w_1 \rightarrow \mathrm{A}w_2\mathrm{U} \mid \mathrm{C}w_2\mathrm{G} \mid \mathrm{G}w_2\mathrm{C} \mid \mathrm{U}w_2\mathrm{A}$$
$$w_2 \rightarrow \mathrm{A}w_3\mathrm{U} \mid \mathrm{C}w_3\mathrm{G} \mid \mathrm{G}w_3\mathrm{C} \mid \mathrm{U}w_3\mathrm{A}$$
$$w_3 \rightarrow \mathrm{GAAA} \mid \mathrm{GCAA}$$

CFGs and RNA

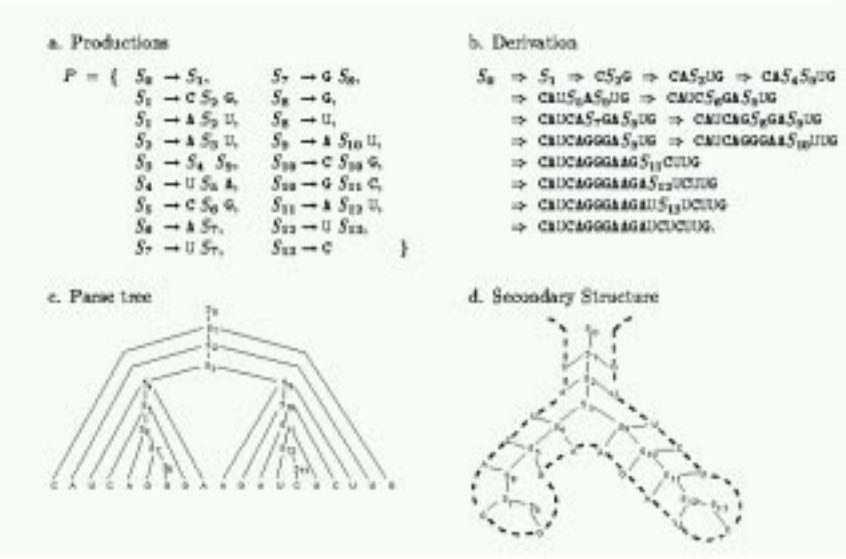


Figure from: Sakakibara et al. *Nucleic Acids Research*, 1994

Stochastic Context Free Grammars

$$s \xrightarrow{0.25} Aw_1U \mid \xrightarrow{0.25} Cw_1G \mid \xrightarrow{0.25} Gw_1C \mid \xrightarrow{0.25} Uw_1A$$

$$w_1 \xrightarrow{0.1} Aw_2U \mid \xrightarrow{0.4} Cw_2G \mid \xrightarrow{0.4} Gw_2C \mid \xrightarrow{0.1} Uw_2A$$

$$w_2 \xrightarrow{0.25} Aw_3U \mid \xrightarrow{0.25} Cw_3G \mid \xrightarrow{0.25} Gw_3C \mid \xrightarrow{0.25} Uw_3A$$

$$w_3 \xrightarrow{0.8} GAAA \mid \xrightarrow{0.2} GCAA$$

Stochastic Grammars?

…the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.

— Noam Chomsky
(famed linguist)

Every time I fire a linguist, the performance of the recognizer improves.

— Fred Jelinek
(former head of IBM speech recgonition group)

Credit for pairing these quotes goes to Dan Jurafsky and James Martin, *Speech and Language Processing*

Three Key Questions

- How likely is a given sequence?
  the Inside algorithm
- What is the most probable parse for a given sequence?
  the Cocke-Younger-Kasami (CYK) algorithm
- How can we learn the SCFG parameters given a grammar and a set of sequences?
  the Inside-Outside algorithm

Chomsky Normal Form

- it is convenient to assume that our grammar is in *Chomsky Normal Form*; i.e all productions are of the form:

  $v \rightarrow yz$ right hand side consists of two nonterminals

  $v \rightarrow A$ right hand side consists of a single terminal

- any CFG can be put into Chomsky Normal Form

Parameter Notation

- for productions of the form $v \rightarrow yz$, we'll denote the associated probability parameters

  $t_v(y, z)$ transition

- for productions of the form $v \rightarrow A$, we'll denote the associated probability parameters

  $e_v(A)$ emission

Determining the Likelihood of a Sequence: The Inside Algorithm

- a dynamic programming method, analogous to the Forward algorithm
- involves filling in a 3D matrix

  $\alpha(i, j, v)$

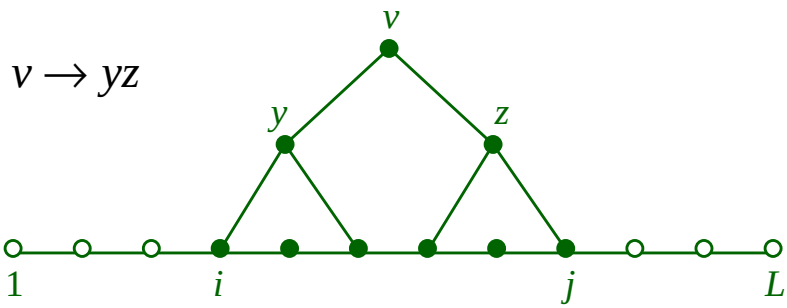
  representing the probability of the all parse subtrees rooted at nonterminal *v* for the subsequence from *i* to *j*

Announcements

- talk today: Harold Varmus, *Defining Tumor Maintenance Functions*, 2pm today in Bascom 272
  - class will end today at 1:45
- class projects
  - I sent email this morning with guidelines for the write-up
  - due Friday, 5/17
- final exam
  - 6-8pm Thursday 5/9
  - room to be announced
- lecture notes
  - all are on the web page now, including those on protein structure prediction

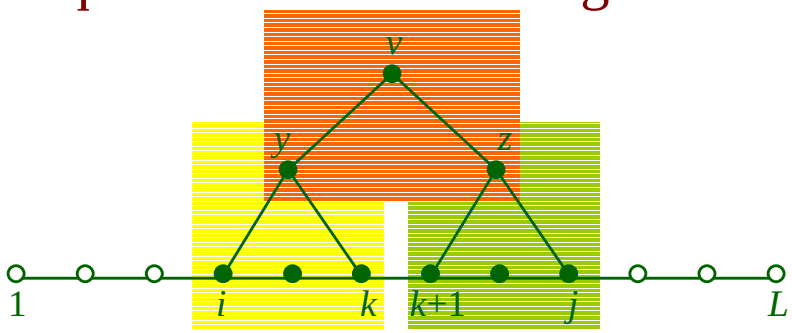
Determining the Likelihood of a Sequence: The Inside Algorithm

$v \rightarrow yz$



- $\alpha(i, j, v)$: the probability of all parse subtrees rooted at nonterminal *v* for the subsequence from *i* to *j*

Determining the Likelihood of a Sequence: The Inside Algorithm



$$\alpha(i, j, v) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=i}^{j-1} \alpha(i,k,y)\, \alpha(k+1, j, z)\, t_v(y,z)$$

M is the number of nonterminals in the grammar

The Inside Algorithm

- initialization (for i = 1 to L, v = 1 to M)

$$\alpha(i,i,v) = e_v(x_i)$$

- iteration (for i = 1 to L - 1, j = i+1 to L, v = 1 to M)

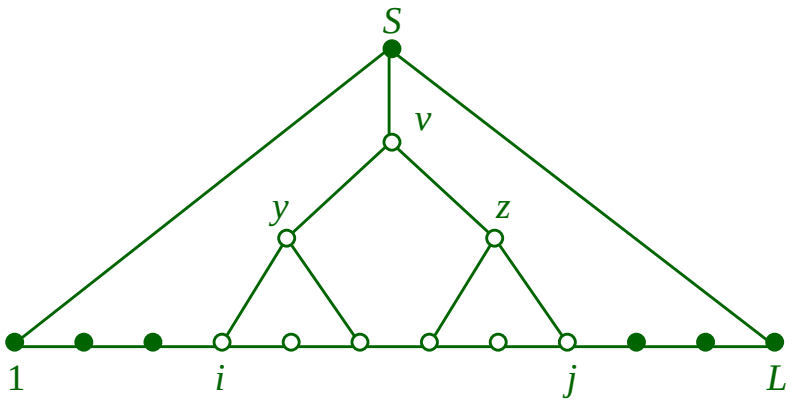
$$\alpha(i, j, v) = \sum_{y=1}^{M} \sum_{z=1}^{M} \sum_{k=i}^{j-1} \alpha(i,k,y)\alpha(k+1, j, z)\, t_v(y,z)$$

- termination

$$\Pr(x) = \alpha(1, L, 1)$$

start nonterminal

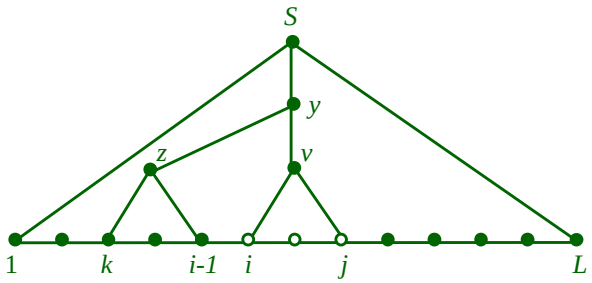
The Outside Algorithm



- $\beta(i, j, v)$: the probability of parse trees rooted at the start nonterminal, exluding the probability of all subtrees rooted at nonterminal *v* covering the subsequence from *i* to *j*

The Outside Algorithm

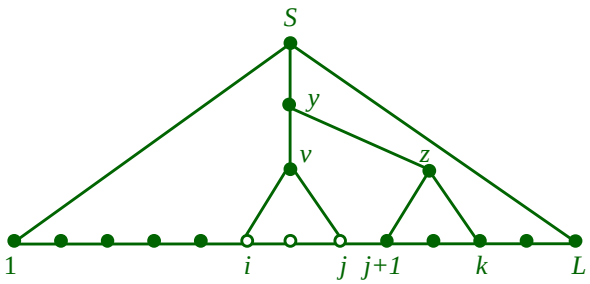
- we can recursively calculate $\beta(i, j, v)$ from β values we've calculated for *y*
- the first case we consider is where *v* is used in productions of the form: $y \rightarrow zv$



$$\sum_{y=1}^{M}\sum_{z=1}^{M}\sum_{k=1}^{i-1}\alpha(k, i-1, z)\beta(k, j, y)\, t_y(z, v)$$

The Outside Algorithm

- the second case we consider is where v is used in productions of the form: $y \rightarrow vz$



$$\sum_{y=1}^{M}\sum_{z=1}^{M}\sum_{k=j+1}^{L}\alpha(j+1,k,z)\beta(i,k,y)\,t_y(v,z)$$

The Outside Algorithm

- initialization

$$\beta(1,L,1)=1 \qquad \text{(the } \textit{start} \text{ nonterminal)}$$

$$\beta(1,L,v)=0 \qquad \text{for } v=2 \text{ to } M$$

- iteration (for $i = 1$ to L, $j = L$ to i, $v = 1$ to M)

$$\beta(i,j,v)=\sum_{y=1}^{M}\sum_{z=1}^{M}\sum_{k=1}^{i-1}\alpha(k,i-1,z)\beta(k,j,y)\,t_y(z,v)+$$

$$\sum_{y=1}^{M}\sum_{z=1}^{M}\sum_{k=j+1}^{L}\alpha(j+1,k,z)\beta(i,k,y)\,t_y(v,z)$$

Announcements

- talk today:

  Keith Dunker, Washington State Univ.

  The Protein Trinity: Linking Function with Intrinsic Disorder

  3:30pm today in Biochemistry B1118
- final exam:
  - 6-8pm Thursday 5/9
  - 115 Psychology
  - you can bring 1-2 pages of notes

Learning SCFG Parameters

- if we know the parse tree for each training sequence, learning the SCFG parameters is simple
  - no hidden state during training
  - count how often each parameter (i.e. production) is used
  - normalize/smooth to get probabilities
- more commonly, there are many possible parse trees per sequence – we don't know which one is correct
  - thus, use an EM approach (Inside-Outside)
  - iteratively
    - determine expected # times each production is used
      - consider all parses
      - weight each by it's probability
    - set parameters to maximize these counts

The Inside-Outside Algorithm

- we can learn the parameters of an SCFG from training sequences using an EM approach called Inside-Outside
- in the E-step, we determine
  - the expected number of times each *nonterminal* is used in parses $c(v)$
  - the expected number of times each *production* is used in parses
    $c(v \to yz)$
    $c(v \to A)$
- in the M-step, we update our production probabilities

The Inside-Outside Algorithm

- the EM re-estimation equations (for 1 sequence) are:

$$\hat{e}_v(A) = \frac{c(v \to A)}{c(v)} = \frac{\sum_{i|x_i=A} \beta(i,i,v) e_v(A)}{\sum_{i=1}^{L}\sum_{j=i}^{L} \beta(i,j,v)\alpha(i,j,v)}$$

cases where *v* used to generate *A*

cases where *v* used to generate any subsequence

$$\hat{t}_v(y,z) = \frac{c(v \to yz)}{c(v)} = \frac{\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\sum_{k=i}^{j-1} \beta(i,j,v) t_v(y,z)\alpha(i,k,y)\alpha(k+1,j,z)}{\sum_{i=1}^{L}\sum_{j=i}^{L} \beta(i,j,v)\alpha(i,j,v)}$$

The CYK Algorithm

- analogous to Viterbi algorithm
- like Inside algorithm but
  - max operations instead of sums
  - retain traceback pointers
- traceback is a little more involved than Viterbi
  - need to reconstruct parse tree instead of recovering simple path

Summary of SCFG Algorithms

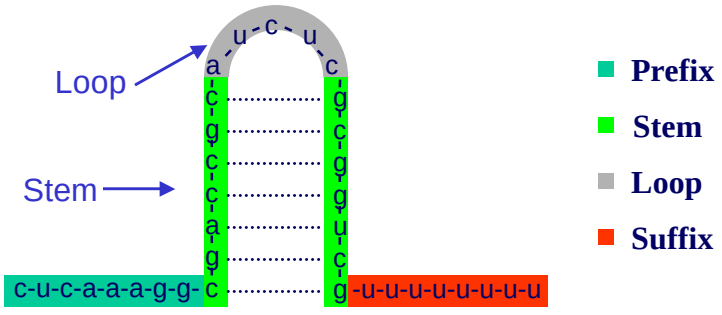
	HMM algorithm	**SCFG algorithm**
optimal alignment	Viterbi	CYK
probability of sequence	forward	inside
EM parameter estimation	forward-backward	inside-outside
memory complexity	$O(LM)$	$O(L^2M)$
time complexity	$O(LM^2)$	$O(L^3M^3)$

Applications of SCFGs

- SCFGs have been applied to constructing multiple alignments and recognizing new instances of:
  - tRNA genes [Eddy & Durbin, 1994; Sakakibara et al., 1994]
  - rRNA subunits [Brown, 2000]
  - terminators [Bockhorst & Craven, 2001]
- trained SCFG models can be used to
  - recognize new instances (Inside algorithm)
  - predict secondary structure (CYK algorithm)
  - construct multiple alignments (CYK algorithm)

Recognizing Terminators with SCFGs

- [Bockhorst & Craven, *IJCAI* 2001]



- a prototypical terminator has the structure above
- the lengths and base compositions of the elements can vary a fair amount

Our Initial Terminator Grammar

START	→	S		
S	→	t_l S t_r	t_r	
S	→	t_l^* S	t_r^* | t_l^* S t_r	t_r^*
S	→	t_l^* S	t_r^* | t_l^* S t_r	t_r^*
S	→	t_l^* S t_r	t_r^*	
S	→	t_l t_r		
	→			
	→		| λ	
S	→			
	→	a | c | g | u		

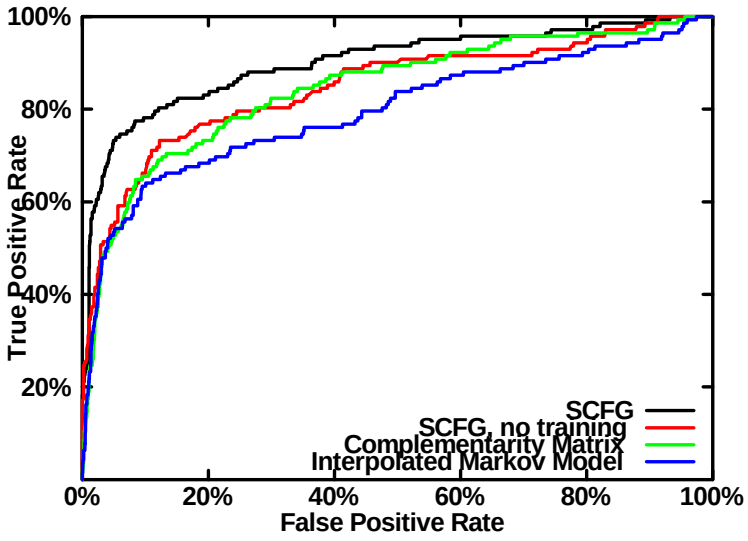
Nonterminals are uppercase, terminals are lowercase

t = {**a,c,g,u**},
t^* = {**a,c,g,u,** λ}

SCFG Experiments

- compare predictive accuracy of
  - SCFG with learned parameters
  - SCFG without learning (but parameters initialized using domain knowledge)
  - interpolated Markov models (IMMs)
    - can represent distribution of bases at each position
    - * cannot easily encode base pair dependencies
  - complementarity matrices
    - Brendel et al., *J Biom Struct and Dyn* 1986
    - ad hoc way of considering base pairings
    - * cannot favor specific base pairs by position

SCFGs vs. Related Methods



Refining the Structure of an SCFG