Stochastic Grammars

Stochastic Context-Free Grammars

I. Holmes

Department of Bioengineering
University of California, Berkeley

Spring semester

Outline

1. Overview of transformational grammars
2. RNA structure
3. Dynamic programming algorithms for SCFGs
4. Beyond SCFGs

- Overview: HMM profiles, HMM genefinders, SCFGs for RNA, repeats, beta-sheets; Natural Language Processing
- What is a transformational grammar? Formal definition: terminals Ω, nonterminals Φ, transformation rules
  - "Language" = set of strings generated by the grammar
  - "Parser" = computer program to **decide** if a given input string is in the language (returns "true" or "false")
  - More generally, we're interested in parsers that compute scores (energies, probabilities) for a given input string
  - These scores are associated with the transformation rules. The grammar is said to be score-*attributed* (Knuth)

- The Chomsky hierarchy of grammars and their associated parsers
  - Regular grammars: finite-state machines (HMMs)
  - Context-free grammars: pushdown automata (SCFGs)
  - Context-sensitive grammars: finite-tape Turing machines (linear-bounded automata)
  - Unrestricted grammars: infinite-tape Turing machines

- The Chomsky hierarchy of grammars and their associated parsers (lower levels)
  - Regular grammars: finite-state machines (HMMs)
  - Context-free grammars: pushdown automata (SCFGs)
    - The **parse tree**; the **inside sequence** and **outside sequence**
    - Chomsky Normal Form; Eddy *et al*'s "RNA Normal Form"
    - Transformations to & consequent universality of Chomsky Normal Form
    - History: Panini's "Ashtadhyayi": a CFG for Sanskrit (3959 rules (sutras), circa.300-700BC). Bishop Robert Lowth's "A Short Introduction to English Grammar" (1762). Chomsky's theory of universal generative grammars (1956 and onwards).

- The Chomsky hierarchy of grammars and their associated parsers (upper levels)
  - Context-sensitive grammars: Turing machines with bounded tape = linear-bounded automata
    - Big category: low-complexity sequence repeats, tandem repeats, other bounded correlations e.g. pseudoknots
    - Lempel-Ziv compression algorithm (allowing local stutter) can be viewed as one of these
    - Limited context sensitivity (e.g. basepair stacking) can be effected using a *lexicalised* SCFG
    - Tree-Adjoining Grammars, Linear-Indexed Grammars, etc.
  - Unrestricted grammars: complete Turing machines

- Summary of Chomsky hierarchy (NB regular $\subset$ context-free $\subset$ context-sensitive $\subset$ unrestricted)

Class	Example rule	Automaton	Complexity
Regular	$S \rightarrow x\ S$	Finite-state	$O(L)$
Context-free	$S \rightarrow T\ U$	Pushdown	$O(L^3)$
Context-sensitive	$S\ T \rightarrow TS$	Linear-bounded	$O(\lvert\Phi\rvert^L)$
Unrestricted	$S\ T \rightarrow U$	Turing machine	Undecidable

Here S, T, U are nonterminals and x is a terminal. "Complexity" refers to the time complexity of parsing a sequence of length L.

- Why RNA is important in evolution and cell biology
  - RNA world; pre- and post-transcriptional regulation; Crick's idea of studying simple examples; ribotechnology

- RNA structure terminology: basepairs, stems, loops, pseudoknots, kissing loops

Terms contributing to the free energy of a folded RNA structure (`scor.berkeley.edu`)

- Hydrogen bonding between bases. Canonical, noncanonical pairs
- Stacking energies due to overlap of π-orbitals of adjacent planar basepairs
  - H-bonding and stacking terms can be combined and measured by direct experiment.
- Unusual configurations: tetraloops, triloops, triple-A platforms
  - Finite number of cases, so also amenable to experimental measurement.

- Entropic cost of closing loops
  - Theory: rods and Gaussian springs. Integrate out displacements, get likelihood ratio (Doi-Edwards, Isambert)
  - Statistics of random walk, $\langle |\Delta \mathbf{x}|^2 \rangle \propto t$, and self-avoiding walk, $\langle |\Delta \mathbf{x}|^2 \rangle \propto t^{1+\epsilon}$
    - Renormalisation (Edwards, de Gennes)
  - Empirical scaling laws fit experimental measurements
- Ligands: solvation, metal ions, small molecules
  - General rules not yet known. Specific small-molecule binding requires conserved motifs (riboswitches).
- Unlike proteins (where amino acid sidechains make multiple contacts), a "**basepair stacking + convex loop penalty**" picture of the free energy seems reasonably accurate as a first approximation (Zuker, Turner, Mathews...)

- The Nussinov algorithm: Finds *strictly nested* foldback structures, i.e. excluding pseudoknots.
  - RNA sequence X, length L, nucleotides $x_1 \ldots x_L$
  - Let $H(x, x') = 1$ if xx' is a canonical Watson-Crick basepair, and 0 otherwise
  - Nussinov recursion finds the structure for $x_i \ldots x_j$ that has the most strictly nested canonical basepairs

$$S(i,j) = \max \begin{pmatrix} S(i+1,j), \\ S(i,j-1), \\ S(i+1,j-1) + H(x_i, x_j), \\ \max_{i \le k \le j}(S(i,k) + S(k,j)) \end{pmatrix}$$

Best structure for X is found by traceback from $S(1, L)$

Nussinov algorithm = parser for **score-attributed grammar**

Rule			Score
S	$\rightarrow$	$S\ x$	0
	$\mid$	$x\ S$	0
	$\mid$	$x\ S\ x'$	H(x,x')
	$\mid$	$S\ S$	0
	$\mid$	ϵ	0

If we allow $H(x, x')$ to be the free energy of basepair formation for xx' (and assume a zero energy cost for any structural feature except basepairs), then this becomes an **energy-attributed grammar**. Same recursion for S now calculates "ground state" energy for a very simple energy model ignoring all but hydrogen bonding terms for strictly nested basepairs.

The partition function for this highly simplified model is

$$\begin{aligned} Z(i,j) &= Z(i+1,j) + Z(i,j-1) + Z(i+1,j-1)\exp(-\beta H(x_i,x_j)) \\ &\quad + \sum_{i \le k \le j} Z(i,k)Z(k,j) \end{aligned}$$

where $\beta = 1/kT$ is an "inverse temperature". NB: have simply changed $H \to \exp(-\beta H)$, $+ \to \times$ and $\max \to +$ in equation for S

We can also have **probability-attributed grammars** or **stochastic grammars**. Relationship between scoring schemes:

Score	Form
Bayes	$P(x)$
Shannon	$h(x) = -\log_2 P(x)$
Boltzmann	$E(x) \sim -\log_e P(x)$

Specifically the Boltzmann probability uses a scaling factor (inverse temperature) and a partition function, $P(x) = \frac{1}{Z} \exp[-\beta E(x)]$, where $Z = \sum_x \exp[-\beta E(x)]$.

Energy-attributed grammar (Zuker, MFOLD). Parse tree for subsequence $X_i \ldots X_j$ must be rooted at W or $V_{X_i X_j}$.

LHS		RHS	Energy
W	$\longrightarrow$	$W\ x$	0
	$\mid$	$x\ W$	0
	$\mid$	$x\ V_{xy}\ y$	$\alpha(x, y)$
	$\mid$	$W\ W$	0
	$\mid$	$x\ y$	$\alpha(x, y)$
	$\mid$	ϵ	0
V_{ab}	$\longrightarrow$	z^n	$h(n)$
	$\mid$	$x\ V_{xy}\ y$	$\alpha_S(x, y \mid a, b)$
	$\mid$	$x\ V_{xy}\ y\ z^n$	$\alpha(x, y) + b(n)$
	$\mid$	$z^n\ x\ V_{xy}\ y$	$\alpha(x, y) + b(n)$
	$\mid$	$z^m\ x\ V_{xy}\ y\ z^n$	$\alpha(x, y) + i(m + n)$
	$\mid$	$x\ V_{xy}\ y\ W\ x'\ V_{x'y'}\ y'$	$\alpha(x, y) + \alpha(x', y')$
	$\mid$	ϵ	0

Here x, y, x', y', z_i are terminals (nucleotides), and z^n is shorthand for an n-nucleotide emission $z_1 \ldots z_n$.

The scoring scheme is as follows:

Free energy term	Meaning
$\alpha(x, y)$	Energy of basepair xy (end of stem, no stacking)
$\alpha_S(x, y \mid a, b)$	Energy of basepair xy stacked on top of basepair ab
$h(n)$	Hairpin loop enclosing n bases
$b(n)$	Asymmetric bulge of n bases
$i(n)$	Interior loop (symmetric bulge) with total of n bases

This scoring scheme was historically formulated in terms of Sankoff's "k-loop decomposition".

NB Zuker's algorithm (grammar), like Nussinov's, excludes pseudoknots.

Strictly, Zuker described the algorithm that finds the lowest-energy parse using the above grammar (CYK).

McCaskill described the algorithm for calculating the partition function (Inside) and thus the posterior probabilities of individual basepairs (Outside).

- Chomsky normal form. (RNA normal form is more useful in practise, but CNF is easier to present.)
  For nonterminals $A, B, C \in \Phi$ and terminals $a \in \Omega$:

Rule			Name
A	$\rightarrow$	$B\ C$	Bifurcation
	$\vert$	a	Emission
	$\vert$	ϵ	Termination

  Probabilities denoted by $P(\text{rule})$, e.g. $P(A \rightarrow BC)$

- Inside algorithm.
  Let $I_A(i,k) = P(x_i \ldots x_{i+k}|A)$ be sum of probabilities for parse trees rooted in A generating sequence $x_i \ldots x_{i+k}$.

$$I_A(i,k) = \left(\sum_B \sum_C \sum_{j=0}^{k} P(A \to BC) I_B(i,j) I_C(i+j, k-j) \right) + \left\{ \begin{array}{ll} 0 & \text{if } k > 1 \\ P(A \to x_i) & \text{if } k = 1 \\ P(A \to \epsilon) & \text{if } k = 0 \end{array} \right\}$$

  NB loopy dependencies, e.g. if $P(A \to AA) \neq 0$ and $P(A \to \epsilon) \neq 0$. It's common to try to avoid these when designing the grammar.

- Cocke-Younger-Kasami (CYK) algorithm.
  Let $Y_A(i,k)$ be probability of ML parse tree rooted in A generating sequence $x_i \ldots x_{i+k}$.

$$Y_A(i,k) = \max \left(\begin{array}{c} \max_B \max_C \max_{j=0}^{k} P(A \rightarrow BC) Y_B(i,j) Y_C(i+j,k-j), \\ \left\{ \begin{array}{ll} 0 & \text{if } k > 1 \\ P(A \rightarrow x_i) & \text{if } k = 1 \\ P(A \rightarrow \epsilon) & \text{if } k = 0 \end{array} \right\} \end{array} \right)$$

  NB similar to Nussinov.

- Outside algorithm.
  Let $O_A(i,k) = P(x_1 \dots x_{i-1}, A, x_{i+k+1} \dots x_L | S)$ be sum of probabilities for incomplete parse trees rooted in S, ending in A and generating outside sequence $x_1 \dots x_{i-1}$ and $x_{i+k+1} \dots x_L$.

$$O_A(i,k) = \sum_B \sum_C \left(\sum_{j=0}^{i} O_C(i-j, j+k) P(C \to BA) I_B(i-j, j) + \sum_{j=0}^{L-i-k} O_C(i, j+k) P(C \to AB) I_B(i+k, j) \right)$$

  with boundary condition $O_A(0,L) = \delta(A = S)$.

- Inside-Outside and posterior probabilities.
  Posterior probability that parse tree contains a subtree with inside sequence $x_i \ldots x_{i+k}$ rooted in state A:

$$P(A_{i,i+k}|X) = \frac{O_A(i,k)I_A(i,k)}{I_S(0,L)}$$

  Posterior probability of bifurcation $A_{i,j+k} \rightarrow B_{i,j}C_{i+j,k}$:

$$P(A_{i,j+k} \rightarrow B_{i,j}C_{i+j,k}|X) = \frac{O_A(i,j+k)P(A \rightarrow BC)I_B(i,j)I_C(i+j,k)}{I_S(0,L)}$$

  etc.

- Parameter estimation: we can write the parse tree likelihood as $\prod_i \theta_i^{n_i}$, where θ_i is the probability of rule i and n_i is the number of times it was applied. As with HMMs, the EM algorithm for SCFGs thus involves computing posterior expectations $\langle n_i \rangle$ for these counts, and setting $\theta_i \propto \langle n_i \rangle$. The posterior expectations are computed using the Inside-Outside algorithm, as shown for rules of the form $P(A \rightarrow BC)$.
- There is a KYC-like analogue to Outside, that can be used to find ML parse tree including a particular subsequence.
- Implementation issues: time complexity is $O(|\Phi|^3 L^3)$, memory is $O(|\Phi| L^2)$.
  - Cubic factors in time complexity arise due to bifurcations, so minimize number of bifurcations for max efficiency (c.f. RNA normal form).

Pair SCFGs, evolutionary SCFGs and tree transducers

- Evolutionary SCFGs: PFOLD (xfold, evofold, etc.)
  - As with Evolutionary HMMs, we can let the terminals be alignment columns
  - Again, terminal emission likelihood $P(A \rightarrow a)$ is implemented as Felsenstein pruning

- Pair SCFGs: Evoldoer. Version of TKF that describes evolution of RNA secondary structure (Holmes 2005).
  - Two kinds of TKF91 links model, recursively nested in a tree
  - Stem sequences rooted in S nonterminals; basepair alphabet Ω^2; ends in an L
  - Loop sequences rooted in L nonterminals; nucleotide alphabet Ω; S's also allowed in sequence
    - Really need to adapt TKF91 model to allow deletion prob to depend on symbol, otherwise S substructures get deleted at same rate as nucleotides
- Pair SCFGs (e.g. Stemloc, QRNA). Heuristic, but with lots of go-faster stripes (pre-aligning & pre-folding sequences).
- Rules for composing Pair SCFGs exist (c.f. string transducers; can view conditionally normalized pair SCFGs as "tree transducers").

Graph grammars

- Tree-adjoining grammars
  - Aravind Joshi (1975, 1985)
- Rivas-Eddy papers
  - A dynamic programming algorithm for RNA structure prediction including pseudoknots. JMB 1999.
  - The language of RNA: a formal grammar that includes pseudoknots. Bioinformatics 2000.
- Graph grammars: easy to describe and simulate, attractive for biology; but how does their DP work?
- Other grammars whose marginals are easy to compute by sum-product DP, e.g.
  - stochastic tree grammars (Abe and Mamitsuka, ISMB 1994)
  - context-sensitive HMMs with finite memory of last N emitted characters (Yoon and Vaidyanathan, 2004)

Discriminative grammars

Discriminative grammars: conditional log-linear models

- Recall HMMs and linear CRFs form a "generative-discriminative pair"
- The analogous discriminative model for SCFGs is a Conditional Log-Linear Model
- This allows a great deal of physics-like parameterization (loop entropies, stacking free energies, terminal mismatch...) without having to introduce many new nonterminals
  - Do, Woods and Batzoglou. "CONTRAfold: RNA secondary structure prediction without physics-based models." Bioinformatics 22:14, pp e90-e98

Summary

- SCFGs