

Grammars and Biological Sequences

Rune Lyngsø

21st of November 2011

Outline

Hidden Markov Models

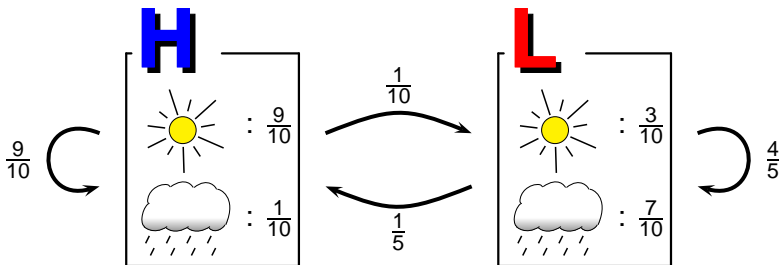
Grammars

Context Free Grammars

The Rest of the Grammars

Hidden Markov Models

Many phenomena can be modelled by an underlying network of hidden states from which observables are emitted with different distributions



States of the Markov chain are not observed directly – observed characters are emitted from the hidden chain of states

Maintaining the State of an HMM

We can maintain progress of an HMM as pair:

$$(\epsilon, \mathbf{H}) \Rightarrow (\odot, \mathbf{H}) \Rightarrow (\odot\odot, \mathbf{L}) \Rightarrow (\odot\odot\text{☁}, \mathbf{H}) \Rightarrow \dots$$

We could also just keep the state at the end of the sequence:

$$\mathbf{H} \Rightarrow \odot\mathbf{H} \Rightarrow \odot\odot\mathbf{L} \Rightarrow \odot\odot\text{☁}\mathbf{H} \Rightarrow \dots$$

At each update we replace a state with an observation and a new state

Fundamentals of Grammars

A formal grammar consists of

- A set of variables \mathbf{V} , e.g. $\{\mathbf{H}, \mathbf{L}\}$
- A set of terminal symbols Σ , e.g. $\{\odot, \text{☁}\}$
- A set of productions \mathbf{P} of the form $x \rightarrow y$ where $x, y \in (\mathbf{V} \cup \Sigma)^*$ and x contains at least one element from \mathbf{V} , e.g. $\mathbf{H} \rightarrow \odot \mathbf{H} \in P$
- A special start variable $S \in \mathbf{V}$, e.g. $S = \mathbf{H}$

Derivations start from the single start variable and keeps applying a suitable production until there are no variables left in the string

Stochastic Grammars

Basics

- Probability associated with each production
- Probabilities of productions replacing variable sums to 1

Example: Grammar probabilities from weather HMM

$$Pr(\mathbf{H} \rightarrow \odot \mathbf{L}) = Pr(\mathbf{H} \rightarrow \mathbf{L}) \cdot Pr(\odot \mid \mathbf{L})$$

Probability of a...

- Derivation is product of probabilities of productions applied
- String is sum of probabilities of derivations generating it

Variations of Regular Grammars

Left regular: productions $U \rightarrow \sigma V$ or $U \rightarrow \sigma$ ($U, V \in \mathbf{V}$ and $\sigma \in \Sigma$)

Extended left regular: productions $U \rightarrow xV$ or $U \rightarrow x$ ($U \in \mathbf{V}$ and $x \in \Sigma^*$)

Right regular: productions $U \rightarrow V\sigma$ or $U \rightarrow \sigma$ ($U, V \in \mathbf{V}$ and $\sigma \in \Sigma$)

Extended right regular: productions $U \rightarrow Vx$ or $U \rightarrow x$ ($U \in \mathbf{V}$ and $x \in \Sigma^*$)

Stochastic Regular Grammars and HMMs

The two formalisms are equivalent in the type of distributions they can describe

One can argue that in their strict formulation, SRGs do not allow separation of transitions and emissions

Context Free Grammars

All productions on the form $U \rightarrow x$ where $U \in \mathbf{V}$ and $x \in (\mathbf{V} \cup \Sigma)^*$

Generating sequences $a^i g^i, i \geq 1$

$$S \rightarrow aSg \mid ag$$

Derivation of sequence aagg:

$$S \Rightarrow aSg \Rightarrow aagg$$

Another Example – Parentheses

Balanced string of parentheses consist of

- An empty string
- A balanced string enclosed between a (and a)
- Two balanced strings concatenated

$$S \rightarrow \epsilon \mid (S) \mid SS$$

String $()()$ can be derived e.g. as

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

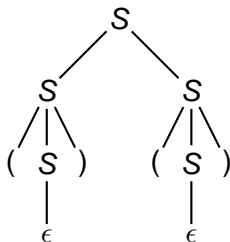
$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()()$$

Parse Trees

An alternative to derivation sequences are parse trees.
For example

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()()$$

would be represented by the parse tree



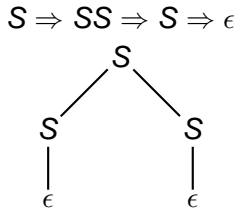
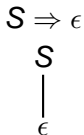
Ambiguous Grammars

Context-free grammars introduce choice of variable to apply production to:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()()$$

These share the same parse tree we have already seen



Distinct parse trees

Chomsky Normal Form

A grammar is in Chomsky Normal Form (CNF) if all productions are

- $S \rightarrow \epsilon$
- $U \rightarrow \sigma, U \in \mathbf{V}$ and $\sigma \in \Sigma$
- $U \rightarrow VW, U \in \mathbf{V}$ and $V, W \in \mathbf{V} \setminus \{S\}$

Any context-free grammar can be converted to normal form

Rewriting $S \rightarrow \epsilon \mid (S) \mid SS$

$S \rightarrow A, A \rightarrow \epsilon \mid (A) \mid AA$

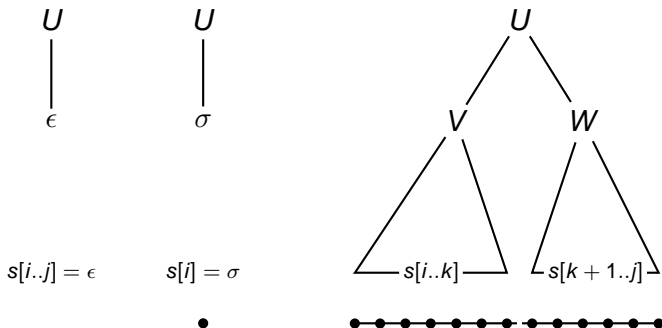
$S \rightarrow A \mid \epsilon, A \rightarrow (A) \mid () \mid AA \mid A$

$S \rightarrow A \mid \epsilon, A \rightarrow BAC \mid BC \mid AA \mid A, B \rightarrow (, C \rightarrow)$

$S \rightarrow A \mid \epsilon, A \rightarrow BD \mid BC \mid AA \mid A, B \rightarrow (, C \rightarrow), D \rightarrow AC$

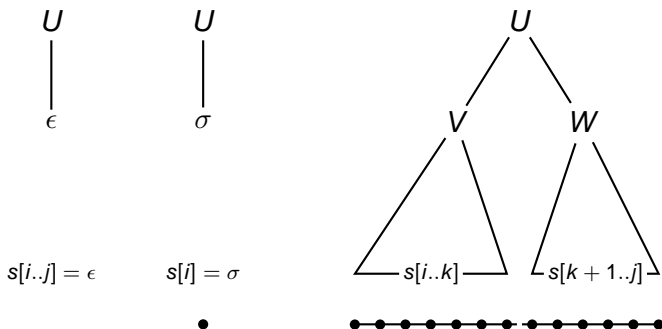
$S \rightarrow \epsilon \mid BD \mid BC \mid AA, A \rightarrow BD \mid BC \mid AA, B \rightarrow (, C \rightarrow), D \rightarrow AD$

Cocke-Younger-Kasami Algorithm



Dynamic programming once again allows efficient computation (though cubic in sequence length, compared to linear for HMMs)

Inside Algorithm

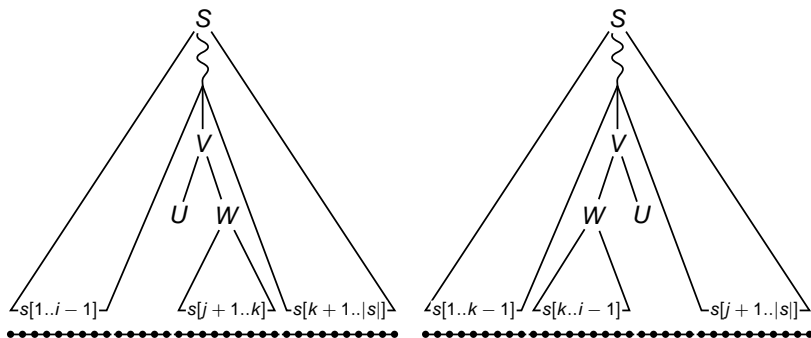


$$I(U, i, j) = \Pr(S \rightarrow \epsilon) \text{ if } U = S \text{ and } j = i - 1$$

$$I(U, i, j) = \Pr(U \rightarrow s[i]) \text{ if } i = j$$

$$I(U, i, j) = \sum_{\substack{U \rightarrow VW \in P \\ i \leq k < j}} \Pr(U \rightarrow VW) I(V, i, k) I(W, k + 1, j) \text{ if } i < j$$

Outside Algorithm



$$O(U, i, j) = 1 \text{ if } U = S, i = 1, \text{ and } j = |s|$$

$$O(U, i, j) = \sum_{\substack{V \rightarrow UW \in P \\ k > j}} \Pr(V \rightarrow UW) O(V, i, k) I(W, j+1, k)$$

$$+ \sum_{\substack{V \rightarrow WU \in P \\ k < i}} \Pr(V \rightarrow WU) O(V, k, j) I(W, k, i-1) \text{ if } i \leq j$$

Parameter Estimation

Annotated Data

Each production probability is set to the frequency with which it is used to replace the variable on the its left hand side

Unannotated Data

Instead of observed frequencies, expected frequencies are computed using the inside/outside algorithms

In both cases a small pseudocount may be added to observed/expected counts

Bioinformatics Example I: RNA Secondary Structures

Two bases introduced by the same production are paired, all other are unpaired

$S \rightarrow L$	one structural component
$ LS$	at least two structural components
$L \rightarrow dFd$	beginning of stem
$ s$	unpaired base
$F \rightarrow dFd$	continuation of stem
$ LS$	end of stem

Despite its simplicity works reasonably well for a single sequence, and is very powerful for alignments of RNA (Knudsen & Hein, 2003; Dowell & Eddy, 2004)

RNA Secondary Structures, continued

How do we design the best possible grammar?

$$S \rightarrow dSd|sS|Ss|SS|\epsilon \quad 17(12)$$

$$S \rightarrow sS|T|\epsilon; T \rightarrow Ts|dSd|TdSd \quad 10(8)$$

$$S \rightarrow LS|L; L \rightarrow dFd|s; F \rightarrow dFd|LS \quad \mathbf{47(45)}$$

$$S \rightarrow sS|dSdS|\epsilon \quad 3(4)$$

$$S \rightarrow dSd|sL|Rs|LS; L \rightarrow dSd; R \rightarrow Rs|\epsilon \quad 34(31)$$

Not obvious what makes a good grammar

Bioinformatics Example II: Alignment with Inversions

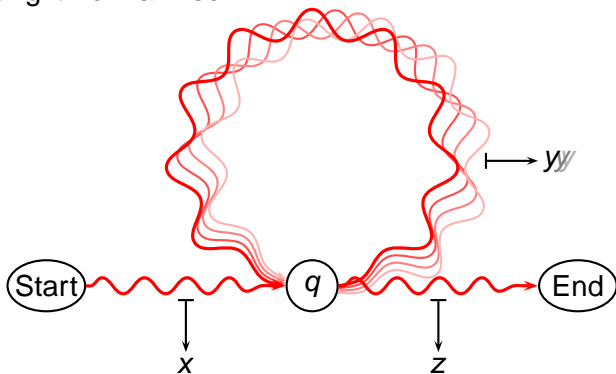
In addition to substitutions, insertions and deletions, also allow inversions:

$$\begin{array}{c}
 \begin{array}{cc}
 \bullet & \xrightarrow{\text{blue}} \bullet \\
 k & \quad \quad \quad l \\
 \bullet & \xrightarrow{\text{green}} \bullet \\
 i & \quad \quad \quad j
 \end{array} \\
 A(i, j; k, l)
 \end{array}
 = \min \left\{ \begin{array}{l}
 \min \left\{ 1 + \begin{array}{cc}
 \bullet & \xrightarrow{\text{blue}} \bullet \\
 k & \quad \quad \quad l \\
 \bullet & \xrightarrow{\text{green}} \bullet \\
 i & \quad \quad \quad j-1
 \end{array}, 1 + \begin{array}{cc}
 \bullet & \xrightarrow{\text{blue}} \bullet \\
 k & \quad \quad \quad l-1 \\
 \bullet & \xrightarrow{\text{green}} \bullet \\
 i & \quad \quad \quad j
 \end{array}, \right. \\
 \left. \delta(s[i], s[j]) + \begin{array}{cc}
 \bullet & \xrightarrow{\text{blue}} \bullet \\
 k & \quad \quad \quad l-1 \\
 \bullet & \xrightarrow{\text{green}} \bullet \\
 i & \quad \quad \quad j-1
 \end{array} \right\} \\
 \min_{a,b} \left\{ 1 + \begin{array}{cc}
 \bullet & \xrightarrow{\text{blue}} \bullet \\
 k & \quad \quad \quad b \\
 \bullet & \xrightarrow{\text{green}} \bullet \\
 i & \quad \quad \quad a
 \end{array} + \begin{array}{cc}
 \bullet & \xleftarrow{\text{blue}} \bullet \\
 l & \quad \quad \quad b+1 \\
 \bullet & \xrightarrow{\text{green}} \bullet \\
 a+1 & \quad \quad \quad j
 \end{array} \right\}
 \end{array} \right.$$

Can be captured by a context-free grammar (but not by an HMM)

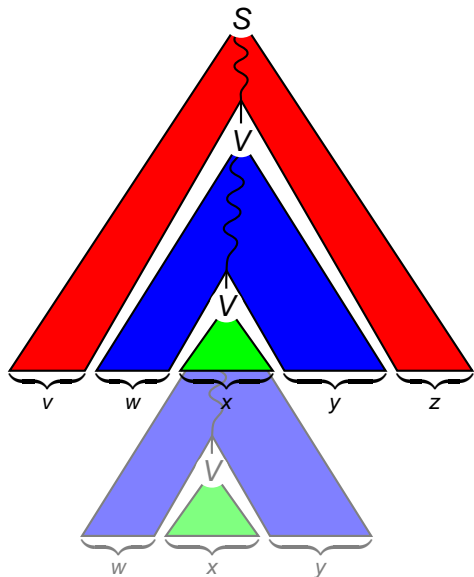
Limitations of Hidden Markov Models

For sufficiently long sequences, there will be a cycle in any path generating it from a fixed HMM



Consequence: Cannot generate e.g. palindromes and $\{\text{☀}^i \text{☁}^i \mid i > 0\}$

Limitations of Context-Free Grammars



When generating sufficiently long strings, there is a variable that is visited twice from root to a leaf

Cannot generate
 e.g. $\{xx \mid x \in \Sigma^*\}$ or
 $\{a^i b^i c^i \mid i > 0\}$

Rest of the Chomsky Hierarchy

Context Sensitive (Non-Contracting) Grammars

- Productions $\alpha V \beta \rightarrow \alpha x \beta$ with $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$ and $x \neq \epsilon$
- Equivalent to $\alpha V \beta \rightarrow x$ with $|\alpha V \beta| \leq |x|$

Analyse s by considering all sequences not longer than s
 CSG for $\{xx \mid x \in \{a, b\}^*\}$:

$$S \rightarrow aAS \mid bBS \mid C$$

$$Aa \rightarrow aA$$

$$Ab \rightarrow bA$$

$$Ba \rightarrow aB$$

$$Bb \rightarrow bB$$

$$AC \rightarrow CA$$

$$BC \rightarrow CB$$

$$CA \rightarrow aC$$

$$CB \rightarrow bC$$

$$C \rightarrow \epsilon$$

Rest of the Chomsky Hierarchy

Context Sensitive (Non-Contracting) Grammars

- Productions $\alpha V \beta \rightarrow \alpha x \beta$ with $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$ and $x \neq \epsilon$
- Equivalent to $\alpha V \beta \rightarrow x$ with $|\alpha V \beta| \leq |x|$

Analyse s by considering all sequences not longer than s

Unrestricted Grammar

- Productions $\alpha V \beta \rightarrow x$ with $\alpha, \beta, x \in (\mathbf{V} \cup \Sigma)^*$

Cannot guarantee that s can be analysed in general

Chomsky Hierarchy

Proposed by Noam Chomsky in 1956 – consists of Regular, Context Free, Context Sensitive, and Unrestricted Grammars

Other Suspects

Linear Grammars

- Productions $U \rightarrow \sigma$, $U \rightarrow V\sigma$, and $U \rightarrow \sigma V$

Can generate palindromes:

$$S \rightarrow aA \mid bB \mid \epsilon$$

$$A \rightarrow Sa$$

$$B \rightarrow Sb$$

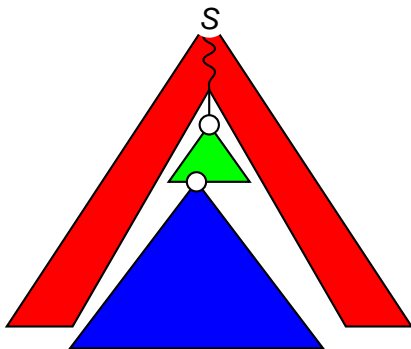
Other Suspects

Linear Grammars

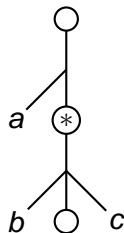
- Productions $U \rightarrow \sigma$, $U \rightarrow V\sigma$, and $U \rightarrow \sigma V$

Tree Adjoining Grammars

Tree adjoining grammars can expand at edges

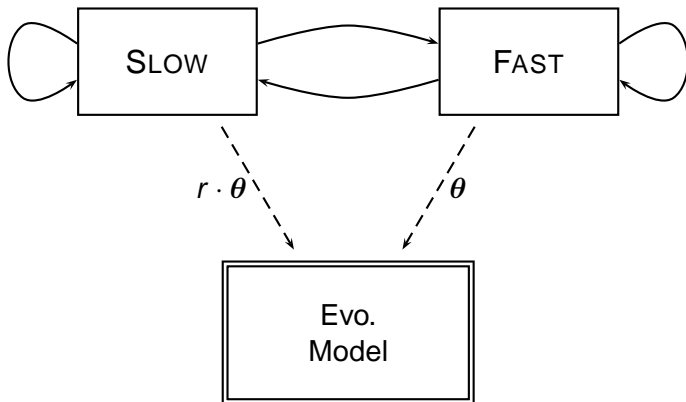


Tree for $\{a^i b^j c^i \mid i > 0\}$



Hierarchical Modelling – Example I

Conserved regions in a genome



Hierarchical Modelling – Example II

Conserved RNA Secondary Structure

$$S \rightarrow L \mid LS, L \rightarrow dFd \mid s, F \rightarrow dFd \mid LS$$



((((((. ((((((.)))))))
 GGUUA - **A**GGCAU**U**GCCACCUACUUCGUGGC**A** - - - UC
 UGGUUA**A**GGCAU**U**GCCACCUACUUCGUGGC**A**UCUCU
 GGUCG - - - - - **G**CUAGGCACAGACAAAAG**C** - - - - -
 GGUU - **A**GGCUC**U**CCACA - UUCG - - UGUGG**A**UCUA -

Single Nuc.
Evo. Model

Paired
Evo. Model