

Phylogenetic trees

Likelihood computations and inference

I. Holmes

Department of Bioengineering
University of California, Berkeley

Spring semester

Outline

- 1 Likelihoods
- 2 Posterior probabilities
- 3 Factor graphs

Tree Notation

Notation and model: let...

x_n	be (actual) state of node n
y_n	be (observed) character at leaf node n
$X = \{x_n\}$	be the set of all node states
$Y = \{y_n\}$	be the set of all observed characters at leaf nodes
Y_n	be the set of all y_m descended from node n
$\overline{Y}_n = Y \setminus Y_n$	be the set of all remaining y_m
t_{mn}	be evolutionary “distance” (time) from m to n

Root node is node 1.

For convenience, number nodes with parents < kids

Conditional & Prior Probabilities

Parent x_p , child x_c :

$$P(x_c|x_p) = M(t_{pc})_{x_px_c} = \exp(\mathbf{R}t_{pc})_{x_px_c}$$

Leaf node c :

$$P(y_c|x_c) = \delta_{x_cy_c}$$

i.e. observation y_c fully specifies state x_c . We can drop the x_c :

$$P(y_c|x_p) = \sum_{x_c} P(x_c|x_p)P(y_c|x_c) = M(t_{pc})_{x_py_c}$$

Assumption: Ur-ancestor was in evolutionary equilibrium

$$P(x_1) = \pi_{x_1}$$

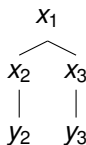
(Root node is always node 1)

Felsenstein's Algorithm: Two Taxa

- Felsenstein's pruning algorithm: our first DP
 - Consider tree T_1 : 1 with the following dependencies



between state variables X and observations Y :



- Joint likelihood of all nodes:

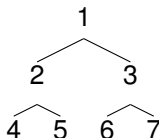
$$P(X, Y) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(y_2|x_2)P(y_3|x_3)$$

- Marginal likelihood of leaves:

$$\begin{aligned} P(Y) &= \sum_X P(X, Y) \\ &= \sum_{x_1} P(x_1)P(y_2|x_1)P(y_3|x_1) \end{aligned}$$

Felsenstein's Algorithm: Four Taxa

Now consider tree T_2 :



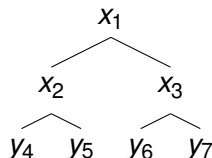
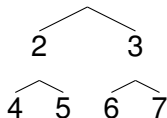
Joint likelihood of all nodes:

$$P(X, Y)$$

$$= P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2)P(x_5|x_2)P(x_6|x_3)P(x_7|x_3)\prod_{n=4}^7 P(y_n|x_n)$$

Felsenstein's Algorithm: Rearranging Terms

Tree T_2 : 1 and model



Marginal likelihood of leaves:

$$\begin{aligned}
 P(Y) &= \sum_X P(X, Y) \\
 &= \sum_{x_1} \sum_{x_2} \sum_{x_3} P(x_1) P(x_2|x_1) P(x_3|x_1) P(y_4|x_2) P(y_5|x_2) P(y_6|x_3) P(y_7|x_3) \\
 &= \sum_{x_1} P(x_1) \left[\left(\sum_{x_2} P(x_2|x_1) [P(y_4|x_2) P(y_5|x_2)] \right) \left(\sum_{x_3} P(x_3|x_1) [P(y_6|x_3) P(y_7|x_3)] \right) \right]
 \end{aligned}$$

Terms in square brackets have the form:

$$F_n(x_n) = P(\{y_i : i \text{ descended from } n\} | x_n) = P(Y_n | x_n)$$

Felsenstein's Algorithm: Generalization

- Let C_n be the set of immediate children of n (so e.g. $C_1 = \{2, 3\}$)
- Rule to compute F_n is then

$$F_n(x_n) = \begin{cases} \prod_{c \in C_n} \left(\sum_{x_c} P(x_c | x_n) F_c(x_c) \right) & \text{if } n \text{ is internal} \\ P(y_n | x_n) & \text{if } n \text{ is a leaf} \end{cases}$$
$$P(Y) = \sum_{x_1} P(x_1) F_1(x_1)$$

- This is the **pruning** algorithm (Felsenstein, 1981).

Felsenstein's Algorithm: Recursion

$$F_n(x_n) = \begin{cases} \prod_{c \in C_n} \left(\sum_{x_c} P(x_c | x_n) F_c(x_c) \right) & \text{if } n \text{ is internal} \\ P(y_n | x_n) & \text{if } n \text{ is a leaf} \end{cases}$$
$$P(Y) = \sum_{x_1} P(x_1) F_1(x_1)$$

- Term after “ \prod ” sign in F_n can be written $E_c(x_p) = P(Y_c | x_p) = \sum_{x_c} P(x_c | x_p) F_c(x_c)$ where p is parent of c
- An instance of the **sum-product algorithm** on a factor graph

Pulley Principle

- “Pulley principle” for reversible models.



- Using $\pi_i M(t)_{ij} = \pi_j M(t)_{ji}$ and $\mathbf{M}(t)\mathbf{M}(t') = \mathbf{M}(t + t')$

$$\begin{aligned}
 P(y_2, y_3) &= \sum_{x_1} \pi_{x_1} M(t_{12})_{x_1 y_2} M(t_{13})_{x_1 y_3} \\
 &= \sum_{x_1} \pi_{y_2} M(t_{12})_{y_2 x_1} M(t_{13})_{x_1 y_3} \\
 &= \pi_{y_2} M(t_{12} + t_{13})_{y_2 y_3} \\
 &= \pi_{y_3} M(t_{12} + t_{13})_{y_3 y_2}
 \end{aligned}$$

- $P(y_2, y_3)$ depends only on $t_{12} + t_{13}$
- Can slide root node (like a pulley) w/out affecting likelihood
- Corollary: can re-root tree at any node, including any leaf

Alignment Probability

- So far we have looked at $P(C|T)$, the probability of an individual alignment column, conditioned on a tree
- Probability of an entire alignment conditioned on tree, $P(A|T)$, is product of column probabilities:

$$P(A|T) = \prod_{C \in A} P(C|T)$$

- Denote maximum likelihood tree by $T_{\text{ML}} = \operatorname{argmax}_T P(A|T)$
- Note that so far we have neglected the indel history that is also (partially) specified by the alignment

Ancestral State Reconstruction on a Phylogeny

- Motivation:
 - Probability distribution of ancestral state, $P(x_n | Y)$
 - Probability distribution of $p \rightarrow c$ branch, $P(x_p, x_c | Y)$
- Recall tree notation:
 - $Y = \{y_i\}$ is the set of all leaf states
 - Y_n contains all y_i descended from node n
 - \overline{Y}_n contains all y_i **not** descended from node n
 - Note $Y_1 \equiv Y$, since node 1 is the root node.

Probabilities of Ancestral States

- Felsenstein's pruning algorithm computes $F_n(x_n) = P(Y_n | x_n)$
 - and thus $P(Y) = \sum_{x_1} \pi(x_1) F_1(x_1)$
- We will now give recursions for $G_n(x_n) = P(x_n, \overline{Y_n})$.
- With these we can easily get posterior probabilities for...
 - State of ancestral node
 - State of ancestral branch

State of Ancestral Node

$$F_n(x_n) = P(Y_n | x_n)$$

$$G_n(x_n) = P(x_n, \overline{Y_n})$$

$$\begin{aligned} P(x_n | Y) &= \frac{P(x_n, Y)}{P(Y)} \\ &= \frac{1}{P(Y)} P(x_n, \overline{Y_n}) P(Y_n | x_n) \\ &= \frac{1}{P(Y)} G_n(x_n) F_n(x_n) \end{aligned}$$

State of Ancestral Branch

- Joint state of ancestral branch



$$\begin{aligned} P(x_p, x_c | Y) &= \frac{P(x_p, x_c, Y)}{P(Y)} \\ &= \frac{1}{P(Y)} P(x_p, \overline{Y_p}) P(x_c | x_p) P(Y_c | x_c) P(Y_s | x_p) \\ &= \frac{1}{P(Y)} G_p(x_p) P(x_c | x_p) F_c(x_c) E_s(x_p) \end{aligned}$$

where $E_s(x_p) = \sum_{x_c} P(x_c | x_p) F_s(x_s)$ as before

Peeling Recursion

- Recursion for $G_c(x_c) = P(x_c, \overline{Y_c})$

$$G_c(x_c) = \begin{cases} \sum_{x_p} G_p(x_p) P(x_c | x_p) E_s(x_p) & \text{if } c > 1 \text{ (not root)} \\ P(x_c) & \text{if } c = 1 \text{ (root)} \end{cases}$$

- aka Elston-Stewart peeling algorithm

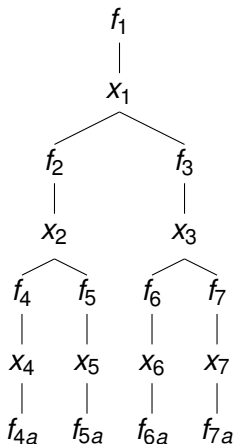
Degenerate characters

IUPAC ambiguity code	Possibilities
R	A G
Y	C T
M	A C
K	G T
S	C G
W	A T
H	A C T
B	C G T
V	A C G
D	A G T
N or X	A C G T

- Recall the hidden state x_i behind observation y_i
- $P(y_i|x_i) = A_{x_i y_i}$ reflects ambiguous y_i
- Can be extended to read quality scores, etc.
- $P(X, Y)$ is product of π_i 's, M_{ij} 's and A_{ij} 's

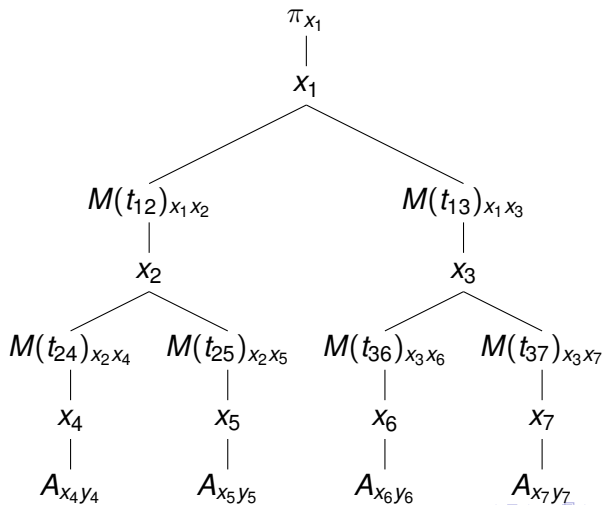
Bigraph representation of $P(X, Y)$

2-coloring: variables $X = \{x_i\}$, functions $\mathcal{F} = \{f_n\}$

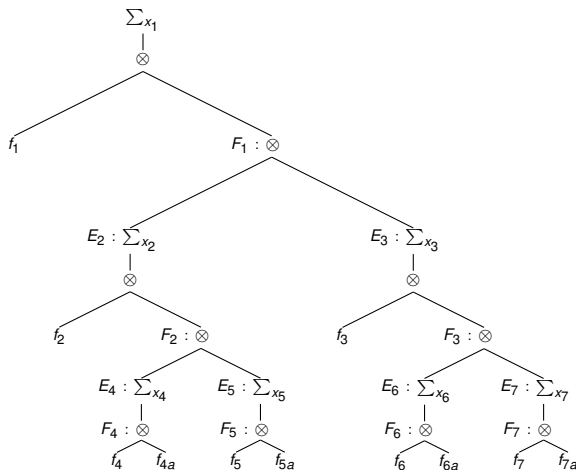


$$\begin{aligned}
 f_1(x_1) &= \pi_{x_1} \\
 f_2(x_1, x_2) &= M(t_{12})_{x_1 x_2} \\
 f_3(x_1, x_3) &= M(t_{13})_{x_1 x_3} \\
 f_4(x_2, x_4) &= M(t_{24})_{x_2 x_4} \\
 f_5(x_2, x_5) &= M(t_{25})_{x_2 x_5} \\
 f_6(x_3, x_6) &= M(t_{36})_{x_3 x_6} \\
 f_7(x_3, x_7) &= M(t_{37})_{x_3 x_7} \\
 f_{4a}(x_4) &= A_{x_4 y_4} \\
 f_{5a}(x_5) &= A_{x_5 y_5} \\
 f_{6a}(x_6) &= A_{x_6 y_6} \\
 f_{7a}(x_7) &= A_{x_7 y_7}
 \end{aligned}$$

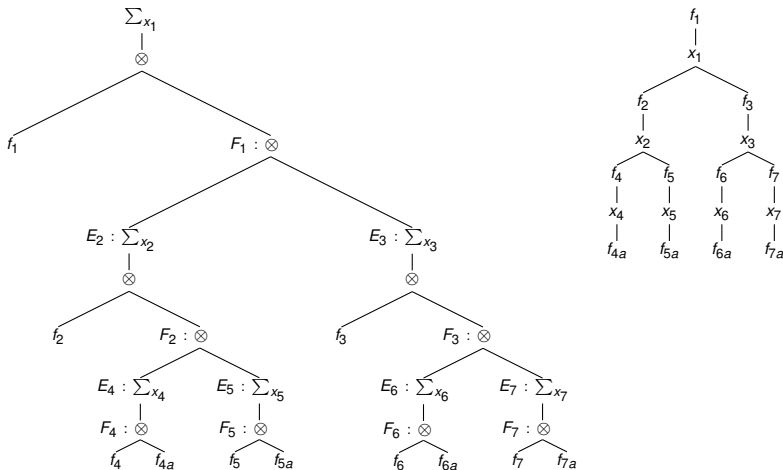
$$\begin{aligned}
 P(x_1 \dots x_7; y_4 \dots y_7) &= f_1(x_1) f_2(x_1, x_2) f_3(x_1, x_3) \\
 &\quad \times f_4(x_2, x_4) f_5(x_2, x_5) f_6(x_3, x_6) f_7(x_3, x_7) \\
 &\quad \times f_{4a}(x_4) f_{5a}(x_5) f_{6a}(x_6) f_{7a}(x_7)
 \end{aligned}$$

Bigraph representation of $P(X, Y)$ showing functions

Expression tree of $P(Y)$



Expression tree of $P(Y)$ and factor graph of $P(X, Y)$



Factor Graphs

- The idea of computing marginals and posteriors on a graphical network of variables has been generalized
 - “**Factor graphs**”, “Graphical models”, “Bayesian networks”, “Markov random fields”
 - “**Sum-product algorithm**”, “Message-passing”, “Belief propagation”
- Here we use the Factor Graph variant of the formalism
 - Kschichang, Frey & Loeliger, IEEE 47:2, February 2001.

The summary operator

- The “summary” or “not-sum” operator, $\sum_{\sim\{\}} f(\cdot)$
 - Suppose $V \subseteq W \subseteq X$.
 - The **summary of $f(W)$ w.r.t. V** is obtained by summing $f(W)$ over all $x_i \notin V$. The result is a function of V .
 - For example...

$$\sum_{\sim\{x_3, x_4\}} f(x_2, x_3, x_4, x_5, x_6) = \sum_{x_2} \sum_{x_5} \sum_{x_6} f(x_2, x_3, x_4, x_5, x_6)$$

- Define the summary of f w.r.t. its own arguments as f itself:

$$\sum_{\sim\{W\}} f(W) \equiv f(W)$$

- **The summaries of a joint likelihood are the marginals:**
If $g(X) = P(X)$, then $\sum_{\sim\{V\}} g(V) = P(V)$

Message-passing

- Factor graphs contain
 - **function nodes** (f_i)
 - **variable nodes** (x_i)
 - An edge $f_i - x_j$ indicates x_j is a parameter of f_i
- Computation proceeds by **message-passing**
- A “message” is actually a pre-computed function over some subset of the variables in the graph
- The message from node a to node b is written $\mu_{a \rightarrow b}$

Message-passing

- Message from f_i to x_j :

$$\mu_{f_i \rightarrow x_j} = \sum_{\sim \{x_j\}} f_i \prod_{k \neq j} \mu_{x_k \rightarrow f_i}$$

where $\{\mu_{x_k \rightarrow f_i}\}$ are the incoming messages to f_i

- Message from x_i to f_j :

$$\mu_{x_i \rightarrow f_j} = \prod_{k \neq j} \mu_{f_k \rightarrow x_i}$$

where $\{\mu_{f_k \rightarrow x_i}\}$ are the incoming messages to x_i

Message-passing

$$\mu_{f_i \rightarrow x_j} = \sum_{\sim \{x_j\}} f_i \prod_{k \neq j} \mu_{x_k \rightarrow f_i}$$
$$\mu_{x_i \rightarrow f_j} = \prod_{k \neq j} \mu_{f_k \rightarrow x_i}$$

The message sent from a node n on an edge e is...

- the product of:
 - 1 the local function at n
(or the unit function, $x_n \rightarrow 1$, if n is a variable node);
 - 2 all messages received at n on edges *other* than e ,
- summarized for the variable associated with e .

Phylogenetic message-passing

For a phylogenetic tree, the messages are as follows
($c - p - s$ denoting child—parent—sibling)

- Leaves-to-root:

$$\mu_{f_{na} \rightarrow x_n} = f_{na}(x_n)$$

$$\mu_{x_n \rightarrow f_n} = F_n(x_n)$$

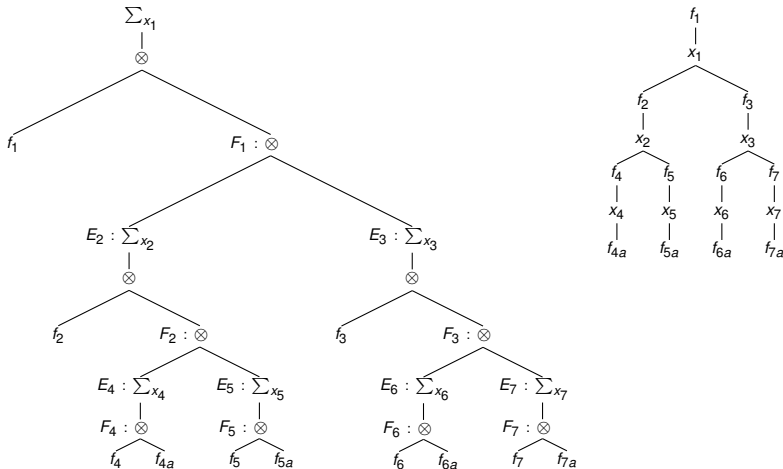
$$\mu_{f_c \rightarrow x_p} = E_c(x_p)$$

- Root-to-leaves:

$$\mu_{f_n \rightarrow x_n} = G_n(x_n)$$

$$\mu_{x_p \rightarrow f_c} = G_p(x_p)E_s(x_p)$$

Expression tree of $P(Y)$ and factor graph of $P(X, Y)$



Factor Graphs

- Many applications in biology...
 - Markov models
 - Phylogenetic trees
 - Hidden Markov models
 - Inference on ontologies
 - Markov random fields (images, gene networks, ...)
- ...and beyond...
 - Error-correcting codes

Changing the Operators

- The “sum” and “product” operators in the sum-product algorithm are arbitrary
 - The sum-product algorithm works by factorising terms like

$$\sum_{x_1} \sum_{x_2} f(x_1) f(x_2) = \left(\sum_{x_1} f(x_1) \right) \left(\sum_{x_2} f(x_2) \right)$$

- This in turn relies on the distributive law,

$$a \times (b + c) = a \times b + a \times c$$

and can be applied to any semiring with operators (\otimes, \oplus)

Changing the Operators: ML inference

- For example, consider

$$a \oplus b \equiv \max(a, b)$$

$$a \otimes b \equiv ab$$

- The pruning algorithm now calculates $\max_X P(X, Y)$
- It is straightforward to recover $\operatorname{argmax}_X P(X, Y)$
 - i.e. the MAP (**Maximum A Posteriori**) ancestral state history
 - note that $\operatorname{argmax}_X P(X, Y) = \operatorname{argmax}_X P(X|Y)$
 - this may not give the same results as $\operatorname{argmax}_{x_i} P(x_i|Y)$

Changing the Operators: Parsimony

- Another example: consider

$$a \oplus b \equiv \min(a, b)$$

$$a \otimes b \equiv a + b$$

and set $M(t)_{ij} = A_{ij} = 1 - \delta_{ij}$ and $\pi_i = 0$

- Then M counts the number of substitutions, and pruning returns the **most parsimonious** imputation of $\{x_i\}$
- A number of neat results for parsimony exist
 - e.g. a branch-and-bound algorithm to find the most parsimonious tree
- but parsimony is v restrictive “model”; strong assumptions:
 - all substitutions equally likely, no back-substitutions...
 - parsimony not really compatible w/ probabilistic methods

Summary

- Message-passing on phylogenetic trees
 - Felsenstein's pruning algorithm
 - Elston-Stewart peeling algorithm
- Pulley principle & reversibility