Phylogeny
Felsenstein’s Maximum Likelihood Approach

Rolf Backofen
Lehrstuhl für Bioinformatik
Institut für Informatik
Albert-Ludwigs-Universität Freiburg

Course Bioinformatics I
built on January 15, 2019
Maximum Likelihood Trees

- Method originally introduced by Felsenstein (1981)
- **So far:** given distances ⇒ clustering between data
- **Now:** directly build tree *with* associated distances
- **Problem:** “scoring” of trees (set of sequences).

- 3 subtasks: 1.) scoring in general
  2.) required for scoring: Markov process
  3.) scoring *en detail*
1.) Scoring in General

- Calculate the likelihood of data, given tree
  \[ \text{data} = \text{sequences} + \text{multiple alignment} \]

- Wanted: likelihood of tree:
  \[ L(\text{tree}) = Pr[\text{data} | \text{tree}] \]

- **Example:** consider tree with inner nodes filled

  ![Tree Diagram](tree_diagram.png)

  \[ p_{\text{AC} \rightarrow \text{CA}}(d_1) \cdot p_{\text{AC} \rightarrow \text{AT}}(d_2) \cdot p_{\text{CA} \rightarrow \text{CT}}(d_3) \cdot p_{\text{CA} \rightarrow \text{CG}}(d_4) \]

- Assume given probabilities for transitions
  \[ p_{\text{AC} \rightarrow \text{CA}}(d) \ldots \]

- **Then tree likelihood can be calculated via product of edges**

- **Later:**
  - without filled inner nodes ⇒ can be solved by DP
  - estimation of branch length
  - finding the best topology
2.) How to get $p_{AC \to CA}(d)$?

⇒ Markov Process

- **First:** split long sequences into single-position events

\[
Pr\left[\begin{array}{c|cc}
\text{data} & d_1 & d_2 \\
\text{CA} & d_3 & d_4 \\
C & T & C \\
\end{array}\right] = Pr\left[\begin{array}{c|cc}
\text{data} & d_1 & d_2 \\
\text{C} & d_3 & d_4 \\
A & T & G \\
\end{array}\right] \times Pr\left[\begin{array}{c|cc}
\text{data} & d_1 & d_2 \\
\text{C} & d_3 & d_4 \\
A & T & G \\
\end{array}\right]
\]

⇒ assume independence of positions

- **Hence:** instead of $p_{AC \to CA}(d)$
  we have only to define

\[p_{A \to A}(d), \; p_{A \to C}(d), \; \ldots\]

reminds of Markov chains

\[
\begin{align*}
\Delta t & \to A \\
\Delta t & \to A \\
\Delta t & \to C \\
\Delta t & \to C \\
\end{align*}
\]

\[
\sum \Delta t = d
\]
Markov Process and Markov Chain

- Markov process: basically $\lim \Delta t \to 0$ for a Markov chain covering time period $\Delta t \Rightarrow \lim_{\Delta t \to 0} (P_{\Delta t})^{\frac{t}{\Delta t}}$

- **Required**: be a continuous stochastic variable (function) $X(t)$

- $t \geq 0$ is a real-valued time parameter (time)
- $X(t)$ can take values from a state space $Q = \{1, \ldots, n\}$.

  *in our case: $Q$ represents $\{A, C, G, U\}$*
Markov Process

Definition (Markov Process)

A (time-homogeneous) Markov process is a triple \((Q, \pi, P(t))\), where 
\(\pi = (\pi_1, \ldots, \pi_n)\) is the initial distribution (i.e., \(\pi_i = Pr[X(0) = i]\)), and \(P(t)\) is an \(n \times n\) matrix 
\[
P(t) = \begin{pmatrix}
p_{1,1}(t) & \cdots & p_{1,n}(t) \\
\vdots & \ddots & \vdots \\
p_{n,1}(t) & \cdots & p_{n,n}(t)
\end{pmatrix}
\]
of transition probability functions. These transition probabilities have to satisfy the following properties:

1. \(\forall i, t : \sum_j p_{i,j}(t) = 1\)
2. Markov property: 
\[
Pr[X(t + s) = j \mid X(s) = i] = Pr[X(t) = j \mid X(0) = i] \quad (\text{MP})
\]
\[= p_{i,j}(t).\]
Markov Process – Properties

- **Markov Property:**

\[
\begin{align*}
X(0) & \quad 0 \quad t \quad t + s \\
X(t) & \quad = \quad X(0) \quad \rightarrow \quad X(t + s)
\end{align*}
\]

⇒ immediately implies that matrices can be multiplied

- example for one entry: \( p_{1,4}(t + s) = \sum_j p_{1,j}(t) \cdot p_{j,4}(s) \)

- overall: property (MP') : \( P(t + s) = P(t)P(s) \)

- **Stationary distribution:** analogous to Markov chains

  - Markov chain: \( \lim_{t \to \infty} \pi P^t = \pi^* \)
  
  - Markov process: \( \lim_{t \to \infty} \pi P(t) = \pi^* \)

⇒ again assumed to exist

- **Final assumption:** \( \lim_{t \to 0} P(t) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = I \)
Determination of $P(t)$

- We want to calculate $P(t)$ from transition rates.
- Rates determined via derivatives:

$$\Rightarrow \text{consider } P'(t) = \begin{pmatrix} \frac{\partial p_{1,1}}{\partial t}(t) & \ldots & \frac{\partial p_{1,4}}{\partial t}(t) \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{4,1}}{\partial t}(t) & \ldots & \frac{\partial p_{4,4}}{\partial t}(t) \end{pmatrix}.$$

- Then we have:

$$P'(t) = \lim_{\Delta t \to 0} \frac{P(t + \Delta t) - P(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{P(t)P(\Delta t) - P(t)I}{\Delta t} \quad \text{(by MP')}$$

$$= P(t) \lim_{\Delta t \to 0} \frac{P(\Delta t) - P(0)}{\Delta t}$$

$$= P(t) \Lambda$$

where $\Lambda = \begin{pmatrix} -\lambda_1 & \lambda_{1,2} & \lambda_{1,3} & \lambda_{1,4} \\ \lambda_{2,1} & -\lambda_2 & \lambda_{2,3} & \lambda_{2,4} \\ \lambda_{3,1} & \lambda_{3,2} & -\lambda_3 & \lambda_{3,4} \\ \lambda_{4,1} & \lambda_{4,2} & \lambda_{4,3} & -\lambda_4 \end{pmatrix}$

matrix $\Lambda$ of transition rates.
Which form does $\lambda$ have?

- look at $p_{i,j}(t)$ ($i \neq j$) and $p_{i,i}(t)$

$\lambda_{ij}$ is the rate from state $i$ **into** state $j$.

$\lambda_{i,i}$ is the rate **out of** state $i \Rightarrow -\lambda_i$, where $\lambda_i = \sum_{j \neq i} \lambda_{i,j}$

- another argument that $\lambda_{i,j}$ are transition rates

$\Rightarrow$ Taylor expansion of $p_{i,j}(t)$ at 0:

$$p_{i,j}(\Delta t) = p_{i,j}(0) + \frac{\partial p_{i,j}}{\partial t}(0)\Delta t + \frac{1}{2!} \frac{\partial^2 p_{i,j}}{\partial t^2}(0)(\Delta t)^2 + \ldots$$

$$= 0 + \lambda_{i,j}\Delta t + \frac{1}{2!} \frac{\partial^2 p_{i,j}}{\partial t^2}(0)(\Delta t)^2 + \ldots .$$

$$\approx \lambda_{i,j}\Delta t \quad \text{(for small $\Delta t$)}$$
Determination of $\mathbf{P}(t)$ (Cont.)

- Where are we:  
  - given transition rates $\Lambda$
  - we know $\mathbf{P}'(t) = \mathbf{P}(t)\Lambda$

- How to calculate $\mathbf{P}(t)$? $\Rightarrow$ solve differential equation

- $\Lambda$ is constant, hence if $\mathbf{P}(t)$ would be a function instead of a matrix
  $\Rightarrow$ solution would be $\mathbf{P}(t) = e^{\Lambda t}$

- But: $e^{\Lambda t}$ not defined for matrices

---

recall: solution of linear differential equation

example: $f'(x) = 2 \cdot f(x) \Rightarrow f(x) = e^{2x}$ and $f'(x) = 2e^{2x}$

---

so: look for another definition of $e^x$ $\Rightarrow$ $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$

for a matrix $\Lambda t$: $\Rightarrow$ $e^{\Lambda t} = \sum_{n=0}^{\infty} \frac{(\Lambda t)^n}{n!}$ $\Rightarrow$ that’s defined!!
Solution of Infinity Sum

- Problem: find a closed form for \( e^{\Lambda t} = \sum_{n=0}^{\infty} \frac{(\Lambda t)^n}{n!} \)

  \( \Rightarrow \) depends on the form of \( \Lambda \)

- Specific form used by felsenstein:

  \[ \Lambda = \begin{pmatrix}
  -(u_2+u_3+u_4) & u_2 & u_3 & u_4 \\
  u_1 & -(u_1+u_3+u_4) & u_3 & u_4 \\
  u_1 & u_2 & -(u_1+u_2+u_4) & u_4 \\
  u_1 & u_2 & u_3 & -(u_1+u_2+u_3)
\end{pmatrix} \]

- \( u_i \) is the rate into nucleotide \( i \), independent from what was before

- Now consider \( P = I + \Lambda/u \), where \( u = u_1 + u_2 + u_3 + u_4 \).

- Simple calculation shows that \( P \) is of the form

  \[ P = \begin{pmatrix}
  \pi_1 & \pi_2 & \pi_3 & \pi_4 \\
  \pi_1 & \pi_2 & \pi_3 & \pi_4 \\
  \pi_1 & \pi_2 & \pi_3 & \pi_4 \\
  \pi_1 & \pi_2 & \pi_3 & \pi_4
\end{pmatrix} \]

- \( \pi_i = \frac{u_i}{u} \).

- Note that \( \pi_1 + \pi_2 + \pi_3 + \pi_4 = 1 \), which implies that \( P \) is stochastic.

- Furthermore, \( P \) is already stationary since \( P^2 = P \Rightarrow \forall n : P^n = P \)
Solved Form

- **Now:** \( P = I + \Lambda/u \) implies \(-u(I - P) = \Lambda\)
- This gives us the solved form for \( e^{\Lambda t} \) as follows:

\[
e^{\Lambda t} = e^{-u(I-P)t} = e^{-ut}Ie^{utP}
\]

\[
= \left[ \sum_{n=0}^{\infty} \frac{I^n(-ut)^n}{n!} \right] \left[ \sum_{n=0}^{\infty} \frac{P^n(ut)^n}{n!} \right]
\]

\[
= I \left[ \sum_{n=0}^{\infty} \frac{(-ut)^n}{n!} \right] \left\{ I + P \left[ \sum_{n=1}^{\infty} \frac{(ut)^n}{n!} \right] \right\}
\]

\[
= e^{-ut}[I + P(e^{ut} - 1)]
\]

\[
= e^{-ut}I + P(e^{-ut}e^{ut} - e^{-ut})
\]

\[
= e^{-ut}I + P(1 - e^{-ut})
\]

- writing down \( P(t) = e^{-ut}I + P(1 - e^{-ut}) \) componentwise, we get

\[
p_{i,j}(t) = e^{-ut}\delta_{i,j} + (1 - e^{-ut})\pi_j,
\]

where the Kronecker \( \delta \)-function is defined by \( \delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \).
Jukes–Cantor model: one of the very first nucleotide substitution models.

is generated by setting \( u_1 = u_2 = u_3 = u_4 = \alpha \),

which gives us the rate matrix

\[
\Lambda = \begin{pmatrix}
-3\alpha & \alpha & \alpha & \alpha \\
\alpha & -3\alpha & \alpha & \alpha \\
\alpha & \alpha & -3\alpha & \alpha \\
\alpha & \alpha & \alpha & -3\alpha \\
\end{pmatrix}.
\]

Then \( u = 4\alpha \) and \( \pi_1 = \pi_2 = \pi_3 = \pi_4 = \frac{1}{4} \). Hence, we have

\[
p_{i,i}(t) = e^{-4\alpha t} + \frac{1}{4}(1 - e^{-4\alpha t})
= \frac{1}{4}(1 + 3e^{-4\alpha t})
\]

and

\[
p_{i,j}(t) = \frac{1}{4}(1 - e^{-4\alpha t}) \quad \text{for } i \neq j.
\]
Overview

Where are we?

1. Scoring ✓

2. Evolutionary model ⇒ Markov process ✓

3. Scoring of complete trees:
   a. scoring for fixed topology
   b. optimizing distance
   c. finding correct topology

⇒ (see book)
Scoring Tree with Fixed Topology

- Example topology:

- Likelihood of tree: depends on the states for inner nodes $s_0, s_5, s_6$
- Given $s_0, s_5, s_6$ then
  \[
  \tilde{L}(s_0, s_5, s_6) = p_{s_0,s_5}(d_5) \cdot p_{s_0,s_6}(d_6) \cdot p_{s_5,a_1}(d_1) \cdot p_{s_5,a_2}(d_2) \cdot p_{s_6,a_3}(d_3) \cdot p_{s_6,a_4}(d_4). 
  \]
- For $s_0, s_5, s_6$ unknown:
  \[
  L = \sum_{s_0} \sum_{s_5} \sum_{s_6} \pi_{s_0} \tilde{L}(s_0, s_5, s_6)
  \]
Optimized Calculation

- Horner’s trick of polynomial evaluation \( \Rightarrow \) pushing sums to the right

\[
L = \sum_{s_0} \prod_{s_0} \left[ \sum_{s_5} p_{s_0,s_5}(d_5) \left( p_{s_5,a_1}(d_1)p_{s_5,a_2}(d_2) \right) \right] \times \left[ \sum_{s_6} p_{s_0,s_6}(d_6) \left( p_{s_6,a_3}(d_3)p_{s_6,a_4}(d_4) \right) \right]
\]

- Then the pattern of this expression follows exactly the pattern of the tree:

\[
\begin{bmatrix} \cdot (\cdots) \end{bmatrix} \begin{bmatrix} \cdot (\cdots) \end{bmatrix}
\]

- Define the \textit{conditional likelihood} \( L_{k,s} \) as the likelihood of the subtree rooted at \( k \), given that node \( k \) has state \( s \)
Recursive Definition

- **Leaf** $i$: $L_{i,s} = \begin{cases} 1 & \text{if the } i\text{th taxon has } s \text{ at this site}, \\ 0 & \text{otherwise}, \end{cases}$

- **Inner nodes:**

  \[ A \leftarrow k \]
  \[ C \leftarrow i \]
  \[ \rightarrow j \rightarrow A \]

  then $L_{i,C}$ and $L_{j,A}$ must already be defined

  \[ \Rightarrow \text{ if } k \text{ is the parent of } i, j, \text{ then define the conditional likelihood} \]

  \[ L_{k,s_k} = \left( \sum_{s_i} p_{s_k,s_i}(d_i)L_{i,s_i} \right) \left( \sum_{s_j} p_{s_k,s_j}(d_j)L_{j,s_j} \right). \]

  \[ \Rightarrow \text{ Dynamic Programming} \]

- **Finally:** total tree likelihood

  \[ L = \sum_{s_0} \pi_{s_0} \cdot L_{0,s_0} \]