Bioinformatics Programming Practical: Task Specification

Tomas Flouri and Alexandros Stamatakis

April 9, 2015

1 Thorne, Kishino, Felsenstein model

1.1 Description

Algorithms—as presented in the winter lecture—for aligning biological sequences are typically not derived from an evolutionary model, but rather use a discrete and pretty *ad hoc* scoring matrix for penalizing mismatches and inserting gaps. In 1991, Thorne, Kishino, and Felsenstein presented a maximum likelihood method for aligning two DNA sequences [1] under an explicit statistical model of evolution.

This statistical model also uses the three well-known types of alignment operations: insertions, deletions, and substitutions. Initially, assume a sequence of size n. To describe the alignment method we first just consider the evolution of a single sequence over time. The procedure is similar to the phylogenetic likelihood models we have learned about in the lectures. It is however extended by insertions and deletions. Also note that, we view the stochastic process in this top-level description as a generating process, that is, for simulating sequences over the course of time.

First, we assign a time-reversible substitution process to each of the n nucleotides. This standard substitution process we already know describes nucleotide substitutions. Now let's see how to model insertions and deletions.

Each nucleotide is also associated with a stochastic deletion variable that is exponentially distributed with parameter μ. If this variable is triggered, then the corresponding nucleotide is removed (deleted) in the course of evolution.

Additionally, each nucleotide is associated with a stochastic insertion variable (called *mortal link*) which is exponentially distributed with parameter $\lambda < \mu$ and placed to the immediate right of each nucleotide (we read sequences from left to right!). If the i-th mortal link (immediately right to the i-th nucleotide) is triggered, a nucleotide is chosen (inserted) from the stationary distribution (denoted as base frequencies $\pi_A, \ldots, \pi_T$ in the maximum likelihood and Bayesian lectures) of the substitution process. Note that, we assume a time-reversible nucleotide substitution process as in the standard phylogenetic likelihood models we have learned about in the lectures. Once we have chosen the nucleotide to be inserted, we place it (together with a new, additional, mortal link to its right) to the right of the mortal link that was triggered. Finally, on the leftmost nucleotide of the complete sequence there is an *immortal link* that can give birth to/generate new nucleotides (including associated mortal links), at the same rate as a *mortal* link but that cannot die.

Hence, for a single sequence we now have stochastic processes that describes insertions, deletions, and substitutions in a single sequence over the course of time. Let us now see how this can be used to do pair-wise optimal stochastic sequence alignment.

1.2 Input

Let $A = a_1a_2 \ldots a_n$ and $B = b_1b_2 \ldots b_m$ be two sequences over the alphabet $\mathcal{A} = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. Let $\overrightarrow{\pi} = (\pi_A, \pi_C, \pi_G, \pi_T)$ be the equilibrium probabilities (denoted as base frequencies in the lectures) of the four nucleotides $\texttt{A}, \texttt{C}, \texttt{G}$ and $\texttt{T}$.

For simplicity, we will use the Felsenstein (typically denoted as F81) standard phylogenetic nucleotide substitution model here. It is simple because the nucleotide transition probabilities for time t can be calculated analytically and we do not need to use an eigenvector/eigenvalue decomposition as for the GTR (General Time-Reversible) model, for instance.

Therefore, we only have one substitution rate parameter δ (instead of 5 for GTR) that we can derive directly from the equilibrium probabilities (base frequencies):

$$\delta = \frac{1}{1 - \pi_A^2 - \pi_C^2 - \pi_G^2 - \pi_T^2}.$$

The transition probability of nucleotide i changing to nucleotide j over time t is defined as

$$P_{i \to j}(t) = \begin{cases} e^{-\delta t} + \pi_j(1 - e^{-\delta t}) & : \quad i = j \\ \pi_j(1 - e^{-\delta t}) & : \quad i \neq j \end{cases}$$

Further, we are also given the *birth rate* λ and *death rate* μ of a so-called birth-death process (see `http://en.wikipedia.org/wiki/Birth-death_process`). We need such a process to model insertions and deletions.

Further, let γ_n be the equilibrium frequency of observing sequences with a length of n nucleotides. The distribution of γ_n obtained under the birth-death model is the geometric distribution

$$\gamma_n = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^n.$$

Finally, in the rest of the text we will use $p_n(t)$ (resp. $\overline{p_n}(t)$) to denote the probability that after time t, a mortal link has exactly n descendants, and one of these is (resp. is not) the original mortal link.

We also use $\zeta_n(t)$ (we use ζ here as the first letter of the Greek word for life) to denote the probability that after time t the **immortal** link has exactly n descendants (including itself).

These probabilities are defined as follows:

$$p_n(t) = e^{-\mu t}[1 - \lambda\beta(t)][\lambda\beta(t)]^{n-1} \qquad n > 0$$
$$\overline{p_n}(t) = [1 - e^{-\mu t} - \mu\beta(t)][1 - \lambda\beta(t)][\lambda\beta(t)]^{n-1} \qquad n > 0$$
$$\overline{p_0}(t) = \mu\beta(t)$$
$$\zeta_n(t) = [1 - \lambda\beta(t)][\lambda\beta(t)]^{n-1} \qquad n > 0$$

where

$$\beta(t) = \frac{1 - e^{(\lambda-\mu)t}}{\mu - \lambda e^{(\lambda-\mu)t}}, \qquad 0 < \lambda < \mu.$$

1.3 Algorithm.

To present the algorithm in its simplest form, we define three matrices M^0, M^1, M^2 each of size $(n+1) \times (m+1)$. Let us also assume that all evolutionary input parameters are given, and do not need to be explicitly estimated/maximized. Hence, the task is to implement a function that computes a ML alignment with the header `TKF91(`$A, B, \lambda, \mu, t, \overrightarrow{\pi}$`);`
Below, we state the rules for the dynamic programming algorithm.

Initialization rules.

$M^0(0,0) = 0.0$

$M^1(0,0) = \gamma_0\zeta_1(t)$

$M^2(0,0) = 0.0$

$$M^0(i,0) = \gamma_i \zeta_i(t) \prod_{j=1}^{i} \pi_{a_j} \overline{p_0}(t) \qquad 1 \leq i \leq n$$
$$M^1(i,0) = 0.0 \qquad 1 \leq i \leq n$$
$$M^2(i,0) = 0.0 \qquad 1 \leq i \leq n$$

$$M^0(0,i) = 0.0 \qquad 1 \leq i \leq m$$
$$M^1(0,i) = 0.0 \qquad 1 \leq i \leq m$$
$$M^2(0,i) = \gamma_0 \zeta_{i+1}(t) \prod_{j=1}^{i} \pi_{b_j} \qquad 1 \leq i \leq m$$

DP rules.

$$M^0(i,j) = \frac{\lambda}{\mu} \pi_{a_i} \overline{p_0}(t) \max\{M^0(i-1,j), M^1(i-1,j), M^2(i-1,j)\}$$
$$M^1(i,j) = \frac{\lambda}{\mu} \pi_{a_i} \max\{P_{a_i \to b_j}(t) p_1(t), \pi_{b_j} \overline{p_1}(t)\} \max\{M^0(i-1,j-1), M^1(i-1,j-1), M^2(i-1,j-1)\}$$
$$M^2(i,j) = \pi_{b_j} \lambda \beta(t) \max\{M^1(i,j-1), M^2(i,j-1)\}$$

Optimal score and alignment output: The score s of the optimal alignment is obtained by

$$s = \max\{M^0(n,m), M^1(n,m), M^2(n,m)\}.$$

The alignment can then be retrieved in the same way as in the Needleman-Wunsch algorithm, by backtracking through the path we have taken to reach the optimal score. Note that, although there may exist exponentially many paths, the task is to output only one such path. However, because the matrices actually contain floating point values, it is highly unlikely that an equality will occur among the maximum values of the three relevant adjacent cells (top, diagonal, left) during backtracking. In the rare case that such an equality occurs, you shall construct the backtracking path by choosing the cell on the path in an anti-clockwise order. That is, if one of the identical maxima is at the top cell you should chose the top cell, then the diagonal cell, and finally the left cell. This type of path selection is not described in the original paper. It is merely a means to ensure that your implementations will output the same results as the reference implementation.

Task specification:

1. Implement function `TKF91(`$A, B, \lambda, \mu, t, \overrightarrow{\pi}$`);` in `C` or `C++`.
2. Vectorize function `TKF91()` using SSE3 and AVX vector intrinsics.
3. Tune function `TKF91()` for maximum performance with respect to cache and computational efficiency.

Things to keep in mind:

- Write clean code with no `clang` compiler warnings
- Use `valgrind` to detect and fix memory leaks
- Remember that you are multiplying probabilities, that is, measures might need to be taken to avoid underflow
- Remember that de-normalized floating point values could be an issue for performance
- Pre-mature optimization is the source of all evil!

References

[1] Thorne, J., Kishino, H., and Felsenstein, J. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution 33*, 2 (1991), 114–124.