alignmentbd User Manual

Benjamin E. Decato

April 25, 2014

Contents

1 Overview

This repository contains three alignment algorithms for DNA sequence alignment. `galignbd` performs global alignment between two DNA sequences, `banded_galignbd` is a more efficient version that requires less space, and `malignbd` implements multiple sequence alignment. Below are descriptions of each program, along with usage. All programs have minimal I/O error checking, so correct specification of arguments is critical at this early stage in development.

2 Theoretical overview

Many alignment algorithms rely on the notion of a scoring function, in which the alignments are rewarded for correct matches and penalized for placing gaps (called "indels" in molecular biology) or allowing a mismatch. The general structure of a scoring function for position x in sequence 1 and y in sequence 2 is:

$$w\binom{x}{y} = \begin{cases} 1 & : x = y \\ -1 & : x \neq y \\ -2 & : \text{x or y indel} \end{cases}$$

In each of the three algorithms implemented in `alignmentbd`, the goal is to maximize the score for the alignment between the sequences and then trace the solution back from the score matrix. All three are famous dynamic programming algorithms, which I will go into now.

(a) **galignbd:**

Global alignment between two sequences. Given two sequences $U = u_1, u_2, ...u_n$ and $V = v_1, v_2, ...v_m$, define $S(i,j) = S(u_1...u_i, v_1...v_j)$. The goal is to maximize the score function $S(n,m)$ using the following recursive definition:

$$S(i,j) = max \begin{cases} S(i-1,j-1) + w\binom{u_i}{v_j} \\ S(i-1,j) + w\binom{u_i}{-} \\ S(i,j-1) + w\binom{-}{v_j} \end{cases}$$

Starting with the base case $S(0,0) = 0$, `galignbd` creates a table of optimal alignment scores for subsets of the two strings. The entry in the $(n,m)^{th}$ cell corresponds to the maximum global alignment score, and the table is traced backwards to find the true alignment.

The time and space complexity for this algorithm as implemented are both quadratic $\mathcal{O}(nm)$, as I iterate over all substrings computing their score and storing it in an n by m table. It is possible to perform global alignment in linear space, but easy implementation of local alignment is sacrificed and the linear space version is not implemented here. Keep an eye out for later commits!

(b) **banded_galignbd:**

This algorithm identifies the global alignment for two strings that differ by at most k mismatches or indels, and is referred to as banded because the score matrix truncates all cells too far away from the theoretical perfect alignment, resulting in a diagonal band data structure. The recursive definition of $S(i,j)$ is identical to global alignment, but in addition to $S(0,0) = 0$ as a base case, $S(i,j) = -\infty$

if $j < L[i]$ or $j > R[i]$ where L and R are the bend at the i^{th} row. That is, the cells outside the k mismatch boundary have a score of $-\infty$.

This algorithm tightens the problem to only closely related sequences, but provides a useful edge: if you know your sequences are fairly close, you can save space and time complexity from quadratic to $\mathcal{O}(kn)$ where n is the larger string.

(c) **malignbd:**

This is an approximation algorithm for multiple sequence alignment. It works as a profile aligner, where sequences are iteratively merged together into profiles, which are snapshots containing the proportion of each nucleotide at each site in the aligned sequence. For example, position 1 of a profile containing three sites could be $A = 0.3333$, $C = 0.3333$, $T = 0$, $G = 0$, $- = 0.3333$ where one sequence contains an A, another a C, and another a deletion at that site.. Given $P = p_1, p_2, .., p_n$ and $Q = q_1, q_2, .., q_n$ where P and Q are profiles, the score for a site between the two is:

$$g\binom{p_i}{q_i} = \sum_{k,l \in \{ACTG-\}} w(k,l) p_i[k] q_j[l]$$

The optimal global alignment between profiles can be calculated using the same dynamic programming recursion as in global alignment with this modified score function:

$$S(p_i, q_j) = max \begin{cases} S(p_{i-1}, q_{j-1}) + g\binom{p_i}{q_j} \\ S(p_{i-1}, q_j) + g\binom{p_i}{-} \\ S(p_i, q_{j-1}) + g\binom{-}{q_j} \end{cases}$$

Order of profile alignment is chosen heuristically based on their distance from each other, calculated by counting the number of mismatches when no alignment is performed on the original sequences in a pairwise manner. After a profile is created from two sequences, the distance from the profile to all other sequences is the average of the distance from each of the profile's aligned sequences to each of the others.

As implemented, `malignbd` runs in $\mathcal{O}(knm)$ time where k is the number of sequences and n, m are the sizes of the two largest sequences. This approximation is a measurable speedup over the naive case of multiple alignment, which operates in $\mathcal{O}(n^k)$ time, where n is the longest sequence.

3 Running the programs

Each of the programs has the option to provide a unique scoring function. This must be done completely or not at all, and can be passed as follows:

```
> galignbd -m 2 -s 1 -i 3 first_sequence.fasta second_sequence.fasta
> banded_galignbd -m 2 -s 1 -i 3 first_sequence.fasta second_sequence.fasta
> malignbd -m 2 -s 1 -i 3 all_sequences.fasta
```

This will provide a score function that penalizes mismatches by 1, indels by 3, and ups the score for a match by 2. For `galignbd` and `banded_galignbd`, the sequences should come in FASTA format in two seperate files. For `malignbd`, they should come in one FASTA file seperated by a newline. There is currently very little I/O error checking: a problem I expect to fix in the future. Therefore be very careful to specify the correct arguments on the command line.