

CS 466 – Introduction to Bioinformatics – Lecture 3

Mohammed El-Kebir

September 6, 2019

Document history:

- 9/7/2018: Initial version.
- 9/9/2018: Included revised analysis of naive fitting alignment.
- 9/6/2019: Fixed off-by-one error in Section 1

Contents

1 Naive Fitting Alignment	1
2 Naive Local Alignment	2
3 Naive Affine Gap Penalty Global Alignment	2

1 Naive Fitting Alignment

In the fitting alignment problem we are given two strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$, a scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$, and are asked to find a substring of \mathbf{w} whose alignment with \mathbf{v} has maximum global alignment score among *all* alignments of \mathbf{v} and *all* substrings of \mathbf{w} .

How do we solve this? A naive approach would be to simply generate all substrings of \mathbf{w} . Each substring \mathbf{w}' corresponds to an instance of the global alignment problem, which can be solved in $O(m|\mathbf{w}'|)$ time. What is the total running time?

We start by observing that if \mathbf{w}' is the empty string, then the optimal alignment score would be trivially 0. So we can assume that $|\mathbf{w}'| \geq 1$, resulting in the following running time.

$$\sum_{i=1}^n \sum_{j=i}^n O(m(j-i+1)) = O(m) \sum_{i=1}^n \sum_{j=i}^n (j-i+1). \quad (1)$$

The number of substrings of length $\ell = 1$ is n . How many substrings are there of length $\ell = 2$? There are $n - 1$ pairs $(i, i + 1)$ where $1 \leq i \leq n - 1$. Thus, there are $n - 1$ substrings

of \mathbf{w} of length $\ell = 2$. Similarly, there are $n - 2$ substrings of \mathbf{w} of length $\ell = 3$ corresponding to pairs $(i, i + 2)$ where $1 \leq i \leq n - 2$, etc. Thus we have the following equation.

$$\sum_{i=1}^n \sum_{j=i}^n (j - i + 1) = \sum_{\ell=1}^n \ell(n - \ell + 1). \quad (2)$$

This can be rewritten as

$$\sum_{\ell=1}^n \ell(n - \ell + 1) = \sum_{\ell=1}^n (\ell n - \ell^2 + \ell) \quad (3)$$

$$= n \sum_{\ell=1}^n \ell - \sum_{\ell=1}^n \ell^2 + \sum_{\ell=1}^n \ell. \quad (4)$$

Using that $\sum_{i=1}^n i = n(n + 1)/2$ and $\sum_{i=1}^n i^2 = n(n + 1)(2n + 1)/6$, we obtain

$$n \sum_{\ell=1}^n \ell - \sum_{\ell=1}^n \ell^2 + \sum_{\ell=1}^n \ell = (n + 1) \cdot \frac{n(n + 1)}{2} - \frac{n(n + 1)(2n + 1)}{6} \quad (5)$$

$$= \frac{n^3 + 3n^2 + 2n}{6}. \quad (6)$$

This amounts to a running time of

$$O(m) \frac{n^3 + 3n^2 + 2n}{6} = O(mn^3). \quad (7)$$

2 Naive Local Alignment

In the local alignment problem we are given two strings $\mathbf{v} \in \Sigma^m$ and $\mathbf{w} \in \Sigma^n$, a scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$, and are asked to find a substring \mathbf{v}' of \mathbf{v} and a substring \mathbf{w}' of \mathbf{w} whose alignment has maximum global alignment score among *all* alignments of *all* substrings of \mathbf{v} and \mathbf{w} .

Similarly to the naive approach for fitting alignment, we could simply generate all substrings of \mathbf{v} and \mathbf{w} . Each pair $(\mathbf{v}', \mathbf{w}')$ of substrings corresponds to an instance of the global alignment problem, which can be solved in $O(|\mathbf{v}'||\mathbf{w}'|)$ time. What is the total running time when considering all pairs of possible substrings? Recall that aligning \mathbf{v} and \mathbf{w} has time $O(|\mathbf{v}||\mathbf{w}|) = O(mn)$. Thus, here we want to compute $O(\sum_{\mathbf{v}'} |\mathbf{v}'| \sum_{\mathbf{w}'} |\mathbf{w}'|)$.

Above, we computed that the sum of the lengths of substrings of \mathbf{w} with length $n = |\mathbf{w}|$ is $(n^3 - 3n^2 + 2n)/6$. That is, $\sum_{\mathbf{w}'} |\mathbf{w}'| = (n^3 - 3n^2 + 2n)/6$. Thus, $\sum_{\mathbf{v}'} |\mathbf{v}'| = (m^3 - 3m^2 + 2m)/6$. This leads to a running time of $O(m^3n^3)$.

3 Naive Affine Gap Penalty Global Alignment

In this case, the edit graph contains $i + j$ incoming edges for each vertex (i, j) . The running time is simply the total number of edges, as each edge $\langle (i', j'), (i, j) \rangle$ requires a constant

time computation that is only performed at the target vertex (i, j) . We thus have

$$\sum_{i=0}^m \sum_{j=0}^n (i + j) = \sum_{i=0}^m \left[i \cdot (n + 1) + \sum_{j=0}^n j \right]. \quad (8)$$

Using that $\sum_{j=0}^n j = \sum_{j=1}^n j = n(n + 1)/2$, we get

$$\sum_{i=0}^m [i \cdot (n + 1) + n(n + 1)/2] = \frac{(m + 1)n(n + 1)}{2} + (n + 1) \sum_{i=0}^m i \quad (9)$$

$$= \frac{(m + 1)n(n + 1) + (n + 1)m(m + 1)}{2} \quad (10)$$

$$= O(mn^2 + nm^2). \quad (11)$$

So if $m = n$, this would lead to a cubic algorithm. This is worse than the $O(mn)$ algorithm presented in class for global alignment with affine gap penalties.