Algorithmic Bioinformatics DD2450, spring 2010, Lecture 2

Lecturer Jens Lagergren
Several current and previous students
will be acknowledged in a separate document.

April 5, 2010

1 Pairwise sequence alignment

Sequence alignment is one of the oldest problems in bioinformatics and considers the problem of describing similarities between two or more sequences. Multiple sequence alignment and genomic alignment are two other common types of sequence alignment.

Sequence alignment has a broad range of applications in bioinformatics. Here are three examples of such applications:

Similarity distances for phylogeny
By measuring the amount of mutations between two genes from a pair of species the evolutionary distance between them can be estimated. Such distance estimates for all pairs from a set of species can be used to construct a phylogenic tree of them.

Searching for a gene from a new genome
When finding a gene in a new genome it may be interesting to see if that gene exists in other already sequenced species.

Aligning genomes as a first step in comparative studies
By aligning two different genomes, regions of interest may be found. As an example, regions where the two genomes are particularly similar are interesting to study, since they often contain elements which are conserved because of their importance to the species.

1.1 Global alignment

In global alignment two DNA sequences are aligned in their full length ignoring the physical structure of the DNA. The evolutionary events taken into account are base substitutions and insertions/deletions (indels). Such mutations can occur for instance through errors made by the DNA polymerase or the cell's repair machinery. The sequences will be strings over the alphabet $\{\texttt{A,T,G,C,-}\}$. The blank symbol, `-`, is needed to represent insertions and deletions. Sometimes alignment of protein sequences might be preferred since codons may change and still encode the same protein.

Definition 1. *The support of a sequence $S \in \{\texttt{A,T,G,C,-}\}^*$ is S with all blank symbols deleted.*

Definition 2. *An alignment of two sequences $X, Y \in \{\texttt{A,T,G,C}\}^*$ is a $2 \times k$ matrix Π such that*

1. *the support of $\Pi_{1,1}, \ldots, \Pi_{1,k}$ is X*
2. *the support of $\Pi_{2,1}, \ldots, \Pi_{2,k}$ is Y*
3. *$\forall i : \neg\,(\Pi_{1,i} = \texttt{-} \;\wedge\; \Pi_{2,i} = \texttt{-})$*

The first two requirements say that the first and the second row of the matrix Π consists of the sequences X and Y respectively, possibly with some blank symbols inserted. The third and final requirement guarantees that no column in Π consists of two blank symbols. This implies that the width k of the matrix Π is at most the sum of the lengths of X and Y.

Example 1. *Given two homologous sequences X and Y with*

$$X = \texttt{GATTAC}$$
$$Y = \texttt{GCCTAAC}$$

a possible alignment Π of X and Y is

$$\Pi = \left[\begin{array}{c} \texttt{GAT-TA-C} \\ \texttt{G-CCTAAC} \end{array}\right]$$

Notation 1. *The following alternative notation for an alignment is used $\Pi = <\Pi_{1,1}, \Pi_{2,1}>, \ldots, <\Pi_{1,|\Pi|}, \Pi_{2,|\Pi|}>$ and $\Pi_i = <\Pi_{1,i}, \Pi_{2,i}>$.*

1.2 Score

Two sequences can be aligned in more than one way. To be able to determine how good a certain alignment is a similarity function $s : \{\texttt{A,C,G,T,-}\}^2 \mapsto \mathbb{R}$ is introduced. The value of $s(a, b)$ should be higher the more likely it is for a to mutate into b.

Definition 3. *The score, σ, of an alignment, Π, is defined as*

$$\sigma(\Pi) = \sum_{i=1}^{|\Pi|} s(\Pi_{1,i}, \Pi_{2,i}).$$

Definition 4. *The optimal global alignment score, γ, of two sequences X and Y is the highest possible score for all alignments, Π, of X and Y.*

$$\gamma(X, Y) = \max \sigma(\Pi).$$

1.3 Dynamic programming solution to the global alignment problem

Global alignment can be formulated as a computational problem in the following manner

Input: Two homologous sequences $X = x_1, \dots, x_m$ and $Y = y_1, \dots, y_n$ with common ancestor Z where X, Y, $Z \in \{\mathtt{A},\mathtt{T},\mathtt{G},\mathtt{C}\}^*$.

Output: Pairs of positions in X and Y that descend from the same position in Z.

Observations

1. If Π, where $|\Pi|$=k, is an optimal alignment of X^i and Y^j then one of following holds
   (a) $\Pi_k = \langle x_i, y_j \rangle$ and Π^{k-1} is an optimal alignment of X^{i-1} and Y^{j-1}.
   (b) $\Pi_k = \langle x_i, \texttt{-} \rangle$ and Π^{k-1} is an optimal alignment of X^{i-1} and Y^j.
   (c) $\Pi_k = \langle \texttt{-}, y_j \rangle$ and Π^{k-1} is an optimal alignment of X^i and Y^{j-1}.
2. Let the matrix $g(i,j) = \gamma(X^i, Y^j)$ represent the optimal global alignment score of X^i and Y^j then $g(i,j)$ can be computed recursively as
$$g(i,j) = \max \begin{cases} g(i-1, j-1) + s(x_i, y_j) \\ g(i-1, j) + s(x_i, -) \\ g(i, j-1) + s(-, y_j) \end{cases}$$
3. Aligning a prefix X^k of X to k gaps gives a score of $\sum_{r=0}^{k} s(x_r, \texttt{-})$
$$\begin{aligned} & g(0,0) = 0 \\ \forall i > 0: \quad & g(i,0) = \textstyle\sum_{r=0}^{i} s(x_r, \texttt{-}) \\ \forall j > 0: \quad & g(0,j) = \textstyle\sum_{r=0}^{j} s(\texttt{-}, y_r) \end{aligned}$$
4. Computing $g(i,j)$ requires the values $g(i-1, j-1)$, $g(i-1, j)$ and $g(i, j-1)$. They will always be available if each row is computed from left to right starting at $g(1,1)$.
5. The optimal global aligning score is the value at $g(m,n)$. By using back-pointers describing which of the three cases that led to the value in each element $g(i,j)$, it is also possible to extract the corresponding alignment that gives the optimal score.

Complexity of the algorithm

Time: $O(mn)$, since there are (m+1)(n+1) cells and each is computed in constant time.

Memory: $O(mn)$, since each cell uses constant memory. To get only the maximum score for all alignments the memory complexity can be reduced to $O(\min(m,n))$ by saving only the previous line of the matrix $g(i,j)$.

Algorithm 1 Optimal global alignment of two sequences using dynamic programming

```
Input: Strings $X$ and $Y$ of length $m$ and $n$ respectively.
Output: Optimal global alignment of $X$ and $Y$.
  // Initialize base cases
  $G[0,0] = 0$
  for $i = 1$ to $m$ do
    $G[i,0] = G[i-1,0] + s(X[i], \texttt{-})$
    $B[i,0] = \uparrow$
  end for
  for $j = 1$ to $n$ do
    $G[0,j] = G[0,j-1] + s(\texttt{-}, Y[j])$
    $B[0,j] = \leftarrow$
  end for
  // Compute optimal values for other cases, save backtrack information
  for $i = 1$ to $m$ do
    for $j = 1$ to $n$ do
      $G[i,j] = G[i-1,j-1] + s(X[i], Y[j])$
      $B[i,j] = \nwarrow$
      if $G[i,j-1] + s(\texttt{-}, Y[j]) > G[i,j]$ then
        $G[i,j] = G[i,j-1] + s(\texttt{-}, Y[j])$
        $B[i,j] = \leftarrow$
      end if
      if $G[i-1,j] + s(X[i], \texttt{-}) > G[i,j]$ then
        $G[i,j] = G[i-1,j] + s(X[i], \texttt{-})$
        $B[i,j] = \uparrow$
      end if
    end for
  end for
  // Construct optimal solution
  $OGA = \emptyset$
  $i = m$
  $j = n$
  while $(i,j) \neq (0,0)$ do
    if $B[i,j] = \nwarrow$ then
      prepend $\langle X[i], Y[j] \rangle$ to $OGA$
      $(i,j) = (i-1, j-1)$
    else if $B[i,j] = \leftarrow$ then
      prepend $\langle \texttt{-}, Y[j] \rangle$ to $OGA$
      $(i,j) = (i, j-1)$
    else
      prepend $\langle X[i], \texttt{-} \rangle$ to $OGA$
      $(i,j) = (i-1, j)$
    end if
  end while
  return $OGA$
```