WIKIPEDIA

Hirschberg's algorithm

In computer science, **Hirschberg's algorithm**, named after its inventor, Dan Hirschberg, is a dynamic programming algorithm that finds the optimal sequence alignment between two strings. Optimality is measured with the Levenshtein distance, defined to be the sum of the costs of insertions, replacements, deletions, and null actions needed to change one string into the other. Hirschberg's algorithm is simply described as a more space-efficient version of the Needleman–Wunsch algorithm that uses divide and conquer.[1] Hirschberg's algorithm is commonly used in computational biology to find maximal global alignments of DNA and protein sequences.

Contents

Algorithm information

Algorithm description

Example

See also

References

Algorithm information

Hirschberg's algorithm is a generally applicable algorithm for optimal sequence alignment. BLAST and FASTA are suboptimal heuristics. If *x* and *y* are strings, where length(*x*) = *n* and length(*y*) = *m*, the Needleman–Wunsch algorithm finds an optimal alignment in O(*nm*) time, using O(*nm*) space. Hirschberg's algorithm is a clever modification of the Needleman–Wunsch Algorithm, which still takes O(*nm*) time, but needs only O(min{*n*, *m*}) space and is much faster in practice.[2] One application of the algorithm is finding sequence alignments of DNA or protein sequences. It is also a space-efficient way to calculate the longest common subsequence between two sets of data such as with the common diff tool.

The Hirschberg algorithm can be derived from the Needleman–Wunsch algorithm by observing that:[3]

1. one can compute the optimal alignment score by only storing the current and previous row of the Needleman–Wunsch score matrix;
2. if $(Z,W) = \operatorname{NW}(X,Y)$ is the optimal alignment of (X,Y), and $X = X^l + X^r$ is an arbitrary partition of X, there exists a partition $Y^l + Y^r$ of Y such that $\operatorname{NW}(X,Y) = \operatorname{NW}(X^l,Y^l) + \operatorname{NW}(X^r,Y^r)$.

Algorithm description

X_i denotes the *i*-th character of X, where $1 \leqslant i \leqslant \operatorname{\mathbf{length}}(X)$. $X_{i:j}$ denotes a substring of size $j - i + 1$, ranging from the *i*-th to the *j*-th character of X. $\operatorname{\mathbf{rev}}(X)$ is the reversed version of X.

X and Y are sequences to be aligned. Let x be a character from X, and y be a character from Y. We assume that $\mathrm{Del}(x)$, $\mathrm{Ins}(y)$ and $\mathrm{Sub}(x, y)$ are well defined integer-valued functions. These functions represent the cost of deleting x, inserting y, and replacing x with y, respectively.

We define $\mathrm{NWScore}(X, Y)$, which returns the last line of the Needleman–Wunsch score matrix $\mathrm{Score}(i, j)$:

```
function NWScore(X, Y)
    Score(0, 0) = 0 // 2 * (length(Y) + 1) array
    for j = 1 to length(Y)
        Score(0, j) = Score(0, j - 1) + Ins($Y_j$)
    for i = 1 to length(X) // Init array
        Score(1, 0) = Score(0, 0) + Del($X_i$)
        for j = 1 to length(Y)
            scoreSub = Score(0, j - 1) + Sub($X_i$, $Y_j$)
            scoreDel = Score(0, j) + Del($X_i$)
            scoreIns = Score(1, j - 1) + Ins($Y_j$)
            Score(1, j) = max(scoreSub, scoreDel, scoreIns)
        end
        // Copy Score[1] to Score[0]
        Score(0, :) = Score(1, :)
    end
    for j = 0 to length(Y)
        LastLine(j) = Score(1, j)
    return LastLine
```

Note that at any point, $\mathrm{NWScore}$ only requires the two most recent rows of the score matrix. Thus, $\mathrm{NWScore}$ is implemented in $O(\min\{\mathrm{length}(X), \mathrm{length}(Y)\})$ space.

The Hirschberg algorithm follows:

```
function Hirschberg(X, Y)
    Z = ""
    W = ""
    if length(X) == 0
        for i = 1 to length(Y)
            Z = Z + '-'
            W = W + $Y_i$
        end
    else if length(Y) == 0
        for i = 1 to length(X)
            Z = Z + $X_i$
            W = W + '-'
        end
    else if length(X) == 1 or length(Y) == 1
        (Z, W) = NeedlemanWunsch(X, Y)
    else
        xlen = length(X)
        xmid = length(X) / 2
        ylen = length(Y)

        ScoreL = NWScore($X_{1:xmid}$, Y)
        ScoreR = NWScore(rev($X_{xmid+1:xlen}$), rev(Y))
        ymid = $\underline{\text{arg max}}$ ScoreL + rev(ScoreR)

        (Z,W) = Hirschberg($X_{1:xmid}$, $y_{1:ymid}$) + Hirschberg($X_{xmid+1:xlen}$, $Y_{ymid+1:ylen}$)
    end
    return (Z, W)
```

In the context of observation (2), assume that $X^l + X^r$ is a partition of X. Index ymid is computed such that $Y^l = Y_{1:\mathrm{ymid}}$ and $Y^r = Y_{\mathrm{ymid}+1:\mathrm{length}(Y)}$.

Example

Let

$$\begin{aligned} X &= \mathrm{AGTACGCA}, \\ Y &= \mathrm{TATGC}, \\ \mathrm{Del}(x) &= -2, \\ \mathrm{Ins}(y) &= -2, \\ \mathrm{Sub}(x,y) &= \begin{cases} +2, & \text{if } x = y \\ -1, & \text{if } x \neq y. \end{cases} \end{aligned}$$

The optimal alignment is given by

```
W = AGTACGCA
Z = --TATGC-
```

Indeed, this can be verified by backtracking its corresponding Needleman–Wunsch matrix:

```
        T   A   T   G   C
     0  -2  -4  -6  -8 -10
A   -2  -1   0  -2  -4  -6
G   -4  -3  -2  -1   0  -2
T   -6  -2  -4   0  -2  -1
A   -8  -4   0  -2  -1  -3
C  -10  -6  -2  -1  -3   1
G  -12  -8  -4  -3   1  -1
C  -14 -10  -6  -5  -1   3
A  -16 -12  -8  -7  -3   1
```

One starts with the top level call to $\mathrm{Hirschberg}(\mathrm{AGTACGCA}, \mathrm{TATGC})$, which splits the first argument in half: $X = \mathrm{AGTA} + \mathrm{CGCA}$. The call to $\mathrm{NWScore}(\mathrm{AGTA}, Y)$ produces the following matrix:

```
       T   A   T   G   C
    0  -2  -4  -6  -8 -10
A  -2  -1   0  -2  -4  -6
G  -4  -3  -2  -1   0  -2
T  -6  -2  -4   0  -2  -1
A  -8  -4   0  -2  -1  -3
```

Likewise, $\mathrm{NWScore}(\mathrm{rev}(\mathrm{CGCA}), \mathrm{rev}(Y))$ generates the following matrix:

```
      C   G   T   A   T
    0 -2  -4  -6  -8 -10
A  -2 -1  -3  -5  -4  -6
C  -4  0  -2  -4  -6  -5
G  -6 -2   2   0  -2  -4
C  -8 -4   0   1  -1  -3
```

Their last lines (after reversing the latter) and sum of those are respectively

```
ScoreL      = [ -8 -4  0 -2 -1 -3 ]
rev(ScoreR) = [ -3 -1  1  0 -4 -8 ]
Sum         = [-11 -5  1 -2 -5 -11]
```

The maximum (shown in bold) appears at `ymid = 2`, producing the partition $Y = \mathrm{TA} + \mathrm{TGC}$.

The entire Hirschberg recursion (which we omit for brevity) produces the following tree:

```
              (AGTACGCA,TATGC)
              /               \
        (AGTA,TA)             (CGCA,TGC)
         /     \              /        \
     (AG, )   (TA,TA)      (CG,TG)     (CA,C)
              /   \        /   \
           (T,T) (A,A)  (C,T) (G,G)
```

The leaves of the tree contain the optimal alignment.

See also

- Longest common subsequence

References

1. Hirschberg's algorithm (http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Hirsch/).
2. "The Algorithm" (http://www.cs.tau.ac.il/~rshamir/algmb/98/scribe/html/lec02/node10.html).
3. Hirschberg, D. S. (1975). "A linear space algorithm for computing maximal common subsequences". *Communications of the ACM*. **18** (6): 341–343. CiteSeerX 10.1.1.348.4774 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.348.4774). doi:10.1145/360825.360861 (https://doi.org/10.1145%2F360825.360861). MR 0375829 (https://www.ams.org/mathscinet-getitem?mr=0375829). S2CID 207694727 (https://api.semanticscholar.org/CorpusID:207694727).