Chapter 11.5-11.7

2015. 03. 17

ISA Lab

이호범

Contents

1. **String similarity**
   - **Approximate occurrence of *P* in *T***

2. **Local alignment**

String similarity

- **Approximate occurrence of *P* in *T***
  - *T* = Text (very long strings)
  - *P* = Patten (short strings)

T	…	t	c	a	c	b	d	b	d	j	…
P	-	-	-	-	c	x	d	-	-	-	-

String similarity

- **Approximate occurrence of *P* in *T***
  - Another important of global alignment
  - Definition
    - Given a parameter δ, a substring *T'* of *T* is said to be an ***approximate occurrence*** of *P* if and only if the optimal alignment of *P* to *T'* has value at least δ
  - Definition(on page 227)
    - $V(i, j)$
      - the value of the optimal alignment of prefixes $S_1[1..i]$ and $S_2[1..j]$
      - ~~$V(0, j) = \sum_{1 \le k \le j} s(-, S_2(k))$~~ → $V(0, j) = 0$
      - $V(i, 0) = \sum_{1 \le k \le i} s(S_2(k), -)$

String similarity

- **Approximate occurrence of *P* in *T***
  - *T* = Text (very long strings)
  - *P* = Patten (short strings)

T	…	t	c	a	c	b	d	b	d	j	…
P	-	-	-	-	c	x	d	-	-	-	-
score	0	0	0	0	2	-1	3	0	0	0	0

In this exmple, the value of 0 means that end space free is applied.

- And it is associated with *V(0, j) =0*

String similarity

- **Approximate occurrence of *P* in *T***
  - Theorem 11.6.2
    - There is an approximate occurrence of P in T ending at position j of T
      - if and only if $V(n, j) \geq \delta$

i \ j		1	2	…	k	…	j	…	m
		0	0	…	0		0	…	0
1									
…									
…									
…									
n							$\geq \delta$		

String similarity

- **Approximate occurrence of *P* in *T***
  - Theorem 11.6.2
    - Moreover, $T[k..j]$ is an approximate occurrence of P in T
      - if and only if $V(n, j) \geq \delta$
      - and there is a path of backpointers from cell (n, j) to cell $(0, k)$

$i \backslash j$		1	2	…	k	…	j	…	m
		0	0	…	0		0	…	0
1									
…									
…									
…									
n							$\geq \delta$		

String similarity

- **Approximate occurrence of *P* in *T***
  - $T[4..6]$ is an approximate occurrence of P in T
    - if and only if $V(3, 6) \geq \delta$

T	…	t	c	a	c	b	d	b	d	j	…
P	-	-	-	-	c	x	d	-	-	-	-
score	0	0	0	0	2	-1	3	0	0	0	0

j / *i*		t	c	*a*	***c***	***b***	***d***	*b*	*d*	*j*
		0	0	0	0	0	0	0	0	0
c										
x										
d							$\geq \delta$			

Local alignment

- **Local alignment: finding substrings of high similarity**
  - Given two strings S_1 and S_2, find substrings α and β of S_1 and S_2, respectively,
    - whose similarity(optimal global alignment value) is maximum
    - over all pairs of substrings from S_1 and S_2.

Local alignment

- **Local alignment: finding substrings of high similarity**
  - S_1: pqraxabcstvq   α: axabcs
  - S_2: xyaxbacsll   β: axbacs
  - match:2 / mismatch: -2 / space: -1

  - Optimal(global) alignment

a	x	a	b	-	c	s
a	x	-	b	a	c	s
2	2	-1	2	-1	2	2

  - Furthermore, over all choices of pairs of substrings, those two substrings have maximum similarity

Local alignment

- **Computing local alignment**
  - Naive
    - $\Theta\ (n^2\, m^2)$ pairs of substrings
    - $O\ (kl)$ bound on the time to align strings of lengths k and l
    - Result time: $O\ (n^3 m^3)$

  - The local alignment problem can be solved in $O\ (nm)$ time
    - Obtained by Temple Smith and Michael Waterman

Local alignment

- **Computing local alignment**
  - Assume that the global alignment of two empty strings has value zero
  - The local suffix alignment problem
    - a more restricted version of the local alignment problem
    - to find a (possibly empty) suffix α of $S_1[1..i]$ and a (possibly empty) suffix β of $S_2[1..j]$
      - such that $V(\alpha, \beta)$ is the maximum over all pairs of suffixes of $S_1[1..i]$ and $S_2[1..j]$ ($i \leq n$ and $j \leq m$)
    - $v(i, j)$
      - the value of the optimal local suffix alignment for the given index pair i, j

Local alignment

- **Computing local alignment**
  - The local suffix alignment problem
    - S_1 = abcxdex
    - S_2 = xxxcde
    - match: 2 / mismatch or space: -1
    - $v(3, 4)$

-	a	b	c
x	x	x	c
-1	-1	-1	2

    - $v(6, 6)$

a	b	c	x	-	d	e
	x	x	x	c	d	e
-1	-1	-1	2	-1	2	2

Local alignment

- **Computing local alignment**
  - The local suffix alignment problem
    - S_1 = abcxdex
    - S_2 = xxxcde
    - match: 2 / mismatch or space: -1
    - $v(3, 4)$

-	a	b	c
x	x	x	c
-1	-1	-1	2

> the meaning of 3 is that we consider only 3 characters of S_1

- $v(6, 6)$

a	b	c	x	-	d	e
	x	x	x	c	d	e
-1	-1	-1	2	-1	2	2

Local alignment

- **Computing local alignment**
  - The local suffix alignment problem
    - S_1 = abcxdex
    - S_2 = xxxcde
    - match: 2 / mismatch or space: -1
    - $v(3, 4)$

-	a	b	c
x	x	x	c
-1	-1	-1	2

> the meaning of 4 is that we consider only 4 characters of S_2

- $v(6, 6)$

a	b	c	x	-	d	e
	x	x	x	c	d	e
-1	-1	-1	2	-1	2	2

Local alignment

- **Computing local alignment**
  - Theorem 11.7.1
    - the relationship between the local alignment problem and the local suffix alignment problem as follows.
    - v^*
      - the value of an optimal solution to the local alignment problem
    - $\boldsymbol{v^* = \max[v(i, j): i \le n, j \le m]}$

      (> Proof is at textbook on 232)

Thus a solution to the local suffix alignment problem solves the local alignment problem

Local alignment

- **Computing local alignment**
  - Theorem 11.7.2

    In other words,
    - If i', j' is an index pair maximizing $v(i, j)$ over all i, j pairs,
    - then a pair of substrings solving the **local suffix alignment problem** for i', j' also solves the **local alignment problem**

Local alignment

- **How to solve the local suffix alignment problem**
  - Theorem 11.7.3
    - For $i > 0$ and $j > 0$, the proper recurrence for $v(i, j)$ is

$$v(i, 0) = v(0, j) = 0$$

$$\begin{aligned} v(i, j) = \max[&0, v(i\text{-}1, j\text{-}1) + s(S_1(i), S_2(j)), \\ &v(i\text{-}1, j) + s(S_1(i), \text{-}), v(i, j\text{-}1) + s(\text{-}, S_2(j))] \end{aligned}$$

    - The zero in the recurrence acting to “restart” the recurrence

Local alignment

- **How to solve the local suffix alignment problem**
  - S_1 = abcxdex
  - S_2 = xxxcde
  - match: 2 / mismatch or space: -1

$v(i, j)$		a	b	c	x	d	e	x
	0	0	0	0	0	0	0	0
x	0							
x	0							
x	0							
c	0							
d	0							
e	0							

Local alignment

- **How to solve the local suffix alignment problem**
  - S_1 = abcxdex
  - S_2 = xxxcde
  - match: 2 / mismatch or space: -1

$v(i, j)$		a	b	c	x	d	e	x
	0	0	0	0	0	0	0	0
x	0	0						
x	0							
x	0							
c	0							
d	0							
e	0							

Local alignment

- **How to solve the local suffix alignment problem**
  - S_1 = abcxdex
  - S_2 = xxxcde
  - match: 2 / mismatch or space: -1

$v(i, j)$		*a*	*b*	*c*	*x*	*d*	*e*	*x*
	0	0	0	0	0	0	0	0
x	0	0	0	0	2	1	0	2
x	0	0	0	0	2	1	0	2
x	0	0	0	0	2	1	0	2
c	0	0	0	2	1	1	0	1
d	0	0	0	1	1	3	2	1
e	0	0	0	0	0	2	5	4

Local alignment

- **How to solve the local suffix alignment problem**
  - S_1 = abcxdex
  - S_2 = xxxcde
  - match: 2 / mismatch or space: -1

$v(i, j)$		*a*	*b*	*c*	*x*	*d*	*e*	*x*
	0	0	0	0	0	0	0	0
x	0	0	0	0	2	1	0	2
x	0	0	0	0	2	1	0	2
x	0	0	0	0	2	1	0	2
c	0	0	0	2	1	1	0	1
d	0	0	0	1	1	3	2	1
e	0	0	0	0	0	2	**5**	4

Find largest value in any cell in the table, that is cell(*i**, *j**)

Local alignment

- **How to solve the local suffix alignment problem**
  - S_1 = abcxdex
  - S_2 = xxxcde
  - match: 2 / mismatch or space: -1

v(*i*, *j*)		*a*	*b*	***c***	***x***	***d***	***e***	*x*
	0	0	0	0	0	0	0	0
x	0	0	0	0	2	1	0	2
x	0	0	0	0	2	1	0	2
x	0	0	**0**	0	2	1	0	2
c	0	0	0	2	1	1	0	1
d	0	0	0	1	1	3	2	1
e	0	0	0	0	0	2	**5**	4

Tracing back from cell (*i**, *j**) until an entry*(i', j')* is reached zero.

Local alignment

- **How to solve the local suffix alignment problem**
  - S_1 = abcxdex
  - S_2 = xxxcde
  - match: 2 / mismatch or space: -1

$v(i, j)$		*a*	*b*	***c***	***x***	***d***	***e***	*x*
	0	0	0	0	0	0	0	0
x	0	0	0	0	2	1	0	2
x	0	0	0	0	2	1	0	2
x	0	0	**0**	0	2	1	0	2
c	0	0	0	2	1	1	0	1
d	0	0	0	1	1	3	2	1
e	0	0	0	0	0	2	**5**	4

Then the optimal local alignment substrings are α of $S_1[i'..\, i^*]$ & β of $S_2[\, j'..j^*]$

Local alignment

- **How to solve the local suffix alignment problem**
  - Theorem 11.7.4
    - For two strings S_1 and S_2 of lengths n and m,
    - the local alignment problem can be solved in $O(nm)$ time
      - the same time as for global alignment