Dynamic programming

CMPSC 465 - Yana Safonova

Sequence alignment

Edit distance recurrence

Look at the rightmost column:

Case 1

$$\begin{matrix} x_1 & \cdots & x_{i-1} & x_i \\ y_1 & \cdots & y_j & - \end{matrix}$$

Contributes 1 to the cost plus the cost of alignment $\begin{matrix} x_1 & \cdots & x_{i-1} \\ y_1 & \cdots & y_j \end{matrix}$

$$E(i,j) = 1 + E(i-1,j)$$

Case 2

$$\begin{matrix} x_1 & \cdots & x_i & - \\ y_1 & \cdots & y_{j-1} & y_j \end{matrix}$$

Contributes 1 to the cost plus the cost of alignment $\begin{matrix} x_1 & \cdots & x_i \\ y_1 & \cdots & y_{j-1} \end{matrix}$

$$E(i,j) = 1 + E(i,j-1)$$

Case 3

$$\begin{matrix} x_1 & \cdots & x_{i-1} & x_i \\ y_1 & \cdots & y_{j-1} & y_j \end{matrix}$$

$$E(i,j) = \begin{cases} E(i-1,j-1) & \text{if } x_i = y_j \\ 1 + E(i-1,j-1) & \text{otherwise} \end{cases}$$

Edit distance table: the complete version

$$E(i,j) = \min\{1 + E(i-1,j), 1 + E(i,j-1), \mathrm{diff}(i,j) + E(i-1,j-1)\},$$

$$\begin{array}{ccccccc}
E(0,0) & & E(0,1) & \cdots & E(0,n-1) & & E(0,n) \\
 & \searrow & \downarrow & & & & \\
E(1,0) & \rightarrow & E(1,1) & \cdots & \cdots & & \cdots \\
\vdots & & & & & & \\
 & & & \searrow & \downarrow & \searrow & \downarrow \\
E(m-1,0) & & & \rightarrow & E(m-1,n-1) & \rightarrow & E(m-1,n) \\
 & & & \searrow & \downarrow & \searrow & \downarrow \\
E(m,0) & & & \rightarrow & E(m,n-1) & \rightarrow & E(m,n)
\end{array}$$

Pseudocode

```
def EDIT_DISTANCE(x, y):
    for i = 0, ..., m:
        E(i, 0) = i;
    for j = 0, ..., n:
        E(0, j) = j;
    for i = 1, ..., m:
        for j = 1, ..., n:
            E(i, j) =
              min{1 + E(i − 1, j), 1 + E(i, j − 1), diff(i, j) + E(i − 1, j − 1)};
    return E(m, n);
```

Running time: O(mn)
Memory usage: O(mn)

Example

Indices / letters		0	1	2	3	4	5
		–	A	C	G	T	A
0	–	0	1	2	3	4	5
1	A	1	0	1	2	3	4
2	G	2	1	1	1	2	3
3	G	3	2	2	1	2	3
4	T	4	3	3	2	1	2

→ - insertion to AGGT
↓ - insertion to ACGTA
↘ - match / mismatch

Each path from the green cell to the purple cell corresponds to an alignment

edit distance between AGGT and ACGTA = 2

Edit distance modification

We use an extra table prev to record where each entry of $E(i,j)$ was coming from:

$$\mathrm{prev}(i,j) = \begin{cases} (i-1,j) & \text{if } E(i,j) = 1 + E(i-1,j) \\ (i,j-1) & \text{if } E(i,j) = 1 + E(i,j-1) \\ (i-1,j-1) & \text{if } E(i,j) = \mathrm{diff}(i,j) + E(i-1,j-1) \end{cases}$$

def PRING_ALIGNMENT(x, y, prev)**:**

- Set $i = m, j = n$;
- **while** $i \geq 1$ *and* $j \geq 1$**:**
  - **if** $\mathrm{prev}(i,j) = (i-1, j-1)$**:**
    - $\mathrm{print\_back}\binom{y_i}{x_i}$;
    - $i = i - 1$, $j = j - 1$;
  - **if** $\mathrm{prev}(i,j) = (i-1, j)$**:**
    - $\mathrm{print\_back}\binom{-}{x_i}$;
    - $i = i - 1$;
  - **if** $\mathrm{prev}(i,j) = (i, j-)$**:**
    - $\mathrm{print\_back}\binom{y_j}{-}$;
    - $j = j - 1$;

Example: how to reconstruct the best alignment?

Indices / letters		0	1	2	3	4	5
		–	A	C	G	T	A
0	–	0 (-, -)	1 (0, 0)	2 (0, 1)	3 (0, 2)	4 (0, 3)	5 (0, 4)
1	A	1 (0, 0)	0 (0, 0)	1 (1, 1)	2 (1, 2)	3 (1, 3)	4 (1, 4)
2	G	2 (1, 0)	1 (1, 1)	1 (1, 1)	1 (1, 2)	2 (2, 3)	3 (2, 4)
3	G	3 (2, 0)	2 (2, 1)	2 (2, 2)	1 (2, 2)	2 (2, 3)	3 (3, 4)
4	T	4 (3, 0)	3 (3, 1)	3 (3, 1)	2 (3, 3)	1 (3, 3)	2 (4, 4)

```
ACGTA
|.||
AGGT-
```

From edit distance to alignment score

In edit distance, editing operations can have identical costs:

Match = 0

Mismatch = 1

Insertion / Deletion = 1

$$E(i,j) = \min\{1 + E(i-1,j), 1 + E(i,j-1), \mathrm{diff}(i,j) + E(i-1,j-1)\},$$

where

$$\mathrm{diff}(i,j) = \begin{cases} 1 & \text{if } x_i \neq y_j \\ 0 & \text{otherwise} \end{cases}$$

From edit distance to alignment score

In edit distance, editing operations can have different costs:

Match = reward R, R > 0

Mismatch = penalty P1, P1 < 0

Insertion / Deletion = penalty P2, P2 < 0

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1)\}$$

where

$$\mathit{diff}(i, j) = \text{R if } x_i = y_j \text{ or P1 otherwise}$$

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–					
1	A					
2	G					

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3				
2	G	–6				

Score(i, 0) = Score(0, i) = –3 * i

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2			
2	G	–6				

–3 – 3 or 0 + 2 or –3 – 3

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1		
2	G	–6				

2 – 3 or –3 – 1 or –6 – 3

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	
2	G	–6				

–1 – 3 or –6 + 2 or –9 – 3

Tie. The choice depends on implementation

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	–7
2	G	–6	–1	1		

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	–7
2	G	–6	–1	1	–2	

1 – 3 or –1 – 1 or –4 – 3

Tie. The choice depends on implementation

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	–7
2	G	–6	–1	1	–2	–2

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	–7
2	G	–6	–1	1	–2	–2

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	–7
2	G	–6	–1	1	–2	–2

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

		0	1	2	3	4
		–	A	C	A	G
0	–	0	–3	–6	–9	–12
1	A	–3	2	–1	–4	–7
2	G	–6	–1	1	–2	–2

ACAG
A--G

Local alignment vs global alignment

A short string S

Long string G1: A (repeated many times) + S + C (repeated many times)

Long string G2: G (repeated many times) + S + T (repeated many times)

```
G1 =   AAAAAAAAAAAAAAA...AGCGCGAGCGT...CCCCCCCCCCCCCCCCC
G2 =   GGGGGGGGGGGGGGG...AGCGCGAGCGT...TTTTTTTTTTTTTTTTT
```

Local alignment vs global alignment

A short string S

Long string G1: A (repeated many times) + S + C (repeated many times)

Long string G2: G (repeated many times) + S + T (repeated many times)

```
G1 =   AAAAAAAAAAAAAAA...AGCGCGAGCGT...CCCCCCCCCCCCCCCCC
G2 =   GGGGGGGGGGGGGGG...AGCGCGAGCGT...TTTTTTTTTTTTTTTTT
```

The optimal alignment will depend on values mismatch (P1) and insertion / deletion (P2) penalties
If P1 < P2, then As will be aligned with Gs and Cs with Ts

Local alignment vs global alignment

A short string S

Long string G1: A (repeated many times) + S + C (repeated many times)

Long string G2: G (repeated many times) + S + T (repeated many times)

```
G1 =   AAAAAAAAAAAAAAA...AGCGCGAGCGT...CCCCCCCCCCCCCCCCC
G2 =   GGGGGGGGGGGGGGG...AGCGCGAGCGT...TTTTTTTTTTTTTTTTT
```

- How can we modify the scoring function to find the local similarities?
- I.e., can we find a local alignment instead of the global alignment (end-to-end)

Local alignment recurrence

Match = reward R, R > 0

Mismatch = penalty P1, P1 < 0

Insertion / Deletion = penalty P2, P2 < 0

Restart if score < 0

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

where

$$\mathit{diff}(i, j) = \text{R if } x_i = y_j \text{ or P1 otherwise}$$

Local alignment recurrence

Match = reward R, R > 0

Mismatch = penalty P1, P1 < 0

Insertion / Deletion = penalty P2, P2 < 0

Restart if score < 0

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

where

$$\mathit{diff}(i, j) = \text{R if } x_i = y_j \text{ or P1 otherwise}$$

This modification will allow us to similar aligns substrings of the original strings

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0				
2	G	0				

Score(i, 0) = Score(0, i) = 0

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2			
2	G	0				

0 – 3 or 0 + 2 or 0 – 3 or 0

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i,j) = \max\{\text{P2} + \text{Score}(i-1,j), \text{P2} + \text{Score}(i,j-1), \mathit{diff}(i,j) + \text{Score}(i-1,j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0		
2	G	0				

2 – 3 or 0 – 1 or 0 – 3 or 0

No link to the previous cell!

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	
2	G	0				

0 – 3 or 0 + 2 or 0 – 3 or 0

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i,j) = \max\{\text{P2} + \text{Score}(i-1,j), \text{P2} + \text{Score}(i,j-1), \mathit{diff}(i,j) + \text{Score}(i-1,j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	0
2	G	0	0	1	0	4

0 – 3 or 2 + 2 or 0 – 3 or 0

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	0
2	G	0	0	1	0	4

How to retrieve the actual alignment?

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	0
2	G	0	0	1	0	4

How to retrieve the actual alignment?

- Find a cell with the highest score

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	0
2	G	0	0	1	0	4

How to retrieve the actual alignment?

- Find a cell with the highest score
- Do backtracking

Alignment using scoring scheme

Match = 2
Mismatch = –1
Insertion / deletion = –3

$$\text{Score}(i, j) = \max\{\text{P2} + \text{Score}(i-1, j), \text{P2} + \text{Score}(i, j-1), \mathit{diff}(i, j) + \text{Score}(i-1, j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	0
2	G	0	0	1	0	4

How to retrieve the actual alignment?

- Find a cell with the highest score
- Do backtracking until reach 0

Alignment using scoring scheme

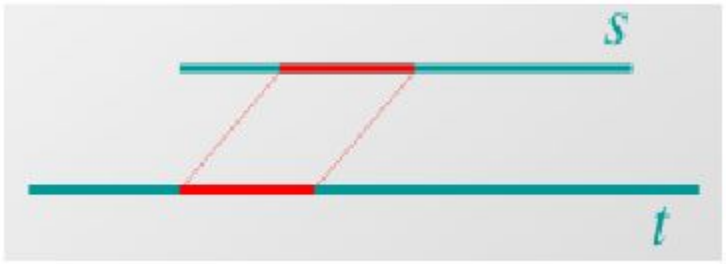
Match = 2
Mismatch = –1
Insertion / deletion = –3

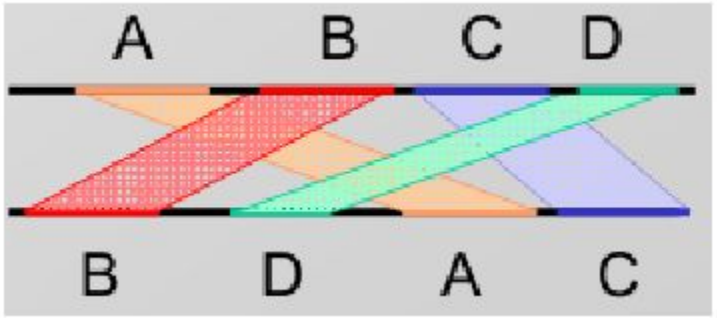
$$\text{Score}(i,j) = \max\{\text{P2} + \text{Score}(i-1,j), \text{P2} + \text{Score}(i,j-1), \mathit{diff}(i,j) + \text{Score}(i-1,j-1), \mathbf{0}\}$$

		0	1	2	3	4
		–	A	C	A	G
0	–	0	0	0	0	0
1	A	0	2	0	2	0
2	G	0	0	1	0	4

ACAG
--AG

Local alignment applications





- Detection of similar fragments of dissimilar strings
- Detection of rearrangements

Global vs local alignments

Strings A, B, C

G1 = A + B + C

G2 = B + C + A

Global vs local alignments

Strings A, B, C

G1 = A + B + C

G2 = B + C + A

A = GCGCGA
B = TGGCA
C = GATGC

The same scoring scheme:

Global:

```
GCGC-GATG-GCAGATGC
..|. |||| || | .|.
TGGCAGATGCGC-G-CGA
```

Local:

```
GCGCGATGGCAGATGC------
      ||||||||||
------TGGCAGATGCGCGCGA
```