

Principles of Computer Science II

Sequence Similarity

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 23



Edit Distance

- ▶ We looked for repeating patterns within DNA sequences.
- ▶ How can we measure the similarity between different sequences?
- ▶ We use the notion of Vladimir Levenshtein introduced in 1966
- ▶ **Edit distance** – the minimum number of editing operations needed to transform one string into another (insert/delete symbol or substitute one symbol for another).

Alignment of ATATATAT vs TATATATA

```
A T A T A T A T -
  : : : : : : :
- T A T A T A T A
```



Sequence Similarity

Alignment of ATATATAT vs TATAAT

```
A T A T A T A T
  : : : : : : :
- T A T A - A T
```



Sequence Similarity

Alignment of TGCATAT vs ATCCGAT

```
TGCATAT
  ↓ delete last T
TGCATA
  ↓ delete last A
TGCAT
  ↓ insert A at the front
ATGCAT
  ↓ substitute C for G in the third position
ATCCAT
  ↓ insert a G before the last A
ATCCGAT
```

Five operations.



Sequence Similarity

Alignment of TGCATAT vs ATCCGAT

TGCATAT
↓ insert A at the front
ATGCATAT
↓ delete T in the sixth position
ATGCAAT
↓ substitute G for A in the fifth position
ATGCCAT
↓ substitute C for G in the third position
ATCCGAT

Four operations.



Edit Distance

- ▶ Vladimir Levenshtein defined the notion of **Edit distance**
- ▶ Did not provide an algorithm to compute it.



Edit Distance Algorithm using Dynamic Programming

- ▶ Assume two strings:
 - ▶ v (of n characters)
 - ▶ w (of m characters)
- ▶ The alignment of v, w is a two-row matrix such that
 - ▶ first row: contains the characters of v (in order)
 - ▶ second row: contains the characters of w (in order)
 - ▶ spaces are interspersed throughout the table.
- ▶ Characters in each string appear in order, though not necessarily adjacently.

A	T	-	G	T	T	A	T	-
A	T	C	G	T	-	A	-	C

- ▶ No column contains spaces in both rows.
- ▶ At most $n + m$ columns.



Edit Distance Algorithm using Dynamic Programming

A	T	-	G	T	T	A	T	-
A	T	C	G	T	-	A	-	C

- ▶ **Matches** – columns with the same letter,
- ▶ **Mismatches** – columns with different letters.
- ▶ Columns containing one space are called **indels**
 - ▶ Space on top row: **insertions**
 - ▶ Space on bottom row: **deletions**

$$\# \text{ matches} + \# \text{ mismatches} + \# \text{ indels} < n + m$$



Representing the rows

v	A	T	-	G	T	T	A	T	-
w	A	T	C	G	T	-	A	-	C

- ▶ One way to represent v
 - ▶ AT-CGTAT-
- ▶ One way to represent w
 - ▶ ATCGT-A-C
- ▶ Another way to represent v
 - ▶ AT-CGTAT-
 - ▶ 122345677
 - ▶ number of symbols of v present up to a given position
- ▶ Similarly, to represent w
 - ▶ ATCGT-A-C
 - ▶ 123455667



Representing the rows

v	A	T	-	G	T	T	A	T	-
w	A	T	C	G	T	-	A	-	C

v	1	2	2	3	4	5	6	7	7
w	1	2	3	4	5	5	6	6	7

can be viewed as a coordinate in 2-dimensional $n \times m$ grid:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} \begin{pmatrix} 4 \\ 5 \end{pmatrix} \begin{pmatrix} 5 \\ 5 \end{pmatrix} \begin{pmatrix} 6 \\ 6 \end{pmatrix} \begin{pmatrix} 7 \\ 6 \end{pmatrix} \begin{pmatrix} 7 \\ 7 \end{pmatrix}$$

The entire alignment is simply a path:

$$(0, 0) \rightarrow (1, 1) \rightarrow (2, 2) \rightarrow (2, 3) \rightarrow (3, 4) \rightarrow (4, 5) \rightarrow (5, 5) \rightarrow (6, 6) \rightarrow (7, 6) \rightarrow (7, 7)$$

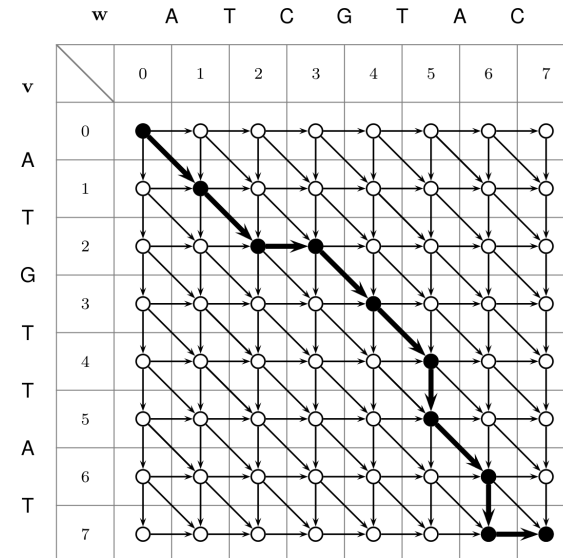


Edit distance graph

- ▶ **Edit graph**: a grid of n, m size.
- ▶ The edit graph will help us in calculating the edit distance.
- ▶ Alignment: a path from $(0, 0)$ to (n, m) .
- ▶ Every alignment corresponds to a path in the edit graph.
- ▶ Diagonal movement at point i, j correspond to column $\begin{pmatrix} v_i \\ w_j \end{pmatrix}$
- ▶ Horizontal movement correspond to column $\begin{pmatrix} - \\ w_j \end{pmatrix}$
- ▶ Vertical movement correspond to column $\begin{pmatrix} v_i \\ - \end{pmatrix}$



Edit distance graph



\swarrow \searrow \rightarrow \swarrow \searrow \downarrow \swarrow \downarrow \rightarrow
 A T - G T T A T -
 A T C G T - A - C



Profile most-frequent k-mer

```
1 def edit_distance(s1, s2):
2     m=len(s1)+1
3     n=len(s2)+1
4
5     tbl = {}
6     for i in range(m): tbl[i,0]=i
7     for j in range(n): tbl[0,j]=j
8     for i in range(1, m):
9         for j in range(1, n):
10            cost = 0 if s1[i-1] == s2[j-1] else 1
11            tbl[i, j] = min(tbl[i, j-1]+1, tbl[i-1, j]+1,
12                           tbl[i-1, j-1]+cost)
13
14     return tbl[i, j]
```



Profile most-frequent k-mer

```
1 def levenshteinDistance(s1, s2):
2     if len(s1) > len(s2):
3         s1, s2 = s2, s1
4
5     distances = range(len(s1) + 1)
6     for i2, c2 in enumerate(s2):
7         distances_ = [i2+1]
8         for i1, c1 in enumerate(s1):
9             if c1 == c2:
10                distances_.append(distances[i1])
11            else:
12                distances_.append(1 + min((distances[i1],
13                                           distances[i1 + 1], distances_[-1])))
14        distances = distances_
15    return distances[-1]
```

