

Algorithms for Computational Biology

Zsuzsanna Lipták

Masters in Molecular and Medical Biotechnology
a.a. 2015/16, fall term

String Distance Measures

Similarity vs. distance

Two ways of measuring the same thing:

1. How **similar** are two strings?
 2. How **different** are two strings?
-
1. **Similarity**: the **higher** the value, the closer the two strings.
 2. **Distance**: the **lower** the value, the closer the two strings.

Similarity vs. distance

Example

s = TATTACTATC

t = CATTAGTATC

- number of equal positions: $|\{i : s_i = t_i\}| = 8$ (out of 10)
80% **similarity** ($s = t$ if 100%, i.e. if **high**)
- number of different positions: $|\{i : s_i \neq t_i\}| = 2$ (out of 10)
Hamming **distance** 2 ($s = t$ if 0, i.e. if **low**)

(Note that both are defined only if $|s| = |t|$.)

Alignment score and edit distance

Edit operations

- **substitution**: a becomes b , where $a \neq b$
- **deletion**: delete character a
- **insertion**: insert character a

Often one views alignments in this way: thinking about the **changes** that happened turning one string into the other, e.g.

ACCT

CACT

2 substitutions

ACCT--

--CACT

2 deletions,
1 substitution,
2 insertions

-ACCT

CA-CT

1 insertion,
1 deletion

The edit distance

Edit distance, also called **Levenshtein distance**, or unit-cost edit distance (Levenshtein, 1965)

Definition

The edit distance $d(s, t)$ is the **minimum** number of edit operations needed to transform s into t .

Example

$s = \text{TACAT}$, $t = \text{TGATAT}$

- $\text{TACAT} \xrightarrow{\text{subst}} \text{GACAT} \xrightarrow{\text{del}} \text{GAAT} \xrightarrow{\text{ins}} \text{TGAAT} \xrightarrow{\text{ins}} \text{TGATAT}$

The edit distance

Edit distance, also called **Levenshtein distance**, or unit-cost edit distance (Levenshtein, 1965)

Definition

The edit distance $d(s, t)$ is the **minimum** number of edit operations needed to transform s into t .

Example

$s = \text{TACAT}$, $t = \text{TGATAT}$

- $\text{TACAT} \xrightarrow{\text{subst}} \text{GACAT} \xrightarrow{\text{del}} \text{GAAT} \xrightarrow{\text{ins}} \text{TGAAT} \xrightarrow{\text{ins}} \text{TGATAT}$ 4 edit op's

The edit distance

Edit distance, also called **Levenshtein distance**, or unit-cost edit distance (Levenshtein, 1965)

Definition

The edit distance $d(s, t)$ is the **minimum** number of edit operations needed to transform s into t .

Example

$s = \text{TACAT}$, $t = \text{TGATAT}$

- $\text{TACAT} \xrightarrow{\text{subst}} \text{GACAT} \xrightarrow{\text{del}} \text{GAAT} \xrightarrow{\text{ins}} \text{TGAAT} \xrightarrow{\text{ins}} \text{TGATAT}$ 4 edit op's
- $\text{TACAT} \xrightarrow{\text{ins}} \text{TGACAT} \xrightarrow{\text{subst}} \text{TGATAT}$

The edit distance

Edit distance, also called **Levenshtein distance**, or unit-cost edit distance (Levenshtein, 1965)

Definition

The edit distance $d(s, t)$ is the **minimum** number of edit operations needed to transform s into t .

Example

$s = \text{TACAT}$, $t = \text{TGATAT}$

- $\text{TACAT} \xrightarrow{\text{subst}} \text{GACAT} \xrightarrow{\text{del}} \text{GAAT} \xrightarrow{\text{ins}} \text{TGAAT} \xrightarrow{\text{ins}} \text{TGATAT}$ 4 edit op's
- $\text{TACAT} \xrightarrow{\text{ins}} \text{TGACAT} \xrightarrow{\text{subst}} \text{TGATAT}$ 2 edit op's

The edit distance

Edit distance, also called **Levenshtein distance**, or unit-cost edit distance (Levenshtein, 1965)

Definition

The edit distance $d(s, t)$ is the **minimum** number of edit operations needed to transform s into t .

Example

$s = \text{TACAT}$, $t = \text{TGATAT}$

- $\text{TACAT} \xrightarrow{\text{subst}} \text{GACAT} \xrightarrow{\text{del}} \text{GAAT} \xrightarrow{\text{ins}} \text{TGAAT} \xrightarrow{\text{ins}} \text{TGATAT}$ 4 edit op's
- $\text{TACAT} \xrightarrow{\text{ins}} \text{TGACAT} \xrightarrow{\text{subst}} \text{TGATAT}$ 2 edit op's
- $\text{TACAT} \xrightarrow{\text{ins}} \text{TGACAT} \xrightarrow{\text{subst}} \text{TGAGAT} \xrightarrow{\text{subst}} \text{TGATAT}$

The edit distance

Edit distance, also called **Levenshtein distance**, or unit-cost edit distance (Levenshtein, 1965)

Definition

The edit distance $d(s, t)$ is the **minimum** number of edit operations needed to transform s into t .

Example

$s = \text{TACAT}$, $t = \text{TGATAT}$

- $\text{TACAT} \xrightarrow{\text{subst}} \text{GACAT} \xrightarrow{\text{del}} \text{GAAT} \xrightarrow{\text{ins}} \text{TGAAT} \xrightarrow{\text{ins}} \text{TGATAT}$ 4 edit op's
- $\text{TACAT} \xrightarrow{\text{ins}} \text{TGACAT} \xrightarrow{\text{subst}} \text{TGATAT}$ 2 edit op's
- $\text{TACAT} \xrightarrow{\text{ins}} \text{TGACAT} \xrightarrow{\text{subst}} \text{TGAGAT} \xrightarrow{\text{subst}} \text{TGATAT}$ 3 edit op's

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

- TACAT $\xrightarrow{\text{subst}}$ GACAT $\xrightarrow{\text{del}}$ GAAT $\xrightarrow{\text{ins}}$ TGAAT $\xrightarrow{\text{ins}}$ TGATAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGAGAT $\xrightarrow{\text{subst}}$ TGATAT

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

- TACAT $\xrightarrow{\text{subst}}$ GACAT $\xrightarrow{\text{del}}$ GAAT $\xrightarrow{\text{ins}}$ TGAAT $\xrightarrow{\text{ins}}$ TGATAT -TAC-AT
TGA-TAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGAGAT $\xrightarrow{\text{subst}}$ TGATAT

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

- TACAT $\xrightarrow{\text{subst}}$ GACAT $\xrightarrow{\text{del}}$ GAAT $\xrightarrow{\text{ins}}$ TGAAT $\xrightarrow{\text{ins}}$ TGATAT -TAC-AT
TGA-TAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT T-ACAT
TGATAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGAGAT $\xrightarrow{\text{subst}}$ TGATAT

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

• TACAT $\xrightarrow{\text{subst}}$ GACAT $\xrightarrow{\text{del}}$ GAAT $\xrightarrow{\text{ins}}$ TGAAT $\xrightarrow{\text{ins}}$ TGATAT -TAC-AT
TGA-TAT

• TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT T-ACAT
TGATAT

• TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGAGAT $\xrightarrow{\text{subst}}$ TGATAT ???

Alignments vs. edit operations

But every alignment corresponds to a series of operations:

- match \mapsto do nothing
- mismatch \mapsto substitution
- gap below \mapsto deletion
- gap on top \mapsto insertion

Example

T-ACAT-

TGAT-AT

Alignments vs. edit operations

But every alignment corresponds to a series of operations:

- match \mapsto do nothing
- mismatch \mapsto substitution
- gap below \mapsto deletion
- gap on top \mapsto insertion

Example

T-ACAT-

TGAT-AT

TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT $\xrightarrow{\text{del}}$ TGATT $\xrightarrow{\text{subst}}$ TGATA $\xrightarrow{\text{ins}}$ TGATAT

Alignments vs. edit operations

Take the following scoring function: $match = 0$, $mismatch = -1$, $gap = -1$.
If alignment \mathcal{A} corresponds to the series of operations \mathcal{S} , then:

$$\text{score}(\mathcal{A}) = -|\mathcal{S}|$$

where $|\mathcal{S}| =$ no. of operations in \mathcal{S} .

Example

- TACAT $\xrightarrow{\text{subst}}$ GACAT $\xrightarrow{\text{del}}$ GAAT $\xrightarrow{\text{ins}}$ TGAAT $\xrightarrow{\text{ins}}$ TGATAT
-TAC-AT
TGA-TAT
- TACAT $\xrightarrow{\text{ins}}$ TGACAT $\xrightarrow{\text{subst}}$ TGATAT
T-ACAT
TGATAT

Minimum length series of edit operations

We are looking for a series of operations of **minimum length** (= shortest):

$$\text{dist}(s, t) = \min\{|\mathcal{S}| : \mathcal{S} \text{ is a series of operations transforming } s \text{ into } t\}$$

N.B.

There may be more than one series of op's of minimum length, but the **length** is unique.

Exercises on edit distance

Exercises

- If t is a substring of s , then what is $dist(s, t)$?
- What is $dist(s, \epsilon)$?
- If we can transform s into t by using only deletions, then what can we say about s and t ?
- If we can transform s into t by using only substitutions, then what can we say about s and t ?

What is a distance?

A **distance function** (metric) on a set X is a function $d : X \times X \rightarrow \mathbb{R}$ s.t. for all $x, y, z \in X$:

1. $d(x, y) \geq 0$, and $(d(x, y) = 0 \Leftrightarrow x = y)$ (non-negative, identity of indiscernibles)
2. $d(x, y) = d(y, x)$ (symmetric)
3. $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

What is a distance?

A **distance function** (metric) on a set X is a function $d : X \times X \rightarrow \mathbb{R}$ s.t. for all $x, y, z \in X$:

1. $d(x, y) \geq 0$, and $(d(x, y) = 0 \Leftrightarrow x = y)$ (non-negative, identity of indiscernibles)
2. $d(x, y) = d(y, x)$ (symmetric)
3. $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality)

Examples

- Euclidean distance on \mathbb{R}^2 : $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
where $x = (x_1, x_2), y = (y_1, y_2)$
- Manhattan distance on \mathbb{R}^2 : $d(x, y) = |x_1 - y_1| + |x_2 - y_2|$
- Hamming distance on Σ^n : $d_H(s, t) = \{i : s_i \neq t_i\}$.

The edit distance is a metric

Claim: The edit distance is a metric (distance function).

Proof: Let $s, t, u \in \Sigma^*$ (strings over Σ):

1. $dist(s, t) \geq 0$: to transform s to t , we need 0 or more edit op's. Also, we can transform s into t with 0 edit op's if and only if $s = t$.
2. Since every edit operation can be inverted, we get $dist(s, t) = dist(t, s)$.
3. (by contradiction) Assume that $dist(s, u) + dist(u, t) < dist(s, t)$, and \mathcal{S} transforms s into u in $dist(s, u)$ steps, and \mathcal{S}' transforms u into t in $dist(u, t)$ steps. Then the series of op's $\mathcal{S}' \circ \mathcal{S}$ (first \mathcal{S} , then \mathcal{S}') transforms s into t , but is shorter than $dist(s, t)$, a contradiction to the definition of $dist$.

(**Exercise:** Show that the Hamming distance is a metric.)

Computing the edit distance

Note first that we can assume that edit operations happen left-to-right. As for computing an optimal alignment, we look at what happens to the last characters. Transforming s into t can be done in one of 3 ways:

1. transform $s_1 \dots s_{n-1}$ into t and then delete last character of s
2. if $s_n = t_m$: transform $s_1 \dots s_{n-1}$ into $t_1 \dots t_{m-1}$
if $s_n \neq t_m$:
transform $s_1 \dots s_{n-1}$ into $t_1 \dots t_{m-1}$ and substitute s_n with t_m
3. transform s into $t_1 \dots t_{m-1}$ and insert t_m

So again we can use Dynamic Programming!

Computing the edit distance

We will need a DP-table (matrix) E of size $(n + 1) \times (m + 1)$ (where $n = |s|$ and $m = |t|$).

Definition: $E(i, j) = \text{dist}(s_1 \dots s_i, t_1 \dots t_j)$

Computation of $E(i, j)$:

- Fill in first row and column: $E(0, j) = j$ and $E(i, 0) = i$
- for $i, j > 0$: now $E(i, j)$ is the **minimum** of 3 entries plus 1 (top and left) or plus 0/plus 1, depending on whether current chars are the same or different
- return entry on bottom right $E(n, m)$
- backtrack for shortest series of edit operations

Algorithm for computing the edit distance

Algorithm *DP algorithm for edit distance*

Input: strings s, t , with $|s| = n, |t| = m$

Output: value $dist(s, t)$

1. **for** $j = 0$ to m **do** $E(0, j) \leftarrow j$;

2. **for** $i = 1$ to n **do** $E(i, 0) \leftarrow i$;

3. **for** $i = 1$ to n **do**

4. **for** $j = 1$ to m **do**

$$E(i, j) \leftarrow \min \begin{cases} E(i-1, j) + 1 \\ E(i-1, j-1) & \text{if } s_i = t_j \\ E(i-1, j-1) + 1 & \text{if } s_i \neq t_j \\ E(i, j-1) + 1 \end{cases}$$

5. **return** $E(n, m)$;

Analysis

- **Space:** $O(nm)$ for the DP-table
- **Time:**
 - computing $dist(s, t)$: $3nm + n + m + 1 \in O(nm)$
(resp. $O(n^2)$ if $n = m$)
 - finding an optimal series of edit op's: $O(n + m)$
(resp. $O(n)$ if $n = m$)

Again alignment vs. edit distance

sim(*s*, *t*) vs. *dist*(*s*, *t*)

Recall the scoring function from before:

match = 0, *mismatch* = -1, *gap* = -1. Then we have:

$$\textit{sim}(s, t) = -\textit{dist}(s, t)$$

(This seems obvious but it actually needs to be proved. Formal proof see Setubal & Meidanis book, Sec. 3.6.1.)

Again alignment vs. edit distance

sim(s, t) vs. *dist(s, t)*

Recall the scoring function from before:

match = 0, *mismatch* = -1, *gap* = -1. Then we have:

$$\textit{sim}(s, t) = -\textit{dist}(s, t)$$

(This seems obvious but it actually needs to be proved. Formal proof see Setubal & Meidanis book, Sec. 3.6.1.)

General cost functions

General cost edit distance: different edit operations can have different cost (but some conditions must hold, e.g. $\text{cost}(\text{insert}) = \text{cost}(\text{delete})$, why?).

Also computable with same algorithm in same time and space.

LCS distance

Given two strings s and t ,

$$LCS(s, t) = \max\{|u| : u \text{ is a subsequence of } s \text{ and } t\}$$

is the length of a longest common subsequence of s and t .

Example

Let $s = \text{TACAT}$ and $t = \text{TGATAT}$

LCS distance

Given two strings s and t ,

$$LCS(s, t) = \max\{|u| : u \text{ is a subsequence of } s \text{ and } t\}$$

is the length of a longest common subsequence of s and t .

Example

Let $s = \text{TACAT}$ and $t = \text{TGATAT}$, then we have $LCS(s, t) = 4$.

$s = \text{TACAT}$, $t = \text{TGATAT}$

LCS distance

Given two strings s and t ,

$$LCS(s, t) = \max\{|u| : u \text{ is a subsequence of } s \text{ and } t\}$$

is the length of a longest common subsequence of s and t .

Example

Let $s = \text{TACAT}$ and $t = \text{TGATAT}$, then we have $LCS(s, t) = 4$.

$s = \text{TACAT}$, $t = \text{TGATAT}$

LCS-distance

$$d_{LCS}(s, t) = |s| + |t| - 2LCS(s, t)$$

LCS distance

Given two strings s and t ,

$$LCS(s, t) = \max\{|u| : u \text{ is a subsequence of } s \text{ and } t\}$$

is the length of a longest common subsequence of s and t .

Example

Let $s = \text{TACAT}$ and $t = \text{TGATAT}$, then we have $LCS(s, t) = 4$.

$s = \text{TACAT}$, $t = \text{TGATAT}$

LCS-distance

$$d_{LCS}(s, t) = |s| + |t| - 2LCS(s, t)$$

Example

We have $d_{LCS}(s, t) = 5 + 6 - 2 \cdot 4 = 3$.

LCS distance

$$d_{LCS}(s, t) = |s| + |t| - 2LCS(s, t)$$

N.B.

There may be more than one longest common subsequence, but the *length* $LCS(s, t)$ is unique! E.g. $s' = TAACAT$, $t' = ATCTA$, then $LCS(s', t') = 3$, and ACA, TCA, TCT, ACT are all longest common subsequences.

LCS distance

In the examples above, we have $d_{LCS}(s, t) = 5 + 6 - 2 \cdot 4 = 3$, and $d_{LCS}(s', t') = 6 + 5 - 2 \cdot 3 = 5$.

Exercise (*)

(1) Prove that d_{LCS} is a metric. (2) Find a DP-algorithm that computes $LCS(s, t)$.

(*) means: for particularly motivated students

Summary: Similarity and distance

Similarity measures for strings

- $sim(s, t)$ - score of an optimal alignment of s, t
- percent similarity (only for equal length strings!)

Distance measures for strings

- edit distance (Levenshtein distance) - minimum no. of edit operations to transform s into t
- Hamming distance (only for equal length strings!)
- LCS distance
- (q -gram distance)

Summary: Similarity and distance

- two ways of expressing the same thing (similarity vs. distance)
- **similarity**: the **higher** the value, the more similar the strings
- **distance**: the **lower** the value, the more similar the strings
- optimal alignment \cong minimum length edit transformation
- both computable in quadratic time and quadratic space