Bioinformatics Algorithms

(Fundamental Algorithms, module 2)

Zsuzsanna Lipták

Masters in Medical Bioinformatics
academic year 2017/18, spring 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?

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

- percentage of equal positions: $|\{i \, : \, s_i = t_i\}| = 8$ out of $10 = 80\%$
  $s = t$ if 100% similar, i.e. if highest possible
  This is called percent similarity in biology.

Similarity vs. distance

Example

s = TATTACTATC

t = CATTAGTATC

- percentage of equal positions: $|\{i : s_i = t_i\}| = 8$ out of $10 = 80\%$
  $s = t$ if 100% similar, i.e. if highest possible
  This is called percent similarity in biology.
- number of different positions: $|\{i : s_i \neq t_i\}| = 2$ (out of 10)
  $s = t$ if 0, i.e. if lowest possible
  This is called Hamming distance of the two strings.

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

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

```
ACCT
CACT
```

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

```
ACCT
CACT
```

2 substitutions

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

```
ACCT        ACCT--
CACT        --CACT
```

2 substitutions

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

`ACCT` `CACT`	`ACCT--` `--CACT`
2 substitutions	2 deletions, 1 substition, 2 insertions

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

ACCT CACT	ACCT-- --CACT	-ACCT CA-CT
2 substitutions	2 deletions, 1 substition, 2 insertions	

From alignments to distance

Edit operations

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

One often views alignments in this way: thinking about the changes that happened turning one string into the other (evolution, typos, ...). E.g.

ACCT CACT	ACCT-- --CACT	-ACCT CA-CT
2 substitutions	2 deletions, 1 substition, 2 insertions	1 insertion, 1 deletion

The edit distance

(Unit cost) edit distance, also called Levenshtein distance (Levenshtein, 1965).

Definition

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

Example

s = TACAT, t = TGATAT

The edit distance

(Unit cost) edit distance, also called Levenshtein distance (Levenshtein, 1965).

Definition

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

Example

s = TACAT, t = TGATAT

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

The edit distance

(Unit cost) edit distance, also called Levenshtein distance (Levenshtein, 1965).

Definition

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

Example

s = TACAT, t = TGATAT

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

The edit distance

(Unit cost) edit distance, also called Levenshtein distance (Levenshtein, 1965).

Definition

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

Example

s = TACAT, t = TGATAT

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

Minimum length series of edit operations

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

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

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 $d_{edit}(s, t)$?
- What is $d_{edit}(s, \epsilon)$?
- 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?
- If we can transform s into t with k edit operations, then what can we say about $d_{edit}(s, t)$?

What is a distance?

The mathematical formalization of *distance* is *metric*:
A metric on a set X is a function $d : X \times X \to \mathbb{R}$ 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?

The mathematical formalization of *distance* is *metric*:
A metric on a set X is a function $d : X \times X \to \mathbb{R}$ 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.

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

1. $d_{edit}(s, t) \geq 0$: 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 $d_{edit}(s, t) = d_{edit}(t, s)$.
3. (by contradiction) Assume that $d_{edit}(s, u) + d_{edit}(u, t) < d_{edit}(s, t)$, and $\mathcal{S}$ transforms s into u in $dist(s, u)$ steps, and $\mathcal{S}'$ transforms u into t in $d_{edit}(u, t)$ 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 $d_{edit}(s, t)$, a contradiction to the definition of d_{edit}.

The edit distance is a metric

Claim: The edit distance is a metric.

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

1. $d_{edit}(s,t) \geq 0$: 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 $d_{edit}(s,t) = d_{edit}(t,s)$.
3. (by contradiction) Assume that $d_{edit}(s,u) + d_{edit}(u,t) < d_{edit}(s,t)$, and $\mathcal{S}$ transforms s into u in $dist(s,u)$ steps, and $\mathcal{S}'$ transforms u into t in $d_{edit}(u,t)$ 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 $d_{edit}(s,t)$, a contradiction to the definition of d_{edit}.

Exercise: Show that the Hamming distance is a metric.

Alignments vs. edit operations

Every alignment corresponds to a series of edit 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

Every alignment corresponds to a series of edit operations:

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

Example

```
T-ACAT-
TGAT-AT
```

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

Alignments vs. edit operations

Every alignment corresponds to a series of edit operations:

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

Example

```
T-ACAT-
TGAT-AT
```

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

(By convention, we apply the edit operations from left to right.)

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

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

- TACAT $\stackrel{\text{ins}}{\to}$ TGACAT $\stackrel{\text{subst}}{\to}$ TGATAT

- TACAT $\stackrel{\text{ins}}{\to}$ TGACAT $\stackrel{\text{subst}}{\to}$ TGAGAT $\stackrel{\text{subst}}{\to}$ TGATAT

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

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

```
-TAC-AT
TGA-TAT
```

- TACAT $\overset{\text{ins}}{\to}$ TGACAT $\overset{\text{subst}}{\to}$ TGATAT
- TACAT $\overset{\text{ins}}{\to}$ TGACAT $\overset{\text{subst}}{\to}$ TGAGAT $\overset{\text{subst}}{\to}$ TGATAT

Alignments vs. edit operations

Not every series of operations corresponds to an alignment:

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

```
-TAC-AT
TGA-TAT
```

- TACAT $\stackrel{\text{ins}}{\rightarrow}$ TGACAT $\stackrel{\text{subst}}{\rightarrow}$ TGATAT

```
T-ACAT
TGATAT
```

- TACAT $\stackrel{\text{ins}}{\rightarrow}$ TGACAT $\stackrel{\text{subst}}{\rightarrow}$ TGAGAT $\stackrel{\text{subst}}{\rightarrow}$ 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

Fact

Every minimum-length series of operations corresponds to an alignment.

Proof (sketch):

Show that in a minimum-length series of edit operations, each position of each string is involved in at most one operation.

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 $\stackrel{\text{subst}}{\rightarrow}$ GACAT $\stackrel{\text{del}}{\rightarrow}$ GAAT $\stackrel{\text{ins}}{\rightarrow}$ TGAAT $\stackrel{\text{ins}}{\rightarrow}$ TGATAT

```
-TAC-AT
TGA-TAT
```

- TACAT $\stackrel{\text{ins}}{\rightarrow}$ TGACAT $\stackrel{\text{subst}}{\rightarrow}$ TGATAT

```
T-ACAT
TGATAT
```

Optimal alignment score vs. edit distance

Theorem

With the scoring function:
match = 0, mismatch = -1, gap = -1, we have:

$$sim(s, t) = -d_{edit}(s, t).$$

Moreover, we get the same optimal alignments / minimum-length series of edit operations.

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

Computing the edit distance

Note first that we can assume that (a) edit operations happen left-to-right, and (b) every character is involved in at most one edit operation. For computing an optimal alignment, we consider what happens to the last characters. Then 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

Computing the edit distance

Note first that we can assume that (a) edit operations happen left-to-right, and (b) every character is involved in at most one edit operation. For computing an optimal alignment, we consider what happens to the last characters. Then 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 $1_1 \dots t_{m-1}$ and substitute s_n with t_m

Computing the edit distance

Note first that we can assume that (a) edit operations happen left-to-right, and (b) every character is involved in at most one edit operation. For computing an optimal alignment, we consider what happens to the last characters. Then 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 $1_1 \dots t_{m-1}$ and substitute s_n with t_m
3. transform s into $t_1 \dots t_{m-1}$ and insert t_m

Computing the edit distance

Note first that we can assume that (a) edit operations happen left-to-right, and (b) every character is involved in at most one edit operation. For computing an optimal alignment, we consider what happens to the last characters. Then transforming s into t can be done in one of 3 ways:

1. transform $s_1 \ldots s_{n-1}$ into t and then delete last character of s
2. if $s_n = t_m$: transform $s_1 \ldots s_{n-1}$ into $t_1 \ldots t_{m-1}$
   if $s_n \neq t_m$:
   transform $s_1 \ldots s_{n-1}$ into $1_1 \ldots t_{m-1}$ and substitute s_n with t_m
3. transform s into $t_1 \ldots t_{m-1}$ 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: $E(i,j) = d_{edit}(s_1 \ldots s_i, t_1 \ldots t_j)$

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 $E(n,m)$
- backtrace for a 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 $d_{edit}(s, t)$

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

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

return $E(n, m)$;

Analysis

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

General cost function

General cost edit distance

Different edit operations can have different cost (but some conditions must hold, e.g. cost(insert) = cost(delete), why?).

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 $s =$ TACAT and $t =$ TGATAT, then we have $LCS(s, t) = 4$.
s = TACAT, t = 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 = TACAT, t = 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 $s =$ TACAT and $t =$ TGATAT, then we have $LCS(s,t) = 4$.
s = TACAT, t = 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 example above, we have $d_{LCS}(s',t') = 6 + 5 - 2 \cdot 3 = 5$.

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

Exercise

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