

## Bioinformatics Algorithms

(Fundamental Algorithms, module 2)

Zsuzsanna Lipták

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

### The $q$ -gram distance

## The $q$ -gram distance

- In many situations, edit distance is a good model for differences / similarity between strings.
- But sometimes, other distance functions serve the purpose better.

### Motivations for using $q$ -gram distance

1. If two parts of a sequence are exchanged (e.g. two paragraphs, two long substrings, two genes), then one can argue that the resulting strings still have high similarity; however, the edit distance will be big. The  $q$ -gram distance can be more appropriate in this case.
2. The edit distance needs quadratic computation time, but this is often too slow. The  $q$ -gram distance can be computed in linear time.

2 / 19

## What is a $q$ -gram?

Let  $\Sigma$  be the alphabet, with  $|\Sigma| = \sigma$ .

**Def.**

A  $q$ -gram is a string of length  $q$ .

**Note**

$q$ -grams are also called  $k$ -mers,  $w$ -words, or  $k$ -tuples. Typically,  $q$  (or  $k$ ,  $w$ , etc.) is small, much smaller than the strings we will want to compare.

We will fix  $q$ , and use the number of occurrences of  $q$ -grams to compute distances between strings.

## Occurrence count

Let  $s$  be a string of length  $n \geq q$ , and  $u$  be a  $q$ -gram. The **occurrence count** of  $u$  in  $s$  is

$$N(s, u) = |\{i : s_i \dots s_{i+q-1} = u\}|,$$

the number of times  $q$ -gram  $u$  occurs in  $s$ .

**Ex.**

Let  $s = \text{ACAGGGCA}$ , then

$N(s, \text{AC}) = N(s, \text{AG}) = N(s, \text{GC}) = 1$ ,  $N(s, \text{CA}) = N(s, \text{GG}) = 2$ , and for all other  $q$ -grams  $u$  over  $\Sigma$ ,  $N(s, u) = 0$ .

3 / 19

4 / 19

## $q$ -gram profile

Fix some enumeration of  $\Sigma^q$ , i.e. some order in which we want to list all  $q$ -grams (e.g. the lexicographic order).

**Def.**

Let  $s$  be a string over  $\Sigma$ ,  $|s| \geq q$ . The  **$q$ -gram profile** of  $s$ ,  $P_q(s)$  is an array of size  $\sigma^q$ , where the  $i$ th entry is

$$P_q(s)[i] = N(s, u_i),$$

and  $u_i$  is the  $i$ th  $q$ -gram in the enumeration.

**Example:**

Let  $\Sigma = \{A, C, G, T\}$  and  $q = 2$ .

Let

$s = \text{ACAGGGCA}$ ,

$t = \text{GGGCAACA}$ ,

$v = \text{AAGGACA}$ .

Then the  $q$ -gram profiles of  $s, t, v$  are shown on the right.

$u$	$P_q(s)$	$P_q(t)$	$P_q(v)$
AA	0	1	1
AC	1	1	1
AG	1	0	1
AT	0	0	0
CA	2	2	1
CC	0	0	0
CG	0	0	0
CT	0	0	0
GA	0	0	1
GC	1	1	0
GG	2	2	1
GT	0	0	0
TA	0	0	0
TC	0	0	0
TG	0	0	0
TT	0	0	0

Notice that the sum of all entries of  $P_q(s) = |s| - q + 1 =$  total number of  $q$ -gram occurrences in  $s =$  number of distinct positions in  $s$  where a  $q$ -gram starts.

5 / 19

6 / 19

## q-gram distance

(Introduced by Ukkonen, 1992)

**Def.:** Given two strings  $s, t$ , the  $q$ -gram distance of  $s$  and  $t$  is

$$\text{dist}_{q\text{-gram}}(s, t) = \sum_{u \in \Sigma^q} |N(s, u) - N(t, u)|.$$

**Equivalent def.:** Given two strings  $s, t$ , the  $q$ -gram distance of  $s$  and  $t$  is

$$\text{dist}_{q\text{-gram}}(s, t) = \sum_{i=1}^{\sigma^q} |P_q(s)[i] - P_q(t)[i]|,$$

which is the **Manhattan distance**<sup>1</sup> of the  $q$ -gram profiles of  $s$  and  $t$ .

<sup>1</sup>The Manhattan distance, or  $L_1$ -distance, of two vectors  $x, y \in \mathbb{R}^n$  is defined as  $\sum_{i=1}^n |x_i - y_i|$ .

7 / 19

## q-gram distance

In the previous example ( $q = 2$ ,  $s = \text{ACAGGGCA}$ ,  $t = \text{GGGCAACA}$ , and  $v = \text{AAGGACA}$ ), we have

$$\text{dist}_{2\text{-gram}}(s, t) = 2, \text{dist}_{2\text{-gram}}(s, v) = 5, \text{ and } \text{dist}_{2\text{-gram}}(t, v) = 5.$$

Note that it is possible to have distinct strings with  $q$ -gram distance 0, e.g.

$$\text{for } w = \text{AGGGCACA}, \text{ we have } \text{dist}_{2\text{-gram}}(s, w) = 0.$$

8 / 19

## q-gram distance

In the previous example ( $q = 2$ ,  $s = \text{ACAGGGCA}$ ,  $t = \text{GGGCAACA}$ , and  $v = \text{AAGGACA}$ ), we have

$$\text{dist}_{2\text{-gram}}(s, t) = 2, \text{dist}_{2\text{-gram}}(s, v) = 5, \text{ and } \text{dist}_{2\text{-gram}}(t, v) = 5.$$

Note that it is possible to have distinct strings with  $q$ -gram distance 0, e.g.

$$\text{for } w = \text{AGGGCACA}, \text{ we have } \text{dist}_{2\text{-gram}}(s, w) = 0.$$

**(Don't just believe this, double check it!)**

8 / 19

## The q-gram distance is a pseudo-metric

### Lemma

The  $q$ -gram distance is a pseudo-metric, i.e. it is non-negative, symmetric, and obeys the triangle inequality (but it is possible to have  $x \neq y$  with  $\text{dist}_{q\text{-gram}}(x, y) = 0$ ).

### Proof:

The three properties follow from the fact that the Manhattan metric is a metric. The example above shows that  $\text{dist}_{q\text{-gram}}(x, y) = 0$  does not imply  $x = y$ .

### Exercise:

Prove the lemma explicitly.

9 / 19

## Connection to edit distance

### q-gram Lemma

Let  $d_{\text{edit}}(s, t)$  denote the (unit-cost) edit distance of  $s$  and  $t$ . Then

$$\frac{\text{dist}_{q\text{-gram}}(s, t)}{2q} \leq d_{\text{edit}}(s, t).$$

### Proof

Every edit operation contributes to the  $q$ -gram distance at most  $2q$ : Consider the simplest case, a **substitution** in position  $i$  of  $s$ , where character  $s_i$  is substituted by character  $x$ , and let  $s'$  be the resulting string. If  $q \leq i \leq |s| - q + 1$ , then there are exactly  $q$   $q$ -grams of  $s$  affected by the substitution:  $s_{i-q+1} \dots s_i$ , up to  $s_i \dots s_{i+q-1}$  (otherwise fewer); the counts of all these are **decremented** by 1, while the counts of the **new**  $q$ -grams  $s_{i-1} \dots x, s_i \dots x s_{i+q}$ , etc. are **incremented** by 1. Therefore,  $\text{dist}_{q\text{-gram}}(s, s') \leq 2q$  (it could be less because these  $q$ -grams need not be all distinct). For a **deletion**, the number of  $q$ -grams whose count is decremented is at most  $q$ , while those whose count is incremented is at most  $q - 1$ ; for an **insertion** the other way around.—The claim follows by induction on the number of edit operations.

10 / 19

## Connection to edit distance

### Examples

With the earlier examples, we have

1. Exchange of two long substrings:  $d_{\text{edit}}(s, t) = 6, d_{\text{edit}}(s, w) = 4$  (compare to:  $\text{dist}_{q\text{-gram}}(s, t) = 2, \text{dist}_{q\text{-gram}}(s, w) = 0$ , with  $q = 2$ ).
2. The  $q$ -gram distance is at most  $2q$  times edit distance ( $q$ -gram lemma):  $d_{\text{edit}}(s, v) = 2$  (compare to:  $\text{dist}_{q\text{-gram}}(s, v) = 5 \leq 8 = d_{\text{edit}}(s, v) \cdot 2q$ , with  $q = 2$ )

Based on the  $q$ -gram lemma and the fact that the  $q$ -gram distance can be computed in linear time, we can use the  $q$ -gram distance as a filter for edit distance computations.

11 / 19

## Computation of the $q$ -gram distance

### Basic ideas

- Use a **sliding window** of size  $q$  over  $s$  and  $t$
- Use an array  $d_q$  of size  $\sigma^q$
- First slide a window over  $s$ , increment respective entry for every  $q$ -gram seen
- Then slide over  $t$ , decrement respective entry for every  $q$ -gram seen
- Now  $d_q[r] = N(s, u_r) - N(t, u_r)$ .
- Sum up the absolute values of the entries:  
 $dist_{q\text{-gram}}(s, t) = \sum_i |d_q[i]|$

We will see: This algorithm runs in **linear time**.

**But:** how do we know where to find the entry for the current  $q$ -gram?  
 This is called **ranking** (coming soon)

12 / 19

## Computation of the $q$ -gram distance

### Algorithm for computing $q$ -gram distance

**input:** Strings  $s, t$  of length  $|s| = n$  and  $|t| = m$

**output:**  $dist_{q\text{-gram}}(s, t)$

1. initialize  $d_q[0 \dots \sigma^q - 1]$  with 0s
2. for  $i = 1, \dots, n - q + 1$ :  $r \leftarrow rank(s_i \dots s_{i+q-1})$   
 $d_q[r] \leftarrow d_q[r] + 1$
3. for  $i = 1, \dots, m - q + 1$ :  $r \leftarrow rank(t_i \dots t_{i+q-1})$   
 $d_q[r] \leftarrow d_q[r] - 1$
4.  $d \leftarrow 0$
5. for  $i = 0 \dots \sigma^q - 1$ :  $d \leftarrow d + |d_q[i]|$ .
6. return  $d$

13 / 19

### Goal

Given  $q$ -gram  $u$ , we want to know which entry of the array  $u$  corresponds to.

**Ex.:** Where is the  $q$ -gram CG? In position 6.

### Ranking functions

- A **ranking function** is a bijection  $rank : \Sigma^q \rightarrow [0 \dots \sigma^q - 1]$ .
- $rank(u)$  gives us the position of  $u$  in the enumeration of  $\Sigma^q$
- needs to be very efficiently computable
- the ranking function we use will give us **constant time** per  $q$ -gram of  $s$

$r$	$u_r$	$d_q(s)$
0	AA	0
1	AC	1
2	AG	1
3	AT	0
4	CA	2
5	CC	0
6	CG	0
7	CT	0
8	GA	0
9	GC	1
10	GG	2
11	GT	0
12	TA	0
13	TC	0
14	TG	0
15	TT	0

14 / 19

## Ranking function

- **Basic idea:** We will interpret the  $q$ -gram itself as a number: a number base  $\sigma$ . In our case:  $\sigma = 4$ .
- **First,** we assign numbers  $0, \dots, \sigma - 1$  (here:  $0, 1, 2, 3$ ) to the characters:

$$f : A \mapsto 0, C \mapsto 1, G \mapsto 2, T \mapsto 3$$

- **Second,** we extend this to strings: e.g. CG becomes  $12_4 = 1 \cdot 4^1 + 2 \cdot 4^0 = 6_{10}$ . (i.e. 12 in base 4 equals 6 in base 10.)
- **In general,** for  $u = u_1 \dots u_q$ , the  $rank(u)$  is given by:

$$rank(u) = f(u_1) \cdot \sigma^{q-1} + f(u_2) \cdot \sigma^{q-2} + \dots + f(u_{q-1}) \cdot \sigma^1 + f(u_q) \cdot \sigma^0.$$

- **E.g.**  $rank(\text{CATT}) = 1 \cdot 4^3 + 0 \cdot 4^2 + 3 \cdot 4 + 3 \cdot 1 = 64 + 0 + 12 + 3 = 79$ .

15 / 19

## Sliding window

### Crucial trick

The rank of the  $q$ -gram starting in position  $i + 1$  can be computed from the rank of the  $q$ -gram starting in position  $i$  in constant time.

### Example

Let  $s = \text{GACATTGACGAT}$ , and let  $q = 4$ . Let's compare the rank of CATT and ATTG, two consecutive  $q$ -grams:

$$\begin{aligned} rank(\text{CATT}) &= 1 \cdot 4^3 + 0 \cdot 4^2 + 3 \cdot 4^1 + 3 \cdot 4^0 \\ rank(\text{ATTG}) &= 0 \cdot 4^3 + 3 \cdot 4^2 + 3 \cdot 4^1 + 2 \cdot 4^0 \end{aligned}$$

So  $1 \cdot 4^3$  has to be subtracted, the rest multiplied by 4, and finally  $2 \cdot 4^0 = 2$  added.

16 / 19

## Sliding window

In general:

$$\begin{aligned} rank(s_i \dots s_{i+q-1}) &= f(s_i) \cdot \sigma^{q-1} + f(s_{i+1}) \cdot \sigma^{q-2} + \dots + f(s_{i+q-1}) \\ rank(s_{i+1} \dots s_{i+q}) &= f(s_{i+1}) \cdot \sigma^{q-1} + \dots + f(s_{i+q-1}) \cdot \sigma + f(s_{i+q}) \end{aligned}$$

Therefore, if  $rank(s_i \dots s_{i+q-1}) = C$ , then

$$rank(s_{i+1} \dots s_{i+q}) = (C - f(s_i) \cdot \sigma^{q-1}) \cdot \sigma + f(s_{i+q})$$

**Ex.**  $rank(\text{ATTG}) = (rank(\text{CATT}) - 1 \cdot 4^3) \cdot 4 + 2 \cdot 4^0 = (79 - 64) \cdot 4 + 2 = 62$ .  
 Double check:  $rank(\text{ATTG}) = 0 \cdot 4^3 + 3 \cdot 4^2 + 3 \cdot 4 + 2 = 48 + 12 + 2 = 62$ .

17 / 19

## Analysis

- computing the rank of the first  $q$ -gram:  $O(q)$  time
- computing rank of the  $(i + 1)$ st  $q$ -gram, given the rank of the  $i$ th  $q$ -gram: constant time

18 / 19

## Analysis (cont.)

Computing the  $q$ -gram distance of two strings  $s, t$  of length  $n$  resp.  $m$ :

- initialize array  $d_q$ :  $O(\sigma^q)$  time
- slide window of size  $q$  over  $s$ : there are  $n - q + 1$  windows, for each, we have to compute its rank  $r$  and then update the entry  $d_q(r)$ ; rank of first window takes  $O(q)$  time, for all following windows  $O(1)$ , while updating entry is always constant time -  $O(n)$  time
- slide window of size  $q$  over  $t$ : similarly,  $O(m)$  time
- compute sum of absolute values:  $O(\sigma^q)$  time

Thus,

- Total **time**:  $O(n + m + \sigma^q)$
- Total **space**:  $O(\sigma^q)$  (for the array  $d_q$ )
- If we choose  $q \leq \log_{\sigma}(n), \log_{\sigma}(m)$ , then  $\sigma^q = O(n + m)$ , so we have **linear time and space**  $O(n + m)$ .

19 / 19