Efficient algorithm to get cyclic Hamming distance

Oscar Cabrera

ocg.steppenwolf@gmail.com

Independent Researcher

March 2024

Abstract

In this text we use the cross-correlation method using the fast Fourier transform to present a simple algorithm, with a time complexity of $O(N \log N)$, in order to calculate the exact value of the cyclic Hamming distance between two sequences of length N over any alphabet of size k. It may be useful in applications where we only want to know that value, like in sequence alignment, without taking into account insertions and deletions.

1 Introduction

There are many string metrics for measuring the edit distances between two sequences, the most common is the Levenshtein distance [1], which allows insertions, deletions and substitutions. Another useful metric is the Hamming distance [2], which allows only substitutions. In this text we will focus on the creation of a simple and efficient algorithm to get the minimum cyclic Hamming distance between two sequences using the well-known techniques of cross-correlation and FFT; although it is probably a known method, it is not easy to find an explicit and simple algorithm to do this task [3]. It will be especially useful for sequence alignment applications, in situations where only substitutions are allowed and we want to know the exact Hamming distance, not only a cross-correlation measure [4].

Let's make a quick summary of basic concepts before present our algorithm. We have the following scenario: given two sequences $x = \{x_0, \ldots, x_{n_x-1}\}$ and $y = \{y_0, \ldots, y_{n_y-1}\}$ over an alphabet of size k, where $n_x \geq n_y$, we want to get the cross-correlation values $(x \star y)_n$. As both sequences should have equal length $N = n_x$, in case $n_x > n_y$ it is a common practice to do a zero-extension of y. First of all, the (circular) cross-correlation for two sequences $x, y \in C^N$ is defined by:

$$(x \star y)_n = \sum_{m=0}^{N-1} x_m \cdot y^*_{(m-n)_{\bmod N}} \qquad n = 0, \ldots, N-1 \tag{1}$$

The Discrete Fourier Transform (DFT) of x is denoted by $\mathcal{F}\{x\} = X = \{X_0, \ldots, X_{N-1}\}$, where X_k is calculated as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1 \tag{2}$$

The Inverse Discrete Fourier Transform (IDFT) of X is denoted by $\mathcal{F}^{-1}\{X\} = x$, where x_n is calculated as:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn} \qquad n = 0, \ldots, N-1 \tag{3}$$

As both DFT and IDFT have similar expressions, any implementation to calculate DFT can easily calculate IDFT (the asterisk means the complex conjugate):

$$\mathcal{F}^{-1}\{X\} = \frac{1}{N}\mathcal{F}^{*}\{X^{*}\} \tag{4}$$

The cross-correlation theorem satisfies the following (the multiplication is pointwise):

$$x \star y = \mathcal{F}^{-1}\{\mathcal{F}\{x\} \cdot \mathcal{F}^{*}\{y\}\} \tag{5}$$

The most common practice to calculate $x \star y$ is the use of the cross-correlation theorem, but computing DFT or IDFT directly from the definition has a time complexity of $O(N^2)$, too slow to be practical. The use of Fast Fourier Transform (FFT) to compute the DFT and IDFT reduces the time complexity to $O(N \log N)$. The value of N is not a problem for current FFT algorithms [5].

2 Algorithm

The real scenario is slightly different to the previous one, because we are interested in he cyclic Hamming distance instead of the cross-correlation.

Definition 2.1. *The Hamming distance $d(x, y)$ between two sequences of equal length is defined as the minimum number of substitutions required to change one string into the other.*

$$d(x, y) = \#\{j : x_j \neq y_j, j = \{0, ..., N-1\}\} \tag{6}$$

Definition 2.2. *Given two sequences x and y, the number of coincidences $c(x, y)$ is the number of positions at which the corresponding symbols are equal.*

$$c(x, y) = \#\{j : x_j = y_j, j = \{0, ..., N-1\}\} = N - d(x, y) \tag{7}$$

Definition 2.3. *The cyclic Hamming distance $d_{min}(x, y)$ is the minimum Hamming distance between x and each possible rotation of y,*

$$d_{min}(x, y) = \min_{j \in [0..N-1]} \{d(x, r_j(y))\}, \tag{8}$$

where $r_j(y)$ is the j-th rotation of y.

Given two sequences $x = \{x_0, \ldots, x_{n_x-1}\}$ and $y = \{y_0, \ldots, y_{n_y-1}\}$ over an alphabet of size k, where $n_x \geq n_y$, we want to get the cyclic Hamming distance $d_{min}(x, y)$. As both sequences should have equal length $N = n_x$, in case $n_x > n_y$ we will make a k-extension of y (i.e. filling with a value equal to k, which is outside of our alphabet).

The direct computation of this has a time complexity of $O(N^2)$, so we should think of a more efficient way to do it. Let's see if we can take benefit of the cross-correlation method, giving us a time complexity of $O(N \log N)$.

Proposition 2.1. *Given two binary sequences x and y, and encoding symbol 0 as number 0 and symbol 1 as number 1, the cross-correlation $x \star y$ gives the number of coincident ones between both sequences.*

Theorem 2.1. *Given two sequences x and y, we can denote $c(x, y)$ as the following sum:*

$$c(x,y) = \sum_{i=0}^{k-1} v^{(i)}(x) \star v^{(i)}(y) \tag{9}$$

$$v_j^{(i)}(z) = \begin{cases} 1 & \text{if } z_j = i \\ 0 & \text{if } z_j \neq i \end{cases} \qquad j = 0, ..., N-1$$

Proof. We can decompose (7) as the following sum, making the comparisons separately for each symbol of our alphabet:

$$c(x,y) = \sum_{i=0}^{k-1} \#\{j : x_j = i \wedge y_j = i, j = \{0, ..., N-1\}\}$$

We can also denote sequences x and y by a sum of binary vectors multiplied by each symbol, where a value of 1 in the vector means that the symbol appears in that position in the sequence.

$$x = \sum_{i=0}^{k-1} i \cdot v^{(i)}(x) \qquad y = \sum_{i=0}^{k-1} i \cdot v^{(i)}(y)$$

As the number of coincidences of ones between $v^{(i)}(x)$ and $v^{(j)}(y)$ must be 0 $\forall i \neq j$ because they refer to different symbols, we can denote $c(x, y)$ by:

$$c(x,y) = \sum_{i=0}^{k-1} \#\{j : v_j^{(i)}(x) = 1 \wedge v_j^{(i)}(y) = 1, j = \{0, ..., N-1\}\}$$

Applying Proposition 2.1 to the content of the sum we finally come to expression (9). □

In summary:

1. We compute in (9) the number of coincidences $c(x, y)$ (keep in mind that it is a vector), with a time complexity of k times $O(N \log N)$.
2. Then we apply (7) for the computation of the Hamming distance $d(x, y)$ (again, it is a vector).
3. We finally apply (8) to get the cyclic Hamming distance $d_{min}(x, y)$ and the offsets of y over x where we obtain that value. We can allow all rotations of y or not (see Algorithm 1 for a simple implementation, without optimizations).

Algorithm 1 Compute cyclic Hamming distance $d_{min}(x, y)$, requiring $n_x \geq n_y$

Input: x, y, n_x, n_y, k, $cyclic$
Output: d_{min}, $offsets$

$N \leftarrow n_x$
$offsets \leftarrow [\,]$ ▷ Empty array
if $n_x > n_y$ **then**
    $y[n_y, ..., n_x - 1] \leftarrow k$ ▷ k-Fill to make equal length
$c[0, ..., N-1] \leftarrow 0$ ▷ Vector initialization for $c(x, y)$
for $i = 0$ to $k - 1$ **do**
    for $j = 0$ to $N - 1$ **do** ▷ Create binary sequences $v_x \equiv v^{(i)}(x)$ and $v_y \equiv v^{(i)}(y)$
        if $x[j] = i$ **then**
            $v_x[j] \leftarrow 1$
        else
            $v_x[j] \leftarrow 0$
        if $y[j] = i$ **then**
            $v_y[j] \leftarrow 1$
        else
            $v_y[j] \leftarrow 0$
    $corr \leftarrow \mathbb{R}e\{\texttt{IFFT}(\texttt{FFT}(v_x) \cdot \texttt{FFT}^*(v_y))\}$ ▷ Cross-correlation vector in $O(N \log N)$
    $c \leftarrow c + corr$ ▷ Accumulate vector values
if $cyclic = \texttt{True}$ **then** ▷ All rotations allowed?
    $range \leftarrow N - 1$
else
    $range \leftarrow n_x - n_y$
$c_{max} \leftarrow \max\{c[0, ..., range]\}$ ▷ Find d_{min} and offsets
for $j = 0$ to $range$ **do**
    if $c[j] = c_{max}$ **then**
        $offsets.\texttt{Append}(j)$
$d_{min} \leftarrow N - c_{max}$

3 Example

Let's suppose an example of sequence alignment for DNA: we have sequences CCGATTCC and CCA over an alphabet of 4 symbols {C,G,A,T}. Doing the alignment by hand, we get $d_{min}(x, y) = 6$ for offsets {0,1,6,7} (see Table 1, best results in bold).
If we are not interested in all rotations of y around x, but only in those inside the windowed space of x ($cyclic = \texttt{False}$ in Algorithm 1), then we obtain the same value for $d_{min}(x, y)$ but only for offsets {0,1}.

Table 1: Manual sequence alignment

Offset	C	C	G	A	T	T	C	C	$c(x,y)$	$d(x,y)$
0	C	C	A	-	-	-	-	-	2	**6**
1	-	C	C	A	-	-	-	-	2	**6**
2	-	-	C	C	A	-	-	-	0	8
3	-	-	-	C	C	A	-	-	0	8
4	-	-	-	-	C	C	A	-	0	8
5	-	-	-	-	-	C	C	A	1	7
6	A	-	-	-	-	-	C	C	2	**6**
7	C	A	-	-	-	-	-	C	2	**6**

Now, let's use Algorithm 1 step by step:

1. In our example we initially have $k = 4$, $n_x = 8$ and $n_y = 3$, then $N = 8$. We encode alphabet {C,G,A,T} as {0,1,2,3}, and sequences as $x = \{0, 0, 1, 2, 3, 3, 0, 0\}$ and $y = \{0, 0, 2, 4, 4, 4, 4, 4\}$. The initial vector of coincidences is $c(x, y) = \{0, 0, 0, 0, 0, 0, 0, 0\}$.

2. Iteration $i = 0$. Take in mind that $corr = v^{(i)}(x) \star v^{(i)}(y)$ and $c(x, y)$ is the accumulated number of coincidences.

Offset	0	1	2	3	4	5	6	7
$v^{(0)}(x)$	1	1	0	0	0	0	1	1
$v^{(0)}(y)$	1	1	0	0	0	0	0	0
corr	2	1	0	0	0	1	2	2
$c(x, y)$	2	1	0	0	0	1	2	2

3. Iteration $i = 1$. Observe that, when one of the vectors is a zero vector, the cross-correlation is also a zero vector.

Offset	0	1	2	3	4	5	6	7
$v^{(1)}(x)$	0	0	1	0	0	0	0	0
$v^{(1)}(y)$	0	0	0	0	0	0	0	0
corr	0	0	0	0	0	0	0	0
$c(x, y)$	2	1	0	0	0	1	2	2

4. Iteration $i = 2$.

Offset	0	1	2	3	4	5	6	7
$v^{(2)}(x)$	0	0	0	1	0	0	0	0
$v^{(2)}(y)$	0	0	1	0	0	0	0	0
corr	0	1	0	0	0	0	0	0
$c(x, y)$	2	2	0	0	0	1	2	2

5. Iteration $i = 3$. In this final iteration we get the best result of $d_{min}(x, y) = 6$ for offsets {0,1,6,7}, marked in bold. Same results of Table 1, as we expected.

Offset	**0**	**1**	2	3	4	5	**6**	**7**
$v^{(3)}(x)$	0	0	0	0	1	1	0	0
$v^{(3)}(y)$	0	0	0	0	0	0	0	0
$corr$	0	0	0	0	0	0	0	0
$c(x, y)$	2	2	0	0	0	1	2	2
$d(x, y)$	**6**	**6**	8	8	8	7	**6**	**6**

4 Conclusion

We have successfully done a simple and efficient algorithm to know the exact value of the cyclic Hamming distance between two sequences, in order to be used in sequence alignment applications or other situations. Take in mind that the time complexity is $O(N \log N)$ as long as $k \ll N$, but that will usually be the case.

References

[1] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", Soviet Physics Doklady, 10(8): 707-710, 1966.

[2] R.W. Hamming, "Error detecting and error correcting codes", The Bell System Technical Journal. 29 (2): 147–160. 1950.

[3] S. Hosangadi, "Distance Measures for Sequences", arXiv, 1208.5713, 2012.

[4] A. Rockwood, D. Crockett, J. Oliphant, K. Elenitoba-Johnson, "Sequence alignment by cross-correlation". J Biomol Tech., 16(4):453-8. PMID: 16522868, PMCID: PMC2291754, 2005.

[5] J.W. Cooley, J.W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Mathematics of Computation, 19: 297–301, 1965.