# 2022 S.T. Yau High School Science Award (Math)

# Research Project

## The Team

Name of team member: Tianyi Ren

School: Hangzhou No.2 High School of Zhejiang Province

Province, Country: Zhejiang, China

Name of supervising teacher: Rui Dong

School/Institution: Tsinghua University

Province, Country: Beijing, China

**Title of Research Project**: Realization of virus universe with applications

**Date**: 2022/09/13

# Realization of virus universe with applications

Tianyi Ren

**Abstract**

In recent years, the studies of virus genomes attract more and more attention. Among various methods to study biological sequences, alignment-free methods are taking an increasingly important role because large-scale data requires high computational efficiency, while the traditional alignment-based methods fail to achieve. In this paper, we transform the viral genomes into vectors called natural vectors using mathematical techniques, and analyze the properties of these vectors in order to study the corresponding biological sequences. We first show that the corresponding natural vectors from the same families will gather together based on the convex hull principle, then the $k$-NN algorithm is applied to the classification of virus genomes. In this paper, we also consider $k$-mer natural vectors and propose a training process to obtain the optimal value of the weights for different $k$-mers. Results suggest that our algorithm to train the weights successfully outperforms the previous manual selected weights. We further study the SARS-CoV-2 database and develop a method to identify new variants from known variants. It provides insight into the epidemiological research of COVID-19, especially on the prevention of new pandemic caused by unknown and new variants.

**Keywords: natural vectors; $k$-NN; weight training; new variant identification**

# Contents

# 1 Introduction

The virus genomic universe is a constantly updating space, because new virus genomes emerge and are added to known databases all the time. It is important to analyze the similarity and dissimilarity between those viral genomes to gain a better understanding of the relationship between them, which may inspire us to cure related diseases and develop vaccines. In practical applications, this virus genomic universe is constructed by converting sequences to vectors in order to apply the mathematical techniques. We hope to use better transformation methods and develop better metrics from the perspective of mathematics, so that the distances between these genomes can represent their biological similarity accurately. In this universe, similar genomic sequences (such as from the same family or genus) will be clustered together in the corresponding mathematical space. Therefore, a well-defined distance will be able to help us in many aspects, such as family classification and new variant identification. In particular, the genome space is constructed in a high-dimensional Euclidean space, and all genomes in the space are uniquely represented as high-dimensional points, i.e., vectors. In this project, we use the natural vector method to achieve this [1]. It is proved that there is a one-to-one correspondence between a sequence and its natural vector as long as the order is sufficiently high. In practice, we use natural vectors to construct the genome universe and we further develop $k$-mer natural vector to do the classification.

Here is a brief introduction to our project. Chapter 3 introduces the basic framework of natural vectors to convert an arbitrary DNA sequence to a vector. We also show that each natural vector uniquely determines one DNA sequence, which is the basis of following discussions. Moreover there are two ways to increase the dimension and are used in different situations, which consider high-order and $k$-mer, respectively. Chapter 4 shows the gathering property of natural vectors from the same family using convex hull analysis. In other words, the convex hulls of different viral families are constructed based on the corresponding natural vectors and are disjoint in high dimensional space.

Chapter 5 and 6 are the main applications of our work. In chapter 5 we use the 1-NN method to predict which family a new sequence should belongs to. We first compute the distance using $k$-mer natural vectors where $k$ ranges from 1 to a certain $K$. We further propose an algorithm to train the weights of different $k$-mer natural vectors, so that the weighted sum of distance can better reflects the similarity between the corresponding sequences accurately. In Chapter 6, we apply the proposed method on the identification of new SARS-CoV-2 variants. We compare the distance from a viral sequence to the sequences in the variant group and not in the variant group. Then kernel density estimation is used to plot those distributions (as random variables) to show that the two distributions of distances are different. This inspires us to use a way similar to hypothesis

testing to detect new variants. The accuracy of our method achieves an accuracy of over 99.5% in the detection of new variant in SARS-CoV-2 based on current data.

## 2 Data

This article uses four datasets. Dataset 1 and 2 consist of all available virus reference sequences. Dataset 3 and 4 include SARS-CoV-2 genomes from GISAID.

Dataset 1 is the latest whole virus reference sequence database downloaded from NCBI (National Center for Biotechnology Information, `https://ftp.ncbi.nlm.nih.gov/refseq/release/viral`) on June 30, 2022. Sequences with unknown family information or containing ambiguous letters (such as N) are removed. In other words, we only keep the high-quality genome viruses in our study. If a sequence is the only one in a family, it is also removed to avoid problems in classification. After the cleaning process, there are 11,559 sequences in the Dataset 1.

Dataset 2 is a previous whole virus reference sequence database from [2], containing all sequences submitted to `ftp.ncbi.nlm.nih.gov/genomes/Viruses` before March 31, 2020. There are 7,382 sequences that pass the same criteria as for dataset 1. Dataset 2 is only for training in section 5.

Dataset 3 is based on the well-acknowledged database of SARS-CoV-2 (`https://gisaid.org`). It is worth noticing that here we only consider the main variants of concern (VOC), i.e., Alpha, Beta, Gamma, Delta, Omicron, instead of all the variants. We delete sequences containing ambiguous letters. The same sequences that appear multiple times in the dataset are considered only once, because the information remains the same if the genome sequences are identical. After data cleaning, there are 287,182 sequences that pass and are submitted before June 30, 2022.

Dataset 4 is a subset of Dataset 3, in which we randomly select 400 sequences from each of the five variants. The random sampling is repeated three times to eliminate the sampling error. In other words, we have three versions of Dataset 4, each of which are randomly and independently sampled from Dataset 3 with the same size (2,000 sequences). The introduction of Dataset 4 is due to the scale requirement of $k$-NN method, which Dataset 3 fails to satisfy because of the large number of sequences.

# 3 The natural vector method and its properties

## 3.1 Natural vectors

When analyzing biological sequences, such as DNA, multiple sequence alignment (MSA) is often used to compare the sequences. However, it is well known that alignment takes a lot of computing sources and is extremely time-consuming, which makes it difficult to analyze multiple genome data simultaneously. To understand these sequences effectively and comprehensively, we apply the concept of natural vectors proposed by Yau et al. [1]. Theoretically, every DNA can be represented by $S = s_1...s_n$, where $s_i$ can be any element in $\{A, T, C, G\}$. If we consider its 0-order central moment to $m$-order central moment, we can create a $4(m+1)$ dimensional vector as follows:

**Definition 3.1.** *Consider any DNA sequence $S = s_1 s_2 ... s_n$, define*

$$w_k(i) = \begin{cases} 1, & s_i = k \\ 0, & else \end{cases} \tag{1}$$

*where $k, s_i \in \{A, T, C, G\}$. Then the $m$-order natural vector is defined as*

$$(n_A, n_C, n_G, n_T, \mu_A, \mu_C, \mu_G, \mu_T, D_2^A, D_2^C, D_2^G, D_2^T, ..., D_m^A, D_m^C, D_m^G, D_m^T)$$

*where*

$$n_k = \sum_{i=1}^{n} w_k(i), n = n_A + n_T + n_C + n_G \tag{2}$$

$$\mu_k = \sum_{i=1}^{n} \frac{i}{n_k} w_k(i) \tag{3}$$

$$D_j^k = \sum_{i=1}^{n} \frac{(i-\mu_k)^j}{n_k^{j-1} n^{j-1}} w_k(i) \tag{4}$$

*(If $n_k = 0$, we define $\mu_k = 0$ and $D_j^k = 0$ for all $j$.)*

The normalization factor in Definition 3.1, $\frac{1}{n_k^{j-1} n^{j-1}}$, is designed in the way that when $j \to +\infty$, $D_j^k \to 0$.

For example, consider the sequence $ACGGT$, we have

$$\begin{cases} n_A = 1, \\ n_C = 1, \\ n_G = 2, \\ n_T = 1, \end{cases}$$

by counting the number of four letters. Their average positions are

$$\begin{cases} \mu_A = 1, \\ \mu_C = 2, \\ \mu_G = \frac{3+4}{2} = \frac{7}{2}, \\ \mu_T = 5. \end{cases}$$

Therefore, we calculate the 2-order central moment

$$\begin{cases} D_2^A = 0, \\ D_2^C = 0, \\ D_2^G = \frac{(3-\frac{7}{2})^2+(4-\frac{7}{2})^2}{2\times 5} = \frac{1}{20}, \\ D_2^T = 0, \end{cases}$$

and the 2-order natural of $ACGGT$ is $(1, 1, 2, 1, 1, 2, \frac{7}{2}, 5, 0, 0, \frac{1}{20}, 0)$.

### 3.2 One to one correspondence

In this subsection, we will show that it is possible to summarize the information in a DNA sequence into its natural vector of high order, which in reverse determines the corresponding DNA sequence uniquely ( [1]). This fact can be illustrated by Theorem 3.3. Newton identity (Lemma 3.2) is applied to prove this theorem.

**Lemma 3.2.** *[Newton identity] Let* $p(z) := s_0 z^n + s_1 z^{n-1} + ... + s_n$ $(s_0 = 1)$ *and its roots are* $z_1, ..., z_n$, $P_k := \sum_{i=1}^{n} z_i^k$, *then*

$$P_d s_0 + P_{d-1} s_1 + ... + P_1 s_{d-1} + d s_d = 0, \ \forall d \leq n.$$

*Proof.* See [3]. □

**Theorem 3.3.** *Let* $s$ *be a sequence whose* 0*-order elements are* $n_A, n_C, n_G, n_T$ *respectively. Let* $m = \max\{n_A, n_C, n_G, n_T\}$, $v$ *is the* $m$*-order natural vector of* $s$*, then there is no other sequence* $\tilde{s}$ *whose* $m$*-order natural vector is also* $v$*. Moreover, given the* $m$*-order natural vector* $v$*, we can calculate its corresponding sequence* $s$*.*

*Proof.* Based on $v$, we can easily get $n, n_A, n_C, n_T, n_G$. Thus we only need to show that from $v$, $n_A$ locations of element $A$ in total $n$ locations are determined uniquely, and therefore we can determine the whole sequence. $(C, G, T$ are similar.) We use $z_i$ $(i = 1, ..., n_A)$ to denote the locations of $A$ $(z_1 < z_2 < ... <$

$z_{n_A}$) in the sequence. Denote $\tilde{z}_k = z_k - \mu_k$, the following $T_1, ..., T_{n_A}$ can be calculated from $v$:

$$\begin{cases} T_1 = \tilde{z}_1 + ... + \tilde{z}_{n_A} \\ T_2 = \tilde{z}_1^2 + ... + \tilde{z}_{n_A}^2 \\ ... \\ T_{n_A} = \tilde{z}_1^{n_A} + ... + \tilde{z}_{n_A}^{n_A}. \end{cases}$$

Let $p(z) = s_0 z^{n_A} + s_1 z^{n_A - 1} + ... + s_{n_A} = (z - \tilde{z}_1)...(z - \tilde{z}_{n_A})$, and $s_0 = 1$. By Lemma 3.2, we have

$$\begin{cases} T_1 s_0 + s_1 = 0, \\ T_2 s_0 + T_1 s_1 + 2s_2 = 0, \\ T_3 s_0 + T_2 s_1 + T_1 s_2 + 3s_3 = 0, \\ ... \\ T_{n_A} s_0 + T_{n_A - 1} s_1 + \cdots + n_A s_{n_A} = 0. \end{cases}$$

We can solve $s_i$ by $T_i$:

$$\begin{cases} s_1 = -T_1, \\ s_2 = \frac{T_1^2 - T_2}{2}, \\ s_3 = \frac{-T_1^3 + 2T_3 - 3T_1 T_2}{6}, \\ ... \\ s_{n_A} = \sum\limits_{\substack{\sum_{i=1}^{n_A} i m_i = n_A \\ m_i \geq 0}} \prod_{i=1}^{n_A} \frac{(-T_i)^{m_i}}{m_i! i^{m_i}} \end{cases}$$

Therefore, we get the parameter of $p(z)$ and the corresponding solutions $\tilde{z}_1, ..., \tilde{z}_{n_A}$ can be calculated by enumerating $(l - \mu_A)$ where $l$ is a integer ranging from 1 to $n$. Therefore, the positions of all $A$ are determined. Similarly we can obtain the positions of $C$, $G$, $T$ and therefore the entire sequence. □

### 3.3 $k$-mer natural vectors

Besides increasing the orders of natural vector, we also applied the $k$-mer natural vector idea in this project [4]. $k$-mer natural vector is based on the combination of $k$-mer and the original natural vector. In the $k$-mer natural vector method, we replace the four kinds of nucleotides $\{A, T, C, G\}$ by $4^k$ elements, that is, treating all possible $k$-mer subsequence $s_1...s_k$ ($s_i \in \{A, T, C, G\}$) of length $k$ as a unit. (These units are denoted by $l_1$,...,$l_{4^k}$.) Then we calculate the natural vector of sequence

$$(n_{l_1}, ..., n_{l_{4^k}}, \mu_{l_1}, ..., \mu_{l_{4^k}}, D_2^{l_1}, ..., D_2^{l_{4^k}}, ..., D_m^{l_1}, ..., D_m^{l_{4^k}})$$

similarly as in the original natural vector, where $n_{l_j}$ is the number of occurrence of the $j$th string in our sequence (corresponding to the previous $n_A$) and so on. We usually take first three orders in practice ($m = 2$), and therefore the dimension of a $k$-mer natural vector is $3 \times 4^k$. Compared with 1-mer natural vectors, $k$-mer vectors include the relationship between adjoining sites.

We take the sequence $ACGCGCGT$ as an example and calculate its 2-mer natural vector to show how this works. There are 4 kinds of 2-mer ($AC$,$CG$,$GC$,$GT$) in the sequence and we may denote them by $l_1, ..., l_4$ ($l_5, ..., l_{16}$ are set to 0 automatically because they do not appear in the example sequence), thus we have

$$\begin{cases} n_1 = 1, \\ n_2 = 3, \\ n_3 = 2, \\ n_4 = 1, \\ \mu_1 = 1, \\ \mu_2 = \frac{2+4+6}{3} = 4, \\ \mu_3 = \frac{3+5}{2} = 4, \\ \mu_4 = 7, \\ D_2^1 = 0, \\ D_2^2 = \frac{(2-4)^2+(4-4)^2+(6-4)^2}{3\times 7} = \frac{8}{21}, \\ D_2^3 = \frac{(3-4)^2+(5-4)^2)}{2\times 7} = \frac{1}{7}, \\ D_2^4 = 0, \end{cases}$$

Therefore the final 2-mer natural vector of $ACGCGCGT$ is

$(1, 3, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 4, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \frac{8}{21}, \frac{1}{7}, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

# 4 Geometry of natural vectors

In this section, we show that natural vectors of the sequences from the same family actually gather together in the genome space. In fact, it can be illustrated from the perspective of convex hull principle.

**Definition 4.1.** *$y_1, ..., y_m \in R^n$. A convex combination of $y_1, ..., y_m$ is defined as $\lambda_1 y_1 + ... + \lambda_m y_m$, where $\lambda_i \in [0, 1]$ and $\sum_{i=1}^{m} \lambda_i = 1$.*

**Definition 4.2.** *The set $P \subset R^n$ is called a convex set if $\forall\ y_1, ..., y_m \in P$, every convex combination of $y_1, ..., y_m$ is still located in $P$.*

**Definition 4.3.** *Let $P \subset R^n$. The convex hull of $P$, $Conv(P)$, is defined as the minimum convex set that contains $P$. Equivalently, we can define the convex hull as the set of all convex combination of points in $P$.*

We illustrate the gathering of natural vectors from the same family by the convex hull principle, that is, the natural vectors of sufficiently high order from different families form disjoint convex hulls. On the other hand, if the two convex hulls intersect, we believe that those two families may share some similar biological characteristics.

We need an algorithm to check whether two convex hulls are disjoint or not in order to show the gathering property of sequences from the same family. Actually, the problem can be transformed to solving the following equations:

$$\begin{cases} \sum_{i=1}^{s} \lambda_i a_i - \sum_{j=1}^{t} \beta_j b_j = 0, \\ \sum_{i=1}^{s} \lambda_i = 1, \\ \sum_{j=1}^{t} \beta_j = 1, \\ 0 \le \lambda_i \le 1, i = 1, ..., s, \\ 0 \le \beta_j \le 1, j = 1, ..., t, \end{cases}$$

where $a_1, ..., a_s$ and $b_1, ..., b_t$ are natural vectors from two different families. This equation is solved by *linprog* function in MATLAB.

During programming in MATLAB, it is of great significance to avoid the numerical error because high-order elements of natural vectors tend to zero. Therefore, we need to do the maximum normalization for each dimension to get consistent scale. More specifically, if we denote the $j$-th dimension of the $i$-th natural vector as $v_i^{(j)}$, instead of using $v_i^{(j)}$ directly in the computation, we actually consider $N * \frac{v_i^{(j)}}{\max\limits_l |v_l^{(j)}|}$ after normalization, where $N$ is a large number to make the values not tend to zero. For the reference sequences of all viruses, we choose $N = 10,000$ and for SARS-CoV-2, we choose $N = 10^{15}$ because the sequences are relatively much shorter. As is proved in Lemma 4.4, it is easy to show that the maximum normalization preserve the disjoint properties of convex hull.

**Lemma 4.4.** *$P_1, P_2 \subset R^n$ are two sets. $A \in GL_n(F)$, i.e. $A$ is a $n \times n$ reversible matrix. $AP_i := \{Ax | x \in P_i\}$. Then $Conv(P_1)$ and $Conv(P_2)$ are disjoint if and only if $Conv(AP_1)$ and $Conv(AP_2)$ are disjoint.*

*Proof.* If $Conv(P_1)$ and $Conv(P_2)$ are not disjoint, there exist $\lambda_1, ..., \lambda_s, \beta_1, ..., \beta_t \in$

$R$ satisfying

$$\begin{cases} \sum_{i=1}^{s} \lambda_i = 1, \\ \sum_{j=1}^{t} \beta_j = 1, \\ 0 \le \lambda_i \le 1, i = 1, ..., s, \\ 0 \le \beta_j \le 1, j = 1, ..., t, \end{cases}$$

and $a_1, ..., a_s \in P_1,\ b_1, ..., b_t \in P_2$ such that

$$\sum_{i=1}^{s} \lambda_i a_i - \sum_{j=1}^{t} \beta_j b_j = 0.$$

Therefore,

$$\sum_{i=1}^{s} \lambda_i A a_i - \sum_{j=1}^{t} \beta_j A b_j = 0$$

and $Conv(AP_1)$ and $Conv(AP_2)$ are not disjoint. The other direction is the same since $A$ is reversible. □

By Lemma 4.4, we could magnify the high-order elements approaching zero by acting on them using a matrix with large coefficents such that they will not ignored in programming. Then the disjoint properties are preserved. We first consider Dataset 1, where 11,559 sequences from 123 families form 123 convex hulls. The disjoint status of $\binom{123}{2} = 7,503$ pairs of convex hull are checked and shown in Table 1. Results show that these 7503 pairs of convex hulls of 33-order natural vector do not intersect, in other words, these pairs are disjoint in the $4\times(33+1) = 136$ dimensional space. Detailed results are shown below. (Disjoint percentage is defined as $\frac{\text{The number of disjoint pairs}}{\text{The number of all pairs}}$ where the denominator is 7,503 in this case.)

| **order** | **disjoint pairs** | **disjoint percentage** |
|---|---|---|
| 2 | 6759 | 90.0% |
| 3 | 7040 | 93.8% |
| 4 | 7337 | 97.8% |
| 5 | 7378 | 98.3% |
| 6 | 7421 | 98.9% |
| 7 | 7435 | 99.1% |
| ... | ... | ... |
| 33 | 7503 | 100% |

Table 1: The disjoint percentage of families on Dataset 1 increases as order increases, and equals 100% when order=33.

As shown in the Table 1, even for natural vectors of low order, over 90% of the pairs are disjoint. Only a few families that share a great deal of biological similarities require high order to detect the difference. This is a strong evidence that those 123 families in Dataset 1 are generally distinct from each other.

The convex hull principle of natural vectors is true not only on the family level of viruses, but also on the variant level of viruses. We consider Dataset 3 and find that natural vectors of sequences from different variants form disjoint convex hull. In the Dataset 3, 287,182 sequences are from 5 variants. The result of convex hull analysis shows that 5 convex hulls are mutually disjoint for 23-order natural vectors in a 96-dimensional space ($96 = 4 \times (23 + 1)$).

# 5 Virus classification: $k$-NN algorithm and weight training technique

This section mainly discusses how to determine the family that a DNA sequence belongs to via calculating the distance between different families. A well known supervised algorithm to solve this problem is $k$ nearest neighbors method ($k$-NN method) [5] and here we select $k = 1$ to achieve the goal.

Given a series of natural vectors $u_1, ..., u_m$ with known family information and a query natural vector $v$, 1-NN method is done by the following steps. First, we calculate the distance $d_i := dis(u_i, v)$ (the choice of metric will be discussed later). Then, we take $i$ from $\{1, ..., m\}$ such that $d_i$ is the smallest number among $\{d_1, ..., d_m\}$. Finally, we predict the family of $v$ to be the family that $u_i$ belongs to.

We prefer to consider $k$-mer natural vectors instead of high-order natural vectors because high orders in the original natural vector approach zero and therefore increasing orders will not significantly improve the result. For each $k$, $D_k$ is defined to represent the Euclidean metric for $k$-mer natural vectors.

To include more information of the sequence, we combine the distances of multiple $k$ together and the weighted sum will refine the distance so that it describes the similarity between sequences more accurately. We denote $k$ from 1 to $K$ in the $k$-mer method.

$$Dis_K = \sum_{i=1}^{K} a_i D_i$$

Then the problem becomes the choice of $a_i$. Three candidates of distances are considered here. Based on previous experiments [2], $a_i = \frac{1}{2^i}$ is a good weight in the sense that the percentage of the correct predictions are relatively high (corresponding distance denoted by $Dis_K^1$), and sometimes $a_i = \frac{1}{i^2}$ is also

satisfactory (corresponding distance denoted by $Dis_K^2$). We also consider the original $k$-mer weight: $\begin{cases} a_i = 1, \ i = K \\ a_i = 0, \ otherwise \end{cases}$ and denote it by $Dis_K^3$.

We apply these three distances on Dataset 1 and evaluate it using the leave-one-out cross validation (LOOCV): assume that there are $n$ sequences in the dataset, for each sequence, we take the remaining $n-1$ sequences as the training set to test it. By this method, we can make a prediction of the family of one sequence and compare it to its true family. The percentage of the correct predictions is defined as the accuracy for the distance. Accuracy results are shown in the following table:

| $K$ | $Dis_K^1$ | $Dis_K^2$ | $Dis_K^3$ |
|---|---|---|---|
| 1 | 79.80% | 79.80% | 79.80% |
| 2 | 83.31% | 82.48% | 83.46% |
| 3 | 84.05% | 83.33% | 78.17% |
| 4 | 83.58% | 83.68% | 72.07% |
| 5 | 84.19% | 84.62% | 68.05% |
| 6 | 85.76% | 86.50% | 57.15% |
| 7 | 86.84% | 87.45% | 46.66% |
| 8 | 87.50% | 87.02% | 41.17% |
| 9 | 87.66% | 83.76% | 36.68% |

Table 2: The 1-NN accuracy of $Dis^1(a_i = \frac{1}{2^i})$, $Dis^2(a_i = \frac{1}{i^2})$ and $Dis^3$**(pure** $K$**-mer,** $a_i = 1$ **only when** $i = K$**)** on Dataset 1

Results in Table 1 show that, the pure $K$-mer weight is the worst and the weight $\frac{1}{2^i}$ is the best for most cases. For the weight $\frac{1}{2^i}$, the accuracy is relatively satisfying when $K$ is high.

We further develop the weight training technique to determine a suitable weight without trials and experience to increase accuracy. Motivated by gradient descent method [6] in optimization problems, we first randomly generate a weight for each dimension and then train the weights for multiple iterations to get the optimal solution.

In order to adapt different scales for the weights of different dimensions, we apply multiplication instead of addition in the training. More precisely, if $w_n \in R^n$ is the current weight and $v \in R^n$ is the direction, then the new weight is calculated by:

$$w_{n+1} = w_n \odot (1_n + v)$$

where $1_n = (1, ..., 1) \in R^n$ and Hadamard product $\odot$ is the element wise product defined by $(a_{ij}) \odot (b_{ij}) = (a_{ij} b_{ij})$.

**Definition 5.1.** *$v$ is called a non-decreasing direction on $w$ if*

$$F(w) \leq F(w \odot (1_n + v)).$$

In order to find a non-decreasing direction, we consider three candidates, which include a random direction, its reverse direction, and the same direction that we just moved from. Among those three directions, we choose the one that improve the accuracy most as our direction for this step.

To be more specific, we propose the weight training algorithm as follows:

**Algorithm 1** Weight training algorithm

**Require:** iteration number $N$, $K$ (the largest $k$ for $k-$mer), distribution $X$.

1: Initialization: Randomly select $K$ numbers from uniform distribution $U[0, 1]$ as the initial weight $w_0$.
2: **for** $i = 1$ to $N$ **do**
3: Randomly select $K$ numbers from $X$ to form a direction $v_i^{random}$.
4: If there are non-decreasing directions on $w_{i-1}$ in $v_i^{random}$,$-v_i^{random}$,$v_i^{mem}$, choose the one with the highest accuracy as the direction for step $i$, denoted as $v_i^{chosen}$. Then let $v_{i+1}^{mem} = v_i^{chosen}$ and $w_i = w_{i-1} \odot (1 + v_{chosen})$. If not, $v_{i+1}^{mem} = v_i^{mem}$ and $w_i = w_{i-1}$.
5: **end for**

During training, it is important to watch out for over-fitting. In this problem, "over-fitted" means that the the weight trained by one dataset is only suitable for itself, but on a new set (as the test), the performance is bad. Therefore, if new sequences are found and added to the input, the performance of the weight will decrease dramatically. In this paper, a natural way is applied to check if our method is over-fitted. First, we train our weight by Dataset 2, a previous dataset for whole virus reference sequences. Then, we apply this weight to Dataset 1, the latest dataset for whole virus reference sequences. If the weight also performs well on Dataset 1, it is safe to say that the weight is stable and is not over-fitted.

Different choices of $X$ such as normal distributions, Cauchy distributions and uniform distributions are considered here. However, experimental results show that their performances are quite similar while the uniform distribution is slightly better with the simplest form. Therefore, we choose the uniform distribution to be $X$ with variance is selected to be $0.3^2$ (Thus $X$ is chosen as $U[-0.52, 0.52]$).

Given the iteration number $N = 100$, the largest value of $k$, i.e., $K = 9$, and a random weight with initial accuracy is 45.67%, after the training process, we obtain the following weight (all weight divided by the first item to make $\overline{a}_1 = 1$)

$$\overline{a}_1 = 1$$
$$\overline{a}_2 = 3.7 \times 10^{-1}$$
$$\overline{a}_3 = 7.2 \times 10^{-2}$$
$$\overline{a}_4 = 5.9 \times 10^{-3}$$
$$\overline{a}_5 = 2.2 \times 10^{-2}$$
$$\overline{a}_6 = 1.6 \times 10^{-2}$$
$$\overline{a}_7 = 1.2 \times 10^{-2}$$
$$\overline{a}_8 = 5.7 \times 10^{-3}$$
$$\overline{a}_9 = 2.9 \times 10^{-4}$$

The accuracy of this weight on the training set (Dataset 2) is 88.44%. And we apply the result to Dataset 1 as the testing set. The result is shown as follows: (When $K < 9$, the distance is truncated, indicating only considering $\overline{a}_1$,...,$\overline{a}_K$. This distance is denoted by $\overline{Dis}_K$.)

| $K$ | $\overline{Dis}_K$ |
|---|---|
| 1 | 79.80% |
| 2 | 82.97% |
| 3 | 83.55% |
| 4 | 83.75% |
| 5 | 84.96% |
| 6 | 86.46% |
| 7 | 87.66% |
| 8 | 88.10% |
| 9 | 88.16% |

Table 3: The 1-NN accuracy of $\overline{Dis}_K$ on Dataset 1 increases as $K$ increases

From Table 3, we can see that the weight trained by Dataset 2 actually works on Dataset 1 with the best performance of 88.16%. For most cases, new weight performs better than the fixed weight $a_i = \frac{1}{2^i}$, which is the best weight without any training.

Therefore, the weight training provides us a way to automatically choose good weights and guarantees our accuracy of correct prediction. Besides, virus classification can be done in the level of not only families but also variants. Take SARS-CoV-2 as example. There are many variants in SARS-CoV-2 and we can also use 1-NN algorithm to classify them. Since there are 287,182 sequences in

Dataset 3 and applying 1-NN is too time-consuming, we apply 1-NN to Dataset 4. The results are good for all the three random sets in Dataset 4. Here in Table 5 we select the first random dataset for Dataset 4 as an example.

| $K$ | $Dis_K^1$ | $Dis_K^2$ | $Dis_K^3$ |
|---|---|---|---|
| 1 | 95.2% | 95.2% | 95.2% |
| 2 | 97.9% | 97.6% | 97.85% |
| 3 | 99.05% | 99% | 99.2% |
| 4 | 99.35% | 99.3% | 99.35% |
| 5 | 99.35% | 99.4% | 99.4% |
| 6 | 99.85% | 99.85% | 99.85% |
| 7 | 99.85% | 99.85% | 99.85% |
| 8 | 99.85% | 99.85% | 99.9% |
| 9 | 99.85% | 99.85% | 99.9% |

Table 4: The 1-NN accuracy of $Dis_K^1$, $Dis_K^2$, and $Dis_K^3$ on one set of Dataset 4

Different from 1-NN results of Dataset 1 (Table 2), the accuracy shown in Table 4 is good enough when $K$ is large. Therefore, it is unnecessary to use weight training technique for Dataset 4. In section 6, we will take $Dis_9^3$ as the metric to identify new variant for SARS-CoV-2 since it is the best weight shown in Table 4.

# 6 New variant identification

In this section, we develop a method to identify new variants and apply to Dataset 4 (a random subset of SARS-CoV-2 genomes).

First, we define a distance between a point and a set as below, and then use it to detect viruses from new variants.

**Definition 6.1.** *$(R^n, d)$ is a metric space, $v \in R^n$, $P \subset R^n$, we define*

$$D(v,P) = \begin{cases} \min\limits_{u\in P} d(v,u), & v \notin P, \\ \min\limits_{u\in P\setminus\{v\}} d(v,u), & v \in P. \end{cases}$$

*The choice of $d$ is $Dis_9^3$ for this paper.*

We denote $P_1, ..., P_5$ as the sets of natural vectors of the five variants of SARS-CoV-2. Let $v$ be a random sequence not from $P_i$, then $d(v, P_i)$ is a random variable denoted by $Y_i^{(\text{not})}$. Similarly, let $v$ be a random sequence from $P_i$, then $d(v, P_i)$ can be denoted by $Y_i^{(\text{in})}$. We claim that those two variables, $Y_i^{(\text{not})}$

and $Y_i^{(\text{in})}$ have different distributions and it can be applied to determination of whether a sequence is from $P_i$ or not.

Given a random set in Dataset 4, for each $i$, we have 400 samples for $Y_i^{(in)}$ and 1600 samples for $Y_i^{(\text{not})}$. The densities of $Y_i^{(in)}$ and $Y_i^{(\text{not})}$ are plotted by kernel density estimation [7] in Figure 1, using Beta variant as an example. (The densities of other variants are listed in the supplemental materials.) Kernel density estimation is a method to plot the density function, which is a more direct visualization graph than histogram. Its idea is simple: given a period $[x-h, x+h]$ and $N$ is the sample size, we can approximate $2hf(x)$ by $\frac{1_{]x-h,x+h]}(x_i)}{N}$, where $x_1, ..., x_N$ are samples and $1_A$ is the indicator function defined by

$$1_A(x) = \begin{cases} 1, & x \in A, \\ 0, & x \notin A. \end{cases}$$

Denote $K(x) = \frac{1}{2}1_{[-1,1]}(x)$, then

$$2hf(x) \approx \frac{2K(\frac{x_i - x}{h})}{N},$$

$$f(x) \approx \tilde{f}(x) := \frac{1}{hN} K(\frac{x_i - x}{h}).$$

However, $\tilde{f}$ is not smooth enough and may be inaccurate if the data is not sufficient. Therefore, in kernel density estimation, $K$ is changed to $K_G$, the density function of a zero-mean normal distribution similar to $K$. We take $\frac{1}{hN} K_G(\frac{x_i - x}{h})$ as the approximated density.

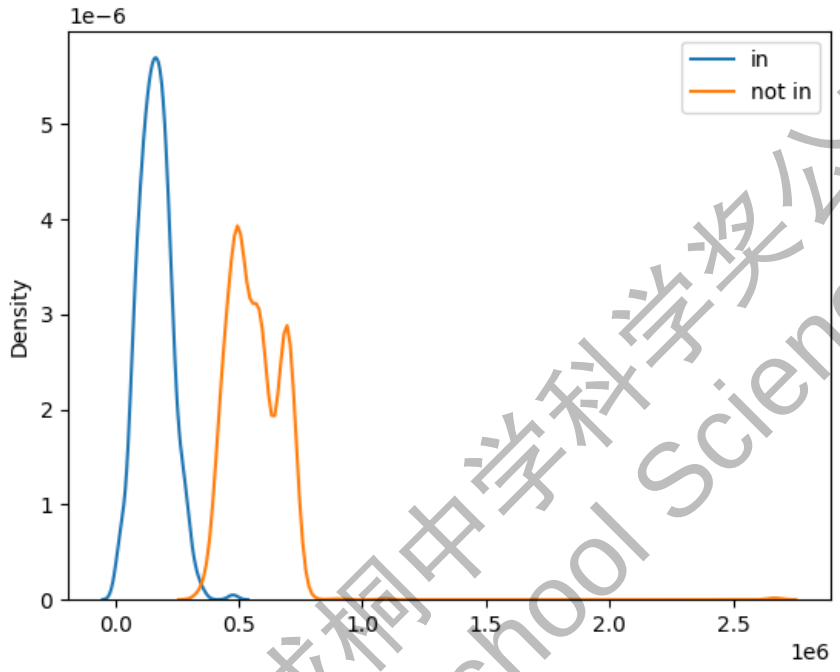


Figure 1: Different distributions of $d(v, P_i)$ for variant Beta of SARS-CoV-2. For $Y^{\text{in}}$, $v \in P_i$; for $Y^{\text{not}}$, $v \in P_i^c$. $P_i = \{\text{sequences in Beta variant}\}$

We see from Figure 1 that the significant difference between the distribution of $Y^{(\text{in})}$ and $Y^{(\text{not})}$, which offers us a way to detect new variants based on its location in the distributions.

**Definition 6.2.** *$f_1, ..., f_n \in R$, then the upper $\alpha$ quantile of $f_1, ..., f_n$ is defined by*

$$\min\{x | Count(f_i \leq x) \geq (1 - \alpha)n\}.$$

**Definition 6.3.** *A upper $\alpha$ distance bound for a set $P_i$ is defined as the upper $\alpha$ quantile of $\{D(v, P_i) | v \in P_i\}$, denoted by $B_i^\alpha$.*

Based on what we have got so far, we propose the following algorithm to detect sequences from new variants.

**Algorithm 2** New variant identification algorithm

**Require:** $\alpha$ in the upper quantile

1: Calculate $B_i^\alpha$ $(i \in I)$ defined in Definition 6.3 ($I$ is the index set of given variants).
2: Given a new sequence $v$, calculate $d_i := D(v, P_i)$ $(i \in I)$ where $D$ is defined in Definition 6.1.
3: If $d_i > B_i^\alpha$ for each $i$, we claim that $v$ is a sequence from new variant.

We apply this algorithm to the identification of new variants in Dataset 4 (SARS-CoV-2), and evaluate it by the following two approaches.

The first approach takes the time order into account. To be more specific, this approach identifies a variant based on the variants found before it. This way can better reflect the cases in reality. New variants are detected successively by predicting whether a new sequence belongs to any known variant. For our dataset, the well-acknowledged chronological order of the five variants is Alpha→Gamma→Beta→Delta→Omicron. Take Delta as an example. For each sequence in Delta variant, we calculate its distance to the three variants before it (Alpha, Gamma, and Beta), and then compare the distances to the threshold respectively. If our algorithm suggests that the sequence doesn't belong to Alpha, Gamma, or Beta, then we predict the new sequence is actually a new (Delta) variant.

We take $\alpha = 0.01$ and test the algorithm, the accuracy is shown as follows (all three random sets in Dataset 4 are considered):

| New variant | Random set 1 | Random set 2 | Random set 3 |
|---|---|---|---|
| **Gamma** | 100% | 100% | 100% |
| **Beta** | 100% | 100% | 99.75% |
| **Delta** | 100% | 100% | 100% |
| **Omicron** | 99.75% | 99.5% | 99.75% |

Table 5: Accuracy of new variant identification on Dataset 4 when $\alpha = 0.01$ (Approach 1)

The second approach ignores the time order of variants. We simply identify one variant based on the information of all other variants. For example, we can take Alpha, Beta, Gamma, Omicron as given variants and predict whether Delta is a new variant. It may not be the case in reality but it is also a way to check the effect of our method. Take $\alpha = 0.01$ and the result is shown as follows:

| New variant | Random set 1 | Random set 2 | Random set 3 |
|---|---|---|---|
| **Alpha** | 100% | 100% | 100% |
| **Gamma** | 100% | 100% | 99.75% |
| **Beta** | 100% | 100% | 99.75% |
| **Delta** | 99.75% | 99.75% | 99.75% |
| **Omicron** | 99.75% | 99.5% | 99.75% |

Table 6: Accuracy of new variant identification on Dataset 4 when $\alpha = 0.01$ (Approach 2)

The results in Table 5 and 6 show that under both evaluation method, our algorithm achieves good performance in the identification of new variants. The parameter, $\alpha = 0.01$, indicates that nearly 1% of sequences that are from a known family will be predicted to not from its family. And the high accuracy shows that there is low possibility that a new variant is not detected.

The analysis above can be rewritten in the language of hypothesis testing. Type I error means rejecting a true case while type II error means accepting a wrong case.

| | **Result: a new variant** | **Result: not a new variant** |
|---|---|---|
| **Reality: a new variant** | correct | Type I Error |
| **Reality: not a new variant** | Type II Error | correct |

Table 7: New variant identification from the perspective of hypothesis testing

The two evaluation results actually suggest the probability of type I error. (Accuracy=1-P(Type I Error)). We find that the probability of type I error is pretty small. And the type II error is approximately bounded by $\alpha$ we choose. If a sequence is predicted to be from a new variant but actually not, then the distance between it and the variant should be larger than $(1-\alpha)$ of the distances between it and sequences from the same variant. Therefore, approximately $P(Type\ II\ Error) \leq \alpha$ (we use "approximately" because new sequences may be slightly different from known sequences even for the same variant). It shows that the probability of both type I and II error of our algorithm is sufficiently low, and is a good way for the identification of new variants.

# 7 Discussion and conclusion

Based on the convex hull principle, we have constructed the genome space of virus dataset in 136-dimensional Euclidean space and all pairs of convex hulls are pairwise disjoint. For 1-NN classification, unlike the previous human efforts,

that is, using weights given by trials and our experience, we propose a new technique to train the weight in this paper. The weights trained by our algorithm, are better than that with human intervention. We train the weights by the old dataset in 2020 (Dataset 2) and apply it to the latest dataset (Dataset 1). Under this new training weight, the accuracy of classification has been improved from 87.66% to 88.16%. Finally, we focus on SARS-CoV-2 dataset and develop an algorithm to detect new variants from known variants, which achieves an accuracy of over 99.5%.

Two new methods are proposed in this paper, i.e., weight training method and new variant identification method, to improve the accuracy of prediction on virus classification. Weight training method offers us a way to obtain the optimal weights for $k$-mer natural vector method, and integrates the information of different $k$s to describe the similarity between biological sequences. New variant identification method is of great importance for epidemic control from the prompt identification of new variants.

The paper can be further improved in the following aspects. First, the parameters of weight training technique, including distributions of random directions and the number of iterations, can be discussed and more candidate distributions can be considered to improve the accuracy. Second, since the dataset for SARS-CoV-2 is too large, some analysis is done in random subsets instead of the whole dataset. Thus more time-efficient algorithms might be proposed in the future to solve this issue. Third, for new variants detection, so far we can only apply the method on the known sequences on GISAID, but the performance on unknown variants remains unknown. Thus the evaluation on the new variants in the future would help us assess the method.

## References


[1] M. Deng, C. Yu, Q. Liang, R. He, and S.-T. Yau, "A novel method of characterizing genetic sequences: Genome space with biological distance and applications," *PloS one*, vol. 6, 03 2011.

[2] N. Sun, S. Pei, L. He, C. Yin, R. L. He, and S. S.-T. Yau, "Geometric construction of viral genome space and its applications," *Computational and Structural Biotechnology Journal*, vol. 19, 2021.

[3] M. Mosse, "Netwon's identities," 01 2019.

[4] J. Wen, R. H. Chan, S.-C. Yau, R. L. He, and S. S. Yau, "K-mer natural vector and its application to the phylogenetic analysis of genetic sequences," *Gene*, vol. 546, 2014.

[5] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, pp. 21–27, 1967.

[6] A. Cauchy, "Methode generale pour la resolution des systemes d'equations simultanees," 1847.

[7] S. Weglarczyk, "Kernel density estimation and its application," *ITM Web of Conferences*, vol. 23, p. 00037, 01 2018.

## Acknowledgements

I would thank my advisor, Rui Dong, from Tsinghua University, for her valuable guidance over the whole process of this work. She taught me the backgrounds of this project, such as the signifinance of virus genomic studies and the mathematical methods of building up suitable models. During this project I usually had some difficulties and felt upset to overcome them. It was her patience and wisdom that helped me out of trouble and depression. She also rectified many mistakes in my paper. My work would not be succesful if there were not her polish.

I would also express my gratitude to my parents and friends. They offered me spiritual supports and impetus to solve my problems.