Robust Representation and Efficient Feature Selection Allows for Effective Clustering of SARS-CoV-2 Variants: Paper Reproduction dan Improvements

Erland Rachmad Ramadhan[a)]
Master Program of Mathematics, Department of Mathematics,
Faculty of Mathematics and Natural Science,
Universitas Indonesia

(Dated: 4 November 2023)

The COVID-19 pandemic has led to an abundance of genomic data on the SARS-CoV-2 virus, presenting a unique opportunity for detailed analysis. This research benefits biologists, policymakers, and authorities in making informed decisions to control virus spread and prepares for future pandemics. Despite the challenge posed by the virus's diverse variants and mutations, this paper focuses on clustering spike protein sequences, crucial for understanding variant behavior. Utilizing a k-mers based approach, the original author of the paper create a fixed-length feature vector for spike sequences. The proposed feature selection method enables efficient clustering of spike sequences, demonstrating higher F_1 scores for clusters in both hard and soft clustering methods.

Keywords: COVID-19; SARS-CoV-2; Spike Protein Sequences; Clustering Analysis; Feature Selection; k-mers

I. INTRODUCTION

The SARS-CoV-2 virus, responsible for COVID-19, has a rapidly spreading genomic sequence worldwide. This genetic information is crucial for understanding outbreak dynamics, designing analyses, drugs, and vaccines, and monitoring changes in viral effectiveness over time. The virus's spike protein, particularly its S region, plays a key role in infection and exhibits significant genomic variation. To efficiently analyze this variation, the original author of the paper propose a focus on amino acid sequences encoded by the spike region using machine learning and clustering methods[1]. By converting these sequences into numeric vectors through k-mers, the original author of the paper aim to reduce data dimensionality and enhance analysis efficiency. The proposed approach integrates feature selection and clustering to gain insights into the virus's evolutionary dynamics, overcoming challenges posed by the vast number of available genomic sequences. The significance of the S protein makes it a potential target for therapeutic interventions and vaccine development. The methodology proposed ensures meaningful analytics, laying the foundation for effective strategies to combat the COVID-19 pandemic.

II. ALGORITHMS

In this section, the proposed algorithm is discussed in detail. The discussion start with the description of k-mers generation from the spike sequences. Then, the generation of feature vector representation from the k-mers information will be described. After that, discussion on different feature selection methods will be given in detail. Finally, the detail of the application of clustering approaches on the final feature vector represetation will be explained.

[a)]Electronic mail: mwkerr1916@icloud.com

A. k-mers Generation

Given a spike sequence, the first step is to compute all possible k-mers. The total number of k-mers that can be generated for a spike sequence are described as follows.

$$N - k + 1 \tag{1}$$

where N is the length of the spike sequence ($N = 1274$ for this paper dataset). The variable k is a user-defined parameter ($k = 3$ is chosen using standard validation set approach[2]).

B. Fixed-Length Feature Vector Generation

Since most of the Machine Learning (ML) models work with a fixed-length feature vector representation, the k-mers information is needed to be converted into the vectors. For this purpose, a feature vector Φ_k is generated for a given spike sequence a (i.e., $\Phi_k(a)$). Given an alphabet Σ (characters representing amino acids in the spike sequences), the length of $\Phi_k(a)$ will be equal to the number of possible k-mers of a. More formally,

$$\Phi_k(a) = |\Sigma|^k \tag{2}$$

Since there are 21 unique characters in Σ (namely $ACDEFGHIKLMNPQRSTVWXY$), the length of each frequency vector is $21^3 = 9261$.

C. Low Dimensional Representation

Since the dimensionality of data is high after getting the fixed length feature vector representation, different supervised and unsupervised methods is applied to obtain a low dimensional representation of data to avoid the problem of the *curse of dimensionality*[3,4]. Each of the methods for obtaining a low dimensional representation of data is discussed below:

1. Random Fourier Features

The first method that is used is an approximate kernel method called Random Fourier Features (RFF)[5]. It is an unsupervised approach, which maps the input data to a randomized low dimensional feature space (euclidean inner product space) to get an approximate representation of data in lower dimensions D from the original dimensions d. More formally:

$$z : \mathbb{R}^d \rightarrow \mathbb{R}^D \tag{3}$$

In this way, the inner product between a pair of transformed points is approximated. More formally:

$$f(x, y) = \langle \phi(x), \phi(y) \rangle \approx z(x)'z(y) \tag{4}$$

In Equation 4, z is low dimensional (unlike the lifting ϕ). Now, z acts as the approximate low dimensional embedding for the original data. Then, z can be used as an input for different ML tasks like clustering and classification.

2. Least Absolute Shrinkage and Selection Operator (Lasso) Regression

Lasso regression is a supervised method that can be used for efficient feature selection. It is a type of regularized linear regression variants. It is a specific case of the penalized least squares regression with an L_1 penalty function. By combining the good qualities of ridge regression[6,7] and subset selection, Lasso can improve both model interpretability and prediction accuracy[8]. Lasso regression tries to minimize the following objective function:

$$\min(\text{Sum of square residuals} + \alpha \times |\text{slope}|) \tag{5}$$

where $\alpha \times |\text{slope}|$ is the penalty term. In Lasso regression, the absolute value of the slope is chosen in the penalty term rather than the square (as in ridge regression[7]). This helps to reduce the slope of useless variables exactly equal to zero.

3. Boruta

The last feature selection method that is used is Boruta. It is a supervised method that is made all around the random forest (RF) classification algorithm. It works by creating shadow features so that the features do not compete among themselves but rather they compete with a randomized version of them[9]. It captures the non-linear relationships and interactions using the RF algorithm. It then extract the importance of each feature (corresponding to the class label) and only keep the features that are above a specific threshold of importance. The threshold is defined as the highest feature importance recorded among the shadow features.

D. Clustering Methods

In the original work, five different clustering methods (both hard and soft clustering approaches) namely k-means[10], k-modes[11], Fuzzy c-means[12,13], agglomerative hierarchical clustering, and Hierarchical density-based spatial clustering of applications with noise (HDBSCAN)[14,15] (note that is is a soft clustering approach). For the k-means and k-modes, default parameters are used. For the fuzzy c-means, the clustering criterion used to aggregate subsets is a generalized least-squares objective function. For agglomerative hierarchical clustering, a bottom-up approach is applied, which is acknowledged as the agglomerative method. Since the bottom-up procedure starts from anywhere in the central point of the hierarchy and the lower part of the hierarchy is developed by a less expensive method such as partitional clustering, it can reduce the computational cost[16].

HDBSCAN is not just density-based spatial clustering of applications with noise (DBSCAN) but switching it into a hierarchical clustering algorithm and then obtaining a flat clustering based in the solidity of clusters. HDBSCAN is robust to parameter choice and can discover clusters of differing densities (unlike DBSCAN)[15].

III. EXPERIMENTAL SETUP AND DATA PREPARATION

In this paper reproduction, the author utilizes only three clustering methods—k-means, k-modes, and Fuzzy c-means. HDBSCAN and agglomerative clustering are omitted due to their lack of parallelization, resulting in prolonged runtime execution, particularly for high-dimensional data such as genome sequences or fixed-length feature vector representations derived from raw data. For all clustering methods except k-means, the input data consists of low-dimensional representations obtained from processing the raw data. This approach is adopted to address the extended runtime execution associated with high-dimensional raw data input. The quality of the clustering algorithms is assessed using the F_1 score. The experiments are conducted on a Core i7 system with a MacOS operating system, 16GB of memory, and a 1.7GHz processor. The algorithm is implemented in Python, and the code is accessible at `https://github.com/erland-ramadhan/sars-cov2-variants-results-reproduction.git`.

IV. RESULTS AND DISCUSSION

TABLE I. Variant-wise F_1 (weighted) score for different clustering methods. (Reproduction work)

Methods	F_1 Score (Weighted) for Different Variants				
	Alpha	Beta	Delta	Gamma	Epsilon
K-Means	0.182	0.314	0.426	0.897	0.355
K-Means + Boruta	0.182	0.314	0.423	0.897	0.355
K-Means + Lasso	0.987	0.998	0.997	0.999	0.997
K-Means + RFF	0.917	0.0	0.0	0.659	0.320
K-Modes + Boruta	0.899	0.761	0.689	0.977	0.933
K-Modes + Lasso	0.994	0.970	0.997	0.999	0.995
K-Modes + RFF	0.178	0.0	0.0	0.981	0.320
Fuzzy + Boruta	0.186	0.316	0.414	0.897	0.356
Fuzzy + Lasso	0.976	0.998	0.985	0.997	0.972
Fuzzy + RFF	1.0	0.0	0.0	0.659	0.0

TABLE II. Variant-wise F_1 (weighted) score for different clustering methods. (Original work)

Methods	F_1 Score (Weighted) for Different Variants				
	Alpha	Beta	Delta	Gamma	Epsilon
K-Means	0.359	0.157	0.611	0.690	0.443
K-Means + Boruta	0.418	0.105	0.610	0.690	0.652
K-Means + Lasso	0.999	0.007	0.840	0.999	0.774
K-Means + RFF	1.0	0.0	0.288	1.0	1.0
K-Modes + Boruta	0.999	0.316	0.860	0.999	0.857
K-Modes + Lasso	0.999	0.173	0.917	0.998	0.076
K-Modes + RFF	1.0	0.0	0.0	0.613	1.0
Fuzzy + Boruta	0.357	0.154	0.613	0.690	0.443
Fuzzy + Lasso	0.999	0.314	0.647	0.999	0.816
Fuzzy + RFF	0.439	0.0	0.0	1.0	0.0

A. Contingency Table

B. Runtime Comparison

[1] Z. Tayebi, S. Ali, and M. Patterson, "Robust representation and efficient feature selection allows for effective clustering of sars-cov-2 variants," Algorithms **14** (2021), 10.3390/a14120348.

[2] P. A. Devijver and J. Kittler, *Pattern recognition: A statistical approach* (Prentice-Hall, London, GB, 1982).

[3] S. Ali, H. Mansoor, N. Arshad, and I. Khan, "Short term load forecasting using smart meter data," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy '19 (Association for Computing Machinery, New York, NY, USA, 2019) p. 419–421.

[4] H. Mansoor, S. Ali, I. Khan, N. Arshad, M. A. Khan, and S. Faizullah, "Short-term load forecasting using ami data," (2022), arXiv:1912.12479 [eess.SP].

[5] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, Vol. 20, edited by J. Platt, D. Koller, Y. Singer, and S. Roweis (Curran Associates, Inc., 2007).

[6] R. W. K. Arthur E. Hoerl and K. F. Baldwin, "Ridge regression: some simulations," Communications in Statistics **4**, 105–123 (1975), https://doi.org/10.1080/03610927508827232.

[7] G. C. McDonald, "Ridge regression," WIREs Computational Statistics **1**, 93–100 (2009), https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wics.14.
[8] R. Muthukrishnan and R. Rohini, "Lasso: A feature selection technique in predictive modeling for machine learning," in *2016 IEEE International Conference on Advances in Computer Applications (ICACA)* (2016) pp. 18–20.
[9] M. B. Kursa and W. R. Rudnicki, "Feature selection with the boruta package," Journal of Statistical Software **36**, 1–13 (2010).
[10] A. M. Fahim, A. M. Salem, F. A. Torkey, and M. A. Ramadan, "An efficient enhanced k-means clustering algorithm," Journal of Zhejiang University-SCIENCE A **7**, 1626–1633 (2006).
[11] S. S. Khan and A. Ahmad, "Cluster center initialization algorithm for k-modes clustering," Expert Systems with Applications **40**, 7444–7456 (2013).
[12] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," Computers & Geosciences **10**, 191–203 (1984).
[13] M. L. D. Dias, "fuzzy-c-means: An implementation of fuzzy *c*-means clustering algorithm." (2019).
[14] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*, edited by J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013) pp. 160–172.
[15] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," Journal of Open Source Software **2**, 205 (2017).
[16] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, "Efficient agglomerative hierarchical clustering," Expert Systems with Applications **42**, 2785–2797 (2015).