COVID-19 mRNA Degradation Prediction

Noor Rafiq, Jefferson Pan

Motivation

The COVID-19 pandemic has killed over a million people and brought along major societal disruption. The timely creation of a vaccine is necessary to solving this global health crisis. One of the leading candidates for a vaccine is an mRNA-based vaccine as they are cheaper and quicker to produce. mRNA vaccines best parallel a natural infection, making them more efficient and effective (1). In fact, mRNA is taken up by 95% of the host cells that it meets (1). Furthermore, mRNA vaccines are chemical-based, so they do not need to be grown in eggs or cells, which take longer to grow than an mRNA strand. This property makes them strong candidates for fast development at a large scale (1).

However, a major hurdle to developing an RNA vaccine is the fragility of RNA molecules. They have a propensity to spontaneously degenerate. Without understanding the causes behind this fragility, worldwide deployment of a COVID-19 vaccine would be a major logistical issue. Currently, delivery operators have prepared for a large-scale vaccine shipment by establishing a large network of freezers and cold-storage facilities capable of maintaining temperatures as low as -85 C (2). However, use of such units, as well as the need for rapid delivery, introduces a number of major risks that could negatively impact distribution (2). Therefore, developing a vaccine that is less sensitive to temperature and less likely to spontaneously degrade could reduce the potential harm of failures in the distribution network. This project intends to create a model to predict the reactiveness of RNA molecules in order to better understand the factors that lead to stability and more quickly reach a COVID-19 vaccine.

Background

RNA is a genetic molecule that contributes to cell protein synthesis and regulation. It is a single-stranded molecule made up of bonded ribonucleotides, each of which contains an arrangement of the four nitrogenous bases adenine, guanine, urasil, and cytosine. Like DNA, the nucleotides of RNA are connected in two different ways. One way is through stronger covalent bonds that form the chain of nucleotides. The other way is through weaker hydrogen bonds that form between complements of nucleotides with adenine and uracil being one pair and guanine and cytosine being another. Along with this paired structure, RNA can be of different shapes, including a multiloop, an internal loop, a bulge, a hairpin loop, a dangling end and an external loop. Furthermore, RNA uses the sugar ribose, which contributes to its unstable nature. RNA molecules are prone to complex base pairing between different sequences, which enables the formation of various three-dimensional structures (3).

Although there are three different types of RNA, the one used in vaccine development is mRNA, or messenger RNA. mRNA's responsibility is to carry instructions from the DNA about protein synthesis to different parts within a cell. Because certain proteins only need to be made at certain moments, mRNA is unstable and quickly breaks down (3). It's nature as the carrier for instructions regarding protein synthesis is what makes mRNA a highly preferred vaccine candidate. As soon as mRNA is taken up by a host cell, that cell can immediately begin protein production, allowing for quicker generation of virus antibodies (4).

The data for this project was generated by EteRNA, a crowdsourcing puzzle computer game in which players generate different RNA molecules in order to better understand how they fold. The highest-scoring results are generated in a lab and studied (5). To date, EteRNA has pioneered multiple novel research campaigns, including OpenCRISPR, OpenTB, and OpenVaccine, the Kaggle competition that this project is a part of that challenges Kaggle data scientist to develop a model that predicts the stability of RNA molecules generated by EteRNA. The dataset for this project will be composed of 3,000 different RNA molecules generated by EteRNA users (6).

Related Work

There were over a thousand entries in the OpenVaccine competition. Fortunately, some of the participants documented their solutions. Within the top ten solutions, a Graph Neural Networks (GNN) approach was the most popular with more than half using some variation. GNN is suited for this problem because RNA molecules can be depicted as graphs with the nucleotides acting as the nodes and covalent bonds and hydrogen acting as edges. In general, the way GNN works is by each node undergoing a process called “message passing”. Each node passes messages to the nodes it shares an edge with. The node takes a sum of all of its received messages, maybe also altering the weight of the message depending on the edge. The node then updates its internal weights depending on the sum of the messages.

Target Task

The target task is to create a model that predicts RNA reactivity, which is based on reactivity in a pH 10 solution, a 50 °C environment, a Magnesium solution, and a combination of those three environments. There are 2400 RNA molecules to train this model, and there are 629 RNA molecules to test it. The training data are molecules that are 107 nucleotides long. The features include each molecule’s nucleotide sequence, a string that indicates which nucleotides are paired and a string that indicates the predicted structure of each nucleotide with ‘M’ indicating multiloop, ‘I’ indicating internal loop and so on. The training data also includes reactivity data for the first 68 nucleotides in the sequence, which was found by doing tests on synthesized RNA molecules in a lab. The testing data are RNA molecules that are 130 nucleotides long, and the goal of this project is to predict the reactivity of the first 91 nucleotides. The real values of reactivity for the testing data will also be found in a laboratory setting. It is due to the limitations of the laboratory testing that we cannot get the reactivity for all of the nucleotides in the data.

An intuitive figure showing WHY Target Task is necessary and important

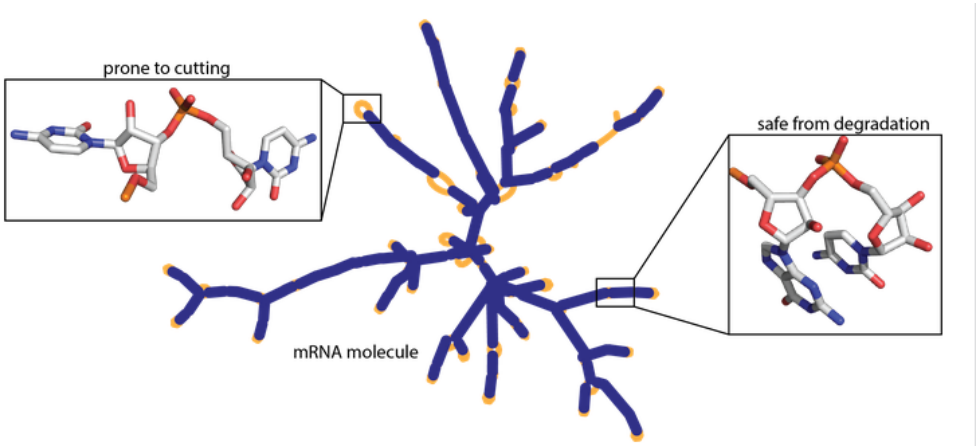

Fig. 1 Diagram of an mRNA molecule. This figure shows the complexity of an RNA molecule. Note the loop structure and the different types of connections between nucleotides.

Proposed Solution

A k-nearest neighbors regression algorithm was proposed for this project. Although k-nearest neighbors is traditionally a classification algorithm, it is possible to use it for regression because there seem to be classes or ranges of values within the reactivity data. Specifically, the reactivity values can be grouped into a class of values less than one, a class of single digit values and a class of values greater than one thousand. Furthermore, k-nearest neighbors is easy to implement and is an unexplored approach in the Kaggle competition. The novelty of this approach is this project's main contribution to the problem.

Implementation

The specifics of the implementation varied as different approaches to calculating distance or preprocessing were evaluated (a total of three different implementations were examined), but the general approach is as follows. First, the competition led by OpenVaccine provides a labeled training set of 2400 molecules, and an unlabeled testing set of 629 molecules. In order to measure the model's performance, the labeled training set was split into a training set of 1800 molecules and a testing set of 600 molecules.

After defining these sets, the features were processed. This processing varied depending on the approach being explored. In the first implementation, the model took in the features of molecule sequence (a string of length 68 with characters A, U, C and G representing the nucleotides present at each part of the sequence) and structure (a string of length 68 with characters ., (and) representing paired bases (each pair of opening and closing parentheses indicates a base pair)). The structure feature was processed by counting the number of pairs present, and the sequence string was not processed.

After processing, the reactivity values of each testing point were predicted. In the first implementation, this was done by finding the *k* nearest neighbors for each base in the sequence of each test point. Specifically, the distance between a base and the corresponding base of every training point was computed. Then, the first *k* bases with the closest distances were stored as the *k* nearest neighbors for that base. The distance metric was calculated using a

number of factors. First, the Hamming distance between both 68-length sequences was found, as well as the Hamming distance between the substring of the sequence around the base being evaluated. The difference between the number of pairs in each point was also calculated. The sum of the squares of these three values was taken, and the square root of this sum was used as the distance metric. However, when running the model, it was found that doing these calculations for each base of each testing point was too costly, and the model took too long to run. For larger datasets and longer sequences, such a model would not be practical. Therefore, the processing and distance determination were altered.

The second implementation explored was to first process the sequence feature using *k*-means clustering, and thus reduce the size of the training dataset from 2400 sequences to 500. The goal was to decrease the number of calculations needed for each *k*-nearest-neighbors search. Scikit-learn's *k*-means clustering library was fit to the training dataset in order to produce 500 centroids, which were then assigned reactivity values as the average of the data points that made up the cluster around each centroid. However, this algorithm did not produce good clustering results. Each of the 500 centroids were too homogenous to make strong predictions. Moreover, although the original reactivity values for each data point had potential values ranging from the negatives to the thousands, the articulation of these extremes was not present in the reduced dataset. Thus, it was concluded that this processing would not predict reactivity accurately, and the processing and distance determination were again altered.

The final implementation of the model involved *k*-mer frequencies. A *k*-mer is defined as all of the subsequences of a certain genomic sequence of length *k*. For this model, 3-mers were used. Since RNA contains only four different nucleotides, the total number of possible 3-mers is 64, and they are enumerated as follows:

UUU	UUA	UUC	UUG	UAU	UAA	UAC	UAG	UCU	UCA	UCC	UCG	UGU	UGA	UGC	UGG
AUU	AUA	AUC	AUG	AAU	AAA	AAC	AAG	ACU	ACA	ACC	ACG	AGU	AGA	AGC	AGG
CUU	CUA	CUC	CUG	CAU	CAA	CAC	CAG	CCU	CCA	CCC	CCG	CGU	CGA	CGC	CGG
GUU	GUA	GUC	GUG	GAU	GAA	GAC	GAG	GCU	GCA	GCC	GCG	GGU	GGA	GGC	GGG

The sequence for each molecule was thus converted into a frequency count of each 3-mer, reducing it from a string of length 68 to an integer array of length 64. The way in which *k* nearest neighbors was calculated was also altered. Instead of finding the *k* nearest neighbors for each base of each point, the *k* nearest neighbors were found for each point. In addition, the Euclidean distance was used instead of the Hamming distance.

After finding the *k* nearest neighbors for a point, the reactivity values were predicted by taking the average of the reactivities of the *k* neighbors. Since 3 different reactivities had to be predicted (reactivity, reactivity after incubation with magnesium at pH 10, and reactivity after incubation with magnesium at 50°C), 3 separate predictions were made. Essentially, 3 values were predicted for each of the 68 bases of each test point.

As specified by the OpenVaccine competition, the error metric used was the mean columnwise root mean squared error (MCRMSE). The formula is as follows:

$$\text{MCRMSE} = \frac{1}{N_t}\sum_{j=1}^{N_t}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij} - \hat{y}_{ij})^2}$$

where N_t is the number of prediction columns (3 for this model), n is the number of predictions made per column (68 for this model), y_{ij} is the actual reactivity value, and $\hat{y}_{ij}$ is the predicted reactivity value. Since multiple k values were tested, the one with the smallest error was chosen as the best k value.

Data Summary

The data is comprised of the mRNA molecule designs created by players in the game Eterna along with their respective reactivities, which were found in a laboratory setting. The features include a string sequence of the amino acids, a structure string designating which amino acids are paired and a predicted loop type string. The variables to be predicted include general reactivity, reactivity in a pH 10 solution and reactivity in a 50 degree solution. Each of these are arrays where each index represents the reactivity for the corresponding amino acid in the position of the sequence string. Since the mRNA molecules were conceived by people, they may not correspond to mRNA molecules that occur in nature, which means they may not correspond to mRNA molecules that could be involved with the COVID-19 vaccine. To counteract this, the organizers of this competition skewed their selection of mRNA molecules to prioritize diversity to try to minimize the bias that might be associated with human conceived mRNA molecules.

Experimental Results

The top scoring entries in the OpenVaccine Kaggle competition scored in the 0.340-0.350 range. By increasing k, the only hyperparameter of our model, the best performance of the model was 0.849 corresponding to a k value of 25. Further increases in k resulted in a plateaued MCRMSE.

k	MCRMSE
3	1.18
8	1.03
10	0.881

13	0.871
15	0.861
20	0.856
25	0.849

Fig. 2 A table showing results of changing k.

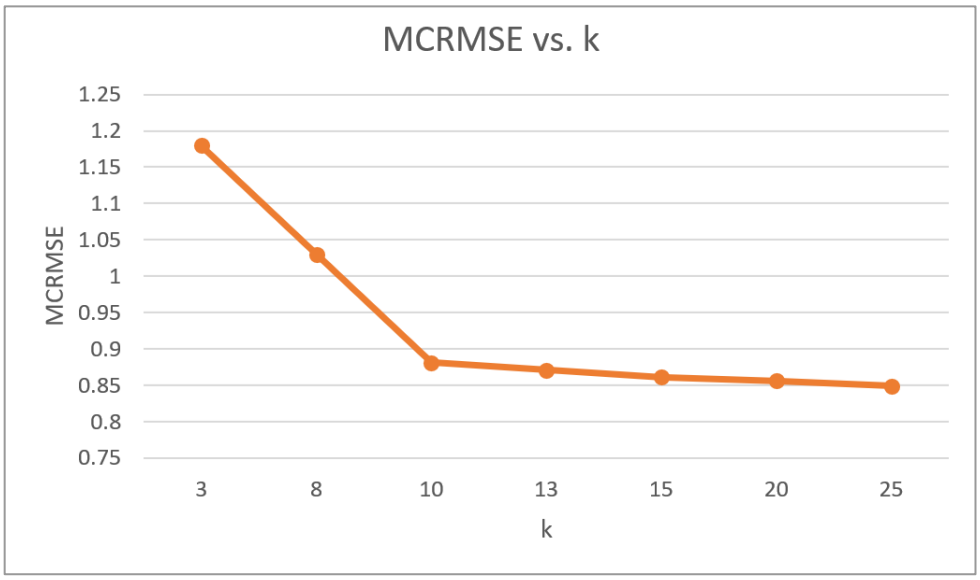

Fig. 3 Graph showing plateauing relationship between k and MCRMSE.

Experimental Analysis

The poor performance of the KNN-regression model can largely be attributed to a phenomenon called the Curse of Dimensionality - the principle that, as the number of features in a dataset increases, the space between each point increases exponentially. This phenomenon can be demonstrated by Figure 4, in which in a one dimensional space, the data is close, but as the number of features increases, the space between each feature increases as well. This phenomenon is problematic for KNN, which makes its predictions based on the distance

between data points. Because the data set had sixty four features, the data space was too large, and molecules that were similar and should have been close together were not due to the Curse of Dimensionality. Within KNN, a sign of the Curse of Dimensionality is the occurrence of hubs, or nodes that are represented as neighbors a disproportionate number of times. As the data gets further apart within the defined space, it becomes more likely that a test data point is not close to any data point in the training set, and that the closest neighbor, the hub, is actually far away and is not similar to the test point. Figure 5, a bar graph of the most frequently occurring KNN, shows that this problem is indeed occurring within the model. The most frequent molecule (molecule number 1253) occurs over sixty times, which means it was one of the nearest neighbors for roughly ten percent of the testing data.

The model implemented was a hybrid between a Naive Bayes Classifier and a KNN Regressor. Like a Naive Bayes Classifier, this model preprocessed the data by counting the frequencies of “words,” which were 3-mers for the genome sequence. A novel domain this model could be applied to is a variation of this problem in which a molecule simply needs to be classified between “stable” and “unstable”. Then, this model might be able to be adapted to more of a Naive Bayes Classifier with more success.

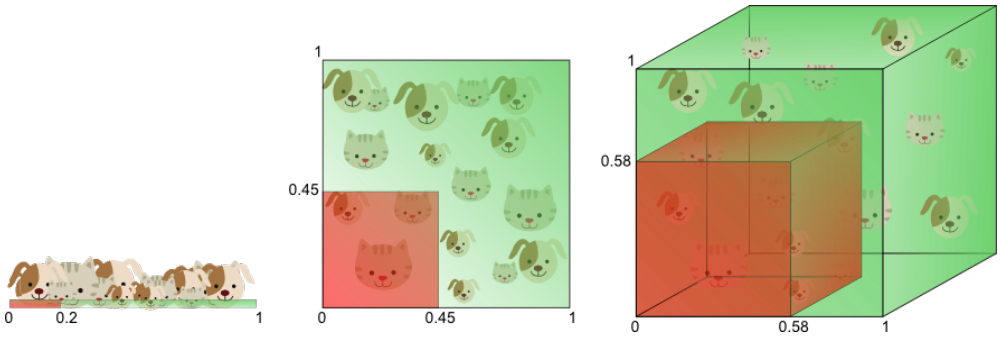

Fig. 4 Illustration of The Curse of Dimensionality

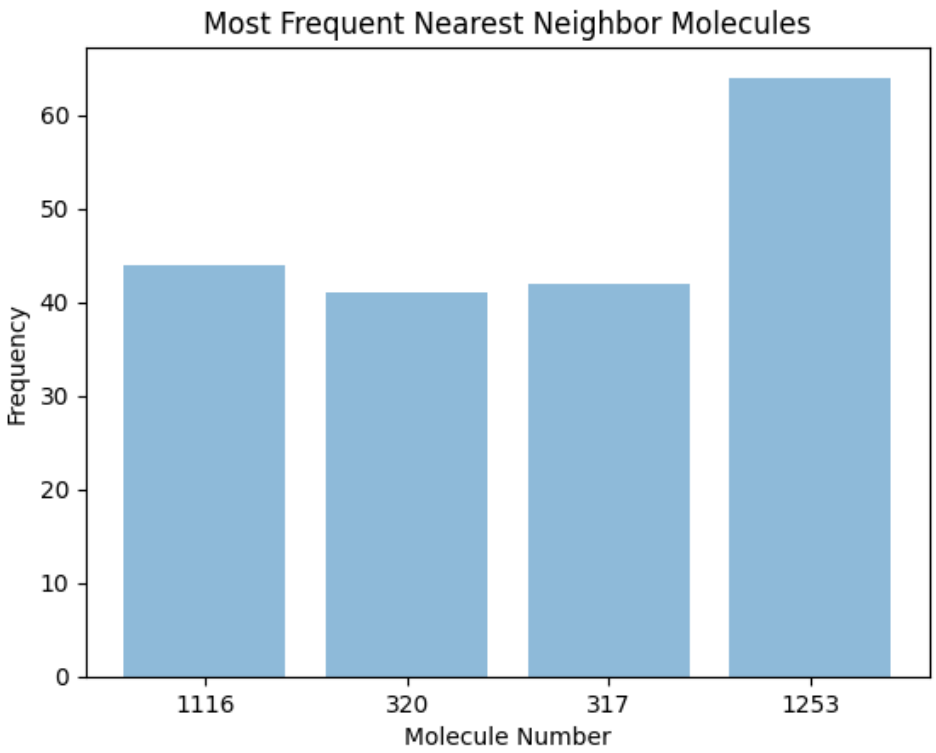

Fig. 5 Most frequently occurring molecules in KNN

Conclusion and Future Work

When using *k*-mer frequency counts as the feature and Euclidean distance as the distance metric, the *k*-nearest neighbor regression model performed best with a *k* of 25, producing an error of 0.849. The performance of this model, as compared to other models scored in the competition, was attributed to the Curse of Dimensionality phenomenon, as well as the method of prediction implemented (in which the prediction was the averages of the *k* neighbors' reactivities, without any form of weighting or normalization).

In order to reduce the effect of the Curse of Dimensionality when using *k* nearest neighbors on this dataset, a preprocessing method other than *k*-mers frequency counts should be used. Specifically, a method that minimizes the number of features and reduces the input space as much as possible should be prioritized. If the Curse of Dimensionality persists despite these efforts, the influence of hubs should also be reduced. Research has been made into potential ways of addressing this issue, and the results of these investigations should be incorporated into the model. In addition, the other information given about the molecules should be considered, including the structure and predicted loop types that OpenVaccine includes with its dataset. Furthermore, since potential reactivity values can range from the negatives to the thousands simply averaging the reactivities of the *k* nearest neighbors for a point is insufficient. Implementing such actions as weighting, normalization, or a majority vote could improve the accuracy of predictions made. Finally, more traditional regression methods should be explored, such as convolutional neural networks. Since *k* nearest neighbors regression depends on the testing dataset being similar to the training dataset, this model may behave more poorly as

different molecules (for example, molecules of differing lengths) are examined and predicted. Less restrictive models may have better performance on varying testing data.

References

https://www.nature.com/articles/d41586-020-02762-y - Mega

https://jamanetwork.com/journals/jama/fullarticle/2770485 (1)
https://www.wsj.com/articles/from-freezer-farms-to-jets-logistics-operators-prepare-for-a-covid-19-vaccine-11598639012 (2)
https://courses.lumenlearning.com/microbiology/chapter/structure-and-function-of-rna/ (3)
https://www.modernatx.com/mrna-technology/science-and-fundamentals-mrna-technology (4)
https://www.cnn.com/interactive/2012/08/tech/gaming.series/research.html (5)
https://eternagame.org/labs/10027854 (6)
https://medium.com/dair-ai/an-illustrated-guide-to-graph-neural-networks-d5564a551783 (7)
https://arxiv.org/pdf/1812.08434.pdf (8)