Classification of Gene Families Using Machine Learning

Moein Hasani*
moein.hasani@usask.ca
University of Saskatchewan
Saskatoon, Saskatchewan, Canada

Michael Horsch†
horsch@cs.usask.ca
University of Saskatchewan
Saskatoon, Saskatchewan, Canada

ABSTRACT

This project has investigated the effectiveness of several Machine Learning models in the classification of DNA sequences. The purpose is to classify the sequences in the dataset into seven gene classes. The models used in this project are Random Forests, Support Vector Machine, and Logistic Regression. Data has been processed using the K-mer counting method with K values of 3, 5, and 7. The final results show the maximum F1 score of 0.963 can be achieved on this dataset with the Logistic Regression model. Furthermore, the experiments suggest that the Random Forest model can be used with various K values while the other two models work well only with higher K values. My implementation of the work is available at github.com/Moeinh77/Gene-Classification-Python.

KEYWORDS

SVM, Random Forests, Naive Bayes, Logistic Regression, DNA sequence, K-mer counting

ACM Reference Format:
Moein Hasani and Michael Horsch. . Classification of Gene Families Using Machine Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

1.1 Topic

DNA sequence classification has been a topic of interest in Bioinformatics research for many years [2]. Recognizing different sequences and classifying them is a step closer to the treatment of many genetic disorders. Now that we have the more computational power and modern Machine Learning (ML) algorithms, this task is easier and more accurate than in the past.

During recent years Deep Learning (DL) has gained much popularity for data science projects and Artificial Neural Networks (ANNs) how performed very well on different datasets such as sequence datasets like the one used in this project. However, there's a downside to the DL approach, the models usually take longer times to train in comparison with classical ML models, and training DL models often require a GPU. Consequently, this project investigates the usefulness of simpler statistical models for the classification of sequences. The results of the classification indicate that less computationally costly models such as Random Forests (RF) can perform quite well on the sequence classification task. There are three models used in this work, I have compared the results of each model with other models, and I have used the F1 score to validate my results further.

1.2 Previous Work

There are several papers that have researched the effectiveness of the ML models on DNA sequence classification, however as mentioned before they mostly utilize DL models. In [6] and [5] Convolutional Neural Networks (CNNs) are used to classify sequences. 1-D Convolutions are used to extract the information by moving the convolution window from each of the sequences. Furthermore, Recurrent Neural Networks (RNNs) are has been adopted as method of classifying the DNA or Protein sequences in works such as [3] and [1]. RNNs seem appropriate for the sequence data since they are made for processing sequential data types.

There are however certain works that have made use of ML classical models. In [8] authors have decided to use models such as Multi-layer Perceptron, SVM (Support Vector Machine), RF, Logistic regression (LogReg), AdaBoost, and XGboost. The authors have shown in their results that all the selected models have performed relatively well with an accuracy of over 90% on all their selected datasets. specifically, the SVM model has managed to result in the highest accuracy in almost all the datasets. Also, SVM and tree models have been shown to be effective for this task by other works such as [9] as well.

Moreover, [8] has used to different methods for preprocessing the sequence data. As mentioned by the authors, the K-mer counting method for preprocessing would be a better choice for handling a dataset with thousands of examples. Other works like [6] and [7] utilize this method of preprocessing as well.

2 MODELS

2.1 Random Forests

Decision Trees (DTs) are models that try to classify the data by asking questions. Each node in a tree asks a question based on a feature, each branch of the tree is the answer to that question and each leaf node is the part of the data divided based on that answer. At each node, all the attributes (one at a time) are used to split the data, the feature that makes the remaining information in the rest of the data the littlest is chosen for that node. One criterion for deciding if the feature that is selected for division of the data in a node is appropriate is Gini Impurity which is calculated as follows: $G(p1, ..., pn) = \Sigma_i^n pi(1 - pi)$.

The idea of the RF is that we train multiple DTs instead of one. Each DT can be prone to over-fitting especially when the tree becomes deeper. However, in RF we use trees with relatively lower depth, and this aids with solving the over-fitting problem. RF can be used with the classification task or the Regression task. When

*
†

used in classification, the majority vote of the trees will be the final answer and when used in Regression, the mean of tree outputs will be the answer. For training an RF model, there is a method called bagging, or bootstrap aggregation. Each DT receives a sample set as big as the whole data, although with replacement of samples from the dataset. This means that instead of using all N samples, we put aside some and instead we duplicate some others. Additionally, each DT only has access to a random subset of features, and this causes the DTs to have variation. Typically in classification, when we have f features, each DT will randomly use $\sqrt{f}$ of the features. So, the final result is each tree will create predictions based on a certain set of samples and features, and this results in the creation of an uncorrelated set of trees. The variety of structures in these trees will produce decisions that will keep each individual tree from making mistakes due to over-fitting.

2.2 Support Vector Machine

SVM is a type of supervised learning algorithm that can be used for both Classification and Regression tasks. In classification, SVM uses margins to find the decision boundary. If we start from a position that is between the two classes of data, the margins are pushed in both directions until each reaches a data point, the boundary will be the line between the margins that are at an equal distance from both margins. If we imagine we have two classes, 1 and -1, we will have three parallel lines which are indicated by $w^T x = 1$ the margin on class 1 side, $w^T x = 0$ the boundary, and $w^T x = -1$ the margin on class -1 side. The distance between the first and the third line indicates the margin. SVM tries to find the w that maximizes the margin. There are two types of margins that we can set the model to use, one is hard margin and the other is the soft margin. In the hard margin, we try to find the margins in a way that no data point is misclassified and we try to maximize the margin by enforcing $max(0, 1 - y_i(w^T x_i)) = 0$. y_i indicates the label (1 or -1) and x_i indicates the data point. On the other hand, in soft-margin situation we allow misclassification but we try to minimize the number of misclassified samples by minimizing the $\Sigma_i max(0, 1 - y_i(w^T x_i)) + \lambda||w||^2$. In the previous equation, λ is a hyperparameter which can be a regularization strength factor. In multi-dimensional spaces, instead of using a line, SVM creates a hyper-plane that separates data points from each other with a similar method that was explained before. When our data is not linearly separable, SVM will use a method called the kernel trick. Kernels are used to transform the data. We can use the kernels to take the data to other spaces, and in those new space the data might be easier to separate. The kernel functions return the similarity between data points by calculating the inner product between two points in a suitable feature space. One of the kernels which are mostly used for SVM is the Radial Basis Function (RBF) kernel. This kernel calculates the similarity of two data points with the help of the Euclidean distance. The formula for RBF is $K(x, y) = exp(-D_{xy}/\sigma^2)$ and D_{xy} shows the Euclidean distance between two data points x and y, and σ is a hyperparameter. Calculating the distance like this makes the RBF somehow similar to a weighted K Nearest Neighbor method in which the near points have more effect on the classification of a new data point.

2.3 Logistic Regression

The LogReg model is often used in scenarios when we want to estimate the probability of an event in a binary situation such as a win or lose with labels of "0" or "1". LogReg models a binary dependent variable with the help of a logistic function. This function is also called Sigmoid Function which is defined like this:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

The distance of data point x from the line described by f, converted to a probability, is calculated using $y(x, w) = f(w^T x)$. We can write the error term for the LogReg in the following form: $error(w, X) = \Sigma_i^n (y_i - y(x, w))^2$. Since the Sigmoid function has a gradient, we can use Gradient Descent for optimizing the LogReg.

This model can be applied to situations with more than two classes as well. In those situations, the One-vs-All scheme can be used or we can use cross-entropy loss for optimizing it.

3 DATA

3.1 Dataset

Each DNA sequence consists of letters "A", "C", "T", and "G". , these letters are called nucleotides. The data has 4380 DNA sequences related to humans, 1682 for chimpanzees, and 820 samples for dogs. There are seven gene classes, displayed with their labeling in Table 1, that are predictable based on the provided data. The dataset is imbalanced and some classes have more data associated with them and some have fewer samples. You can observe the data distribution between these seven classes in Figure 1. As we can see, the data is imbalanced, thus I have used the F1 score to evaluate the results from each model. I have divided the data into training and test splits with 70% of the data dedicated to training and 30% to test. The data is available at kaggle.com/nageshsingh/dna-sequence-dataset.

3.2 Data Processing

In bioinformatics, K-mers are sub-strings of a sequence with a length of k. Using K-mers we convert each of the K nucleotides into a word by moving a sliding window of size K over the sequence. For example, with K = 3 the sequence 'CCAGCTG' turns into ['cca', 'cag', 'agc', 'gct', 'ctg']. K-mer method has been used by other works such as [4] and [6] has been proved to be effective. We make a dictionary of these K-mer words, and we assign a number to each word in the dictionary. Then we for each word we add a column to our dataset. Every time that word is repeated in the sequence, we add one to its count. So instead of feeding each sequence to the model, we feed the models a matrix whose columns are each K-mer count and its rows are related sample in the dataset.

4 RESULTS AND SETTINGS

4.1 Metric

The results are evaluated using the F1 score which is defined as:

$$F1 = \frac{2}{recall^{-1} + precision^{-1}} \tag{2}$$

Since our data is imbalanced we must take the Precision and the Recall of the results into account so that accuracy does not mislead us with just predicting the class with the highest number

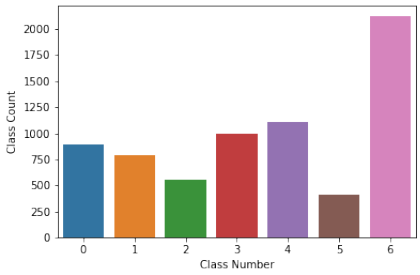

Figure 1: Distribution of the classes in the dataset

Table 1: Gene Families in the dataset and their labeling

Gene family	Label Number
G protein coupled receptors	0
Tyrosine kinase	1
Tyrosine phosphatase	2
Synthetase	3
Synthase	4
Ion channel	5
Transcription factor	6

Table 2: The results of different classifiers in respect different values of K

	K values					
	3		5		7	
	Test F1-score and the Training Time					
Trees	**Random Forest**					
50	0.854	0.65 sec	0.889	1.4 sec	0.915	6.3 sec
100	0.856	1.3 sec	0.902	2.9 sec	0.931	13.2 sec
200	**0.859**	2.67 sec	0.902	5.8 sec	0.932	26.1 sec
C	**Support Vector Machine**					
1	0.60	1.7 sec	0.824	14.1 sec	0.899	170 sec
10	0.732	1.3 sec	0.912	10.1 sec	0.945	150 sec
50	0.773	1.3 sec	**0.919**	9.7 sec	0.932	141 sec
C	**Logistic Regression**					
0.1	0.585	1.3 sec	0.905	5.3 sec	**0.963**	15.9 sec
1	0.589	1.3 sec	0.893	5.7 sec	0.959	16.2 sec
10	0.585	1.3 sec	0.884	5.7 sec	0.954	18.6 sec

Table 3: How K size increases the vocabulary size (unique combination of nucleotides)

K	Vocabulary Size
3	104
5	1303
7	16951

of samples since the accuracy will be high because that class has a lot of samples.

4.2 Settings

The models are trained on an Intel Core-i7 9750H, Python programming language has been the language of choice and sklearn library has been adopted for defining them.

The sklearn library offers many choices for configuring an RF model, among these parameters, there are $n_estimators$, which indicate how many trees will be used for voting, and $max_features$ which is states how many of the features should be selected randomly for each DT. I have decided to try three different values of 50, 100, and 200 for the number of estimators for the RF model. And the rest of the hyperparameters are the default that the sklearn library offers.

For the SVM model, the RBF kernel has been selected as it is often selected as the default choice for the Kernel. The Gamma value of SVM has been set to the 'scale' in SVM implementation of sklearn which is equal to $1/(feature\_number * X.var())$. There's the C hyperparameter which is the inverse of regularization strength, I have experimented with three different values of C which are 1, 10, and 50. And the rest of the hyperparameters are the default that the sklearn library offers.

And as for the LogReg classifier, I have tried 0.1, 1, and 10 for the C hyperparameter. The C here is the same as in SVM, the bigger the C the smaller the regularization effect. The rest of the settings are on defaults.

4.3 Results

In this section, I have provided the test results from training the introduced models, each has been trained with three different K values of 3, 5, and 7. The accuracy of each model and the time required to achieve the accuracy has been stated in Table 2.

RF model shows that it can generally be trained quickly in a matter of seconds even with a dataset with thousands of rows and columns. And as expected as the number of estimators in RF increases the results get better. The best spot seems to be the 100 estimators version since it achieves almost the same results as the 200 estimators result in a shorter amount of time. When $K = 3$ we see a score of almost 0.85 and as the K increases this model achieves better scores. With $K = 3$, the Rf model attains better scores compared to the other two models.

SVM on the other hand has demonstrated that even though it can achieve RF-like results, it will take much more time to be trained. As the C hyperparameter increases the training time of this model has decreased and the score seems to have increased. So bigger C values with RBF kernel can produce more accurate and more efficient results for SVM in this task.

As for the LogReg model, the choice of C does not seem to affect the training time here. Smaller values of C show better score in our work. We can observe that when $K = 3$ the LogReg model does not perform as well as other two classifiers but with $K = 5$ and $K = 7$ the results are acceptable.

5 DISCUSSION

The overall performance of the models indicates that we can in fact be effective when it comes to classifying DNA sequences by using different ML algorithms. All three models can reach acceptable results on the task, with a maximum F1 score of 0.932 for RF, 0.945 for SVM, and 0.963 for the LogReg model. Each of the models can outperform the other two in one of the K scenarios. Best results when K = 3 belongs to RF, when K = 5 to SVM, and when K = 7 to LogReg. However, I believe that the results display the best option for this work is the RF model since it has a reasonable accuracy with all three of the K values and the training time unlike SVM is very short even with K = 7. Moreover, the conversion of sequential data to a count of K-mers proved to be a useful method. As the K increased, different combinations of nucleotides were created, as shown in Table 3and the models received more information, thus the classification score increased.

6 CONCLUSION AND FUTURE WORK

In this project, I tried experimenting with different ML models for the task of classifying DNA sequences. The method for reading the data (using the K-mer method) showed to be quite useful and this project indicates we can treat sequence processing as an NLP task and many techniques of ML that is used for text processing can be applied to bioinformatics sequence data as well. Experimenting with different values of K in K-mer counting produced displayed that larger values of K work better for our task. The maximum F1 score of 0.963 in this classification task was attained by the LogReg model with C = 0.1 and K = 7. Furthermore, the RF model showed that it is quite flexible with different K values and it can produce decent results with small or bigger values of K. In the future, methods such as using an ensemble of the methods or stacking can be used to further increase the score. It would be also interesting to see what happens if we apply feature reduction methods such as PCA to the task. Since all the features are k-mer counts, what happens if we ignore some of their occurrences.

REFERENCES

[1] Giosuè Lo Bosco and Mattia Antonino Di Gangi. Deep learning architectures for dna sequence classification. In *International Workshop on Fuzzy Logic and Applications*, pages 162–171. Springer, 2016.

[2] Mikhail S Gelfand. Prediction of function in dna sequence analysis. *Journal of Computational Biology*, 2(1):87–115, 1995.

[3] Hemalatha Gunasekaran, K Ramalakshmi, A Rex Macedo Arokiaraj, S Deepa Kanmani, Chandran Venkatesan, and C Suresh Gnana Dhas. Analysis of dna sequence classification using cnn and hybrid models. *Computational and Mathematical Methods in Medicine*, 2021, 2021.

[4] MASANORI Higashihara, JOVAN DAVID Rebolledo-Mendez, YOICHI Yamada, and KENJI Satou. Application of a feature selection method to nucleosome data: accuracy improvement and comparison with other methods. *WSEAS Transactions on Biology and Biomedicine*, 5(5):95–104, 2008.

[5] Feriel Ben Nasr and Afef Elloumi Oueslati. Cnn for human exons and introns classification. In *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 249–254. IEEE, 2021.

[6] Ngoc G Nguyen, Vu Anh Tran, Dau Phan, Favorisen R Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, and Kenji Satou. Dna sequence classification by convolutional neural network. *Journal Biomedical Science and Engineering*, 9(5):280–286, 2016.

[7] Aditi Sakalle, Pradeep Tomar, Harshit Bhardwaj, Divya Acharya, and Arpit Bhardwaj. A lstm based deep learning network for recognizing emotions using wireless brainwave driven system. *Expert Systems with Applications*, 173:114516, 2021.

[8] Xiangxie Zhang, Ben Beinke, Berlian Al Kindhi, and Marco Wiering. Comparing machine learning algorithms with or without feature extraction for dna classification. *arXiv preprint arXiv:2011.00485*, 2020.

[9] Qingda Zhou, Qingshan Jiang, and Dan Wei. A new method for classification in dna sequence. In *2011 6th International Conference on Computer Science & Education (ICCSE)*, pages 218–221. IEEE, 2011.