DNA Classification

Final Project

Machine Learning in Medicine

Assoc. Prof. Tran Giang Son

Pham Tuan Nam - 22BI13326

Nguyen Dong San - 22BI13388

Do Bao Phuc - BA12-149

La Thi Thuy Chi - BA12-026

Thai Doan Kien - 22BI13226

Contents

Abstract

DNA sequence classification is a crucial area of research in bioinformatics, enabling scientists to analyze genomes and detect potential diseases. In this report, we employ some advanced machine learning (ML) algorithms such as Random Forest, Multi-Layer Perceptron (MLP), AdaBoost and a pre-trained nucleotide Transformer to classify DNA sequences.

Additionally, we introduce a feature extraction technique, which is called unigram analysis by representing each amino acid sequence. These extracted features are then used in combination with multiple ML models for classification.

To evaluate the effectiveness of these approaches, we conduct experiments on four DNA datasets related to promoter gene sequences. The results demonstrate that all ML models achieve high accuracy across the datasets. Moreover, the unigram feature extraction method consistently yields the best performance, while the newly introduced approach outperforms existing techniques.

1 Introduction

The first successful isolation of DNA by Friedrich Miescher in 1869 was a groundbreaking step in biology as it laid the foundation for understanding the blueprints of all organic life. DNA, short for deoxyribonucleic acid, is the hereditary material found in the cells of all humans and other living organisms. It carries the genetic instructions necessary for biological traits and evolution. A DNA sequence can be represented in the FASTA format, which consists of a character string using only A, C, G, or T. DNA analysis plays a crucial role in diagnosing diseases, studying the spread of infections, solving crimes, and conducting paternity tests. As a result, it has become a key focus in computational biology [3].

In traditional biology, primers are essential tools for DNA analysis. Primers are short single-stranded nucleotide sequences that are crucial for the initiation phase of DNA synthesis. Synthetic primers are widely used in molecular biology for detecting viruses [1], bacteria [2], and parasites [4]. These primers can be found in human DNA sequences infected by specific viruses. The Polymerase Chain Reaction (PCR) method enables the amplification of DNA fragments from viruses, significantly improving detection capabilities [5].

2 Dataset

2.1 Molecular Biology (Promoter Gene Sequences) Dataset

The Molecular Biology (Promoter Gene Sequences) Dataset is designed for evaluating machine learning algorithms in DNA sequence analysis, specifically for identifying promoter regions. Promoters are DNA sequences located upstream of genes that play a crucial role in initiating transcription by providing binding sites for RNA polymerase and transcription factors. Understanding promoter sequences is essential for gene regulation and expression studies. The dataset consists of both positive samples, which are known promoter sequences, and negative samples, which do not exhibit promoter activity. Each sequence is labeled accordingly, making it suitable for supervised learning tasks. This dataset has been widely used to assess hybrid learning models, such as Knowledge-Based Artificial Neural Networks (KBANN), which combine prior biological knowledge with machine learning techniques. Researchers often apply a leave-one-out cross-validation approach to evaluate classification performance.

3 Method

3.1 Feature Extraction of unigram method

Unigram analysis is one of the simplest yet effective techniques for representing DNA sequences in computational biology. In this method, each nucleotide in a DNA sequence—adenine (A), cytosine (C), guanine (G), and thymine (T)—is treated as an individual feature without considering the context of neighboring nucleotides. This approach allows researchers to extract fundamental statistical properties from DNA sequences, such as nucleotide frequency distributions, which can be useful for classification tasks like promoter prediction, gene identification, and disease detection.

Since a DNA sequence is essentially a string composed of the four nucleotides, unigram analysis involves counting the occurrences of each nucleotide in a given sequence. These counts can then be normalized to represent probabilities or used as input features for machine learning algorithms such as decision trees, support vector machines, and neural networks. Although unigram analysis does not capture sequence dependencies, it provides a straightforward and computationally efficient way to analyze DNA sequences, making it a valuable first step in many bioinformatics applications.

3.2 Machine Learning Algorithms

Unigram analysis introduced feature extraction methods was conducted by training and evaluating 5 machine learning models using feature vectors derived from this techniques. Since the classification tasks in the experiments were binary, each processed feature vector was assigned a label of either 1 or 0, indicating whether it belonged to an infected or uninfected DNA sequence.

To ensure reliable performance evaluation, k-fold cross-validation was applied. The dataset was randomly divided into k equal subsets (folds), and the training and testing process was repeated k times. In each iteration, one fold was designated as the test set, while the remaining k-1 folds were used for training. This approach ensured that every sample was used for training k-1 times and for testing once, allowing for a comprehensive assessment of the model's performance. The final model evaluation was based on the average accuracy across all k runs.

3.2.1 Logistic regression

Logistic regression is a linear model primarily employed for binary classification. Similar to the Multi-Layer Perceptron (MLP) model, it utilizes the sigmoid function for the output layer. To mitigate potential overfitting, L2 regularization is incorporated, adding a penalty term to the loss function. This term, also known as L2 regularization, is controlled by the hyperparameter C, which dictates the strength of regularization—smaller values enforce stronger regularization.

$$\text{Loss} \leftarrow \text{Loss} + \frac{1}{C}\sum_{i=1}^{n} w_i^2 \tag{1}$$

Typically, the input features in vector X are assumed to follow a multivariate normal distribution. However, in practical scenarios, this assumption is often violated. In such cases, logistic regression serves as a robust alternative model .

3.2.2 Nearest Neighbor

The K-Nearest Neighbors (KNN) algorithm is a simple and effective machine learning method used for classification and regression tasks. It classifies a new data point based on the majority class among its K nearest neighbors in the feature space.

3.3 Mathematical Formulation

Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the feature vector and y_i represents the corresponding class label, the classification process follows these steps:

1. Compute the distance between the query point x_q and all points in the dataset using a distance metric such as Euclidean distance:

$$d(x_q, x_i) = \sqrt{\sum_{j=1}^{m}(x_{qj} - x_{ij})^2} \tag{2}$$

2. Select the K closest points to x_q based on the computed distances.

3. Assign the class label based on the majority vote among the K neighbors:

$$y_q = \arg\max_c \sum_{i \in N_K} ⊮(y_i = c) \tag{3}$$

where N_K represents the set of K nearest neighbors, and ⊮ is an indicator function.

KNN is non-parametric and relies on distance-based decision making, making it highly interpretable but computationally expensive for large datasets. Proper selection of K and feature scaling significantly impact its performance.

3.3.1 Decision Tree and Random Forest Algorithm

Before delving into the random forest algorithm, it is essential to understand the foundation upon which it is built—the decision tree classifier. A decision tree is a hierarchical model that simulates human decision-making by iteratively partitioning the input space based on feature values. At each internal node, a splitting criterion is applied to divide the dataset into child nodes, and this process continues until a stopping condition is met. The final classification outcome is determined at the leaf nodes. The splitting decisions aim to maximize impurity reduction, where impurity is commonly measured using metrics such as the Gini Index or entropy. The Gini Index is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2 \tag{4}$$

where p_i represents the proportion of class i in dataset D, and C is the total number of classes. Alternatively, entropy quantifies the uncertainty in a dataset and is given by:

$$H(D) = -\sum_{i=1}^{C} p_i \log_2 p_i \tag{5}$$

To determine the best split at a node, the information gain is calculated as:

$$IG = H(D) - \sum_{j=1}^{k} \frac{|D_j|}{|D|} H(D_j) \tag{6}$$

where D_j represents each subset created after the split. A higher information gain suggests a more effective split.

The random forest algorithm extends decision trees by constructing an ensemble of multiple trees, each trained on a different subset of the dataset. This ensemble approach improves generalization and reduces overfitting. The randomness in the model is introduced through the bootstrap aggregating (bagging) method, where training samples are drawn randomly with replacement from the original dataset to construct individual trees. Furthermore, at each split, only a random subset of features is considered, enhancing diversity among trees. The final classification prediction is obtained through majority voting:

$$\hat{y} = \arg\max_{k} \sum_{b=1}^{B} \nVdash(T_b(x) = k) \tag{7}$$

where $T_b(x)$ is the prediction of the b-th decision tree, and $\nVdash(\cdot)$ is an indicator function. For regression tasks, the final prediction is computed as the average of the individual tree outputs:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{8}$$

where B is the total number of trees in the forest.

Two essential hyperparameters must be carefully tuned for optimal performance. The first is the number of estimators B, which controls how many decision trees are included in the forest. A higher number typically improves accuracy but increases computational cost. The second is the maximum depth of each tree, which must be balanced to prevent overfitting (if too deep) or underfitting (if too shallow). By leveraging the power of ensemble learning, random forests effectively handle complex, high-dimensional, and non-linear classification tasks, making them one of the most widely used machine learning models.

3.3.2 Adaptive Boosting (Adaboost)

Adaptive Boosting (Adaboost) is an ensemble learning technique that improves classification accuracy by sequentially combining multiple weak classifiers to form a strong model. Unlike random forests, where decision trees are trained independently on different subsets of the dataset, Adaboost trains each classifier sequentially using the entire dataset. However, it dynamically adjusts sample weights to prioritize misclassified instances, forcing subsequent classifiers to focus more on difficult cases. This iterative reweighting process enhances the model's ability to handle complex patterns.

In Adaboost, each weak classifier $h_t(x)$ is assigned a weight α_t based on its classification performance. Initially, all training samples are assigned equal weights:

$$w_i^{(1)} = \frac{1}{N}, \quad i = 1, 2, \ldots, N \tag{9}$$

where N is the total number of training samples. After training the weak classifier, the classification error is computed as:

$$e_t = \sum_{i=1}^{N} w_i^{(t)} \nVdash (h_t(x_i) \neq y_i) \tag{10}$$

where y_i is the true label, and $\nVdash(\cdot)$ is an indicator function that equals 1 if the prediction is incorrect and 0 otherwise. The weight of each classifier is then calculated using:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right) \tag{11}$$

which ensures that classifiers with lower error rates receive higher importance in the final model. The sample weights are updated as follows:

$$w_i^{(t+1)} = w_i^{(t)} \exp(-\alpha_t y_i h_t(x_i)) \tag{12}$$

followed by normalization to maintain a proper probability distribution:

$$w_i^{(t+1)} = \frac{w_i^{(t+1)}}{\sum_{j=1}^{N} w_j^{(t+1)}} \tag{13}$$

After training multiple weak classifiers, the final prediction is made using a weighted majority vote:

$$H(x) = \operatorname{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) \tag{14}$$

where T is the total number of weak classifiers.

Similar to random forests, Adaboost has key hyperparameters that must be carefully tuned, including the number of estimators (T) and the maximum depth of individual decision trees. A higher number of estimators improves accuracy but increases computational cost, while excessively deep trees may lead to overfitting. By iteratively refining weak learners, Adaboost effectively constructs a robust ensemble model, making it a powerful and widely used technique in machine learning, particularly in scenarios where improving weak classifiers can significantly enhance predictive performance.

3.4 Pre-trained Nucleotide Transformer

The DNA sequence classification model is designed using a Transformer-based architecture, leveraging a pre-trained nucleotide Transformer as the backbone. The model consists of the following key components:

- **Embedding Layer:** DNA sequences are tokenized and converted into numerical representations using a specialized tokenizer.
- **Transformer Encoder:** A series of self-attention layers process the tokenized sequences to extract meaningful representations.
- **Pooling Layer:** The representation of the `[CLS]` token from the last hidden state is used as the sequence embedding.
- **Dropout Layer:** Regularization is applied to prevent overfitting.
- **Fully Connected Layer:** A linear layer maps the extracted features to a predefined number of output classes.

The model takes as input a DNA sequence, tokenized and padded to a fixed length. The output is a probability distribution over the possible classes, computed as:

$$\mathbf{h}_{cls} = \mathbf{h}[:, 0, :] \tag{15}$$

$$\mathbf{y} = \text{sigmoid}(\mathbf{W} \cdot \mathbf{h}_{cls} + \mathbf{b}) \tag{16}$$

where:

- $\mathbf{h}$ is the hidden state output from the Transformer.
- $\mathbf{h}_{cls}$ is the representation of the `[CLS]` token.
- $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters in the classification head.

The model is trained using a supervised learning approach, with Cross-Entropy Loss as the objective function. AdamW is used as the optimizer, along with mixed precision training to improve computational efficiency. The best-performing model is selected based on validation accuracy and saved for inference.

4 Data Analysis

In this section, we describe the data extraction process from the DNA dataset using unigram features. A unigram approach treats each nucleotide (A, T, C, G) as a separate feature, allowing for frequency-based and binary encoding analysis.

4.1 Feature Extraction Using Unigram

For each DNA sequence, we count the occurrences of nucleotides and transform them into a numerical feature representation. Below is an example table showing the nucleotide count:

Sequence	s10	AMPC
A	38	26
T	27	22
C	26	34
G	15	24

Table 1: Feature representation of nucleotides with first 2 type of DNA.

4.2 One-Hot Encoding Representation

Another common approach is one-hot encoding, where each nucleotide is represented as a binary vector. Below is an example:

DNA	0_a	0_c	0_g	0_t
S10	0	0	0	1
AMPC	0	0	0	1
AROH	0	0	1	0
DEOP2	1	0	0	0
LEU1TRNA	0	1	0	0

Table 2: First four columns of the DNA one-hot encoding representation with an index.

These representations allow machine learning models to effectively process DNA sequence data for classification or prediction tasks.

5 Result

All results presented in this section are the mean accuracy and standard deviation over ten folds of cross-validation. The comparison of using 5 machine learning models and Transformer-based architecture made on the Molecular Biology dataset.

Model	**Mean Accuracy**
Random Forest	48.3%
Logistic Regression	93.0%
AdaBoost	85.0%
Decision Tree	78.0%
Nearest Neighbor	78.0%
Transformer-Based	100%

Table 3: Mean accuracy of different models on the dataset.

The results indicate that Logistic Regression (93.0%) and Transformer-Based models (100%) achieved the highest accuracy on the Molecular Biology dataset, suggesting that the dataset is well-suited for models that leverage linear decision boundaries and deep feature extraction.

Traditional machine learning models such as AdaBoost (85.0%) and Decision Tree (78.0%) performed moderately well, demonstrating their ability to capture some of the dataset's patterns. However, Random Forest (48.3%) significantly underperformed, likely due to an inadequate number of trees or improper feature selection. Nearest Neighbor (78.0%) also showed average performance, indicating that similarity-based methods may not be optimal for this dataset.

Overall, the Transformer-Based model outperformed all others, reinforcing the effectiveness of deep learning for complex pattern recognition in molecular biology.

6 Conclusion

This study compared five traditional machine learning models with a Transformer-based approach for DNA sequence classification. The Transformer model significantly outperformed all others, achieving 100% accuracy, highlighting its ability to capture complex sequence dependencies.

Among traditional methods, Logistic Regression achieved the highest accuracy (93.0%), followed by AdaBoost (85.0%), while Decision Tree and Nearest Neighbor performed moderately (78.0%). Random Forest had the lowest accuracy (48.3%), indicating its limitations in this task.

These results demonstrate that deep learning, particularly Transformer models, is well-suited for DNA classification. However, simpler models like Logistic Regression and AdaBoost remain viable for smaller datasets due to their efficiency and interpretability. Future work could explore hybrid approaches to leverage the strengths of both traditional and deep learning methods.

References

[1] Jens Bukh, Robert H. Purcell, and Roger H. Miller. Importance of primer selection for the detection of hepatitis c virus rna with the polymerase chain reaction assay. *Proceedings of the National Academy of Sciences*, 89(1):187–191, 1992.

[2] Samart Dorn-In, Rupert Bassitta, Karin Schwaiger, Johann Bauer, and Christina S. Hölzel. Specific amplification of bacterial dna by optimized so-called universal bacterial primers in samples rich in plant dna. *Journal of Microbiological Methods*, 113:50–56, 2015.

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

[4] M. Andreína Pacheco, Axl S. Cepeda, Rasa Bernotienė, Ingrid A. Lotta, Nubia E. Matta, Gediminas Valkiūnas, and Ananias A. Escalante. Primers targeting mitochondrial genes of avian haemosporidians: Pcr detection and differential dna amplification of parasites belonging to different genera. *International Journal for Parasitology*, 48(8):657–670, 2018.

[5] Teresita M. Porter and G. Brian Golding. Factors that affect large subunit ribosomal dna amplicon sequencing studies of fungal communities: classification method, primer choice, and error. *PLoS One*, 7(4), 2012.