Project 2: Enhancer Classification Problem

deadline: 09/12/2024

Objective

The goal of Project 2 is to train a classifier capable of predicting enhancer sequences based on DNA sequence data using the frequency of k-mers.

Biological Context

Enhancers are non-coding DNA sequences located on chromatin, sometimes as far as 1Mbp from gene promoters. However, when they are in close 3D proximity to their target promoters, they often result in increased gene expression (Fig.1).

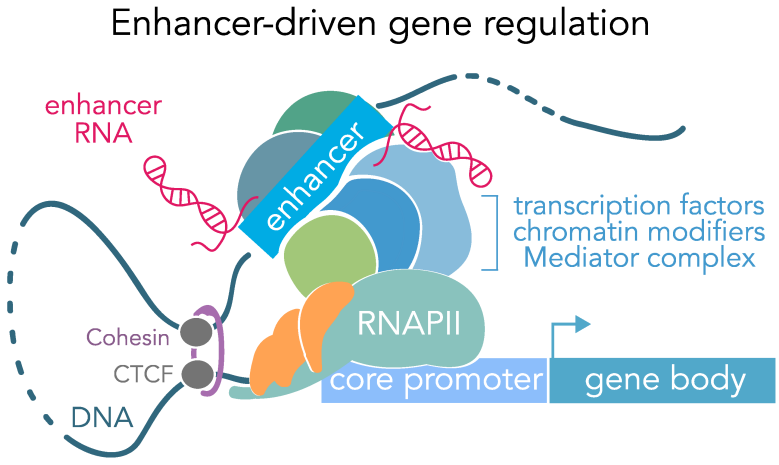

Figure 1: Model of enhancer-driven gene regulation.

For example, most of genetic differences between *Homo sapiens* and primates (like chimpanzees) are located in non-coding regions of the DNA, affecting gene regulation rather than coding regions of genes themselves.

Studying the location and effects of enhancers on transcriptional levels is crucial for better understanding:

- The regulatory mechanisms of gene expression.
- The physiology of diseases linked to mutations that affect gene function.
- Genetic diversity between human individuals or between humans and other species.

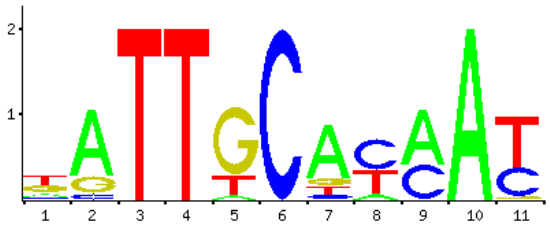

Figure 2: A CEBPB binding motif sequence logo from JASPAR database.

Enhancer sequences, like many other regulatory elements, contain specific sub-sequences called **Transcription Factor Binding Sites (TFBS)** (Fig. 2). These are regions where proteins known as **Transcription Factors (TFs)** bind. These factors play crucial roles, such as recruiting RNA Polymerase II (RNAPolII) to enable transcriptional activation, or mediating enhancer-promoter (E-P) spatial interactions.

Data:

The complete human genome sequence in the GRCh38 (hg38) version should be downloaded in FASTA format from the Gencode project website: `https://www.gencodegenes.org/`. This data will be used to prepare negative samples for classifier training. For more information on the FASTA format, which is used to store nucleotide or amino acid sequences, refer to: `https://en.wikipedia.org/wiki/FASTA_format`.

When working with FASTA files, note the following:

- Sequences may include both lowercase letters (`actg`) and uppercase letters (`ATCG`). In this project, such differences are not significant.
- The symbol `N` represents an unidentified nucleotide. These are especially common in telomeres (chromosome ends) and centromeres (regions where chromatids intersect, forming the middle of the chromosome's "X" structure).

- Only one strand of the DNA (the "sense" strand) is recorded in the FASTA format, as the complementary strand can be easily reconstructed.
- Important! nucleotides/amino acids are indexed starting from **1**!

The coordinates of enhancers in the genome should be obtained from the VISTA database: `https://enhancer.lbl.gov/vista/` - please download *experiments.tsv.gz*.

The following columns in the dataset are particularly important:

- **curation_status:** Indicates whether the enhancer's activity has been experimentally validated (`positive` or `negative`).
- **coordinate_hg38:** Contains the genomic coordinates for the `hg38` assembly, formatted as `chr16:86396481-86397120`.
- **seq_hg38:** Provides the DNA sequence for the specified region in the `hg38` genome.

Preprocessing

Feature Extraction: k-mer frequencies

The classifier will be trained on the frequency of k-mers in DNA sequences, where $k \in [3, 10]$. TFBSs typically range from 6 to 20 bp in length, and the chosen range for k represents a balance between specificity and computational efficiency.

Write a function `count_kmers(str seq)` that:

- Creates a dictionary of all possible k-mers.
- Counts the occurrence of each k-mer in the input sequence by sliding a window of width k with a step of 1.
- Calculates the frequency of each k-mer by dividing its count by the length of the sequence (note: do not divide by the number of k-mers in the sequence to ensure normalization across sequences of varying lengths).

Since it is unknown which DNA strand Transcription Factors bind to, reverse complement k-mers should also be counted. You can use the `Bio.Seq.reverse_complement()` function from the Biopython library for this purpose (Fig 3).

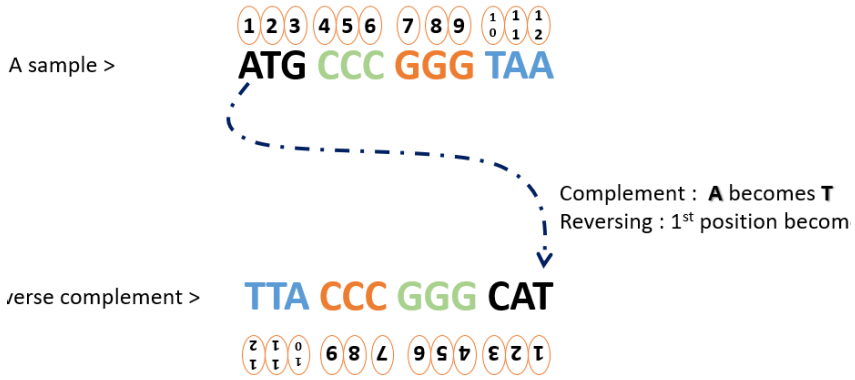

Figure 3: How to make reverse complement of a DNA sequence.

K-mers and their reverse complements are treated as a single feature (e.g., `ATTC` and `GAAT` are considered the same 4-mer). For example:

- The dictionary of all 4-mers will contain 136 features. This number represents half of the total 256 possible combinations (4^4), plus the number of palindromic k-mers (which are identical to their reverse complements).
- The classifier's input will consist of 136 features and a label (`enhancer` or `non-enhancer`).

Positive and Negative Data

- **Positive data:** Records from `experiments.tsv.gz` where `curation_status == 'positive'`.
- **Negative data:** Provided in two variants:
  1. Records where `curation_status == 'negative'`.
  2. Random sequences from the entire genome (GRCh38 FASTA file), ensuring:
     - No overlap with positive sequences.
     - An equal number of negative sequences to positive sequences.
     - Sequence lengths matching the lengths of positive sequences.
     - No `N` symbols in the sequences.

You can use tools like `bedtools` (or its Python library `pybedtools`) for efficient sequence manipulation.

Model Training and Validation

- Use at least one classification algorithm (e.g., Random Forest, SVM).
- Allocate the last 400 positive and the last 400 negative rows from the `experiments.tsv.gz` file to the test set. Additionally, allocate 400 random sequences to the test set if using a random negative dataset. **Do not use these test sequences during model training.**
- Perform **10-fold cross-validation**.
- Train the classifier for at least three different values of k (e.g., 3, 4, 5).

Results and Report

You are required to prepare a comprehensive report detailing the following aspects of the task:

Objective

Clearly (and shortly) state the goal of the task, including the motivation behind it and its biological context.

Methods

Provide a detailed description of the methods used, including:

- **Preprocessing steps:** Explain how the data was prepared.
- **Classifier(s):** Specify the classifier(s) chosen and provide a justification for your choice. If multiple classifiers or parameter configurations were tested, describe them and their differences.
- **Comparison:** Compare different classifiers or configurations, detailing their performance under various conditions.

Performance Metrics

Include the following performance metrics for the classifiers:

- **AUC-ROC**
- **Accuracy**
- **Precision**
- **Recall**
- **F1-score**
- **Confusion Matrix**
- Any additional metrics that provide further insights into the model's performance.

Comparison of Negative Datasets

Provide a detailed comparison of the results obtained using the two types of negative datasets:

- Records where `curation_status == 'negative'`.
- Random sequences extracted from the genome. Ensure that these sequences meet the requirements outlined in the preprocessing section (e.g., no overlap with positive sequences, appropriate lengths, no `N` symbols).

Submission Requirements

Your submission must include:

- A well-structured and comprehensive report uploaded to the **Leon platform** (8 points).
- The complete code used for the task, including scripts for preprocessing, training, and evaluation (8 points).
- The random negative dataset generated as part of the assignment (4 points).