Assignment 2 - Kaggle Part

Antibiotic Resistance Genes (ARGs) Prediction

AIST 4010: Foundation of Applied Deep Learning (Spring 2024)

DUE: **11:59PM (HKT), Mar. 18, 2024**

1 Introduction

The spread of antibiotic resistance has become one of the most urgent threats to global health, which is estimated to cause 700,000 deaths each year globally. Its surrogates, antibiotic resistance genes (ARGs), are highly transmittable between food, water, animal, and human to mitigate the efficacy of antibiotics. Accurately identifying ARGs is thus an indispensable step to understanding the ecology, and transmission of ARGs between environmental and human-associated reservoirs.

In this assignment, you need to design an end-to-end system for ARG prediction. In this task, your system will be given raw sequence protein sequences (translated from genes) as input. First, your system needs to classify the input into ARG or non-ARG. Then if the input is a non-ARG, your system needs to output the class **'nonarg'**. If the input is an ARG, your system needs to go further to predict resistant antibiotic type. Here we have 14 antibiotic families including **'aminoglycoside', 'macrolide-lincosamide-streptogramin', 'polymyxin', 'fosfomycin', 'trimethoprim', 'bacitracin', 'quinolone', 'multidrug', 'chloramphenicol', 'tetracycline', 'rifampin', 'beta_lactam', 'sulfonamide', 'glycopeptide'**. The system should decide which family the predicted ARG is resistant to.

You will train your model on a dataset containing thousands of sequences labeled by non-ARG or ARG with the corresponding families. You will learn more about sequence processing, sequence encoding, and, of course, convolutional or other neural network layers. This assignment is quite different from assignment 1. You will develop skills in processing sequences with deep learning models.

Notice that the choice of models will not be limited anymore. Any architectures are fine, including CNN, LSTM, and Transformer. And **pre-trained** models are recommended to achieve high scores

- Goal: Given a raw protein sequence, predict the non-ARG or ARG family class.
- Kaggle: `https://www.kaggle.com/c/aist4010-spring2024-a2`

2 ARG Prediction

2.1 Protein Sequence Representation

We all know images are represented by many digit values for every pixel. Then how will we represent protein sequences? Protein sequence is typically notated as a string of letters, listing the amino acids starting at the amino-terminal(N) end through to the carboxyl-terminal(C) end. Either a three letter code or single letter

code can be used to represent the 20 naturally occurring amino acids. Table 1 shows the 20 natural amino acid notation. For example, a protein sequence can be represented as 'MKLIEIEKLNKYFNTAIIGAS'.

Table 1: 20 natural amino acid notation

Amino Acid	3-Letter	1-Letter
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

2.2 Protein Sequence Encoding

Image pixel values are natural digits that can be passed to our algorithms to process. However, for the letters of protein sequences, we couldn't directly deal with them. So the first problem you need to solve is how to encode the protein sequences. Actually, there are several ways to achieve it.

Here, **one-hot encoding** is the simplest but very effective solution. By One-hot encoding, the protein sequence can be encoded into a $20 \times L$ matrix. Each row corresponds to the presence of 20 standard amino acid (AAs) while each column is a spot on a protein sequence.

For one-hot encoding, you may need to think about two questions:

- What if we meet a rare amino acid besides the 20 standard AAs?
- How to deal with sequences of different lengths?

Besides, other types of embeddings are also welcome including some pre-trained sequences embeddings.

2.3 Multi-class Classification

This problem may seem fancy, but the ARG prediction is still a multi-class classification: the input to your system is a protein sequence and your model needs to predict the non-ARG or ARG class.

However, noted that the number of non-ARG sequences is a lot larger than that of any single ARG class. Therefore, this dataset is actually an **imbalanced** dataset. You need to think about how to deal with this problem.

The task could be divided to three parts:

- Loading sequence data and labels from raw files.
- Transforming the data to appropriate representations.
- Training a deep learning model for classification.

3 Dataset

The data for the assignment can be downloaded from the Kaggle competition link[1]. The dataset contains thousands of protein sequences with labels.

3.1 File Structure

The structure of the dataset is as follows:

- `train.fasta`: You are supposed to use train data set to train your model for the task.
- `val.fasta`: You are supposed to use val data to validate the classification F-score.
- `test.fasta`: You are supposed to assign classes for images in test data and submit your result.
- `sample-submission.csv`: This is a sample submission file for this competition.

3.2 Loading Sequence Data - Bio.SeqIO

All the protein sequences are stored as the 'FASTA' format. In bioinformatics, the 'FASTA' format is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes. The format also allows for sequence names and comments to precede the sequences.

For one sequence, usually it includes two parts: the description and the sequences as shown in figure 1. The first line beginning with **'>'** symbol is the description while the second line is the protein sequence information. You can easily deal with this type of file by **'Bio.SeqIO'** library[2].

3.3 Loading Data Labels

For an ARG sequence as shown in figure 2, the second part of the description is **'FEATURES'**, and the forth part is the ARG class. In this example, the ARG class is **'quinolone'**.

For a non-ARG sequence as shown in figure 3, the description starts with **'sp'** and doesn't have **'FEATURES'** or ARG classes.

[1] `https://www.kaggle.com/c/aist4010-spring2024-a2/data`
[2] `https://biopython.org/wiki/SeqIO`

Figure 1: A 'FASTA' files example

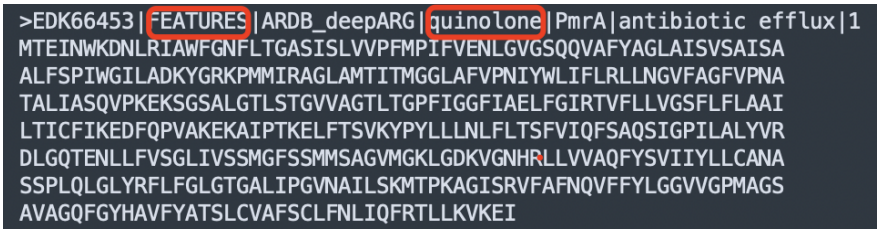

Figure 2: An ARG sequence example

Figure 3: A non-ARG sequence example

4 System Evaluation

4.1 Label Mapping

For evaluation, please mapping your predicted ARG classes as this dict.
arg_dict =
{'aminoglycoside': 0, 'macrolide-lincosamide-streptogramin': 1, 'polymyxin': 2, 'fosfomycin': 3, 'trimethoprim': 4, 'bacitracin': 5, 'quinolone': 6, 'multidrug': 7, 'chloramphenicol': 8, 'tetracycline': 9, 'rifampin': 10, 'beta_lactam': 11, 'sulfonamide': 12, 'glycopeptide': 13}
If the predicted result is non-ARG, please set the label to be **{'nonarg': 14}**.

You may check the `sample-submission.csv` for your reference before submission.

4.2 Evaluation Metric

The evaluation metric is macro F-score. To calculate macro F-score, first we need to calculate macro average of Precision and Recall. Then we apply the F-score equation to calculate it.

$$\begin{aligned} \text{PrecisionMacroAvg} &= \frac{(Prec_1 + Prec_2 + \ldots + Prec_n)}{n} \\ \text{RecallMacroAvg} &= \frac{(Rec_1 + Rec_2 + \ldots + Rec_n)}{n} \\ \text{F1} &= 2 \cdot \frac{\text{PrecisionMacroAvg} \cdot \text{RecallMacroAvg}}{\text{PrecisionMacroAvg} + \text{RecallMacroAvg}} \end{aligned} \tag{1}$$

5 Methods

If you have no idea how to start your assignment, this section can provide some suggestions.

1. Still, the first step is to make sure you **load the datasets** correctly. I believe you're familiar with how to load images. But this time you need to deal with a different type of data: sequences. Please read the instructions in Section 'Dataset' carefully.
2. You may refer to your pipeline in A1 to fastly start training a simple model like **MLP** or **CNN**.
3. Since we usually don't perform data augmentation on biological sequences, you can use hyperparameters tuning and advanced model architectures to improve your performance.
4. These steps are enough to beat the two baselines. But if you want to achieve a higher score, you should apply some **pre-trained protein language models** like ESM[3] and ProtTrans[4]. These models are trained on large protein datasets and could be used for downstream tasks including contact prediction, structure prediction, interaction prediction, etc. They can help you achieve amazing results. In Assignment 2, the pre-trained models are **ALLOWED**.

6 Submission

The following are the deliverables for this assignment:

- **Kaggle submission.** Please submit your predicted results on the Kaggle page with your nickname. Keep it the same with A1. Make sure your nickname appears on the public leaderboard. If your score is higher than the **MLP baseline**, you can obtain at least 60%. If your score is higher than the **CNN baseline**, you can obtain at least 80%. The final score will depend on your ranking. At least the first three students can obtain a full score.
- **Blackboard submission.** The Deadline for the Blackboard submission is **one day later** than the Kaggle submission, so you don't need to rush. Please pack all files in one '.zip' or '.tar.gz' file named 'nickname_SID'. For example, if my nickname is 'lctest' and my SID is '1155123456', I should name my submission file as 'lctest_1155123456.zip'. And it should have these contents:
  - A report describing your model architecture, loss function, hyperparameters, and any other interesting detail led to your best result for the above competition. And cite all references in this report. Please limit the report to **two pages** (including references) and submit it in **'.pdf' format.**

[3] `https://github.com/facebookresearch/esm`
[4] `https://github.com/agemagician/ProtTrans`

– A sub-folder containing **all** your source codes (including data-processing, training, prediction, etc.) in '.ipynb' or '.py' format.

7 Conclusion

Nicely done! Here is the end of Assignment 2 (Kaggle Part), and the beginning of the sequence processing world. As always, feel free to ask us on Piazza if you have any questions. We are always here to help.

I'd like to emphasize some notices here:

- Referring to open-source codes to implement the models is **ALLOWED**.
- Directly calling model APIs is **ALLOWED**.
- Loading pre-trained models is **ALLOWED**.
- Remember to cite the above items if you use them.
- The Kaggle submission deadline is **11:59 PM (HKT), Mar. 18, 2024.** The competition will close exactly at that time. You **CANNOT** use late days for the Kaggle part assignment. Grades will be deducted by 25% for each late day.
- The Blackboard submission deadline is one day later: **11:59 PM (HKT), Mar. 19, 2024.**. You must submit the files required on Blackboard, or you will face a 10% mark deduction.

Good luck and enjoy the challenge!

8 Reference

1. `https://en.wikipedia.org/wiki/Protein_primary_structure`
2. `https://en.wikipedia.org/wiki/FASTA_format`
3. `https://tomaxent.com/2018/04/27/Micro-and-Macro-average-of-Precision-Recall-and-F-Score/`
4. `https://biopython.org/wiki/SeqIO`
5. `https://github.com/facebookresearch/esm`
6. `https://github.com/agemagician/ProtTrans`
7. Junkang Wei, Siyuan Chen, Licheng Zong, Xin Gao, Yu Li, Protein–RNA interaction prediction with deep learning: structure matters, Briefings in Bioinformatics, Volume 23, Issue 1, January 2022