BIEN 410 – Final Project Description (20 pts, DE.3, DE.4)

Course Instructor: Brandon Xia (brandon.xia@mcgill.ca)
Project TA: Leah Pollet (leah.pollet@mail.mcgill.ca)
Code and report due date: November 30th

Goal

The main goal of the project is to give you a chance to play around with some of the algorithms discussed in class, in the context of a creative, large-scale problem. You will be working on implementing an original algorithm to predict protein secondary structure from protein sequence input.

Rules

Your code needs to read in a protein sequence file in FASTA format as input and return an output file containing your best prediction as to which amino acids in the input file belong to alpha-helix secondary structures.

To evaluate your work, we will hold a competition between all the scripts submitted by students in the class: each student's code will be run on the same input sequence file (new data, different from the training data), and the resulting output predictions files will be compared with the correct secondary structure labeling of the input sequences to determine the prediction accuracy of each student's code. Your final grade for the code portion of this project will depend on the performance of your code in this class competition. We cannot afford to change your submissions to ensure that they run properly. So, pay close attention to the specifications in this project description, as any deviations from these guidelines will have you disqualified from the competition, resulting in no marks for the competition portion of this project.

Getting started

We have provided you with a *BIEN410_FinalProject* package containing the data that you need to start writing and testing your code. Documentation on this package can be found in the **Package description** section. You will need to read that section carefully.

To get started with this project, download the *BIEN410_FinalProject* package, unzip it, and modify the *Student_Info.txt* file located in the *src* folder so that the studentName, studentID and programmingLanguage lines reflect your name, McGill student ID and whether you are submitting MATLAB or Python 3 code.

Take a look at the *infile.txt* file located in the *input_file* folder. *infile.txt* is an example of a file that can be used as input to your code. It contains several sequences in FASTA format. Note that the sequences are amino acid sequences, and that each sequence is preceded by a description line in the format ">name | length" with "name" the name of the sequence, and "length" the amino acid length of the sequence. Your code will only be tested on sequences in this format.

Take a look at the *labels.txt* file located in the *training_data* folder. *labels.txt* contains the correct secondary structure labeling for the sequences in *infile.txt*. The *labels.txt* file can be read three lines at time, with lines (1) and (2) in each triplet, identical to a description line and a sequence line in *infile.txt* respectively. Line (3) in each triplet is the secondary structure label for each amino acid in the sequence, with the "H" characters representing alpha-helices and the "-" characters representing non-alpha helix secondary

structures. For instance, if the n^{th} amino acid in a sequence belongs to an alpha-helix, then the n^{th} character of the secondary structure label line is "H". Note that each secondary structure line has the same length as the corresponding amino acid sequence. Moreover, *labels.txt* is in the correct format to be an output from your code: If your code returns a file identical to *labels.txt* when giving it *infile.txt* as input, its accuracy should be 100%.

Try and run the example classifier implemented in the *src_MATLAB* or *src_Python* folder to generate a random output file. To do this you will first have to run *SStrain.m* or *SStrain.py*. The *SStrain* scripts read in the training data in the *training_data* folder (*labels.txt*), compute the fraction of amino-acids labeled as alpha-helix in the training data, and write this fraction to *parameters.txt*. Now that the training is complete, you can make a random prediction, weighed by the fraction of amino-acids in the training data that belong to alpha-helices. To do this you have to run *SSpred.m* or *SSpred.py*. The *SSpred* scripts use the parameters in the *parameters.txt* file to make a secondary structure prediction for the *infile.txt* sequence file located in the *input_files* folder and write the prediction to the *outfile.txt* file located in the *output_files* folder.

Try and evaluate the prediction accuracy of the example classifier. To do this you can run the *testing.py* script located in the *testing* folder. This script takes a prediction file (e.g., *outfile.txt*) and a correct labeling file (e.g., *labels.txt*) as input and prints the accuracy of the prediction. See the **Testing** section for more details.

Get started with your own secondary structure prediction algorithm by editing the *SSpred.m* or *SSpred.py* scripts in located in the *src* folder.

Report

You will write a report with a detailed explanation of your approach and reasoning. The suggested length is between 4 and 5 pages (~400 words per page), but the most important constraint is that the report be clear and concise. The report must be submitted as a pdf document. The report must include the following required components:

1. An explanation of how your code works, and a motivation for your approach.
2. A brief description of the theoretical basis of the approach. If you used algorithms from other sources than class material, briefly describe the algorithm and be sure to cite your sources. You do not need to cite class material.
3. A discussion of the advantages and disadvantages of your approach.
4. If you tried other approaches during the project, describe them briefly and discuss how they compared to your final approach.
5. A brief description of how you would go about improving your code.

Marking Scheme

50% of the project mark will be allotted for performance in the class competition, and the other 50% will be based on your report. The student's code which achieves the best accuracy (smallest total number of misclassification errors) will receive full marks for the tournament. The remaining codes will receive marks according to a linear interpolation based on the accuracy achieved.

Package description

We have provided you with a *BIEN410_FinalProject* package containing the data that you need to start writing and testing your code. The organization of the package is as follows:

```
BIEN410_FinalProject
|
| --- input_file              Folder to store input file to your prediction script.
|     |
|     \ --- infile.txt        Example input file to your prediction script. Note that a different input file, with identical
|                             format, will be used when testing the accuracy of your work.
|
| --- output_file             Folder where your prediction script should write an output file. Testing the accuracy of
|     |                       your work will use the file written to this folder.
|     |
|     \ --- outfile.txt       Placeholder for the file written by your prediction script.
|
| --- src                     Folder to store scripts. Scripts you write should correctly run from this folder and follow
|     |                       the specifications detailed in the Specifications section.
|     |
|     | --- Student_Info.txt  Text file to edit with your information.
|     | --- SSpred.m          Script to edit if you are using MATLAB for this project.
|     | --- SSpred.py         Script to edit if you are using Python 3 for this project.
|     | --- parameters.txt    Text file to store data required by your prediction script (SSpred.py or SSpred.m).
|     |
|     . Additional scripts can be placed here, but only SSpred.py or SSpred.m will be run when testing the accuracy of
|     \ your work. Ensure that results from additional scripts are stored in parameters.txt if needed for prediction.
|
| --- src_MATLAB              Example of a working src folder using MATLAB.
|
|     | --- SStrain.m         Script that writes the fraction of alpha-helix labels in the training data to parameters.txt.
|     |                       Note: this is an example of an additional script that would not be run when testing.
|     |
|     | --- parameters.txt    Output from SStrain.m. Used as input to SSpred.m.
|     |
|     \ --- SSpred.m          Script that reads in input from ../input_file/infile.txt, makes a random prediction of alpha-
|                             helix or non-alpha helix, weighted by the fraction in parameters.txt for each amino acid in
|                             the input and print a prediction file in the correct format to ../output_file/outfile.txt.
|
| --- src_Python              Example of a working src folder using Python 3.
|     |                       SStrain.py, SSpred.py and parameters.txt are equivalent to SStrain.m, SSpred.m and
|     \                       parameters.txt in ../src_MATLAB respectively.
|
| --- training_data           Folder containing labeled data that you can use to guide your predictions.
|     |
|     \ --- labels.txt        The correct alpha-helix labeling of the sequences in ../input_file/infile.txt. Note that if
|                             your prediction script wrote a file identical to labels.txt with input ../input_file/infile.txt,
|                             its accuracy should be 100%.
|
| --- testing                 Folder containing a script you can run to test the accuracy of your work.
|     |
|     \ --- testing.py        Testing script. Details on how to run this script in the Testing section.
\
```

Submission Format

When it is time to submit, you will only submit your *src* folder. This folder should have the following format:

```
src
|
| --- Student_Info.txt              Text file with your information.
| --- SSpred.m OR SSpred.py         Your prediction script.
| --- parameters.txt                Text file to store data required by your prediction script.
|
. You can submit additional files/scripts (example: training scripts), but they will NOT be run when testing the
. accuracy of your work. Ensure that results from additional scripts are stored in parameters.txt prior to submission if
. they are needed by your prediction script. Any submission that does not contain the three files above will be
. considered incomplete. While submitting additional scripts is optional to get full marks on the competition, any
. script discussed in your report should be added to your submission folder for reference and partial marks.
\
```

Finally, compress your src folder, preferably using zip, and submit the compressed folder alongside your report. Note that the three main files should follow the additional guidelines in the **Specification** section.

Specifications

To ensure that we can run your submission, the three main files must meet the following additional constraints:

Student_Info.txt

1. Do not change the name of this file.
2. Your name, McGill ID, and details on the programming language you are using should be noted in this file.

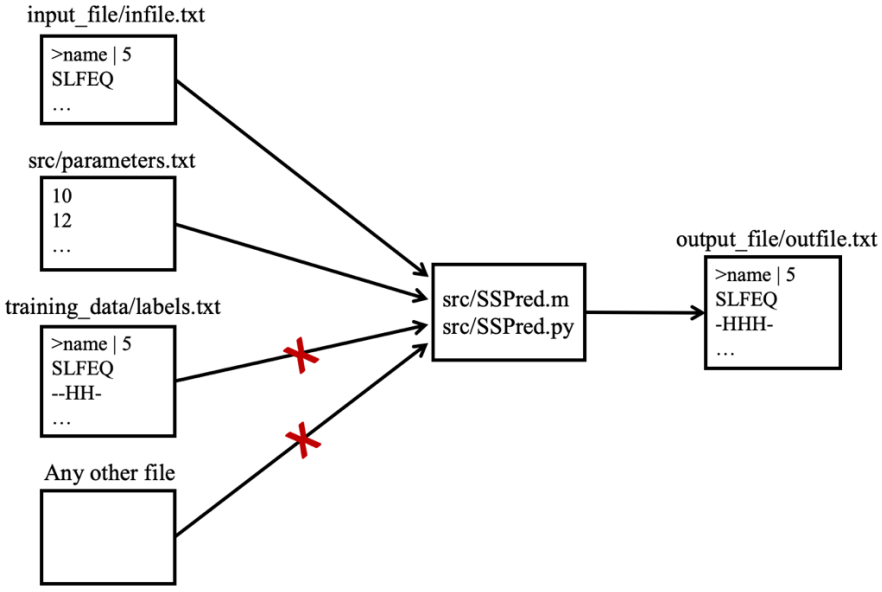
parameters.txt

1. Do not change the name of this file.
2. This file can be in any format you like, as long as it remains a human-readable text file (binary files for instance are not allowed) and does not exceed the size requirement below.
3. This file cannot exceed a size of 500KB.

SSpred.m / *SSpred.py*

1. Do not change the name of this file.
2. Do not change the format of this file: only .m or .py will be accepted as valid formats.
3. You can write your prediction script in either MATLAB, or Python 3.
4. All code implementations must be your own. You cannot use external libraries, packages or downloaded modules. You can use modules that are built into the programming language you choose (e.g., Python standard library). As a room of thumb: we should be able to run your code without requiring additional downloads.
5. We expect well-commented code.

6. You are free to reuse any of the example code in the *BIEN410_FinalProject* package in your submission, as long as you document that you have done so.
7. The inputs to this script are specified so that we can run your submission automatically:
    - When running your prediction script, only two files will be available: *infile.txt* located in the *input_file* folder, and *parameters.txt* located in the *src* folder. Ensure that your script only reads from those files to make a prediction. In particular, this means that your prediction script will not have access to the training data (*training_data* folder). So, ensure that any training is done prior to submission and that parameters obtained from training are stored in *parameters.txt* if they are needed to make a prediction.
    - When testing your code, we will use different sequences in *infile.txt*, but the format of the file will remain identical: *infile.txt* contains several amino acid sequences in FASTA format, with each sequence preceded by a description line in the format ">name | length" with "name" the name of the sequence, and "length" the amino acid length of the sequence. If your script runs on the example *infile.txt* located in the *input_file* folder, it will run on the file that we use for testing.
8. The output to this script is specified so that we can run your submission automatically:
    - Your prediction script should write a file with name *outfile.txt* to the *output_file* folder. The format of *outfile.txt* should match the format of the *labels.txt* file located in the *training_data* folder: Each sequence in *outfile.txt* has three lines. The first two lines are identical to the description and sequence lines in *infile.txt*. The third line contains your secondary structure prediction for each amino acid in the sequence, with the character "H" representing alpha-helix and the character "-" representing non-alpha helix.
    - Note that the description and sequence lines in *outfile.txt* need to be identical to the equivalent lines in *infile.txt* for proper testing of your code. If you can run the *testing.py* script located in the *testing* folder on your prediction file (*outfile.txt*), the file is in the correct format.
9. Below is a summary figure with the inputs and outputs allowed for your script:



10. Use relative paths and not absolute paths in your code. For instance, if reading the *infile.txt* file from your script, you should specify its location as "../input_file/infile.txt" (relative path) and not "/Users/Joe/Desktop/BIEN410_FinalProject/input_file/infile.txt" (absolute path), as the second path only exists on Joe's computer!

Testing

In the testing folder we have provided you with the *testing.py* script, that you can run to test the accuracy of your work. This script takes in two arguments:

The "-p" or "--predictions" argument is the path to a secondary structure prediction file written by your code. Note that this file needs to be in the correct format detailed in this Project Description, or *testing.py* will raise an error. An example prediction file can be obtained by running the example random classifier *SSPred.py* (located in the *src_Python* folder) or *SSPred.m* (located in the *src_MATLAB* folder). Both these scripts will make a random secondary structure prediction for the sequences in *infile.txt* (located in the *input_file* folder) and print out the *outfile.txt* prediction file in the correct format to the *output_file* folder.

The "-l" or "--labels" argument is the path to the file containing the correct alpha-helix labeling for the sequences you used to make a prediction. Note that you were given the *labels.txt* file (located in the *training_data* folder), which contains the correct alpha-helix labeling for the sequences in *infile.txt* (located in the *input_file* folder) in the correct format to be used as an argument for *testing.py.*

The script *testing.py* then simply compares the files specified by the two arguments to compute the accuracy of your prediction script.

For example, to run this script from the terminal:

1. Go to the *testing* folder:
   cd BIEN410_FinalProject/testing
2. Run *testing.py:*
   python3 testing.py -p ../output_file/outfile.txt -l ../training_data/labels.txt
3. If you need help, you can use the --help argument:
   python3 testing.py --help

Academic Integrity

This is an individual project. The exchange of ideas is encouraged but sharing of code and reports is forbidden and will be treated as cheating.