Assignment Goal

- Gain deeper familiarity with MEME for discovering motifs within biological sequences.
- Consider the parameters in the PWM model.

Instructions

- To submit your assignment, log in to the biostat server **mi1.biostat.wisc.edu** or **mi2.biostat.wisc.edu** using your BMI (biostat) username and password.
- Copy all relevant files to the directory **/u/medinfo/handin/bmi776/hw1/<USERNAME>** where **<USERNAME>** is your BMI (biostat) username. Submit all of your Python source code and test that it runs on the biostat server.
- For the rest of the assignment, compile all of your answers in a single file and submit as **solution.pdf**.
- Write the number of late days you used at the top of **solution.pdf**.
- For the written portions of the assignment, show your work for partial credit.

Part 1: MEME Implementation

Write a program, **learn_motif.py**, that takes as input a set of DNA sequences and an integer width **w** and learns an OOPS model for a motif of width **w**.

Implement the EM algorithm from MEME to learn the motif. You should calculate the likelihood $P(X|\theta)$, or the log likelihood $\log P(X|\theta)$, after each iteration and stop the algorithm when the change in this value is less than a fixed threshold. Use the exhaustive subsequence approach described during lecture (slide 32) to choose the starting point for EM. Use pseudocounts of 1 ($d_{c,k} = 1$) when estimating the model parameters.

Your program should be callable from the command line as follows:

```
python learn_motif.py --SEQ=<sequences> --W=W --M=<model>
                      --P=<positions> --SUBSEQ=<subsequences>
```

where

- `<sequences>` is a text file containing DNA sequences one per line. The sequences will all have the same length.
- `w` is the width of the motif to learn.
- `<model>` is the name of the text file into which the program will output the learned motif model (i.e., the probabilities for each nucleotide in each column) in a tab-delimited format with the background frequencies in the first column.
- `<positions>` is the name of the text file into which the program will output the predicted starting location of the motif in each sequence one per line. Use 0 for the first position in a sequence.
- `<subsequences>` is the name of the text file into which the program will output the subsequences corresponding to the motif occurrence in each sequence one per line.

Input files, the template `learn_motif.py` with argument parsing code, and example output can be downloaded from
`https://www.biostat.wisc.edu/bmi776/hw/hw1_files.zip`

To test your program, you may use the file `hw1_example1.txt` to discover the width 7 consensus motif `ATTACGA` and the file `hw1_example2.txt` to discover the width 10 consensus motif `TGGCTTCCCG`. Sample output files for `hw1_example2.txt` are provided in `hw1_files.zip`.

Your program will be evaluated on these example inputs, the sequences in Part 2, and additional datasets that will be kept private.

Part 2: Motif Discovery

We will run your MEME implementation to see how motif finding may be used in a biological study. Suppose the input sequences in the file `hw1_hidden.txt` come from the promoter regions of 100 yeast genes that are differentially expressed in a stress response experiment. We would like to use motif finding to learn more about a candidate regulator (transcription factor) of this expression response.

Use your `learn_motif.py` program to discover the motif of width 8 hidden in the sequences in `hw1_hidden.txt`.

Test that your program runs on the biostat server by running

```
python learn_motif.py --SEQ=hw1_hidden.txt --W=8 --M=model.txt
                --P=positions.txt --SUBSEQ=subsequences.txt
```

from your `handin` directory. Leave the output files in the `handin` directory.

Part 3: Sequence Logo

Construct a sequence logo for the predicted motif sequences from Part 2 by using the WebLogo application (http://weblogo.threeplusone.com/create.cgi). Use the contents of the file `subsequences.txt` as input to the WebLogo application. Use PDF as the output format and select Logo-size as large. Save the logo as `logo.pdf` and submit it in your handin directory.

It is also easy to create sequence logos in Python. Instead of using the web server, you can optionally install the `weblogo` package (https://pypi.python.org/pypi/weblogo) with the command:

```
pip install weblogo
```

After installing, you can generate the logo with the command:

```
weblogo -s large -F pdf < subsequences.txt > logo.pdf
```

This requires having `weblogo` on your PATH or setting an alias as in HW0. See the HW0 instructions for adding `/u/medinfo/bmi776-miniconda3/bin/weblogo` to your PATH or setting a `weblogo` alias. There is also a Python API to use `weblogo` programmatically, but it is not documented.

Part 4: Transcription Factor Search

Search the JASPAR transcription factor binding profile database (http://jaspar.genereg.net/) using the profile matrix that you learned in Part 2. Use the contents of the `model.txt` file for this search. Delete the first (background) column, leaving the nucleotide column, and paste the PWM into the “Align to a custom matrix or IUPAC string” field.

What is the name of the yeast transcription factor that binds to this motif? This is the top-scoring transcription factor with species (taxonomy) code

4932. Are there differences between the logo you generated in Part 3 and the JASPAR logo for this transcription factor?

Search the *Saccharomyces* Genome Database (www.yeastgenome.org/) for this transcription factor, entering its name into the search field. Based on the overview description, what cellular responses is it involved in?

Part 5: PWM Parameters

(A) Suppose you have run a motif finding algorithm and converge to the following solution for a PWM with width 4. The inferred motif matches for your 10 input sequences are shown below, indicated with underlined nucleotides:

CAAACGAA
CCGGAAGG
TGGACCCT
GTCACCAG
ACGTACCG
GGCCCGGT
CAAATGTT
CTTACCAG
CTGATTGA
GGCGTGTA

From these motif starting positions, you estimate the following PWM and background parameters. The background parameters have been estimated using the nucleotides outside of the motif matches. We will not use pseudocounts in this small example because it has been constructed to avoid probabilities of 0.

	$i = 1$	$i = 2$	$i = 3$	$i = 4$
A	0.1	0.4	0.6	0.2
C	0.1	0.1	0.2	0.5
G	0.7	0.1	0.1	0.1
T	0.1	0.4	0.1	0.2

$background$
0.175
0.325
0.325
0.175

Consider extending the PWM to width 5. If you fix the motif starting positions, derive the maximum likelihood estimate parameters (without pseudocounts) for the new PWM column and the updated background parameters.

(B) Compare the difference in log likelihoods of the observed data under the original and extended model. Specifically, compute $\log_{10} P(X|A,\theta') - \log_{10} P(X|A,\theta)$, where X is the sequence data, A is the inferred motif starting positions, θ are the provided parameters of the background and width 4 PWM, and θ' are the parameters of the background and width 5 PWM you estimated in (A). Did extending the PWM width increase, decrease, or not affect the log likelihood?

(C) In (B) you determined how changing the PWM width affected the log likelihood for a single example dataset. Consider whether this is a special case dependent on the example sequences or a general phenomenon. Select **one** of the following options and support your selection with a formal argument or mathematical proof:

- The log likelihood in the optimal PWM model with width W+1 **will always be greater than or equal to** the log likelihood in the optimal model with width W.
- The log likelihood in the optimal PWM model with width W+1 **will always be less than or equal to** the log likelihood in the optimal model with width W.
- The log likelihood in the optimal PWM model with width W+1 **could be greater than, equal to, or less than** the log likelihood in the optimal model with width W depending on the sequence data.