Challenge Problem for Assignment 2
CS 279
Fall 2023

This problem is entirely optional, and you can receive full credit for the assignment without attempting it. We will never deduct points for a challenge problem submission and will award extra credit depending on the quality of your solution.

In this challenge problem, we'll use PyRosetta to investigate how an amino acid chain folds into a structure using the Rosetta energy function (force field).

Question: We'll compare two methods of predicting the final protein structure. **We want you to compare the effectiveness of each method (using PyRosetta energy function values, any pertinent screenshots, and any other figures).** The goal of this problem is to have you explore how PyRosetta predicts protein structures, so feel free to include any observations or insights you had about each method. One way to build a good answer would be to have a couple sentences summarizing your results for Method 1, a couple sentences summarizing your results for Method 2, and 1-2 final paragraphs comparing the two methods. You may also choose to include relevant screenshots of predicted structures to support your observations. **You do not need to include code in your submission.**

For all of the questions, we will be using the extended structure of the amino acid chain in *pdbs/1prb_seq.pdb* as the starting structure. The original PDB structure of this protein is also provided for you under *pdbs/1prb.pdb*.

Some helpful considerations as you write your response are listed below. You do not need to explicitly answer these in your write-up but they can guide your response.

- What are the limitations of knowledge-based protein structure prediction methods?
- If our goal is to find conformations with low free energy, why would we ever want to accept a structure with higher energy than our current estimate?
- Did you see either of the methods converge to a final structure? If so, how long did each method take?

Set-up: You will need PyMOL and PyRosetta 4 to complete this challenge problem. We **strongly recommend** using the in-person LTS machines for this portion. While we **do not recommend** installing locally, as local installation has been difficult historically, the included "Assignment 2 Challenge Problem Set-Up" could be helpful.

Once you have both software installed, open a PyMOL window, navigate to the challenge problem folder, and type on the PyMOL command line: *run PyMOLRosettaServer.py*. This will create a link between PyRosetta and PyMOL so that you can watch the protein progress through the structures.
Then, open a separate terminal window. Any other commands below (e.g., *python3.9 predict.py <arguments>*) will need to be run on the terminal.

Part A: Monte Carlo Methods

Before we start exploring the two methods, we will implement parts of the Monte Carlo algorithm, which we will use to predict the protein structure. Implement acceptMove() and predict() in predictor.py. More instructions are in the figure below. Note that ΔU in the below figure refers to the change in U(c) from the current conformation to the next state (i.e., $\Delta U = U(c') - U(c)$).

In the context of biomolecular simulation and structure prediction, the Metropolis **Monte Carlo** algorithm goes as follows.

1. $c \leftarrow$ select an initial conformation
2. Repeat:
   - $c' \leftarrow$ sample a local random move from c (e.g. change one dihedral angle)
   - if $U(c') \leq U(c)$,

$$\text{assign } c \leftarrow c'$$

else

$$\begin{cases} \text{assign } c \leftarrow c' & \text{with probability } e^{-\Delta U/(k_B \cdot T)} \\ \text{do nothing} & \text{with probability } 1 - e^{-\Delta U/(k_B \cdot T)} \end{cases}$$

In our implementation, we will use this algorithm. Note that the proposed move is accepted if the energy decreases, and is accepted with exponentially decreasing probability if the energy increases. This acceptance profile is called the Metropolis Criterion.

Part B: Sampling Moves

In each iteration of the Monte Carlo algorithm, we sample a move to apply to the protein (this is the first bullet point under step 2 in the above figure). We will explore two possible methods for sampling.

Method 1: Dihedral Angles

sampleMove() in the DihedralPredictor class has been implemented for you. This function takes in a Pose object, and returns a new Pose object which has sampled one of three types of dihedral moves.

Specifically, we select a residue at random, then choose one of the following three moves at random: (1) a dihedral move in φ, (2) a dihedral move in ψ, or (3) a shear move, where you select a Δφ, and then update ψ by −Δφ. A move is defined as a rotation of the dihedral angle.

Changes in dihedral angle will be sampled from a normal distribution centered at 0 with standard deviation 5. It's a good idea to take a look at the code to get a sense of what is happening in this function call.

Run DihedralPredictor for 100000 iterations on *pdbs/1prb_seq.pdb* in terminal and watch it search the conformational space in PyMOL. You can use the following command:
python3.9 predict.py pdbs/1prb_seq.pdb dihedral100k.pdb -dihedral -pymol -100000
The predicted structure will be saved in the “out" folder under the name “dihedral100k.pdb".

Method 2: Fragment Set

We can also approach the final structure of a protein by sampling the backbone geometry from a library of *n*-mers in known protein structures (“fragments”) and applying this to the corresponding residues at each iteration. An *n*-mer is a sequence of *n* amino acids.

sampleMove() in the FragmentPredictor class has been implemented for you, and the fragment sets are available under frags/frag3 (3-mers) and frags/frag9 (9-mers).

Using 9-mer fragment sets, run FragmentPredictor for 100000 iterations on *pdbs/1prb_seq.pdb* in terminal and watch the simulation on PyMOL. You can use the following command:
python3.9 predict.py pdbs/1prb_seq.pdb frag9.pdb -frag9 -pymol -100000
As before, the predicted structure will be saved in the “out” folder under the name “frag9.pdb”. You can also experiment with using 3-mers by changing out “frag9” in the above command with “frag3”.

Note: *The Python commands above are only provided as the starting point for your exploration. Feel free to experiment with different arguments. You may consider changing the number of iterations; using the 3-mer fragment set; adding the -fa flag, which will use the full structure to compute energy; combining multiple methods; thinking about ways to refine the predicted structure; and so on.*
The -pymol flag is not required for prediction; removing “-pymol” from your command will speed up the prediction and still give you the final structure under the “out” folder, but you will not be able to watch the intermediate structures on PyMOL.

Submission Instructions: Please refer to the top of the assignment to ensure that your response lines up with the original question. Remember that one way to build a good answer would be to have a couple sentences summarizing your results for Method 1, a couple sentences summarizing your results for Method 2, and 1-2 final paragraphs comparing the two methods.

You can submit your answer to the question, along with any relevant screenshots, as part of your write-up to Assignment 2 on Gradescope. Please tag the corresponding pages for “Challenge Problem”. We are not expecting more than 2-3 paragraphs. Remember, you **do not** need to submit any code - the paragraphs/explanation will suffice.