Assignment 2 – Atomic-Level Molecular Modeling

CS/BIOE/CME/BIOPHYS/BIOMEDIN 279

Due: October 31, 2017 at 3:00 PM

The goal of this assignment is to understand the biological and computational aspects of macromolecular energy functions and of predicting the three-dimensional structure of a folded protein.

Acknowledgements: Portions of this assignment are based off of the PyRosetta Tutorials.

1 Preliminaries

- Download `assn2.zip` from the course website or copy it into your AFS space (see Accessing Software)
- Obtain access to PyMOL and PyRosetta (see Assignment 2 Setup)

2 Force Fields and Free Energy Calculation

2.1 Potential Energy and Force Fields

Proteins, like all molecules, prefer to occupy energetically stable (i.e. low energy) structures. Thus, in order to predict which conformations a protein is likely to adopt, it's important to understand the forces that act within proteins that govern the stability of the macromolecule.

In computational models of proteins, the potential energy function is described as a *force field*. Most force fields are defined as a sum of a number of potential energy terms. Here, we review a few of the terms that appear in common physics-based force fields and discuss their relevance.

Bond Lengths and Angles

Every covalent bond has a potential energy associated with its length and every pair of adjacent bonds has a potential associated with the angle formed by the bonds. We model the potential energy of each bond length and each bond angle as the energy of a harmonic oscillator, or classical spring.

For individual bonds, the energy is proportional to the bond's squared deviation from its ideal length. For bond angles, the energy is proportional to the angle's squared deviation from its ideal angle.

$$U_b(c) = \sum_{b \in B} k_b (b - b_0)^2$$

$$U_\theta(c) = \sum_{\theta \in \Theta} k_\theta (\theta - \theta_0)^2$$

Dihedral Angle

In addition to the energy between bond angles, there is also a potential associated with the dihedral angles (not just ϕ and ψ, but any relevant dihedral angle).

$$U_\phi(c) = \sum_{\phi \in \Phi} \sum_n k_{\phi,n}(1 + \cos(n\phi - \phi_n))$$

Many computational force fields will only contain terms up to n = 3, but some will also include values of n as high as 6.

Question 1: *Use* `dihedralPeriod.py` *to plot a period of the potential energy associated with a single dihedral. Play around with the weights $k_{\phi,n}$ and the values of ϕ_n. Submit a plot where the minimum energy occurs at* $180°$*, the maximum energy occurs at* $0°$*, a local minimum occurs at* $60°$ *and a local maximum occurs at* $120°$*. Include the values of* `k` *and* `offset` *used.*

Electrostatic Potential

Nearly all atoms in a protein carry some partial charge. These charges cause an attractive or repulsive potential according to Coulomb's Law.

$$U_e(c) = k_e \cdot \sum_i \sum_{j>i} \frac{q_i q_j}{r_{ij}}$$

Note that this sum is over all pairs of atoms, not just those involved in a bond. (k_e is a global constant that will depend on the units used for charge and for distance, but not on the atoms under consideration.)

van der Waals Forces

In addition to the electrostatic potential, atoms can also undergo attraction and repulsion via van der Waals forces. These forces are frequently calculated as follows.

$$U_{vd}(c) = \sum_i \sum_{j>i} \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^6}$$

There is some theoretical justification for why the attractive force would depends on r_{ij}^{-6}, but the choice to make the repulsive force depend on r_{ij}^{-12} is primarily historical for computational

convenience (the square of r_{ij}^{-6}). It should be clear that these forces fall off quickly as the atoms in question are far apart.

Question 2: *Structural biologists often refer to the "hydrophobic effect" during protein folding. This "effect" is said to cause certain regions of a protein to fold such that they are no longer exposed to water. Explain this driving force in terms of the electrostatic force.*

Thus, in all, we can define a force field as follows.

$$U(c) = U_b(c) + U_\theta(c) + U_\phi(c) + U_e(c) + U_{vd}(c)$$

Question 3: *Consider the potential energy function $U(c)$. When computing this function, which term(s) will take the longest to evaluate? Give a brief explanation for your answer. (Assume there are $> 50K$ atoms in the protein.)*

2.2 Free Energy

Recall, proteins are dynamic molecules, which are always in motion. Thus, when we talk about protein structure prediction, we're really interested in finding a set of conformations where the protein will spend most of its time.

Let's consider some set of similar conformations C. (We're defining a conformation as a particular arrangement of all the atoms, as specified by specific coordinates for each.) We frequently refer to such a set of conformations as a macrostate or ensemble (or a conformational state, if the set of conformations corresponds to a well in the potential energy surface). We know the probability that the protein is in some conformation c is Boltzmann distributed according to $U(c)$. This means we can express the probability that the protein will adopt some conformation in the macrostate C as the following sum.

$$\Pr\left(p \text{ adopts some } c \in C\right) \propto \sum_{c \in C} e^{-U(c)/k_B T}$$

Intuitively, a macrostate is well-populated if the energy of each conformation in the macrostate is sufficiently low and the number of structures in the macrostate is sufficiently large. We can define the free energy of the macrostate C as the value G_C that satisfies

$$\Pr\left(p \text{ adopts some } c \in C\right) = e^{-G_C/k_B T}$$

It is important to understand the distinction between minimizing potential energy and minimizing free energy: minimizing potential energy gives the single most stable conformation (c) while minimizing free energy gives the most likely macrostate (C).

Question 4: *Consider Figure 1. Let's say this plot represents the potential energy of a given protein as it adopts different conformations. Which conformational macrostate (X, Y, or Z) will be preferred at $T = 0$K? 100K? 298K? Estimate the percentage present of each macrostate at each temperature. You can approximate the energy profile of each macrostate as a box or a parabola. Note: $k_B = 1.38 \times 10^{-23}$ J/K*

This question should illustrate the fact that the optimal structure of a protein (or any molecule for that matter) varies with temperature.

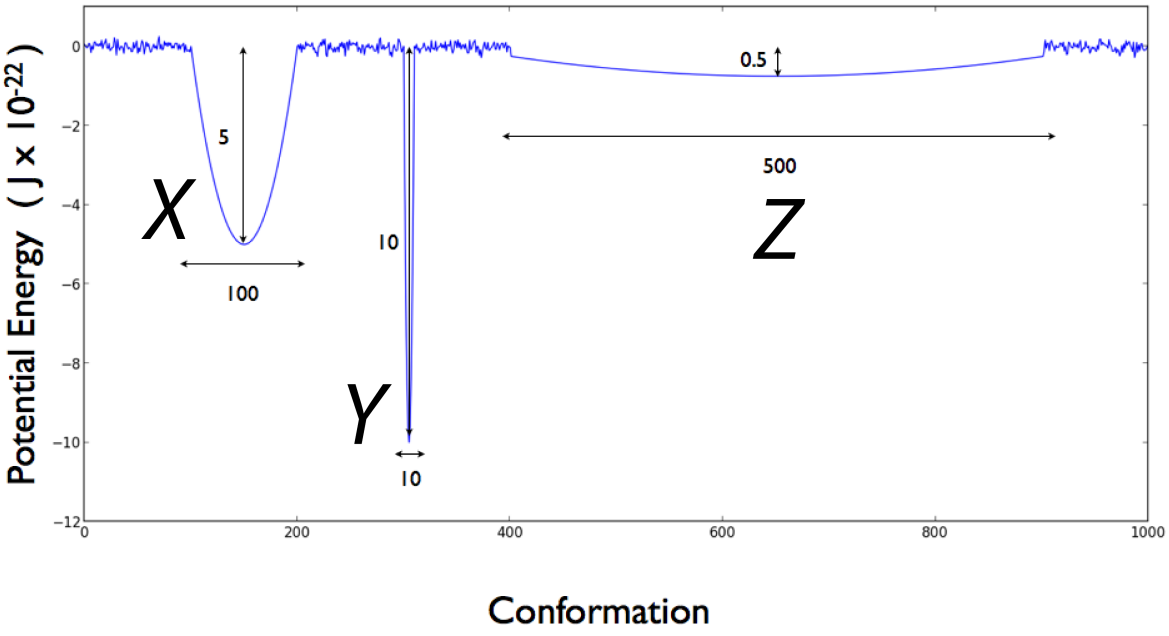

Figure 1: Potential Energy vs. Conformation

2.3 Knowledge-Based Force Fields

In the following exercises, we will be using PyRosetta which utilizes the Rosetta Force Field or Score Function. One key design decision behind Rosetta is the use of a knowledge-based force field. Instead of trying to approximate the potential energy of atomic interactions directly, this force field tries to estimate the free energy of structures by incorporating knowledge of solved protein structures. In other words, conformations are deemed more likely, or lower "energy", if they are similar to conformations that are known to exist in other proteins.

Question 5: *What is one limitation to knowledge-based prediction methods?*

3 Structure Prediction

The predominant belief among structural biologists is that, for the majority of proteins, the amino acid sequence of a protein uniquely defines the three-dimensional structure of the folded protein. This belief is the foundation for protein structure prediction: given the amino acid sequence of a

protein, predict its three-dimensional structure.

Question 6: *Suppose we have a well-specified free energy function $U(c)$ that, for any particular conformation c of a protein, gives the free energy of an ensemble (macrostate) of conformations similar to c. Suppose this energy function has been shown to be accurate across many proteins. Since we want to determine the structures that minimize this free energy function, what prevents us, practically, from evaluating the force field on all reasonable structures to see which is minimum?*

3.1 Monte Carlo Methods

Monte Carlo Methods refer to a large class of algorithms that use random sampling to estimate the solution to computationally complex problems. In the context of protein structure prediction, one common 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 $U(c') < U(c)$,

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

     else

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

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.

Question 7: *Why does the Metropolis Criterion make sense? Namely, if our goal is to minimize free energy, why would we ever want to accept a structure with higher energy than our current estimate?*

Question 8:

(a) Implement `acceptMove()` *in* `Predictor`.
`acceptMove()` *should take in the current energy and the energy of the proposed move, and return* `True` *according to the Metropolis Criterion.*

(b) Implement `predict()` *in* `Predictor`.
Implement the Monte Carlo prediction algorithm with Metropolis Criterion described above. Make calls to `sampleMove()` *and* `acceptMove()` *where appropriate.*

Now that you've implemented the basic framework for the Monte Carlo / Metropolis algorithm, we can start to think about what types of perturbations to the current structure we should explore. First, we will try purely random changes in the dihedral angles.

PyRosetta provides support to interface with PyMOL, so that you can visualize what is happening to a given protein as it is exploring the conformational space. We must initialize PyMOL to listen for calls from PyRosetta. To do this, in PyMOL, run the python script provided in the `assn2` directory.
(Note: If you are running PyMOL through the terminal on FarmShare, you may want to start it using the command `pymol &`, so that you can continue to enter commands.)

```
PyMOL> run PyMOLPyRosettaServer.py
```

Question 9:
`sampleMove()` *in* `DihedralPredictor` *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 $\Delta\phi$, and then update ψ by $-\Delta\phi$.

Changes in dihedral angle will be sampled from a Gaussian 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.

(a) Run `DihedralPredictor` *for 1000 iterations on* `sequence.pdb` *and watch it search the conformational space in PyMOL.*

```
python predict.py pdbs/sequence.pdb dihedral1000.pdb -dihedral
-pymol -1000
```

After 1000 iterations, describe what the structure looks like. Generate a Ramachandran plot using the included script `ramaPlot.py` *from the terminal.*

```
python ramaPlot.py <file1>.pdb [<file2>.pdb ...]
```

Include a screen-shot of both the structure and the Ramachandran plot in your writeup.

(b) Now run the predictor for 100,000 iterations (without watching on PyMOL for speed's sake).

```
python predict.py pdbs/sequence.pdb dihedral100K.pdb -dihedral
-100000
```

Does the structure resemble a folded protein any more than before? Can you distinguish any distinctive secondary structures in PyMOL or in the Ramachandran plot? Include a screen-shot of the structure and of the Ramachandran plot.

Question 10: *As the structure changes according to the algorithm, should we expect it to follow the pathway by which the real protein folds? Why or why not?*

While perturbing the structure one dihedral angle at a time seems like a natural way to sample the search space, you can see that it will take a lot of random samples to converge to anything that

looks like a folded protein. In order to speed this process up, we will introduce a knowledge-based component to the search process.

3.2 Knowledge-Based Sampling

The next predictor will use a library of polypeptide conformations, taken from published structures in the PDB, to inform its random conformational sampling. In particular, at each step, the algorithm will choose an n-mer in the current structure and replace the backbone geometry with the backbone geometry of a corresponding n-mer from the library.

Question 11:
`sampleMove()` *in* `FragmentPredictor` *has been implemented for you. This function is simple and involves a call to PyRosetta's* `ClassicFragmentMover` *object.*

(a) Run the `FragmentPredictor` *for 1000 iterations on* `hras.pdb` *and watch it search the conformational space in PyMOL. Do this for the set of 9-mers and 3-mers.*

```
python predict.py pdbs/sequence.pdb frag9.pdb -frag9 -pymol
-1000
python predict.py pdbs/sequence.pdb frag3.pdb -frag3 -pymol
-1000
```

After 1000 iterations, describe what the structure looks like. Include a screen-shot of the structure and of the Ramachandran plot.

(b) As before, now run each fragment predictor for 100,000 iterations. Observe the final structure in PyMOL and view the Ramachandran plots of the predicted structures. How do these predictions compare to the dihedral prediction? To each other? Be sure to include a screenshot of the structure and Ramachandran plot.

Question 12: *Which method seems to converge to a reasonable structure more quickly? Why is this?*

We can also use these methods for prediction in combination. In general, we start with coarse-grained knowledge-based methods, and then move to more refined sampling.

Question 13:

Run your predictors in succession using the following command.

```
python predict.py out/frag9-100K.pdb frags.pdb -frag3 -100000
python predict.py out/frags.pdb result.pdb -dihedral -100000
```

Include a screenshot of the new structure.

3.3 Side-Chain Packing

So far, the structure prediction algorithms we've used have focused on sampling conformations of the protein backbone. The backbone is crucial in determining secondary and tertiary structure, but it is not the whole story.

As PyRosetta modifies a protein's conformation according to the coarse-grained Rosetta energy function, it does not model the full geometry of side-chains. When the structure is converted back to an all-atom representation, the side-chains can take on highly improbable conformations.

Question 14: *In PyMOL, look at the orientation of the side-chains on* `helix.pdb` *and* `bstrand.pdb`*, and compare them to those from the final prediction* `result.pdb`*.*

```
PyMOL> select result-helix, (result and resi 152-168)
PyMOL> align result-helix, helix
PyMOL> center helix
```

```
PyMOL> select result-bstrand, (result and resi 37-58)
PyMOL> align result-bstrand, bstrand
PyMOL> center bstrand
```

What is the RMS distance between each pair of predicted and crystal structures?

To predict the side-chain orientations more realistically, PyRosetta provides methods which perform similar Monte Carlo sampling on the side-chains. These sampling techniques can also be used in protein design, but here, we will just use them to predict the existing side-chains' orientations.

```
python pack.py <input>.pdb <output>.pdb
```

Question 15: *Look at the side-chains for the protein after packing. What are the root mean squared (RMS) distances now? Is the relative magnitude of change in RMS distance after packing roughly the same or fairly different between the alpha helix and beta strand? Briefly explain why this is the case.*

Question 16: *Compare the final structure (after side-chain packing) to the ground-truth crystal structure (*`hras.pdb`*). Report the RMSD and include a Ramachandran plot.*

Question 17: *Evaluate how the algorithms performed qualitatively. How did each algorithm perform at predicting different secondary structures? Which were predicted more accurately, alpha helices or beta strands? Why do you think this is? How do the methods compare in terms of computational cost? What are some of the pros and cons to each methodology?*

4 Feedback

Again, we'd love some feedback on the assignment! You will receive full credit for this portion for providing any response. We encourage constructive criticism.

Question 18: *What was your favorite aspect of the assignment? What was your least favorite aspect of the assignment? Why? Any suggestions for improvement?*

Question 19: *Approximately how long did this assignment take you? Where did you spend most of this time?*

5 Challenge Question

See assn2_challenge.pdf for the optional challenge question. If you want to attempt the question, it is due a week after this assignment.

6 Submission Instructions

We will make a quiz on Canvas for this assignment, similar to the one for assignment 1. There may be a few days between the release of this assignment and the availability of the quiz.