

Assignment 2 – Atomic-Level Molecular Modeling

CS/BIOE/CME/BIOPHYS/BIOMEDIN 279

Due: **Thursday, October 26, 2023 at 1:00 PM**

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

1 Preliminaries

1. If you are working on your local computer, **make sure you can run PyMOL 2.5 and Python 3.9**. As mentioned in the [software handout](#), we heavily recommend that you have Anaconda for Python 3.9 installed as well.
2. Please start your assignment early so that the course staff can help you troubleshoot any issues in [office hours](#) or [Ed](#).
3. **Please save your code from the “Exercises” listed below**. We’ll ask you to append the code you wrote for the Exercises at the end of your written submission.
4. In Part 4, you will be running a Colab notebook. Historically, students have run into issues on Safari. We have confirmed that the Colab runs successfully in Firefox, Edge, and Chrome.

2 Force Fields and Free Energy Calculation

2.1 Potential Energy and Force Fields

Molecules, including proteins and other biomolecules, prefer to occupy energetically stable (i.e. low-energy) three-dimensional structures. Thus, in order to predict which conformations a protein or other biomolecule is likely to adopt, it’s important to understand the forces acting between its atoms.

One frequently used type of potential energy function for biomolecules, called a molecular mechanics force field, is typically defined as a sum of bond length, bond angle, dihedral angle, electrostatics, and van der Waals terms (see slides from the lecture on ["Energy Functions & their Relationship to Molecular Conformation,"](#) particularly slide 22). Let’s define an individual protein conformation c as a particular arrangement of all the atoms that is specified by precise coordinates for each atom. Then, the potential energy associated with conformation c (defined as $U(c)$) can be modeled as

$$U(c) = U_b(c) + U_\theta(c) + U_\phi(c) + U_e(c) + U_v(c)$$

Note: $U_b(c)$: bond length stretching, $U_\theta(c)$: bond angle bending, $U_\phi(c)$: torsional angle twisting, $U_e(c)$: electrostatic interactions, $U_v(c)$: van der Waals interactions.

Question 1: For a potential energy function defined by such a molecular mechanics force field, which terms (bond length, bond angle, dihedral angle, electrostatics, or van der Waals) will require the most computational time to evaluate? (Assume the molecular system being simulated contains a very large number of atoms.)

2.2 Free Energy

Recall that proteins and other biomolecules are constantly changing shape as their atoms move around. Thus, when we talk about protein structure prediction, we're looking to find a set of similar conformations in which the protein will spend most of its time.

Like in question 1, let c represent an individual protein conformation. We also define a set C of conformations similar to c . We refer to such a set of conformations as a macrostate or conformational state.

We know that the probability that a biomolecule is in some conformation c is Boltzmann distributed according to $U(c)$, where $U(c)$ is the potential energy associated with conformation c . This means we can express the probability that the biomolecule will adopt some conformation in the macrostate C as the following sum.

$$\begin{aligned} \Pr(c \in C) &= \frac{1}{Z} \sum_{c \in C} e^{-U(c)/(k_B \cdot T)} \\ &\propto \sum_{c \in C} e^{-U(c)/(k_B \cdot T)} \end{aligned} \quad (1)$$

k_B : Boltzmann constant

T : temperature

Z is a normalizing constant such that the summed probability across all possible conformations is 1. If we're interested in relative probabilities of macrostates, we don't need to calculate Z .

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

$$\Pr(c \in C) = e^{-G_C/(k_B \cdot T)} \quad (2)$$

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).

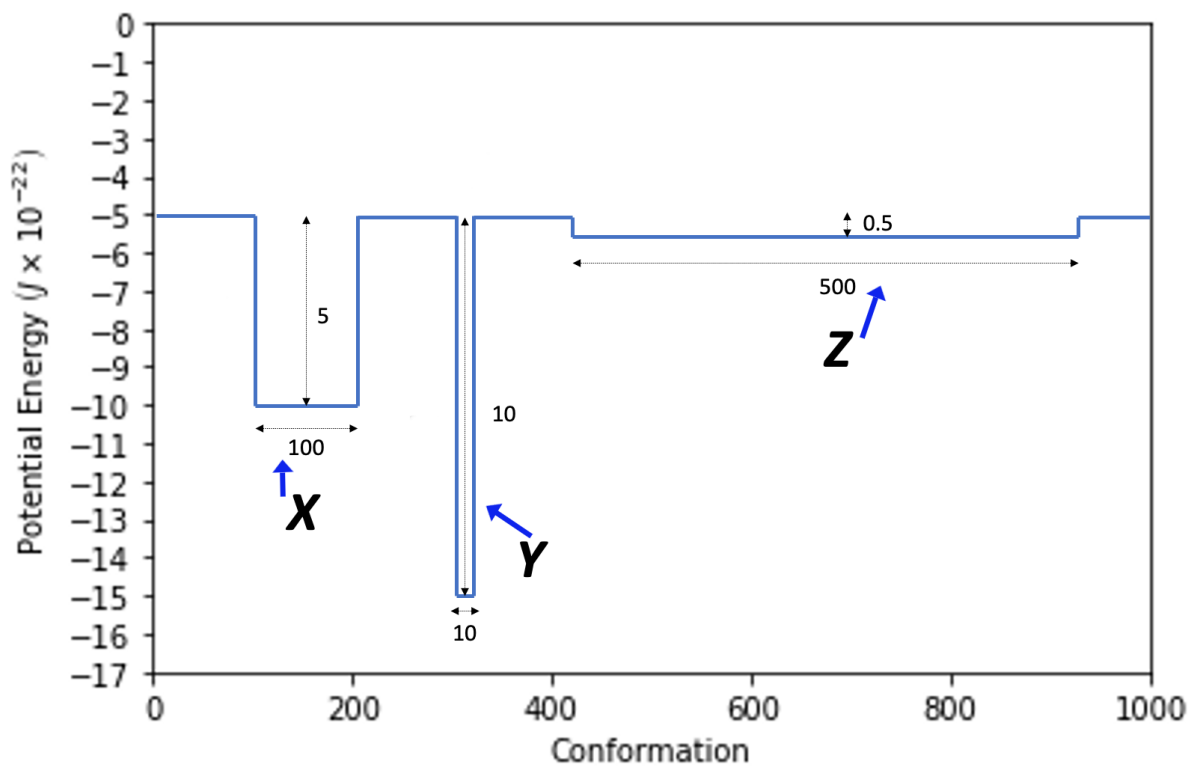


Figure 1: Potential Energy vs. Conformation, labeled.

Question 2: Consider Figure 1. Let's say this plot represents the potential energy of a given protein (and the water surrounding it) as it adopts different conformations.

- (a) Intuitively, which conformational state (X, Y, or Z) would you expect to be preferred at very low temperatures? Which conformational state (X, Y, or Z) would you expect to be preferred at very high temperatures? Why?
- (b) Now, calculate the approximate relative percentage present of each macrostate at each of the temperatures: 1 K, 100 K, and 310 K by using Equation (1). Display your results in a table as shown below. Note that 310 K (equivalent to 37 °C or 98.6 °F) is normal human body temperature.

Temperature (K)	Percentage Present of each Macrostate (%)		
	X	Y	Z
1			
100			
310			

Hints: $k_B = 1.38 \times 10^{-23}$ J/K. You may ignore the time the protein spends outside of conformational states X, Y, and Z. Pay close attention to the sign of the potential energy.

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

3 Deriving a knowledge-based energy function

In this problem, we will build a “knowledge-based” energy function. Instead of approximating the potential energy of atomic interactions and then integrating these to compute free energies, this type of energy function directly estimates the free energy of a candidate structural model by incorporating knowledge of experimentally determined protein structures. Candidate structural models are deemed lower energy (more likely) if they share properties with structures of other proteins. The energy function we will implement is inspired by the paper [Statistical potential for assessment and prediction of protein structures](#) by M.-Y. Shen and A. Sali, which has accrued over 2000 citations.

Our knowledge-based energy function will be based on distances between pairs of residues. We will record the distances between residues in a training set of protein structures from the PDB and give favorable scores to candidate structural models that include inter-residue distances that are observed more frequently than would be expected by random chance. Each pair of residue types (e.g. serine and tryptophan), will have its distances recorded separately. In this way, the energy function will be able to distinguish which types of residues it is favorable to position closely.

Specifically, the overall energy for a candidate structural model will be a sum over energy values for

each pair of residues.

$$G = \sum_{i=1}^N \sum_{j=i+1}^N g(d_{ij}, A_i, A_j) \quad (3)$$

where N is the number of residues in the protein, d_{ij} is the distance between residues i and j , A_i and A_j are the amino acid types of residues i and j , respectively, and g is a per-residue-pair energy function. We'll go into more detail about the per-residue-pair energy function later.

We provide starter code in “statistical_potential.py”. As you complete the exercises below, you should only edit the code in places between the “#####” lines. We've provided unit tests so that you can check the correctness of your code as you implement the components. To run the unit tests, open a terminal window and navigate to your Assignment 2 directory. You can then run the unit tests using the following command.

```
> python test_statistical_potential.py -v
```

Note: You may have to replace `python` with `python3` in order to point to the correct python version.

3.1 Distribution of inter-residue distances in structures from the PDB

First, we will compute distributions (histograms) of the distances between pairs of residues in a large set of structures from the PDB. We will compute a separate distribution for distances between each pair of residue types (i.e., one for distances between aspartate and alanine residues, one for distances between aspartate and arginine residues, etc.).

In the code, we will represent these distributions as a list of counts for various distance bins. The i^{th} entry in the list will store the count for distances between i and $i + 1$ angstroms (\AA). For example, the first entry will correspond to the number of occurrences between 0 and 1 \AA , the second entry will correspond to the number of occurrences between 1 and 2 \AA , and so on. These lists will each have 15 entries, and we will ignore residue pairs with distances over 15 \AA .

Exercise 1: In “statistical_potential.py” implement the function “observed_distribution”. Your code should pass the tests prefixed with “test_observed_distribution” in “test_statistical_potential.py” after implementing this. Plot the observed distribution for aspartate and arginine by running the below command.

```
> python statistical_potential.py ASP ARG
```

Note: While distributions are usually shown as bar charts (histograms), this assignment displays distributions as line graphs to clearly show the difference between the observed and expected distributions.

3.2 Accounting for the expected distribution of inter-residue distances

Given that aspartate and arginine residues typically have opposite charges and presumably interact favorably with one another, it might seem a bit surprising to see that there are many more occurrences of these residues far apart than close together. But is this really unexpected?

We will now consider how many occurrences one would expect in each distance bin if all the residues were positioned randomly. The expected number of occurrences of residue distances between distance 0 and distance d_{ij} is proportional to the volume of a sphere centered at a given point with radius d_{ij} . Therefore, calculating the expected number of occurrences between distances d_{ij} and $d_{ij} + 1$ is equivalent to asking how much volume is located at distances between d_{ij} and $d_{ij} + 1$ from a given point. **Note:** The expected number of occurrences between distance 0 and distance d_{ij} is proportional to a sphere because the distances are measured in 3 dimensions (x,y,z) .

Two important notes: (1) The expected number of occurrences of distances is influenced by the overall number of each residue type in the protein. You do not need to include this in your solution. Elsewhere in the code, we will normalize the entire expected distribution so that the expected and observed counts are the same at 15 Å. (2) In the Shen and Sali paper, they consider a very complex expected count function that accounts for the fact that at distances approaching the size of the protein the expected counts decrease. Do not attempt to model this effect for the purposes of this assignment.

Exercise 2: Implement the function “`expected_distribution`”. Your code should pass all the test cases from Exercise 1 as well as the “`test_expected_distribution`” test case after implementing this. Hint: the volume of a sphere with radius r is $\frac{4}{3}\pi r^3$

Run, the code again for aspartate and arginine. You should now see that there are more occurrences of distances in the 4 to 10 Å range than we would expect by chance.

Question 3: In the plot for aspartate and arginine, you should see that there are fewer observed than expected occurrences for very short distances (0–3 Å). We will soon see that this is the case for every pair of residue types. What physical effect causes this? Hint: How wide is an atom?

3.3 Deriving residue-pair energies

We will now combine the observed and expected distributions we computed above into the per-residue-pair energy function, g . Specifically, the energies will be given by the negative log ratio of the two distributions.

$$g(d_{ij}, A_i, A_j) = -\log \frac{N_{obs}(d_{ij}, A_i, A_j)}{N_{exp}(d_{ij}, A_i, A_j)} \quad (4)$$

Here $N_{obs}(d_{ij}, A_i, A_j)$ is the number of occurrences of residue types A_i and A_j at a distance of d_{ij} in the set of solved protein structures and $N_{exp}(d_{ij}, A_i, A_j)$ gives the corresponding expected count.

Note: \log denotes \log_e , which is also known as the natural log (ln).

This way of combining the two distributions favors distances that are observed more often than expected and results in terms that have the mathematical properties of energies. Following the convention for energies, distances that are more favorable should get lower scores. Distance bins with greater observed than expected counts should also get negative scores.

Exercise 3: Implement the function “`distributions_to_energy`” in the code. Your code should now pass the “`test_distributions_to_energy`” test, in addition to all the tests from the previous questions.

We will now run the code for several pairs of residue types and examine the learned per-residue-pair energy function. Remember, to run the code on, for example, aspartate (ASP) and glutamate (GLU), you can use the command:

```
> python statistical_potential.py ASP GLU
```

Question 4:

- (a) Run your code for aspartate (ASP) and arginine (ARG), and then aspartate (ASP) and glutamate (GLU). What physical interaction explains the difference in energies?
- (b) Run your code for leucine (LEU) and isoleucine (ILE). What “effect” causes these residues to interact favorably?
- (c) Run your code for cysteine (CYS) and cysteine (CYS). What special property of cysteine explains the favorability of positioning cysteine residues very close to one another?

Hint: Slide 43 in "[Biomolecular structure \(including protein structure\)](#)" contains examples of amino acid properties.

Exercise 4: Compute per-residue-pair energy functions for all pairs of residue types by running

```
> python statistical_potential.py compute
```

This will plot all of the energy functions and write them to a file called “energies.txt” in the “starter_code” directory. You don’t have to submit the “energies.txt” file for this exercise, but “energies.txt” is required for the upcoming questions.

3.4 Protein structure prediction

Now that we have code to compute per-residue-pair energies, we can sum them all to get scores for entire proteins. We will use these scores to rank candidate structural models for a set of 20 proteins. For each protein, we’ve provided 300 candidate models, each of which is labeled with an RMSD to an experimentally determined structure (the “native structure”). Ideally, our scoring function will assign the most favorable (lowest) score to models similar to the native structure.

Exercise 5: Implement the “scorer” function. This function computes the energy for a single structural model. Your code should now pass all of the test cases.

After your code passes the test cases, use the below command to run the scoring for all of the test proteins.

```
> python statistical_potential.py score
```

For each protein, the code will print 1) the RMSD of the structural model selected by the energy function, 2) the lowest RMSD model across all candidate models, and 3) the median RMSD of the candidate models. You should see that the energy function performs much better than the median RMSD (which is what would be expected if we picked a random structure), but that the model by no means has perfect performance. For each protein, the code will also produce a scatter plot of RMSD vs. energy for each candidate structural model.

Question 5:

- (a) For what fraction of the proteins does the energy function predict a close to native candidate structural model, where “close to native” is defined as having an RMSD of less than 4.0 Å?
- (b) Include the scatter plots of RMSD vs. energy in your write-up.
- (c) Such scatter plots are commonly included in publications describing knowledge-based energy functions. What relationship between the RMSDs and energies is needed such that one can confidently select a near-native candidate structural model based on the energy function? A simple answer is sufficient.
- (d) Here we are using the energy function to score candidate structural models generated with other software. Energy functions are also often used to guide generation of candidate models (“sampling”). As you saw in lecture, sampling algorithms generally permit transitions that increase the predicted energy. Based on inspection of the scatter plots for the proteins with PDB codes 1bbh and 1cid, why is this essential to arrive at a near-native structural model? **Hint:** What would a curve of best fit look like for these two plots?

Let’s visualize a case where our energy function makes a correct prediction. Open PyMOL and navigate to the assignment directory using the “cd” command. Let’s load the “native” (experimentally determined) structure, a “correct” (close to native) structural model, and an incorrect structural model.

```
PyMOL> load test-pdb-moulder/1mdc/native.pdb
PyMOL> load test-pdb-moulder/1mdc/model2.pdb, incorrect
PyMOL> load test-pdb-moulder/1mdc/model4.pdb, correct
```

Then, align both of the candidate structural models to the native structure. You should see that the “correct” model has a much lower RMSD than the “incorrect” model.

We’ve provided a PyMOL script “visualize.py” to show residue pairs that are assigned particularly favorable per-residue-pair energies by our energy function. (Note that this script depends on “energies.txt” to get the energies, so you must have computed the energies before running this script.) Running the below commands will add lines between residue pairs with highly favorable (less than -0.75) per-residue-pair energy in the structural models.

```
PyMOL> run visualize.py
PyMOL> show_energies correct
PyMOL> show_energies incorrect
```

Question 6: A key region of the protein that is correctly predicted in the correct structural model, but incorrectly predicted in the wrong structural model, involves the residues in the vicinity of residue 18. In the correct structural model, interactions between several pairs of residues in this region are deemed to be highly favorable by the energy function. What “effect” causes these interactions to be highly favorable? What is surrounding residue 18 in the incorrect model that is likely energetically unfavorable?

Hint: A quick way to locate residue 18 is to use the command “select resid 18”.

4 Protein structure prediction with AlphaFold

Now we will do structure prediction using AlphaFold (specifically, AlphaFold2). Like the scoring function that we just created, AlphaFold is a knowledge-based method, in that it was trained to reproduce experimentally-determined structures of proteins, as opposed to directly capturing the underlying physical laws. However, there are several fundamental differences between the knowledge-based potential we just created and AlphaFold that enabled AlphaFold's unprecedented success.

First, AlphaFold does not just score protein conformations, but actually directly outputs 3D structures.

Second, our knowledge-based potential started with a pre-specified functional form, which introduced a variety of constraints on the final scoring function, whereas AlphaFold uses a deep learning approach that can learn far more expressive functions.

Question 7: *Name one assumption that we made in constructing our knowledge-based energy function that limits its ability to represent complex scoring functions. **Note:** There are multiple possible answers, and we're not looking for one specific answer.*

Finally, AlphaFold is able to leverage two types of widely available experimental data that can aid in structure prediction: sequences of related proteins and experimentally determined structures of related proteins. To gain insight into how AlphaFold leverages these data types, we will predict the structure of flavin reductase using AlphaFold with different combinations of input data.

We will use a Google Colab notebook version of AlphaFold called [ColabFold](#). The notebook contains helpful usage information. As mentioned in the Preliminaries section, we have tested and recommend running the Colab in Firefox, Chrome, or Edge.

As a baseline, let's use AlphaFold to predict the structure of flavin reductase using only its sequence. First, we will need to look up the amino acid sequence. Amino acid sequences for most characterized proteins are available from UniProt, a useful resource that organizes a large amount of information about each protein.

[Here](#) is the entry for flavin reductase. You can get the sequence by navigating to the "Sequence" section and clicking "Copy sequence".

Function | P500237 | G_PROTEIN_RECEP_F1_ | PS50262 | G_PROTEIN_RECEP_F1_

Names & Taxonomy

Subcellular Location

Disease & Variants

PTM/Processing

Expression

Interaction

Structure

Family & Domains

Sequence

Similar Proteins

Protein family/group databases

GPCRDB | Search...

Sequence

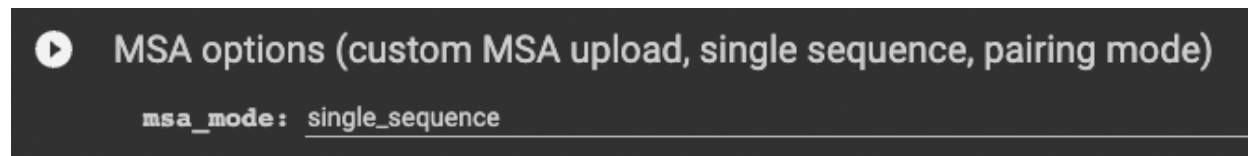
Tools | Download | Add | Highlight | Copy sequence

Length 361
Mass (Da) 41,354

MNESRWTEWR¹⁰ ILNMSSGIYN²⁰ VSERHSCPLG³⁰ FGHYSVVDVC⁴⁰ :
CLACISVDRY¹⁴⁰ LAITKPLSYN¹⁵⁰ QLVTPCLRRI¹⁶⁰ CIILIIWYSC¹⁷⁰ :

Now, let's setup the ColabFold run. Paste the flavin reductase sequence into the "query_sequence"

text box. Then, to disable the use of related sequences (multiple sequence alignments), set “msa_mode” to “single_sequence”.



To run the prediction in ColabFold, navigate to "Runtime" in the Colab toolbar and select "Run all". The results should automatically be downloaded as a zip file. Unzip the file and visualize the top-ranked model using PyMOL. ColabFold predicts 5 potential models, which correspond to 5 .pdb files. We will focus on only the top-ranked model, which includes “rank_001” in the file name. Compare the predicted structure to the experimentally determined structure of flavin reductase with PDB ID: 1HDO. You can download this structure using the [fetch](#) command in PyMOL.

Question 8: *Align the predicted and solved structures to compare them. What is the RMSD between the predicted and experimentally determined structures? Do you notice any differences between the predicted structure and the experimental structure?*
Note: *We are not looking for one specific answer. We expect a general comparison of the two structures.*

In addition to the sequence of the query protein, AlphaFold can make use of information about the sequences of related proteins. Before being input to the model, these sequences are aligned together to form a “multiple sequence alignment” (MSA).

Let’s make a prediction for the same protein, but this time using MSA information. Ensure that the flavin reductase sequence is pasted into the “query_sequence” textbox. Set the “msa_mode” to `mmseqs2_uniref_env` (“MMSeqs2: (UniRef+Environmental)”) and run the prediction again. You can re-run all of the cells after updating the options by clicking “Runtime” - “Restart and run all”.

Question 9: *Again, download the results and compare the “rank_001” model to the experimentally determined structure. What is the RMSD between the predicted and experimentally determined structures? Do you notice any differences between the predicted structure and the experimental structure?*
Note: *We are not looking for one specific answer. We expect a general comparison of the two structures.*

If you like, you can view the sequences used to make the prediction by opening the .a3m file in a text editor.

AlphaFold is also able to make use of experimentally determined structures of related proteins.

AlphaFold has an automated pipeline for fetching template structures from the PDB, but for the sake of interpretability, let’s provide a template manually. In PyMOL fetch PDB ID: 4R1S. The [protein](#) in 4R1S, which is from petunias (the plant), is similar in sequence to flavin reductase and carries out a similar function. In addition to loading the structure into the PyMOL window, this will produce a file “4r1s.cif”. We will have AlphaFold use this model as a template by setting “template_mode” to “custom”. We’ll use the same flavin reductase sequence and the same “msa_mode” as the previous question, so paste the flavin reductase sequence into the “query_sequence” and set the “msa_mode”

to `mmseqs2_uniref_env`. Starting the prediction by clicking “Runtime” and then “Restart and run all” will result in a “Browse” button appearing. Click it and select the file “4r1s.cif”.

Question 10: *Again, download the results and compare the “rank_001” model to the experimentally-solved structure. What is the RMSD between the predicted and experimentally determined structures? Do you notice any differences between the predicted structure and the experimental structure?*

Note: *We are not looking for one specific answer. We expect a general comparison of the two structures.*

Question 11: *Compare the template structure (4r1s) to the experimentally determined structure of flavin reductase. Based on what you see, why are the structures of homologous proteins useful for protein structure prediction? Phrase your answer in terms of the “sequence determines structure, structure determines function” paradigm.*

We note that, while either the multiple sequence information or the templates are sufficient here to give a correct prediction, there are some proteins for which only one or the other source of information is available (and in some especially challenging cases, both types of information are required to make an accurate prediction).

5 Feedback

Again, we’d love some feedback on the assignment! You will receive full credit for this portion for providing any response. Please fill out the assignment [feedback form](#) and provide the code word below. We encourage constructive criticism.

Question 12: *What is the feedback form code word?*

6 Challenge Question

This question is intentionally more challenging than the rest of the homework. It is **not required**, and you can receive full credit for the assignment without attempting it. We will never deduct points for a challenge question submission and will award extra credit depending on the quality of your solution.

Question 13: *The challenge question for this assignment is posted separately on the class website.*

7 Submission Instructions

Similar to Assignment 1, please submit this homework on Gradescope (<https://www.gradescope.com/>). Type up your answers in the text editor of your choice (LaTeX, Word, etc.), making sure to clearly note the question number associated with your answer. When submitting, make sure to tag your answers with the correct questions on Gradescope.

Copy the code for your implementation of `statistical_potential.py` from all the exercises

to the end of your writeup. You only need to include the functions implemented in the exercises: `observed_distribution()`, `expected_distribution()`, `distributions_to_energy()`, and `scorer()`. Then convert the entire document to a pdf. Go to Gradescope, navigate to CS279, “Assignments”, then select **Assignment 2**.

If you completed the challenge question, please carefully read and follow the submission instructions at the bottom of the challenge question document.

If you have any issues submitting, please let the course staff know via Ed or email!