CS612 Homework Assignment 4

Due Mon. April 24, on Gradescope

1. **Representing a sequence as a matrix:** One thing you will have to do for your term project is encode a protein sequence as a set of binary vectors of size 21. This is called "one-hot encoding". You can use other representations, of course. Given the following sequence: "DDHHGFDYNGVMVV", express it as a binary matrix of size 14×21, where 14 is the sequence size and 21 is the "alphabet" of amino acids one letter codes + space. Every position is a binary vector where The one-letter codes for all amino acids is: "ACDEFGHIKLMNPQRSTVWY-" (notice the dash in the end). You should attach your code and the end matrix. You can look here for help: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/. There are also many R functions.

2. **Molecular Dynamic simulations hands-on exercise:** In this assignment you will perform a simple MD simulation of Ubiquitin. You will have to download the NAMD software and you will use VMD for visualizing and processing the results. Download NAMD and VMD at

   http://www.ks.uiuc.edu/Development/Download/download.cgi.

   They are perfectly safe and you can delete them later.

   a. Do the NAMD tutorial at http://www.ks.uiuc.edu/Training/Tutorials/#namd . Do only the part that refers to ubiquitin in a water sphere.
   Attach a screenshot of the final trajectory to your submission.
   b. The log files contain a lot of information about the run of the simulation. Look at the log file and see how the information is output to the file (quite self-explanatory). What is the final potential energy of the system? What is the final bond energy? VdW energy?

3. **Transformations on Molecules:**

   a. Read the PDB file 2EZM from the PDB website. Leave only the C,N and CA atoms (you can either edit the file or use the VMD selection option). These are only the main backbone atoms (except the carboxyl O but it's not important for the dihedral calculations below). Calculate the backbone dihedral angles (ϕ and ψ) for amino acid 10 given the following guidelines:
      - Each atom can be represented as a 3-D points with the x,y,z coordinates taken from the PDB file.
      - A dihedral angle is defined by four atoms - A,B,C,D as follows: Find the normal to the planes defined by A,B,C (or the vectors B-A and C-B) and by B,C,D (or the vectors C-B and D-C) and get the angle between the two normals to get the magnitude of the dihedral angle.
      - The sign of the dihedral angle is determined by the angle between the normal to the plane ABC and the vector D-C. If the angle is $< 90^\circ$ (or the dot product between the two vectors is negative), reverse the sign of the dihedral angle.
      - The vector B-A is obtained by subtracting the x,y,z coordinates of A from those of B (same for C-B, D-C etc.).
      - The cosine of the angle between two vectors is defined as the dot product of the two vectors (after normalization). To normalize a vector divide it by its magnitude, which is defined as $\sqrt{x^2 + y^2 + z^2}$ for a 3D vector. So to get an angle between two vectors – first normalize them, then calculate their dot product and then do $arccos$ on the result (acos in Matlab and R).

- Remember that the dot-product of two vectors $u = [u_1, u_2, u_3]$ and $v = [v_1, v_2, v_3]$ is $u_1 * v_1 + u_2 * v_2 + u_3 * v_3$ and it is a scalar (number), not a vector.
- **Notice that even though I am using degrees here, most programming languages use radians!** To convert from radians to degrees use the following formula: $angle(deg) = angle(rad) * 180/pi$.
- The normal to the plane defined by two vectors is defined as the cross-product of the two vectors (after normalization as defined above). The cross product of two vectors $u = [u_1, u_2, u_3]$ and $v = [v_1, v_2, v_3]$ is a vector perpendicular to v_1 and v_2 and is defined as follows: $v_1 \times v_2 = [u_2 * v_3 - u_3 * v_2, u_3 * v_1 - u_1 * v_3, u_1 * v_2 - u_2 * v_1]$. Notice that $u \times v$ is not necessarily normalized and has to be normalized to calculate the dihedral angle correctly.
- The dihedral angle ϕ is defined as the dihedral angle between C^{i-1}, N, CA, C (where C^{i-1} is the C atom of the previous amino acid) and ψ is defined as the dihedral angle between N, CA, C, N^{i+1} (where N^{i+1} is the N atom of the next amino acid). In other words: ϕ is the rotation angle around the N-CA axis and ψ is the rotation angle around CA-C axis.

  To verify the correctness of your calculations you can upload the molecule on VMD and select $Mouse- > Labels- > Dihedrals$, and then click on the four consecutive atoms in the VMD graphical window. The value of the dihedral angle will appear on that window. You may use MatLab, R, Excel or any reasonable programming language. Attach the code as part of your homework. For this section you may use any programming language you want. For your convenience you can also use the MatLab or R setup below. In any case attach the code you used.

b. Calculate the energy for the structure above with only the backbone atoms. Use a simplified energy function that counts only steric clashes between atoms. The energy is the sum over all the pairs of atoms that are at least 4 atoms apart on the peptide chain (that is – do not share a covalent bond, a planar angle or a dihedral angle). The energy for each pair of atoms is:

$$Eij = \begin{cases} 100/r_{ij}^2 & \text{if } r_{ij} < 3.4 \\ 0 & \text{Otherwise} \end{cases}$$

Where r_{ij} is the distance between two atoms. In this (simplified) function we model each atom as a hard sphere with a radius of 1.7Å, hence the 3.4 - two atoms collide only if their distance is smaller than the sum of their radii. Output the total energy and also report the pairs of severely clashing atoms, whose contribution to the energy is at least 20, and their distances.

c. Rotate the dihedral bond between the N and CA of the last amino acid by 30°. You can perform the rotation using quaternions or matrices as taught in class. If using quaternion, use the N-CA vector as the rotation axis and don't forget to translate the molecule such that N is at the origin. The rotation is equivalent to transforming all the atoms **after** the axis by 30°. Does the structure change much? What atoms move in space as the result of this rotation? What is the RMSD between the original and the modified structure? You can measure the RMSD using VMD or the code you wrote for HW3.

To view the structures you can either type the transformed coordinates back to a PDB file or save the coordinate file with the .crd suffix. In order to view the file in VMD, add a line at the top of the .crd file with anything in it. Read the pdb file and then File→ Load data into molecule. Upload the .crd file and it will appear as another frame.

d. Now rotate the dihedral bond between CA and C of amino acid 40 by 20°. What is the RMSD between the original and the modified structure now? Upload the three structures to vmd or chimera and attach an image of the superimposed structures as part of your homework.

e. Calculate the energy for the two structures you created in c and d above.

General Setup

To calculate the backbone dihedral angles of residue 10 you will need the coordinates of 5 atoms: C^9, N, CA, C, N^{11}. You can extract the coordinates from the attached PDB file into a text file using your favorite text editor or, after saving only the relevant records to a file called, for example, res10.pdb using the unix command: awk '{print \$7" "\$8" "\$9}' res10.pdb > res10.txt. This command prints only fields 7,8, and 9 of each line – these are the x,y,z coordinates of the PDB file.

To extract only the backbone coordinates for the molecule, you can upload it to VMD and click $File- > Savecoordinates- > Dihedrals$

You can use this file as input to R or Matlab (see setup from previous homework assignment for help).

Common troubleshooting:

- Notice that $v_1 \times v_2 = -(v_2 \times v_1)$, so make sure the order of the vectors is correct.
- Make sure your vectors are normalized when calculating the angle between them.

Further explanation for NAMD tutorial

Start with preparing the system here: http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node7.html

Then prepare the water sphere here:

http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node8.html

The files you need are in the build-1-1 directory. For the `wat_sphere.tcl` script to work you may need to copy the files `ubq.psf` and `ubq.pdb` from the `example-output` subdirectory to your working directory in build-1-1. The generated files you will need for your simulation will be named `ubq_ws.psf` and `ubq_wb.pdb`.

Then run your simulation as described here: http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node9.html. Notice that the command line to run the simulation (in your dos/Unix terminal) is: `namd2 ubq_ws_eq.conf > ubq_ws_eq.log&`.

This means that your simulation output is written into a log file named `ubq_ws_eq.log`, which is what I want you to attach to your HW. This is the file where you should find your energy values. Much of the file is all kinds of machine instructions and many many numbers, but you will see that it documents the run of the simulation. For example, you will see lines that look like this:

ETITLE:	TS	BOND	ANGLE	DIHED	IMPRP
ELECT	VDW	BOUNDARY	MISC	KINETIC	
TOTAL	TEMP	POTENTIAL	TOTAL3	TEMPAVG	
ENERGY:	0	1947.9207	1238.5705	263.8635	10.7301
−18781.7697	1984.0070	71.1903	0.0000	0.0000	
−13265.4876	0.0000	−13265.4876	−13265.4876	0.0000	

TS stands for time step, and the second line gives you the energy values in the order that appears in the first line. the second number (1947.9207) corresponds to BOND - the bonded energy, etc.

What I am asking are values taken from the final line, which, when I ran it, was:

ENERGY:	2600	244.0385	671.4530	315.3755	45.0069
−22732.2423	1504.1822	3.1973	0.0000	4342.0934	
−15606.8955	312.9086	−19948.9889	−15586.7004	309.7774	

That is, step 2600. You should get similar but non-identical values.

See also here: http://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-win-html/node12.html.

The snapshot I ask can be obtained by uploading the file `ubq_ws.psf` onto VMD (using File→new Molecule) and then using File→Load data into Molecule and select `ubq_ws_eq.coor` which should appear in your sphere-1-2 directory after the end of the simulation. This is your final snapshot. You can save the molecule image either by a screenshot or by using File→Render and select the default (vmdscene.tga) and click "Start rendering". The .tga suffix can be converted to png or jpg using an image editor.

Please don't attach any other file. They are very big and some are not even text files.

Setup for Matlab

Matlab offers a lot of matrix operations that you would otherwise have to implement yourself. We provide some hints/directions for those of you who choose to implement this assignment in Matlab. The very first step you need to do is to read from an ASCII file that contains the cartesian coordinates of the chain you will manipulate and save these coordinates in a data structure that will represent your chain. The simplest data structure at this point is an array, where positions $3 * (i-1)+1, 3 * (i-1)+2$, and $3 * (i-1)+3$ contain the coordinates of atom i. Note that Matlab starts counting from 1. You can read the coordinates into Matlab using the command:

$cartesian\_coordinates = textread(input\_file,'\%f');$

Where the input file contains the coordinates. You can write a simple program that leaves in only the coordinates or save the PDB file in crd format on VMD, and delete the first dummy line, or use the awk command as above. You can check how many coordinates you have read with the command length(cartesian coordinates). The textread command stores all coordinates read from the backbone.crd file in the cartesian coordinates array. You could manipulate this array with the dihedral rotations you will define. However, it might be more convenient for matrix operations and for clarity to store the cartesian coordinates in a matrix with N rows and 3 columns for each row. In this way, row i contains all the three cartesian coordinates for atom i. Define the number of atoms in the backbone with the command:

$N = length(cartesian\_coordinates)/3;$

Now you need to declare a matrix with the right dimensions. Since you actually need to work with homogeneous coordinates, it might be convenient to declare a matrix with N rows and 4 columns, where the last column contains 1. You can do so with the command:

$backbone\_chain = zeros(N, 4);$

which creates a matrix with N rows and 4 columns initialized to all zeros. You can set the fourth column to 1 with the command:

$backbone\_chain(:, 4) = 1;$

Now you need to place the cartesian coordinates from the array to this matrix. You can do so with the for loop below:

$for\ i = 1 : N$
$for\ j = 1 : 3$
$backbone\_chain(i, j) = cartesian\_coordinates((i-1) * 3 + j);$
$end;$
$end;$

The cartesian coordinates you have just read will serve as a basis for performing rotations, according to what method you choose to represent and manipulate the protein. Say the position of atom i needs to be transformed by your transformation matrix. You can do so by multiplying trans mat with backbone chain(i, :) as in trans

mat * backbone chain(i, :)'. Note that the colon is very convenient as it gives an entire row or an entire column and that backbone chain(i, :)' gives you the transpose that is necessary for the multiplication to be carried out. In any case, your transformation matrices need to have dimension 4x4 to work with homogeneous coordinates. You can initialize an empty 4x4 matrix by writing

$transmat = zeros(4, 4);$.

Then you can set the elements of this matrix to what they should be for the particular method you use. You can evaluate all the cosines and sines that you need in Matlab. Matlab offers you built-in operations such as dot product or vector norm. Make sure that you understand these operations before you apply them.

* I thank Prof. Lydia Kavraki for providing this section.

Setup for R

You can see basic vector and matrix operations in the previous homework setup. I couldn't find an easy way to do cross product in R, so I attach an R file with some functions implemented by me, as well as some basic quaternion stuff. Feel free to use it as a basis for your implementation. Don't forget the "apply" and "sweep" options in R that can save you lots of time instead of using loops which are less efficient.