SCIE2100 Practical 8, Week 11

Aim

In this practical you will consider a computational method for predicting protein secondary structure. You will tinker with its make-up to further your understanding of how such predictions can be performed. You will also evaluate its performance to acquaint yourself with metrics and to get a sense for how well (or poorly) it works.

You will evaluate a single prediction manually and anecdotally by viewing the real three-dimensional structure in the Protein Databank (PDB). You will also evaluate a large number of predictions quantitatively by relying on secondary structure assignments determined from PDB structures. Please read the appendix and familiarize yourself with the theory of the Chou-Fasman algorithm before starting this practical.

Files

For this week's practical you will need this specification, *sequence.py* (with *symbol.py* and *prob.py*) and *sstruct.py*. You will also need to download a couple of data files: *prot2.fa* and *sstr3.fa*.

Secondary structure: the data

We refer to three-class secondary structure, with H=Alpha Helix, E=Beta Sheet, and C=Coil (or everything else).

1. View the two files *prot2.fa* and *sstr3.fa* and verify that protein sequences are matched by their secondary structure assignment.

Load them into your Python shell using `readFastaFile` in *sequence.py*. Print a few of them to check that you can use the same index to retrieve both an amino acid sequence and its secondary structure assignment.

Chou-Fasman: the method

Chou and Fasman presented a detailed study of the presence (and absence) of amino acids in local protein secondary structure elements from a set of resolved protein structures. Their findings were presented as a table of scores that can be used to assess the potential secondary structures that are formed from amino acids. We represent these scores as a Python dictionary that associates each amino acid with a list of scores (see the appendix for the original table). The first three scores for each amino acid correspond to its tendency to be involved in alpha-helices, beta-sheets and turns respectively; we won't use the remaining scores in this prac.

```
cf_dict = {  # Chou-Fasman table
#     P(a), P(b), P(t),    f(i), f(i+1), f(i+2), f(i+3)
'A': ( 142,   83,   66,   0.060,  0.076,  0.035,  0.058 ),    # Alanine
'R': (  98,   93,   95,   0.070,  0.106,  0.099,  0.085 ),    # Arginine
'N': ( 101,   54,  146,   0.147,  0.110,  0.179,  0.081 ),    # Aspartic Acid
'D': (  67,   89,  156,   0.161,  0.083,  0.191,  0.091 ),    # Asparagine
'C': (  70,  119,  119,   0.149,  0.050,  0.117,  0.128 ),    # Cysteine
'E': ( 151,   37,   74,   0.056,  0.060,  0.077,  0.064 ),    # Glutamic Acid
'Q': ( 111,  110,   98,   0.074,  0.098,  0.037,  0.098 ),    # Glutamine
'G': (  57,   75,  156,   0.102,  0.085,  0.190,  0.152 ),    # Glycine
'H': ( 100,   87,   95,   0.140,  0.047,  0.093,  0.054 ),    # Histidine
'I': ( 108,  160,   47,   0.043,  0.034,  0.013,  0.056 ),    # Isoleucine
'L': ( 121,  130,   59,   0.061,  0.025,  0.036,  0.070 ),    # Leucine
'K': ( 114,   74,  101,   0.055,  0.115,  0.072,  0.095 ),    # Lysine
'M': ( 145,  105,   60,   0.068,  0.082,  0.014,  0.055 ),    # Methionine
'F': ( 113,  138,   60,   0.059,  0.041,  0.065,  0.065 ),    # Phenylalanine
'P': (  57,   55,  152,   0.102,  0.301,  0.034,  0.068 ),    # Proline
'S': (  77,   75,  143,   0.120,  0.139,  0.125,  0.106 ),    # Serine
'T': (  83,  119,   96,   0.086,  0.108,  0.065,  0.079 ),    # Threonine
'W': ( 108,  137,   96,   0.077,  0.013,  0.064,  0.167 ),    # Tryptophan
'Y': (  69,  147,  114,   0.082,  0.065,  0.114,  0.125 ),    # Tyrosine
'V': ( 106,  170,   50,   0.062,  0.048,  0.028,  0.053 ),    # Valine
'Y': (  69,  147,  114,   0.082,  0.065,  0.114,  0.125 ),    # Tyrosine
'V': ( 106,  170,   50,   0.062,  0.048,  0.028,  0.053 ),}   # Valine
```

Chou and Fasman also proposed a set of rules to be followed in order to predict alpha helices, beta sheets and turns from amino acid sequence (see appendix).

The Chou-Fasman rules are generally based on “windows” of amino acids.
First, to easily work with windows of “scores”, we assign scores to each position in a sequence `seq` (`index` is used to identify the type of score):

```
def getScores(seq, index = 0):
    """ Create a score list for a sequence by referencing the Chou-Fasman table. """
    return [cf_dict[s.upper()][index] for s in seq]
```

To identify alpha helices we look at 6 residues at a time and check if 4 or more have an alpha score of at least 100:

```
def markCountAbove(scores, width = 6, call_cnt = 4):
    """ Create a list of booleans that mark all positions within a window
        of specified width that have scores above 100.
        scores: a list of scores (one for each position in sequence)
        width: width of window
        call_cnt: required number of positions with score 100 or more
        return: list of "calls" (positions in windows with at least call_cnt)
    """
    above = [False for _ in range(len(scores))]
    cnt = 0 # keep track of how many in the current window that are > 100
    for i in range(len(scores)):
        if scores[i] > 100: cnt += 1
        if i >= width:
            if scores[i - width] > 100: cnt -= 1
        if cnt >= call_cnt:
            for j in range(max(0, i - width + 1), i + 1):
                above[j] = True
    return above
```

Once you understand the above, lets use them (they are available in *sstruct.py*):

2. Consider the following amino acid sequence PNKRKGFSEGLWEIENNPTVKASGY which we can work with in Python like so:

```
> import sequence as seq
> import symbol as sym
> myprot = seq.Sequence('PNKRKGFSEGLWEIENNPTVKASGY', sym.Protein_Alphabet, '2NLU_r76')
> alpha = getScores(myprot, 0)
> calls_a1 = markCountAbove(alpha, width = 6, call_cnt = 4)
> print makesstr(calls_a1, 'H')
-HHHHHHHHHHHHHHHHHHHHHHH—
```

The first rule thus predicts that most of the sequence will fold into a helix. Use PDB and search for entry 2NLU. The above sequence is the C-terminus of 2NLU starting at position 76 and onwards. Use Jmol to view the structure.

(Right-click and Set Picking "Label" to view residue identity.) What does the PDB structure corresponding to that sequence look like? is it helical?

3. According to the original paper by Chou and Fasman, Proline is a helix-breaker (see point 3 in the excerpt from the original paper in the appendix). However, the functionality to consider this property of Proline is not implemented—the method above can still classify it as belonging to a helix. Study the prediction from the exercise above and look at the structure in PDB. In fact, many would argue that the ideal output should be closer to `----HHHHHHHHHHH----------`. Describe a strategy to "break" predicted helices. You will implement this strategy in Python later.

Chou and Fasman suggested that the boundaries of helix calls could be extended both downstream and upstream subject to average scores (see rules in appendix). We implement this process like so:

```python
def extendDownstream(scores, calls, width = 4):
    """ Create a list of booleans that mark all positions that are contained
        in supplied calls list AND extend this list downstream containing a
        specified width average of 100.
    """
    sum = 0.0
    order = range(0, len(calls) - 1, +1)  # we are extending calls downstream
    cnt = 0
    for i in order:  # extend to the right
        if calls[i]: # to extend a call is required in the first place
            cnt += 1
            sum += scores[i] # keep a sum to be able to average
            if cnt >= width:   # only average over a width
                sum -= scores[i - width + 1]
            if not calls[i + 1] and sum + scores[i + 1] > width * 100: # check
                calls[i + 1] = True
        else: # no call, reset sum
            cnt = 0
            sum = 0.0
    return calls
```

The Chou-Fasman rules can be used to predict the assignment beta-sheet as well. In fact it is the same procedure as with alpha helices, except that we use the beta scores in the table and a window width of 5 (instead of 6). Here, we require that 3 positions have a score of 100 (or above; see appendix). The methods above can thus be used once again we modified parameter values (see *sstruct.py* for an example).

4. Describe in your own words the process of applying the Chou-Fasman prediction rules for alpha-helices and beta-sheets. (i.e., in your own words what do the code snippets do?)

Once we have alpha-helices and beta-sheets predicted, a method needs to ensure that they do not overlap. (A residue cannot be in both simultaneously.) Chou-Fasman has another rule for this (see appendix) that we implement with two methods:

```python
def calcRegionAverage(scores, calls):
    """ Determine for each position in a calls list the average score over the region
        in which it is contained. """
    region_avg = []
    sum = 0.0
    cnt = 0
    # First determine the average for each region
    for i in range(len(scores)):  # go through each position
        if calls[i]:              # position is part of a "called" region
```

```
            sum += scores[i]       # add the score of that position to the average
            cnt += 1               # keep track of the number of positions in region
        else:                      # we are outside a "called" region
            if cnt > 0:            # if it is the first AFTER a called region
                region_avg.append(sum/cnt)    # save the average
            sum = 0.0              # reset average
            cnt = 0
    if cnt > 0:                    # if it is the first AFTER a called region
        region_avg.append(sum/cnt)    # save the average
    # with all averages known, we'll populate the sequence of "averages"
    region = 0
    pos_avg = []
    cnt = 0
    for i in range(len(scores)):
        if calls[i]:
            pos_avg.append(region_avg[region])
            cnt += 1
        else:
            pos_avg.append(0)
            if cnt > 0:
                region += 1
            cnt = 0
    return pos_avg

def checkSupport(calls, diff):
    """ Create a list of booleans indicating if each true position is supported
        by a positive score """
    supported = []
    for i in range(len(calls)):   # go through each position
        supported.append(calls[i] and diff[i] > 0)
    return supported
```

So, for a given protein `myprot` we predict its secondary structure by running a series of methods, updating `calls` we make for both alpha-helices and beta-sheets:

```
alpha = getScores(myprot, 0)
beta  = getScores(myprot, 1)

calls_a1 = markCountAbove(alpha, width = 6, call_cnt = 4)
calls_a2 = extendDownstream(alpha, calls_a1, width = 4)
calls_a3 = extendUpstream(alpha, calls_a2, width = 4)

calls_b1 = markCountAbove(beta, width = 5, call_cnt = 3)
calls_b2 = extendDownstream(beta, calls_b1, width = 4)
calls_b3 = extendUpstream(beta, calls_b2, width = 4)

avg_a = calcRegionAverage(alpha, calls_a3)
avg_b = calcRegionAverage(beta, calls_b3)
diff_a = [avg_a[i] - avg_b[i] for i in range(len(avg_a))]
diff_b = [avg_b[i] - avg_a[i] for i in range(len(avg_a))]
calls_a4 = checkSupport(calls_a3, diff_a)
calls_b4 = checkSupport(calls_b3, diff_b)
```

PDB: evaluate predictions mark I

PDB contains over 80000 protein structures/complexes. The structures below are numbered from 1 to 0; choose the structure that corresponds to the last digit of your student number, search PDB for that protein, and view the structure in Jmol.

1. 1EVH	2. 2Q67	3. 1WN2	4. 1RDU	5. 1OWX
6. 2KDE	7. 2CR5	8. 2KDN	9. 2EEI	0. 1VCK

5. Copy the sequence and predict the secondary structure. Visually inspect the structure and *roughly* evaluate the agreement of assignments. Note the colour scheme used in Jmol that indicates the secondary structure (under "Display

options"). How well does the predicted structure correlate with the PDB provided structure?

Submit (Q5): Give the name, amino acid sequence and predicted secondary structure of your chosen PDB structure. Describe and assess the correlation between your predicted structure and the PDB structure, giving a rough evaluation of the percent accuracy of the secondary structure prediction? Break down how you determine this from the structure, e.g., by annotating an image, and specifying accuracy for alpha helix and beta sheet.

DSSP: evaluate predictions mark II

DSSP (Define Secondary Structure of Proteins) is a (standard) system for assigning secondary structure to individual residues from known three-dimensional structure. We collected 2518 protein structures from PDB for which DSSP secondary structure was assigned (refer to prot2.fa and sstr3.fa). To evaluate how well we are able to predict secondary structure we can look at the agreement between what the method is outputting and what sstr3.fa says for every single position in every single sequence.

If we focus on one class, say helix, for a single position, there are four possible outcomes.

- True positive (*tp*): The position is a helix and the prediction is a helix (correct).
- True negative (*tn*): The position is not a helix, and the prediction is not a helix (correct).
- False positive (*fp*): The position is not a helix, but the prediction is a helix (wrong).
- False negative (*fn*): The position is a helix, but the prediction is not (wrong).

From these, we can determine sensitivity *tp*/(*tp*+*fn*) and specificity *tn*/(*tn*+*fp*), but more importantly percent accuracy (*tp*+*tn*)/(*tp*+*tn*+*fp*+*fn*).

6. Complete the code that counts the number of each of these outcomes (beginning at line 241 of *sstruct.py*).

7. What percent accuracy would you expect if you were guessing the assignment of 3-class secondary structure? You can assume that each class is even in size.

   Submit (Q7): Give your base-line accuracy calculations from exercise 7.

8. Determine the accuracies for alpha-helices and beta-sheets for the proteins and structures in *prot2.fa* and *sstr3.fa* respectively.

   Submit (Q8): Give the accuracy assessments from exercise 8.

9. Implement the Proline breaker functionality discussed above. Re-assess the accuracies.

   Submit (Q9): Describe your proposed Proline breaker strategy from exercise 3. Provide your implementation code for your Proline breaker method (from exercise 9), and discuss what difference it makes to the accuracy assessments (i.e., re-run exercise 8)

Assessment

Answer the questions highlighted and marked with Submit above in a PDF or Word document and submit it to Blackboard by the due date.

Appendix: Chou-Fasman table and algorithm

Name	P(a)	P(b)	P(turn)	f(i)	f(i+1)	f(i+2)	f(i+3)
Alanine	142	83	66	0.060	0.076	0.035	0.058
Arginine	98	93	95	0.070	0.106	0.099	0.085
Aspartic Acid	101	54	146	0.147	0.110	0.179	0.081
Asparagine	67	89	156	0.161	0.083	0.191	0.091
Cysteine	70	119	119	0.149	0.050	0.117	0.128
Glutamic Acid	151	037	74	0.056	0.060	0.077	0.064
Glutamine	111	110	98	0.074	0.098	0.037	0.098
Glycine	57	75	156	0.102	0.085	0.190	0.152
Histidine	100	87	95	0.140	0.047	0.093	0.054
Isoleucine	108	160	47	0.043	0.034	0.013	0.056
Leucine	121	130	59	0.061	0.025	0.036	0.070
Lysine	114	74	101	0.055	0.115	0.072	0.095
Methionine	145	105	60	0.068	0.082	0.014	0.055
Phenylalanine	113	138	60	0.059	0.041	0.065	0.065
Proline	57	55	152	0.102	0.301	0.034	0.068
Serine	77	75	143	0.120	0.139	0.125	0.106
Threonine	83	119	96	0.086	0.108	0.065	0.079
Tryptophan	108	137	96	0.077	0.013	0.064	0.167
Tyrosine	69	147	114	0.082	0.065	0.114	0.125
Valine	106	170	50	0.062	0.048	0.028	0.053

Algorithm:

1. Assign all of the residues in the peptide the appropriate set of parameters.
2. Scan through the peptide and identify regions where 4 out of 6 contiguous residues have P(a-helix) > 100. That region is declared an alpha-helix. Extend the helix in both directions until a set of four contiguous residues that have an average P(a-helix) < 100 is reached. That is declared the end of the helix. If the segment defined by this procedure is longer than 5 residues and the average P(a-helix) > P(b-sheet) for that segment, the segment can be assigned as a helix.
3. Repeat this procedure to locate all of the helical regions in the sequence.
4. Scan through the peptide and identify a region where 3 out of 5 of the residues have a value of P(b-sheet) > 100. That region is declared as a beta-sheet. Extend the sheet in both directions until a set of four contiguous residues that have an average P(b-sheet) < 100 is reached. That is declared the end of the beta-sheet. Any segment of the region located by this procedure is assigned as a beta-sheet if the average P(b-sheet) > 105 and the average P(b-sheet) > P(a-helix) for that region.
5. Any region containing overlapping alpha-helical and beta-sheet assignments are taken to be helical if the average P(a-helix) > P(b-sheet) for that region. It is a beta sheet if the average P(b-sheet) > P(a-helix) for that region.
6. To identify a bend at residue number j, calculate the following value
   $$p(t) = f(j)f(j+1)f(j+2)f(j+3)$$
   where the f(j+1) value for the j+1 residue is used, the f(j+2) value for the j+2 residue is used and the f(j+3) value for the j+3 residue is used. If: (1) p(t) > 0.000075; (2) the average value for P(turn) > 1.00 in the tetrapeptide; and (3) the averages for the tetrapeptide obey the inequality P(a-helix) < P(turn) > P(b-sheet), then a beta-turn is predicted at that location.

From Chou and Fasman, *Biochemistry*, **1974**, *13* (2), pp 222–245
DOI: 10.1021/bi00699a002, page 224.

these assignments are compared to the classification of Lewis and Scheraga (1971), it is seen that they assigned Ser as a helix breaker (Ser is helix indifferent, Table I), Phe and Tyr as helix indifferent (Phe is a helix former, Tyr is a helix breaker, Table I), with the other residues in agreement with the present findings. While Ptitsyn and Finkelstein (1970) made no attempts in identifying the β-sheet potential of individual amino acids, their general classification that hydrophobic residues are β formers and that charged residues, as well as Pro, are β breakers appears in reasonable agreement with the present assignments for β-sheet residues in Table I. As the data sampling herein of helical residues (890) is more than double that of the β residues (424) in the 15 proteins, it is not expected that the P_α values will change greatly with the possible exceptions of Met, Trp, and Cys since these residues do not occur frequently in proteins. However, future values of P_β based on large numbers of proteins, as their conformation becomes known, could alter some of the present assignments for β-sheet residues.

With these limitations in mind, one can proceed with the prediction of secondary structures in proteins using Table I and the following predictive rules.

A. Search for Helical Regions. 1. Helix Nucleation. Locate clusters of four helical residues (h_α or H_α) out of six residues along the polypeptide chain. Weak helical residues (I_α) count as 0.5 h_α (*i.e.*, three h_α and two I_α residues out of six could also nucleate a helix). Helix formation is unfavorable if the segment contains $^1/_3$ or more helix breakers (b_α or B_α), or less than $^1/_2$ helix formers.

2. *Helix Termination.* Extend the helical segment in *both* directions until terminated by tetrapeptides with $\langle P_\alpha \rangle < 1.00$. The following helix breakers can stop helix propagation: b_4, b_3i, b_3h, b_2i_2, b_2ih, b_2h_2, bi_3, bi_2h, bih_2, and i_4. Once the helix is