Lecture Notes for 20.320 Fall 2012

Protein Structure Prediction

Ernest Fraenkel

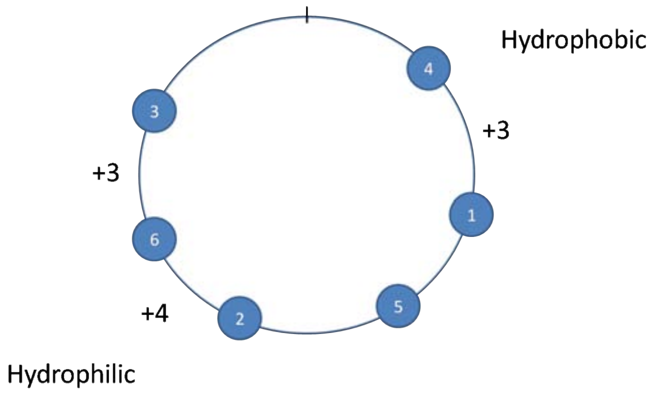
Introduction

We will examine two methods for analyzing sequences in order to determine the structure of the proteins. The first approach, known as the Chou-Fasman algorithm, was a very early and very successful method for predicting secondary structure. The second approach uses sequence homology to compare a sequence of unknown structure to a set of proteins with known structures.

Predicting secondary structure

One of the earliest successes for protein structure prediction was the prediction of the helices of insulin by Schiffer and Edmundson in 1967. Recall that one of the principle driving forces for protein folding is burial of hydrophobic surfaces. In a small protein like insulin, almost every helix is going to have one face on the surface and one on the interior.

So, if we imagine looking down a projection of a helix, what will it look like?



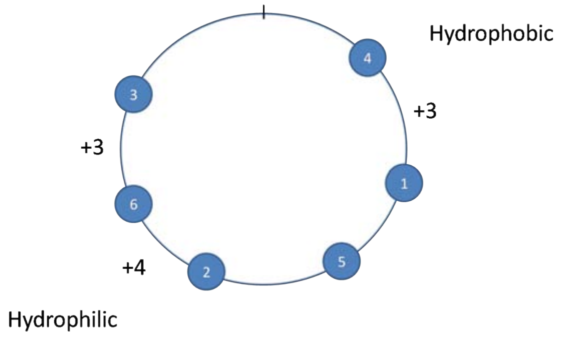
Based on the repeat of an alpha helix, what periodicity do you expect for hydrophobic and hydrophilic residues?

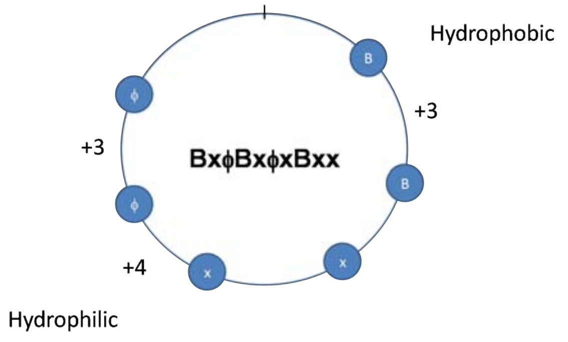
What repeat would we expect for an amphipathic beta sheet?

Now assume you are given a sequence with the following pattern, where B is used to represent a hydropo**b**ic residue and ϕ is used to represent hydrophilic:

BxϕBxϕxBxx

try mapping it onto the helix:





This sequence could represent a **amphipathic** helix.
What about this sequence: BϕBϕBϕBϕ ?
This approach has only limited applicability. Why?

Chou and Fasman Algorithm

Chou-Fasman developed a more robust algorithm motivated by what was understood at the time about protein folding. First, as we already mentioned in the section on "foldamers", some amino acid homopolymers spontaneously form alpha helices. Poly-L-lysine is an interesting example: "At neutral pH, PLL exists as a random coil in solution due to the high charge of the lysine side chains. Above pH 10.6, PLL transforms into the α-helix conformation because the charge on PLL is reduced at a pH above the pK_a (10.5) of the lysine side chains. Upon heating PLL to 51 °C followed by slow cooling to room temperature at a pH above 10.6, the α-helix structure transforms into the β-sheet conformation [16]." (from Blanch and Prausnitz, 2002).

Other homopolymers also make alpha-helices in solution.
This led to the idea that some amino acids had a higher helix propensity for forming helices.

The second important theoretical point was that helices begin with small regions that spontaneously adopt a helical conformation. These are called nuclei. The energy needed to extend a nucleated helix is lower than that needed to create the nucleus in the first place.

So the overall idea was as follows: find regions that might be these sites of nucleation and then see how far they could extend before the helix was broken by residues that don't favor an alpha helical conformation. The full algorithm associates a score with a region that is designed to approximate the probability the sequence is helical. This can then be compared to the probability the sequence is in another state.

Here is a bit more detail on the algorithm:
First, they compiled a database of proteins of known structure. At the time, this was a relatively easy task, because only a handful of protein structures were known. They then classified each amino acid in these structures into helices, beta-sheets, beta-turns and “other” (also called “coil”).

Then they made a frequency table with twenty rows and four columns. Each element is the normalized frequency of that amino acid in the categories Alpha, Beta, Turn and Coil.

What do you expect the frequency of Pro to be in Alpha?
It’s near at the bottom of the list. It’s actually tied with glycine. You might want to think about why glycine is down there.

Here are the main ideas of the algorithm.

1. *Helix Nucleation*. Locate clusters of four out of six residues with a high propensity for forming helices.
   There are special cases for Asp and His which weakly nucleate and for Tyr, Asn, Pro and Gly which are considered helix breakers.
2. *Helix Termination*. Extend the helical segment in *both* directions until terminated by tetrapeptides with low average helical propensity scores.
3. Pro cannot occur in the alpha helix.

It’s not a hard algorithm to implement and it is surprisingly accurate. Three decades of work in the field have only made incremental improvements over this approach.

Some features are worth noting. The algorithm is loosely based on early experiments indicating that helices are nucleated and then grow, but it doesn’t attempt to simulate this process explicitly. It is also based on observations that some amino acids have a higher helical propensity. Again, it does not try to derive these propensities from first-principles. Rather, it derives them statistically from databases.

These kinds of approaches are called knowledge-based. It’s a misleading name. There is a lot of “knowledge” in them in the sense that they use information from databases, but no knowledge in the sense of understanding. Later algorithms are even more of this form. They may do a better job of finding subtle patterns in the data, but they provide no insight into the origins of these patterns.

The field exploded after this algorithm was published and now there are many new versions. **Some use evolutionary information** to aid in determining the secondary structure, and this is a broad concept that is often invoked.

Predicting domain structure

Secondary structure predictions work OK for alpha helices, less well for other structures. It seems likely that alpha helix formation is more likely to be dominated by local forces than beta sheets. Since beta-sheets can have two strands from very different regions of the protein h-bonded together, their stability is based on non-local sequence features that would be missed in the Chou-Fasman approach.

These methods are still far from giving a complete structure. As computer power has increased, more and more attention is given to attempting to predict the full structure of a protein.

This field underwent a revolution with a simple idea. Every two years there is a competition in which the organizers get hold of a set of x-ray crystal structures that are solved but not yet published. They release the sequences of these proteins to the participants and ask them to submit predicted structures. Then there is an independent team that evaluates the results. The competition is called CASP, "Critical Assessment of Techniques for Protein Structure Prediction." This gives a completely objective way to find out which algorithms work and which don't. The most recent CASP in 2010 involved 200 different protein structure prediction groups.

Homology Modeling

Based on the CASP competition it turns out that the most successful way to predict a protein's structure is to use the structure of another protein with similar sequence. There are two challenges:

1. Accurately aligning the sequence with the structure
2. Refining the structure with the new side chains and any insertions/deletions in the sequences.

Sequence Alignments and Searches

Given that proteins can diverge in sequence and preserve structure, how could you find the members of a family?

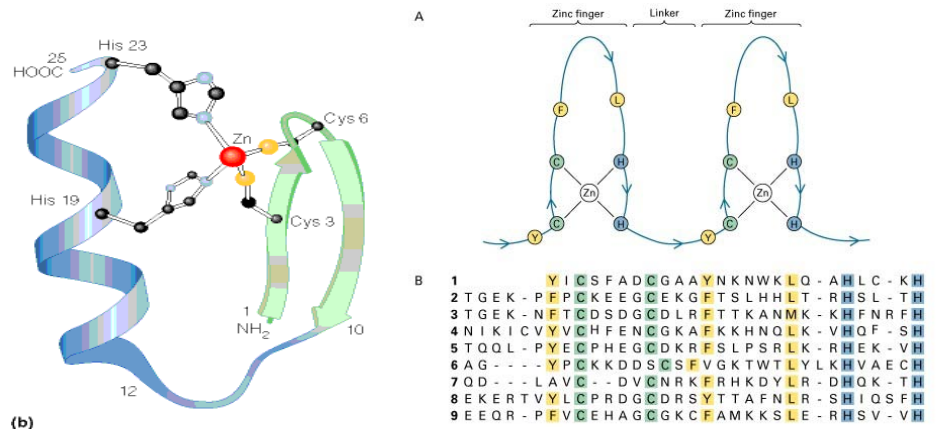
> What algorithm would you use to search for exact sequence matches?
> What is the time complexity of this algorithm?

The real problem is much more challenging than searching for exact matches for two reasons: (1) either sequence may contain regions not present in the other; (2) we don't expect perfect matches, just similar sequences. The exact solutions to this problem are slow when we need to examine large numbers of sequences: O(n^2). The most popular program, BLAST, are a lot faster, but are not guaranteed to find the right answer all the time.

We can see from a structural perspective that not every position in the protein will be equally preserved during evolution. Consider, for example, a class of protein that binds to DNA and regulates transcription that is known as a zinc finger (or more accurately a C_2H_2-style zinc finger). These proteins usually contain a number of small domains (<30 amino acids) and are shown in the figure below. In the middle is a zinc ion. The domains are so small, that they can't fold without the zinc.

Why do you think the zinc finger is not stable without zinc?

The zinc is held in place by four amino acids: two histidines and two cysteines. As you might imagine, these are absolutely conserved in evolution.



```
1           Y I C S F A D C G A A Y N K N W K L Q - A H L C - K H
2 T G E K - P F P C K E E G C E K G F T S L H H L T - R H S L - T H
3 T G E K - N F T C D S D G C D L R F T T K A N M K - K H F N R F H
4 N I K I C V Y V C H F E N C G K A F K K H N Q L K - V H Q F - S H
5 T Q Q L - P Y E C P H E G C D K R F S L P S R L K - R H E K - V H
6 A G - - - - Y P C K K D D S C S F V G K T W T L Y L K H V A E C H
7 Q D - - - L A V C - - D V C N R K F R H K D Y L R - D H Q K - T H
8 E K E R T V Y L C P R D G C D R S Y T T A F N L R - S H I Q S F H
9 E E Q R - P F V C E H A G C G K C F A M K K S L E - R H S V - V H
```

Images of a zinc finger proteins from

Look at the set of sequences that have been collected. When a set of sequences are put together, it is called a multiple sequence alignment (MSA). What do you notice about the zinc finger MSA?

Sequence search strategies

How can we take advantage of MSAs to identify other proteins that are likely to share the same structure?

One thing we could do would be to look for a pattern that all the sequences share. There is a website called "PROSITE" comes up with patterns for protein domains. The one they have for zinc fingers is

C - x(2,4) - C - x(3) - [LIVMFYWC] - x(8) - H - x(3,5) - H

This does a good job of recognizing zinc finger proteins. But what is wrong with it? It suggests that the amino acids L,I,V,M,F,Y,W and C are all equally probable in the position four residues after the second cysteine. It turns out that is far from true. Here are the actual numbers:

```
C:    3 %
F:   88 %
I:    1 %
L:    2 %
M:    0 %
V:    1 %
W:    0 %
Y:    6 %
```

In some cases, this kind of mistake might not hurt. The signal from the locations of the cysteines and histidines is strong, that if you used this pattern to search for zinc fingers you probably would make very few errors, even if you dropped many other parts of the alignment. However, let's say we were working in a genome where C and H were very common. Then this kind of approach would make more mistakes.

What are some of the problems with this approach?

1. The answers are all-or-nothing
2. The pattern doesn't take into account frequencies of amino acids in a domain
3. The pattern doesn't take into account frequencies of amino acids in a genome

There is a much more sensitive approach called a weight matrix. To make things simpler, let's look at recognizing DNA patterns rather than protein patterns. Then we only have to compute the numbers for four bases instead of 20 amino acids.

Let's say we want to find the regions of the DNA bound by the protein GCN4.
We could build a pattern that looks like this: TGA-[CG]-TCA

But it turns out that there are lots of sequences to which GCN4 binds that don't match this pattern:

Consider the valid GCN4 binding sites below:

```
TGACTCC
TGACTCA
TGACAAA
TGACTCA
TTACACA
TGACTAA
TGACTAA
TGACTCA
TGACTCA
TGACTCA
```

How many of these sequences match the consensus TGA-[CG]-TCA?

Homology Modeling

An alternative to the pattern-matching approach described above is to build a more quantitative model based on the frequencies of each base/amino acid at each position. In the examples below we will illustrate the concepts using DNA sequences so that our matrices only need four rows instead of twenty.

We'll start with the valid GCN4 binding sites above. The alignment is 7 bps long, so let's create a 4x7 matrix with the number of occurrences of each base at each position. (If we were working with protein sequences we would, of course, need to have a 20xN matrix to represent an N-residue pattern.)

Counts:
```
A:      0 |      0 |     10 |      0 |      2 |      3 |      9 |
C:      0 |      0 |      0 |     10 |      0 |      7 |      1 |
G:      0 |      9 |      0 |      0 |      0 |      0 |      0 |
T:     10 |      1 |      0 |      0 |      8 |      0 |      0 |
```

It's easy to turn these into frequencies, just divide by the total number of sites, which is ten here.

Frequencies:
```
A:  0.000 |  0.000 |  1.000 |  0.000 |  0.200 |  0.300 |  0.900 |
C:  0.000 |  0.000 |  0.000 |  1.000 |  0.000 |  0.700 |  0.100 |
G:  0.000 |  0.900 |  0.000 |  0.000 |  0.000 |  0.000 |  0.000 |
T:  1.000 |  0.100 |  0.000 |  0.000 |  0.800 |  0.000 |  0.000 |
```

If we had a large enough sample, these frequencies would approximate the corresponding probabilities. The frequency x_{A2} of "A" at position 2, for example, approximates $P(X_2=A)$ in a valid sequence.

How would we compute the probability of seeing the sequence "TGACTCA" if we assume that the probability at any position is independent of all other positions?

So far, we have solved two of our problems. We can now get a number instead of a yes/no answer. We can also be sensitive to the fact that some positions show a preference for one residue yet allow others.

What about the final problem – the fact that some bases might be very common in the genome?

The solution is to ask a slightly different question.
So far, what we have been computing is P(sequence |binding site).

Often, in probabilistic approaches, it's better to ask how likely it is to get a result compared to a null model. For example, let's go back to an example you probably saw in 6.00. Let's say I have two coins. One is fair and one is biased toward "heads" p(H)=.6. If I toss a coin three times, what is the probability of getting three heads for each coin:

fair coin has a prob(H,H,H,H|fair) = 1/16 = .0625
biased coin has a prob(H,H,H,H|unfair) = .6*.6*.6*.6   = .13

Both these probabilities sound like small numbers, but we are not interested in the absolute probability of an event. Rather, we have seen the event and we are trying to decide whether it makes more sense under one model (the coin is fair) or another model (the coin isn't fair).

To evaluate which model is better, we want to look at the ratio of these probabilities:

p(HHH|unfair)/p(HHH|fair) = .13/.06=2.1

So it is more than twice as likely the coin is unfair than fair.

This is called a **likelihood ratio**, and it is a very common test in statistics.

What is the likelihood ratio in our case? We wish to know the ratio of p(sequence|binding site)/p(sequence|genome). We have already seen how to compute the first term from frequencies. How would you compute the second term?

To get the likelihood ratio for the whole sequence you can easily prove (try it) that it's the product of the ratios for each base.

$$Model\_prob = \prod_{i=1}^{w} p_{\text{model}}(b,i)$$

$$Background\_prob = \prod_{i=1}^{w} p_{\text{background}}(b)$$

$$\frac{Model\_prob}{Background\_prob} = \prod_{i=1}^{w} \frac{p_{\text{model}}(b,i)}{p_{\text{background}}(b)} = \prod_{i=1}^{w} odds(b,i)$$

If we assume for convenience that all bases are equally frequent in the genome, then in our case that comes out to:

```
Frequencies
A:  0.000 |  0.000 |  1.000 |  0.000 |  0.200 |  0.300 |  0.900 |
C:  0.000 |  0.000 |  0.000 |  1.000 |  0.000 |  0.700 |  0.100 |
G:  0.000 |  0.900 |  0.000 |  0.000 |  0.000 |  0.000 |  0.000 |
T:  1.000 |  0.100 |  0.000 |  0.000 |  0.800 |  0.000 |  0.000 |
Likelihood
A:      0 |      0 |  4.000 |      0 |  0.800 |  1.200 |  3.600 |
C:      0 |      0 |      0 |  4.000 |      0 |  2.800 |  0.400 |
G:      0 |  3.600 |      0 |      0 |      0 |      0 |      0 |
T:  4.000 |  0.400 |      0 |      0 |  3.200 |      0 |      0 |
```

What would be the likelihood ratio for this sequence “AGACTCC”?

Do we really believe that if we have never seen an “A” in the first position it can **never** occur? Or do we mean that it is extremely unlikely?

I won’t go into the theoretical justification for it, but the way around this problem is to set these zero probabilities to a very small number. These are called pseudocounts.

So instead of saying we found exactly zero sequences with “A” at the first position, we could say that we found that there was a very low frequency of each base. In the example below each base gets a pseudocount of 0.25%.

```
Frequencies
A:  0.0025 |  0.0025 |  0.9926 |  0.0025 |  0.2005 |  0.2995 |  0.8936 |
C:  0.0025 |  0.0025 |  0.0025 |  0.9926 |  0.0025 |  0.6955 |  0.1015 |
G:  0.0025 |  0.8936 |  0.0025 |  0.0025 |  0.0025 |  0.0025 |  0.0025 |
T:  0.9926 |  0.1015 |  0.0025 |  0.0025 |  0.7946 |  0.0025 |  0.0025 |
Likelihood
A:  0.0099 |  0.0099 |  3.9703 |  0.0099 |  0.8020 |  1.1980 |  3.5743 |
C:  0.0099 |  0.0099 |  0.0099 |  3.9703 |  0.0099 |  2.7822 |  0.4059 |
G:  0.0099 |  3.5743 |  0.0099 |  0.0099 |  0.0099 |  0.0099 |  0.0099 |
T:  3.9703 |  0.4059 |  0.0099 |  0.0099 |  3.1782 |  0.0099 |  0.0099 |
```

To compute the matrix with pseudocounts, we added 0.25% to each frequency and then renormalized. In other words, if F(b,i) is the frequency for base *b* at position *i* and *p* is the pseudocount frequency, then the updated frequency F’(b,i) = (F(b,i)+p)/(1+4p)).

The number of pseudocounts to be added is somewhat arbitrary. Laplace’s rule states that we should just add one occurrence to each count. A slightly more principled approach would be to distribute the pseudocounts according to some prior probability distribution of what we expect to see. In practice, the pseudocounts are often distributed according to the background distribution of bases. In this case, we would have F’(b,i) = (F(b,i)+w*q(b))/(1+w), where *w* is an (arbitrary) scaling factor and *q(b)*is the background frequency of base *b*. In our example above, w=.01 and q(A)=q(C)=q(G)=q(T)=.25.

The probabilities can get very small, especially when you are looking at lots of positions. When you manipulate these numbers in the computer they will often be rounded down to zero. The way around this is to use logarithms. This is often done in base 2.

```
Log Likelihood
A: -6.658 | -6.658 |  1.989 | -6.658 | -0.318 |  0.261 |  1.838 |
C: -6.658 | -6.658 | -6.658 |  1.989 | -6.658 |  1.476 | -1.301 |
G: -6.658 |  1.838 | -6.658 | -6.658 | -6.658 | -6.658 | -6.658 |
T:  1.989 | -1.301 | -6.658 | -6.658 |  1.668 | -6.658 | -6.658 |
```

When we are using logarithms, the product them becomes a sum:

$$\frac{Model\_prob}{Background\_prob} = \prod_{i=1}^{w} \frac{p_{model}(b,i)}{p_{background}(b)} = \prod_{i=1}^{w} odds(b,i)$$

$$= \sum_{i=1}^{w} \log\_odds(b,i)$$

So to compute the score for any sequence we simply pick out the right elements of the matrix and add them up

$$\boxed{\text{Try computing the sum of the log likelihood for “\underline{A}GACTCC” and “\underline{T}GACTCC”}}$$

Now we could run this process over and over again for each position in a genome and plot the log-likelihood ratio.

The peaks here are the likely binding sites for the protein. You will see this again in the problem set.

We can do exactly the same thing for a protein sequence. One thing that we need to be careful about is that the alignment we start with is the best possible.
The alignment for zinc finger sequences in the figure is unfortunate because they should have added extra gaps to make sequence #6 work out properly.

Using Sequence Analysis for Structural Prediction
Once you have determined the domain structure of a protein, what you do next really depends on what your goal is. If you need a very rough structure you might be done. For example, let’s say you wanted a rough guess at the active site so you could choose positions to mutate. A crude homology model might be enough. From the model you could generate a sequence alignment and deduce a region that might be near the active residues.

What might you need higher resolution?

A lot of work in the pharmaceutical industry involves taking a lead compound and optimizing it. In some cases, the process can be carefully optimized to improve affinity for a protein through knowledge of the structure. You can look at the candidate compound bound to the protein and explore ways to improve it. For example, you might find that the compound fits well, but leaves an empty hydrophobic pocket.

Why would that be a problem?
What would you like to do to fix it?

The problem with this approach is that sometimes you can get the structure of one protein but be unable to get the structure of a second very similar protein. One of the bottlenecks in protein structure determination is the ability to grow crystals containing the highly pure protein. This process is rather haphazard, and has seen only modest improvements over many years. For unclear reasons you might be able to get the structure of a mouse protein but not be able to get the structure of the human ortholog, for example.

You don't have to give up. If there is a lot of sequence homology, then the mouse protein is probably a good stand-in for the human one. Of course, there can be differences in the details of the active site that matter. A single residue could change the hydrophobic pocket into a hydrophilic one. The modification that made sense before might be exactly the wrong one now.

How could we refine the structure based on homology to more accurately represent the true structure of the target?