Predicting local proteins structure using hidden Markov models

Practical information

You can work in pairs on this project. After completion you should write a report containing:

1. brief introduction to the problem,
2. related research: Take one of the three papers available on the course web page and read it. Discuss what you have done in the context of this paper (about 300 words),
3. method section: Describe your implementation,
4. results and discussion: Document your results. Don't just paste in output from your program. Make meaningful figures/tables summarizing the results. Discuss the results. Don't just say meaningless things like "we see that method A works better than method B". Discuss why you think this is!,
5. conclusions.

Email the report to Feifei within the deadline (Friday, 2008.10.17 at 12.00).

After the deadline you will receive the report of one of the other groups by email. During the presentation on Monday 2008.10.20 at 13-15 you are responsible for asking questions to that group. Prepare a short 15 min presentation. A computer and projector will be available.

If you meet the report deadline, include all parts of the report, hold a presentation and review another group's report you will be given 0-4 bonus point depending on the quality of your report, presentation and the insightfulness of your questions to the other group.

Good luck!

Background

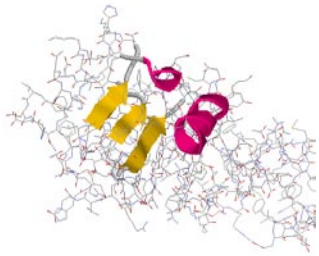
A detailed understanding of the molecular activity of proteins requires knowledge of their three-dimensional structure. However, experimental methods for determining protein structure, such as crystallography or NMR, are expensive and time consuming. For this reason experimental structures are only available for approximately 30 thousand of the 30 million protein sequences known today.

Computational methods for predicting the three-dimensional structure of proteins from sequence have come a long way in the last ten years, however, they are still largely unreliable unless a close sequence homologue of known structure exists. Computational methods for predicting the structure of a *target* sequence given a database of known structure can be divided into three classes:

1. Homology modeling (comparative modeling): A protein with high sequence similarity to the target exists in the structural database (i.e. *template*), and the problem is reduced to aligning the two sequences and transferring the structure of the template to the target.
2. Fold prediction: There is no clear sequence homology between the target and any sequence in the database. However, the fold of the target is represented in the database, and the problem becomes finding the correct fold and fitting the target sequence to this class of protein structures based on the physical/chemical properties of specific amino acids (threading).
3. New fold prediction (*ab initio* prediction): The correct fold is not represented in the database, and the structure of the target has to be modeled *ab initio* (for the beginning) by e.g. assembly using a library of protein fragments and an energy function.

In this project we will investigate a method based on *local descriptors of protein structure*. This approach is applicable both to fold and new fold prediction, however, here we will limit ourselves to model and recognize local substructures and use this to identify the correct fold of a protein domain.

a. Local descriptor: structure



b. Descriptor group: structure

c. Descriptor group: sequence

DESCRIPTOR	FRAGMENT 1	FRAGMENT 2	FRAGMENT 3	FRAGMENT 4	FRAGMENT 5
1qama_#37	35-40 FEIGSG	56-60 TAIEI	83-7 KDILQ	96-102 YKIFGNI	108-16 TDIIRKIVF
1g38a_#46	44-9 LEPACA	68-72 VGVEI	88-92 ADFLL	100-6 DLILGNP	144-52 GAFLEKAVR
1g55a_#9	7-12 LELYSG	31-5 AAIDV	55-9 KTIEG	71-7 DMILMSP	100-4 ----LHILD
1hdoa_#9	7-12 AIFGAT	31-5 TVLVR	53-7 GDVLQ	69-75 DAVIVLL	88-96 SEGARNIVA
1booa_#272	270-5 VDIFGG	291-5 ISFEM	33-7 GDSLE	48-54 SLVMTSP	77-85 LSFAKVVNK
1i9ga_#106	104-8 LEAGA-	128-32 ISYEQ	160-4 SDLAD	175-9 --AVLDM	183-91 WEVLDAVSR
1eg2a_#249	247-51 LDFFA-	268-72 ICTDA	45-9 CDCLD	60-4 QLIIC--	86-94 KRWLAEAER
1ek6a_#8	6-11 LVTGGA	30-4 VVIDN	65-9 MDILD	83-9 MAVIHFA	109-17 LTGTIQLLE
1bxka_#7	5-10 LITGGA	30-4 VVVDK	58-62 VDICD	78-82 --VMHLA	102-10 IVGTYTLLE
1qrra_#7	5-10 MVIGGD	29-33 CIVDN	74-8 GDICD	92-8 DSVVHFG	121-9 VIGTLNVLF
…	…	…	…	…	…	…	…	…	…	…

Figure 1. a) An example of a local descriptor of protein structure consisting of five fragments. b) Descriptors in other proteins that are structurally similar to the descriptor in a). c) The sequence alignment resulting from the structure alignment in a). Each row is one local descriptor named as 'protein domain name'#'central amino acid'.

Local descriptors of protein structure

A local descriptor of protein structure is a set of short backbone fragments centered in three dimensions around a particular amino acid (Figure 1a). A local descriptor is built by

a) identifying all close amino acids within a radius of 6.5 Å (an amino acid is represented as the point on the vector [C_α,C_β] that lies 2.5Å away from C_α),
b) for each close amino acid, adding four sequence neighbors, two from each side, to obtain continuous backbone fragments of five amino acids, and
c) merging any overlapping fragments into segments.

One can compute local descriptors from all amino acids in a representative set of protein domains from the Protein Data Bank (PDB, http://www.rcsb.org/pdb) with less than 40% sequence identity to each other and then construct a library of commonly reoccurring local descriptors by

a) for each local descriptor identifying a group of structurally similar local descriptors and
b) selecting a set of representative, partially overlapping *descriptor groups* (Figure 1b).

By doing this it is possible to obtain a relatively small number of local substructures (approx. 4000 descriptor groups) from which all protein structures in PDB can be

assembled. Each group presents a link between sequence and structure at the local level by providing several examples of sequence fragments that take on the same local structure conformation (Figure 1c).

a) Domain 1l3la1 from fold a4 b) Domain 1h9ka1 from fold b40

Figure 2. Representative structures for the folds studied in this project.

Data

In this project we will model the sequence-structure relationship in descriptor groups using hidden Markov models. We will look at a reduced dataset of six descriptor groups (Table I). All protein domains in one group come from the same fold, and there are two different folds (see SCOP: http://scop.mrc-lmb.cam.ac.uk/scop):

Fold a4: DNA/RNA-binding 3-helical bundle (Figure 2a)
Fold b40: OB-fold (Figure 2b)

Table I. The descriptor groups considered in this project.

Descriptor group	Number of similar descriptors	Number of sequence fragments	Fold
`1cuk_3#18`	42	3	b40
`1h9ka1#27`	39	4	b40
`1hlva1#42`	63	3	a4
`1jjcb3#54`	19	3	b40
`1l3la1#194`	28	3	a4
`1lnwa_#57`	54	3	a4

For each descriptor group you are given three files, e.g.

1. `1h9ka1#27_domains.txt`: This file lists the protein domain name, the local descriptor name and the fold for each members in the descriptor group:

```
1 1a0i_1 1a0i_1#270 b.40
```

```
2 1an8_1 1an8_1#39 b.40
3 1b8aa1 1b8aa1#39 b.40
4 1bkb_2 1bkb_2#93 b.40
5 1bvsa3 1bvsa3#19 b.40
…
```

2. `1h9ka1#27_sequences.txt`: This file lists the sequence of each domain with one or more descriptor in the group; one sequence per line. To more easily read the sequence into Matlab, the amino acids are coded as integers (see the file aminoacids.txt):

```
13 4 12 4 1 3 6 8 8 14 6 10 18 19 6 17 9 6 10 1 12 4 6 9 18
8 6 5 4 18 10 10 4 16 6 15 10 18 12 1 17 12 8 16 15 1 10 11
3 4 5 17 4 17 18 9 4 1 17 10 16 14 19 6 5 5 16 13 20 6 8 6
3 12 3 1 2 17 8 12 13 20 3 6 19 1 2 14 8 16 20 11 4 4 17 13
3 6 16 10 15 7 13 16 5 18 11 5 15
9 9 3 8 16 12 18 9 16 3 10 10 20 1 20 17 8 17 13 20 3 20 9
3 2 15 18 12 5 16 17 17 7 17 10 12 8 3 17 14 9 20 15 6 9 3
20 20 8 16 16 4 11 16 20 4 1 16 14 9 5 9 15 3 3 7 18 3 18 5
6 10 5 20 8 10 12 16 7 17 6 4 20 8 20 6 6 8 17 13 1 14 12
11 20 15 17 7 20 16 16 4 8 17 4 4 10 12 6 14 9 18 9 18 1 6
19 18 19 4 18 9 3 10 6 6 8 9 5 10 19 8 15 3 15 3 6 8 18 14
8 17 1 13 9 9 9 18 3 13 4 10 5 9 10 8 13 9 10 15 16 4 3 18
18 1 18 4 6 18 18 12 5 17 13 9 1 9 10 6 5 4 8 10 13 4 9 8
18 18 10 12 15 1 4 17
8 8 4 9 5 17 1 14 8 10 16 18 16 6 3 18 8 14 10 11 3 11 15 3
20 9 17 8 4 18 13 11 9 20 18 4 4 4 1 9 6 15 10 1 13 6 1 4
18 4 18 19 14 8 10 3 15 20 9 8 8 15 18 9 6
11 8 5 16 18 15 6 4 18 10 4 18 1 10 3 7 1 18 8 4 1 1 6 8 6
20 15 18 12 1 17 13 16 1 10 1 17 10 12 14 6 16 14 1 15 10
18 17 1 11 18 18 15 4 3 16 11 17 10 20 6 5 16
…
```

3. `1h9ka1#27_descriptors.txt`: This file lists the descriptors in the group by giving the start and stop positions of each fragment in the sequence; one descriptor per line.

```
5 11 26 34 36 42 85 91
23 29 33 41 47 53 65 71
21 27 35 43 44 50 70 76
5 11 15 23 26 32 47 53
5 11 15 23 24 30 42 48
…
```

Example:

The fourth line in the domain-file contains the descriptor 1bkb_2#93. This means that the fourth line in the sequence-file contain the sequence of domain bkb_2, and the fourth line of the descriptor-file contain the start and stop positions for the fragments of the descriptor 1bkb_2#93 in the sequence.

Thus the positions 5 11 15 23 26 32 47 53 correspond to the following fragments in the sequence (bold):

8 8 4 9 **5 17 1 14 8 10 16** 18 16 6 **3 18 8 14 10 11 3 11 15** 3 20 **9 17 8 4 18 13 11** 9 20 18 4 4 4 1 9 6 15 10 1 13 6 **1 4 18 4 18 19 14** 8 10 3 15 20 9 8 8 15 18 9 6

or if coded back to amino acids:

5-11 FTAQILS 15-23 DVIQLMDMR 26-32 KTIEVPM 47-53 AEVEVWQ

Note that the descriptor file gives a set of, in this case, four multiple alignments (i.e. the alignments of the corresponding fragments in each descriptor given by their structural similarity). The multiple alignments are intercepted by caps that vary in length from sequence to sequence. Note for example that the gap between the second and third fragment in 1bkb_2#93 (line 4) is of length two, while there is in fact no gap in 1bvsa3#19 (line 5). Note also, that, for simplicity, we consider only descriptor groups without deletions in this project.

Project

The data you need to do this project can be downloaded from the course website: Material.zip.

Although you are free to use any programming language for this project, some Matlab-functions to help with the practical issues of i/o et cetera is provided with the Material.zip file. See "help" in Matlab for how to be able to call these functions (also; see the end of this document). You may want to use the functions hmmviterbi, hmmdecode etc. from Exercise 4 also.

Remember: Play around with the problem and have fun. This is new research you are doing.

Task 1

Propose an architecture for a hidden Markov model that can model the set of multiple alignments, and the gaps between them, given by a local descriptor group. An end state can be incorporated by adding a special symbol (21) which denotes End. This should only be emitted by the end state.

Hint: There are no insertions/deletions within each multiple alignment, in other words, each position in the segments of the alignment should be represented by a hidden state.

Task 2

Build the HMM; one for each group. Estimate the parameters for each descriptor group using the training set.

Task 3

a) Investigate to what degree the HMMs are able to assign the six local substructures to the correct positions in the sequences used to train the models.

b) The files labeled TEST contains sequences that where not used to train the model and descriptors that match the groups we use to train the HMMs. Note that none of these sequence have a statistically significant sequence similarity to any sequences in the training set (i.e. BLAST E-score > 0.05). Investigate to what degree the HMMs are able to assign the six local substructures to the correct positions in these unseen sequences.

   OBS: Note that there is no TEST data for one of the groups (1h9ka1#27) and thus you will have to exclude this group from the current analysis.

c) Can you improve the performance for the sequences in a) by using pseudocounts, i.e. explicitly adding counts to emissions that never has been observed in the training data? What about for the sequences in b)? Explain what you observe.

Task 4

Given four sets of sequences:

A. Sequences used to build the HMM (i.e. the training set).
B. Sequences that match the group modeled by the HMM, but not in A (i.e. the TEST groups given in Task 3b)
C. Sequences from the same fold as sequences used to build the HMM, but not in A.
D. Sequences from a different fold than the sequences used to build the HMM.

Obtain the probabilities that sequences in the different sets were generated by the given HMM. Plot the distribution for each set. Explain what you see.

Task 5

Sequences with low homology to the training set will by nature be more difficult to accurately align to the HMM. One idea for improving the generalization capabilities of the model is to use a different symbol set for describing the sequences by e.g. labeling aminoacids by their physico-chemical properties. What is the motivation behind this idea? Design a new encoding of the aminoacid sequence based on e.g. physico-chemical properties or substitution frequencies. Translate the sequences to this new encoding and repeat Task 3 and 4.

For properties see e.g. http://en.wikipedia.org/wiki/Aminoacid

Task 6

Could you use secondary structure information (helix, coil, sheet et cetera) to improve your predictions? How would you incorporate such information into the HMM?

Could you use a multiple alignment consisting of the target and a set of related proteins (without known structure) to improve the performance? How? Why would this result in better performance?

Do you have other ideas for how to improve the performance of the HMMs?

Do you have any ideas for how to assemble complete protein structures given the HMMs? What is the crucial point in order for this to work?

Some Matlab-functions

You are given a set of Matlab functions to help with the implementation, consult the help function for each of them:

read_group – Reads a group into Matlab
get_group_count – Returns the number of descriptors in the group
get_segments – Returns the segments of a sequence
get_sequence – Returns a sequence from the group
get_path – Returns "the" true path assuming sequentially numbered states where the sequence is in an insert state (state 1) until the first segment. The first position in the first segment is state 2, the next state 3, et cetera. The next insert is the next state in order and so on until End. E.g., assume a sequence K long has two segments, two aa long positioned at 3-4 and 7-8 respectively. The path returned would be 112344567777 .. (*K 7s)*.. 78 where 8 is the end state.