Fold recognition using local descriptors of protein structure and Hidden Markov Models

BY

MINYAN HONG

Molecular Biotechnology and Bioinformatics

Uppsala University, May 15th, 2007

Supervisor: Dr. Torgeir R. Hvidsten

Linnaeus Centre for Bioinformatics (LCB)

Abstract:

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. A local descriptor is a set of continuous backbone fragments that are close in three dimensions. By clustering structurally similar descriptors, we got a link between sequence and structure at the local level. This master thesis used hidden Markov models (HMMs) to model the sequence-structure relationship represented by similar local descriptors (i.e. descriptor groups), and use this to predict protein fold from sequence (i.e. fold recognition). On a set of sequences with no significant sequence similarity to the training set, 74.7% had the correct fold as one of the top 5 predictions, and 33.1% had the correct fold as the top prediction. Moreover, the results showed that the actual amino acid information is necessary for predicting protein structure, while the secondary structure can be used as a complement (e.g., a rough alignment can be found by only using secondary information, but to find the exact alignment we need the additional information of amino acids).

Contents

Introduction

The unprecedented increase in the number of new protein sequences arising from genomics and proteomics highlights directly the need for methods to rapidly and reliably determine the molecular and cellular functions of these proteins (Zhang *et al*. 2003). Protein structure represents a powerful means of discovering function, because structure is well conserved over evolutionary time, and it therefore provides the opportunity to recognize homology that is undetectable by sequence comparison (Steven E. Brenner, 2001).

Structural genomics is a broad initiative of various centers aiming to provide complete coverage of the protein structure space (Iddo Friedberg, *et al.*, 2007). It first and foremost encompasses the technological and methodological advances enabling high-throughput automated treatment of targets, but also target selection, structural determination and analysis (Daniel J Rigden, 2006). Currently structural genomics is a conjoined experimental and computational effort, which is expected to provide a comprehensive repertoire of models of soluble globular protein domains (Steven E. Brenner, 2001).

Because of the massive amounts of protein sequence data that are derived from modern large-scale DNA sequencing efforts, experimental structural genomics using X-ray crystallography or NMR techniques still are lagging far behind the output of protein sequences [1]. Therefore, computational modeling seems to be the only way to close this growing gap.

A number of different computational approaches for protein structure prediction have been developed over the last 30 years (Silvio Carlo Ermanno Tosatto, 2002). It can be divided into three classes: homology modeling, fold recognition (protein threading) and new fold prediction (*ab initio*).

<u>Homology modeling</u>, which is also known as comparative modeling, predicts the three-dimensional structure of a given protein sequence (target) based primarily on its alignment to one or more proteins of known structure (templates) (Marti-Renom *et al.*, 2000). This method become less reliable with decreasing sequence similarity between the target and its template(s) (especially for sequence identity < 25%) (Hvidsten, *et al.*, 2003).

<u>Protein threading</u> is used when there is no clear sequence homology between the target and any sequence in the database, but the fold of the target is represented in the database. Therefore, the target sequence is threaded through the backbone structures of a collection of template proteins (i.e. fold library) and a “goodness of fit” score is calculated for each sequence-structure alignment.

Ab initio methods predict the structure from sequence alone, without relying on similarity at the fold level between the modeled sequence and any of the known structures (Bonneau, *et al.*, 2001). It assumes that the native structure corresponds to the global free energy minimum accessible during the lifespan of the protein and attempt to find this minimum by an exploration of many conceivable protein conformations (András Fiser and Andrej Sali).

In the project, we used hidden Markov models (HMMs) to model the sequence-structure relationship represented by similar local descriptors (i.e. descriptor groups), and use this to predict protein fold from sequence (i.e. fold recognition). Below are the main tasks in the project:

1) Assign local descriptors (substructures of proteins) to whole protein sequences to see whether they can be aligned to the true positions (i.e., alignment).
2) Calculate the probability of a certain sequence being generated by the HMMs, and then retrieve fold from these HMMs to see how the hidden Markov models work in fold recognition.
3) Experiment with and evaluate different amino acid alphabets for estimating parameters in the HMMs (using different substitution groups, chemical properties, etc.) for both alignment and fold recognition.

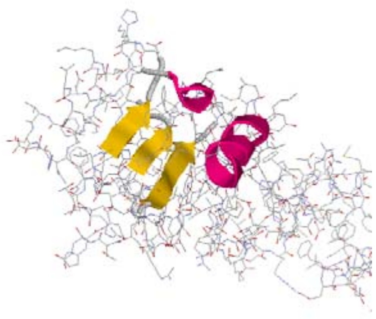
Data

The Structural Classification of Proteins (SCOP) database is a largely manual classification of protein structural domains based on similarities of their amino acid sequences and three-dimensional structures. It provides a comprehensive ordering of all proteins of known structure according to their evolutionary and structural relationships (Lo Conte, *et al*., 2002). The classification includes different hierarchical levels (Tim, *et al*., 1997): the first two levels, family and superfamily, describe near and distant evolutionary relationships; the third, fold, describes geometrical relationships.

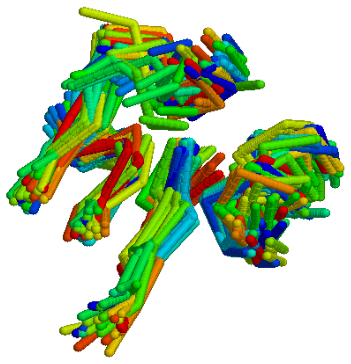
A local descriptor (Figure1. a) of protein structure encompasses short segments of a protein chain that are located around a selected amino acid residue (Kryshtafovych *et al*., 2003). A detailed description of descriptor construction can be found in (Hvidsten *et al*., 2003). Each descriptor is assigned an identification tag which reflects information about the domain of its belonging (according to the ASTRAL nomenclature (Brenner *et al*., 2000)) as well as the number of the central residue

(Hvidsten *et al*., 2003) (e.g. 1e43a2#231 is the descriptor from protein 1e43, chain a, domain 2 with origin at residue number 231). A set of structurally similar descriptors are clustered together to be a group (Figure1. b). In general, a decision on the similarity of descriptors is made by comparing the following parameters: number and length of segments, shape of individual segments, number of geometrically similar segments and the overall fit quality in terms of the RMSD score of their superposition (Hvidsten *et al*., 2003).

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
…	…	…	…	…	…	…	…	…	…	…

Figure1. 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'.

The data we used in the project is divided into two parts: one is the TRAIN set, which contains all non-overlapping groups from a representative set of all known protein structures in PDB (i.e. ASTRAL version 1.63). All sequences in TRAIN have less than 40% sequence identity to each other. Another is the TEST set, which contains descriptors from sequences in a later version of ASTRAL (1.67) that structurally match the TRAIN groups. All sequences in TEST have no significant sequence similarity to sequences in TRAIN (i.e. E-value < 0.05 using BLAST [2]). Moreover, all protein domains in one group (both TRAIN and TEST) come from the same fold. For DATASET1, each protein sequence is composed by 21 amino acids (20 standard amino acids and 1 non-standard amino acid). For both DATASET2 and DATASET3, the secondary structure (H: helix, E: strand, C: coil) are combined with the previous 21 amino acids information, thus we have 61 (20*3+1) symbols for coding a protein sequence. For the TRAIN set, DSSP ([5], Kabsch *et al.*, 1983) was used to obtain the secondary structure from PDB, while for the TEST set, PSIPRED ([6], Jones DT. 1999) was used to predict the secondary structure from sequence.

To more easily read the sequence into MATLAB, the amino acids are coded as integers, see below:

Without secondary structure:

```
A 1; C 2; D 3; E 4; F 5; G 6; H 7; I 8; K 9; L 10; M 11; N
12; P 13; Q 14; R 15; S 16; T 17; V 18; W 19; Y 20; X 21;
```

With secondary structure:

```
A-H 1; A-C 2; A-E 3; C-H 4; C-C 5; C-E 6; D-H 7; D-C 8; D-E
9; E-H 10; E-C 11; E-E 12; F-H 13; F-C 14; F-E 15; G-H 16;
G-C 17; G-E 18; H-H 19; H-C 20; H-E 21; I-H 22; I-C 23; I-E
24; K-H 25; K-C 26; K-E 27; L-H 28; L-C 29; L-E 30; M-H 31;
M-C 32; M-E 33; N-H 34; N-C 35; N-E 36; P-H 37; P-C 38; P-E
39; Q-H 40; Q-C 41; Q-E 42; R-H 43; R-C 44; R-E 45; S-H 46;
S-C 47; S-E 48; T-H 49; T-C 50; T-E 51; V-H 52; V-C 53; V-E
54; W-H 55; W-C 56; W-E 57; Y-H 58; Y-C 59; Y-E 60;
```

For example, a sequence from '1a0fa2#64_sequencesTRAIN',

'`MKLYIYDHCPYCLKARMIFGLKNIPVELHVLLNDDAETPTRMVGQKQVPILQKDDS RYMPESMDIVHYVDKLDGK`' ➔ '`11 9 10 20 8 20 3 7 2 13 20 2 10 9 1 15 11 8 5 6 10 9 12 8 13 18 4 10 7 18 10 10 12 3 3 1 4 17 13 17 15 11 18 6 14 9 14 18 13 8 10 14 9 3 3 16 15 20 11 13 4 16 11 3 8 18 7 20 18 3 9 10 3 6 9`' (after coded, without secondary structure information) ➔ '`32 27 30 60 24 59 8 20 5 37 58 4 28 25 1 43 31 22 13 16 28 25 35 23 38 53 12 30 21 54 29 29 35 8 8 2 10 49`

`37 49 43 31 52 17 41 26 41 53 38 24 30 42 27 8 8 47 45 60 33 39 11 46 31 7 22 52 19 58 52 7 25 28 8 17 26`' (with secondary structure).

Both DATASET1 and DATASET2 have 361 TRAIN groups that belong to 138 different SCOP folds. The corresponding TEST set has 105 groups containing 167 unique sequences with 46 different folds, but DATASET2 introduces secondary structure information. DATASET3 has 1802 TRAIN groups containing 2553 unique sequences with 77 different folds. All folds have at least 5 groups. The corresponding TEST set consists of 399 unique sequences with 53 folds. These sequences match 947 TRAIN groups (i.e. there are 947 TEST groups).

For example; group '1aisa1#66_TEST' in the TEST set corresponds to group '1aisa1#66_TRAIN' in the TRAIN set, i.e. '1aisa1#66_TEST' contains all descriptors from domains in the TEST set that structurally match the descriptor 1aisa1#66.

Methods

Hidden Markov Models

A basic Markov model of a process is a model where each state corresponds to an observable event and the state transition probabilities depend only on the current and predecessor state (Smith *et al*., 2002). A hidden Markov model (HMM) is a doubly embedded stochastic process with an underlying stochastic process that is *not* observable (i.e., hidden), but can only be observed through another set of stochastic processes that produce the sequence of observation (Rabiner, 1989).

Hidden Markov models (HMMs) now provide a coherent theory for profile methods. They are a class of probabilistic models that are generally applicable to time series or linear sequences (Sean R. Eddy., 1998). HMMs have been most widely applied to recognizing words in digitized sequences of the acoustics of human speech (Rabiner, 1989). They were introduced into computational biology in the late 1980s (Churchill, 1989), and for use as profile models in mid-1990s (Krogh *et al.*, 1994).

Now, HMM has many applications in bioinformatics and genomics, like prediction of protein-coding regions in genome sequences, modeling families of related DNA or protein sequences, prediction of secondary structure elements from protein primary sequences, etc.

In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a *hidden* Markov model, the state is not directly visible, but variables influenced by the state are visible.

Each state has a probability distribution over the possible output tokens. Therefore the sequence of tokens generated by an HMM gives some information about the sequence of states. Figure2 shows the rough architecture of a HMM that we are using in our project.

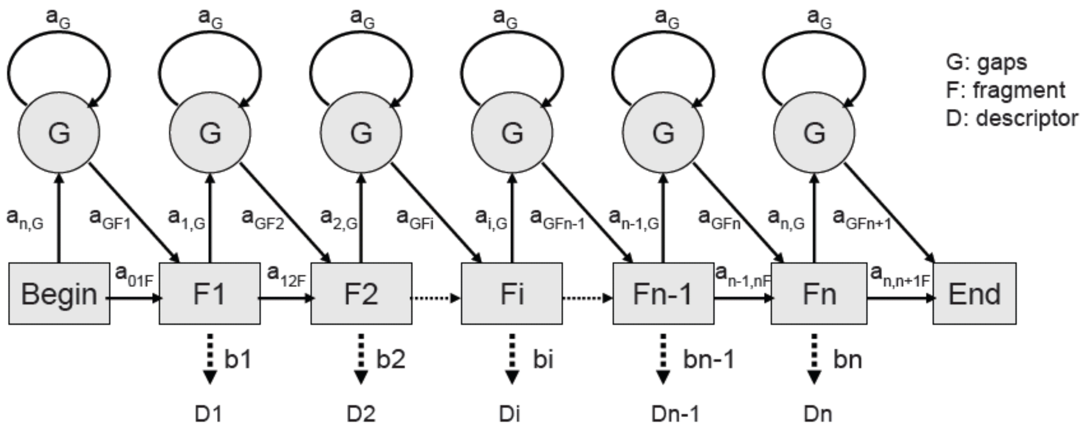


Figure2. Rough architecture of the hidden Markov model (HMM) used in the project. Rectangles and circles in shade (i.e. F and G) are hidden states; D: observable outputs; a: transition probabilities; b: output/emission probabilities.

In order to build a HMM, we need to at least know its two parameters 'transitions probabilities' and 'emissions probabilities'. In our data, the sequences and descriptors of each TRAIN group are known already, so the two parameters for HMM can be estimated from them (i.e. use descriptor information to estimate 'transition probability', use sequence and descriptor information together to estimate 'emission probability').

For example, the descriptor in the sequence '1b8aa1#23' in group '1a0i_1#247_domainsTRAIN' has three segments [or fragment]: '20 26 37 44 69 76'. Their correspond positions are shown in **bold** in the sequence: '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'. In the project, all amino acids that don't construct descriptors are considered as gaps, so if coded back to amino acids, it is '--…--KVAGWVW--…--LWIRDRDG--…--EDVVAVEG--…--'. In the HMM we used in the project, we call each amino acid position in the descriptor, plus the insertion in between segments, a state. The probability of moving from one state to the other state is the transition probability. In Figure2, the bottom line is called the main states. They model the fragments (i.e., local descriptors) of a sequence in our project. The transition probability are always ‘one’ both between main states (e.g., in fragment 20-26 `KVAGWVW` in the example sequence, the probability of moving from K➔V, or V➔A, … is ONE) and from main state to the next insert state. The upper line are called insert states, which are used to model the remaining part of a protein sequence (i.e., all continued gaps are considered as one insert state). The transition probabilities of moving between main states and insert states depend on the number of continued gaps (e.g., if an insert state has 5 continued gaps, the transition probability is 0.2 (1/5) from this insert state to the next state (i.e., first amino acid in the next main state), 0.8 for staying in this insert state (i.e., 0.2 for moving from one gap to the next gap within the insert state, 4 times moves)). The transition probabilities of moving from one amino acid to the forward nonadjacent amino acids or to any previous amino acids are zero. Each state has a probability of emitting certain output symbols (i.e., emission probability: the probability of a certain amino acid being emitted from a certain state), and these output symbols are calculated from fragments (e.g. 20-26 `KVAGWVW`, etc.) or gaps of a modeled protein sequence in our project. So, for the first insert state of the example sequence ‘1b8aa1#23’: ‘`11 20 15 17 7 20 16 16 4 8 17 4 4 10 12 6 14 9 18: MYRTHYSSEITEELNGQKV`’, the emission probabilities of each appeared amino acids are [M: 1/19, Y: 2/19, ..., E: 3/19, …, V: 1/19], and for those amino acids that haven’t appeared here (e.g., A, C, D, etc.), the emission probabilities are zeros. All probabilities mentioned so far are for one descriptor only. For a descriptor group, the probabilities (either transition or emission) will be added over all descriptors, and then averaged. For example, the emission probability of L in the first position in the first segment of sequences in Figure1 is 0.6 (10 descriptors of which 6 have L in the first position).

Because the probabilities for either transitions or emissions may be zero in some cases, we use ‘PSEUDOCOUNT’ in the project to avoid error. PSEUDOCOUNT is used here to get a positive number when having zero probabilities for transitions or emissions. In principle, this pseudo count indicates that there is some situation that is emitting this symbol or moving from a state x to another state y, even though this is not observed in the data. Thus this value should be 1 or larger than 1 (i.e. an integer) in theory. But in terms of mathematics, we can give any positive values to 'pseudocount'. This is needed to avoid 'divided by zero' mistakes. Also, it is used to incorporate prior knowledge into the model. Small pseudocount means that we estimate the parameters based mostly on the data, while higher pseudocount means that we place more confidence in our prior knowledge. Our experience from this project is that the smaller the value, the more reliable the HMM is. For DATASET1 and DATASET2, we used PSEUDOCOUNT ‘0.01’, while for DATASET3, we used the value '0.0001'. Thus, in the above example, the zero probabilities will be changed to a very small positive number (e.g., 0.00003, etc), and the original non-zero probabilities (e.g. 0.2: 1/5) will be changed a bit too.

Statistics Toolbox extends MATLAB to support a wide range of common statistical tasks [4]. It provides five functions for the analysis of hidden Markov models, including the generation of random data (hmmgenerate), maximum likelihood estimation of model parameters (hmmtrain and hmmestimate), calculation of most probable state sequences (hmmviterbi), and calculation of posterior state probabilities (hmmdecode).

In project, first we built HMMs for all TRAIN groups respectively (i.e., for each group in TRAIN set, where the protein sequences and the true paths of these respective sequences are known, we used 'hmmestimate' to estimate the transition and emission probabilities). Then, the function 'hmmviterbi', which uses the Viterbi algorithm, was used (in the alignment part) to compute the most likely sequence of states that the HMM would generate. Finally, in fold recognition, we used 'hmmdecode' to return the logarithm of the probability of the sequence. Because the actual probability of a sequence goes towards 0 rapidly as the length of the sequence increases, the probability of a sufficiently long sequence is less than the smallest positive number a computer can represent. Consequently, 'hmmdecode' returns the logarithm of the probability instead.

Alignment

For alignment, we use the generated HMM (from one TRAIN group) and one sequence from the corresponding group (either TRAIN group itself or TEST group) to estimate the hidden path (i.e. the descriptor situation). Then we align this estimated path to the true path to get a match degree. For example, a path corresponds to the sequence of states in the HMM that generate a descriptor. In the above example with sequence '1b8aa1#23', the descriptor is '20 26 37 44 69 76': '`--…--KVAGWVW--…--LWIRDRDG--…--EDVVAVEG--…--`', so the corresponding path of this sequence is '1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 **2 3 4 5 6 7 8** 9 9 9 9 9 9 9 9 9 9 9 **10 11 12 13 14 15 16 17** 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 18 **19 20 21 22 23 24 25 26** 27 27 27 …'. Each insert state (i.e., continued gaps) corresponds to one number, and each state in the main state corresponds to a unique continued number.

After performing this procedure on all sequences in one group, an average match degree for one group is obtained. This average match degree is considered as the performance in the alignment part of the project. We used three kinds of match degrees in this part, 'Original', 'Replace' and 'DescOnly'.

-- Original: Compare estimated-paths to their corresponding true-paths, and calculate to what degree they match;

-- Replace: First, replace original paths. E.g., PATH1: ‘1 1 1 1 1 2 3 4 5 6 7 7 7 7 8 9 10 10…’ ➔ PATH2: ‘0 0 0 0 0 1 1 1 1 1 0 0 0 2 2 0 0…’. In PATH1, numbers ‘1’, ‘7’ and ‘10’ are for gaps, and the remaining numbers are descriptors. In PATH2, all gap numbers are replaced by ‘0’, and the numbers in one segment are replaced by the same number (i.e., ‘1’ for the first segment in a sequence, ‘2’ for the second segment, and so on). Then we calculate the match degree for these replaced paths, including the gaps;

-- DescOnly: Exclude gap matches for both ‘Original’ and ‘Replace’ (i.e. only consider descriptors’ match) respectively. For example, for original, consider the true path ‘**1 1 1** 1 2 3 4 5 6 **7 7** 8 9 10 11 **11 12**’ and the estimated path ‘**1 1 1** 2 3 4 5 6 7 **7 7** 7 8 9 10 **11 12**’. If we include gaps, then the match score is (3+2+1+1)/17; while if we exclude gaps, the match score is 1/8. After replace, the true path and the estimated path change to ‘**0 0 0** 0 **1 1 1 1** 1 **0 0** 2 **2 2** 0 **0 3**’ and ‘**0 0 0** 1 **1 1 1 1** 0 **0 0** 0 **2 2** 2 **0 3**’ respectively. Similarly, if we include gaps, then the match score is (3+4+2+2+1+1)/17; while if we exclude gaps, the match score is (4+2+1)/8.

Fold recognition

In fold recognition, for each sequence in the TEST set, we match it to all HMMs (i.e., 361 for DATASET1 & DATASET2 and 1802 for DATASET3). We use ‘hmmdecode’ to calculate the probability of this sequence being generated (or emitted) by a certain HMM. Then we pick up the top 30 probabilities, and retrieve folds from each of these top 30 groups. The probability of a sequence being generated by a certain HMM is the product value of the probability that each elements in this sequence is generated by this HMM (i.e., emit probability of that element).

In order to give small classes (folds) an opportunity, we normalize the scores of the predicted folds in TOPS (i.e. top 30 here) in this way: if ‘TOPS = l+m+n+…’, we first normalize them to l/L, m/M, n/N,…, then sort these normalized values and pick up the top five predictions. Here, L, M, N mean 'how many times a certain group that belong to this fold (L, M, N, respectively), appear in the data (i.e. 361 or 1802), and l, m, n, mean 'how many times a certain fold appeared in the TOPS predictions.

Below is an example for the sequence ‘1vhoa2’ in DATASET3.

1vhoa2	LOGPSEQS	PREDICTED	CORRECT
1iwga3#627_TRAIN	-905.131	d.58	c.56
2pii__#74_TRAIN	-914.771	d.58	c.56
1lfwa2#261_TRAIN	-917.844	d.58	c.56
1kgsa2#51_TRAIN	-919.414	c.23	c.56
4tmka_#203_TRAIN	-923.941	c.37	c.56
1b00a_#19_TRAIN	-927.042	c.23	c.56
1mb3a_#22_TRAIN	-927.899	c.23	c.56
1psda3#397_TRAIN	-929.686	d.58	c.56
1gega_#43_TRAIN	-930.272	c.2	c.56
1loua_#27_TRAIN	-930.885	d.58	c.56
1esc__#299_TRAIN	-931.215	c.23	c.56
1a5t_2#44_TRAIN	-931.29	c.37	c.56
1ho1a_#190_TRAIN	-931.562	c.1	c.56
1f5na2#127_TRAIN	-932.743	c.37	c.56
1nksa_#121_TRAIN	-933.074	c.37	c.56
1m8pa3#430_TRAIN	-936.872	c.37	c.56
1d8wa_#256_TRAIN	-937.4	c.1	c.56
1qj4a_#33_TRAIN	-938.456	c.69	c.56
1m4la_#300_TRAIN	-938.63	c.56	c.56
1jfra_#223_TRAIN	-939.066	c.69	c.56
7reqb1#110_TRAIN	-940.967	c.1	c.56
1glqa2#55_TRAIN	-941.341	c.47	c.56
1jf9a_#313_TRAIN	-941.481	c.67	c.56
1k9sa_#86_TRAIN	-941.541	c.56	c.56
1nbwa3#538_TRAIN	-941.759	c.55	c.56
1kpga_#222_TRAIN	-942.103	c.66	c.56
1a8q__#267_TRAIN	-942.231	c.69	c.56
1kvka2#235_TRAIN	-942.576	d.58	c.56
1gpma1#287_TRAIN	-942.793	c.26	c.56
1ldda_#797_TRAIN	-943.391	a.4	c.56

* LOGPSEQS is the ‘logarithm of probability of sequence’ that generated by a certain HMM.

The predicted top 30 folds are: a.4 (1/19); c.1 (3/122); c.2 (1/113); c.23 (4/30); c.26 (1/26); c.37 (5/75); c.47 (1/26); c.55 (1/14); c.56 (2/15); c.66 (1/29); c.67 (1/73); c.69 (3/62); d.58 (6/28). (i.e., numbers in parenthesis are ‘how many times this fold has been predicted in the top 30 predictions / how many groups in TRAIN set are from this fold). After normalization, the top 5 predicted folds are [①d.58; ②c.23 and c.56; ③c.55; ④c.37;].

Then we compare these 5 predicted folds to the known correct folds (i.e., ‘c.56’ in this case) to see how good the HMMs perform for predicting folds. So, in this example, the predicted fold with the most support for sequence ‘1vhoa2’ is ‘d.58’. Thus this sequence is not correctly predicted. But as fold ‘c.56’ is still one of the top 5 predictions (②: c.23 and c.56), then it can still be a useful prediction and we call it a “partially correct” prediction. Moreover, if the true fold ‘c.56’ is not in the top 5 predictions, we say that the prediction failed. GOOD prediction combines ‘correct prediction’ and ‘partially correct prediction’. We predicted fold for all the test sequences and used this procedure to see how many sequences/folds that were predicted correctly.

In addition to using all amino acids, we also re-group (ENCODE) the protein sequences by their physico-chemical properties. We then performed the above two tasks ‘alignment’ and ‘fold recognition’ on the substituted data to obtain new results.

Below are four ENCODE ways

```
Encode 1                        Encode 2
acid: D E                       acid: D E
base: H K R                     base: H K R
hydrophile: C G N Q S T Y       polar: N Q S T Y
hydrophobe: A F I L M P V W     nonpolar: A C F G I L M P V W

Encode 3                        Encode 4
nonpolar: A F G I L M P V W     group_1_H = Helix;
uncharged_polar: C N Q S T Y    group_2_C = Coil;
charged_polar: D E H K R        group_3_E = Strand;
```

Results & Discussion

Alignment

As we met a problem during the alignment procedure for DATASET3, we only got match degrees for DATASET1 and DATASET2 (Table 1 and Table 2).

NOTE: ‘Original (Ori)’ means the original match degree while ‘Replace (R)’ means the match degree after replaced. ‘Ori_DescOnly’ and ‘R_DescOnly’ mean ‘DescOnly’ match degrees for ‘Original’ and ‘Replace’ respectively. (More detail in methods)

Table 1

TRAIN SET	DATASET1	DATASET2				
		Amino acids	ENCODE1	ENCODE2	ENCODE3	ENCODE4
Original (Ori)	0.842632	0.872893	0.803288	0.801343	0.785341	0.685959
Ori_DescOnly	0.702089	0.742978	0.597293	0.596255	0.565784	0.307547
Replace (R)	0.939662	0.951630	0.922837	0.921677	0.915312	0.872533
R_DescOnly	0.757395	0.797810	0.695780	0.692126	0.670803	0.536554

Table 2

TEST SET	DATASET1	DATASET2				
		Amino acids	ENCODE1	ENCODE2	ENCODE3	ENCODE4
Original (Ori)	0.624255	0.643714	0.621845	0.616145	0.618381	0.570327
Ori_DescOnly	0.254307	0.284446	0.277078	0.282819	0.282375	0.178066
Replace (R)	0.829250	0.855373	0.848949	0.848675	0.849566	0.829764
R_DescOnly	0.412008	0.488955	0.470972	0.472235	0.477787	0.419941

The DescOnly match degrees for Original path may illustrate the performance of the HMMs most accurately in the project.

After replace, results are raised in all cases. It's because the replaced path can consider state '**0** 0 **1 1** 1 **0 0**…' and '**0** 1 **1 1** 0 **0 0**…' to have 5 matches, while before replace they were '**1** 1 2 3 4 **5 5**…' and '**1** 2 3 4 5 **5 5**…' respectively, and have only 3 matches. If such situation extends to long sequences, then the difference will become larger.

The fact that the all values of 'Original' & 'Replace' are higher than the 'DescOnly' match degrees respectively is due to high contribution of gaps to the scores. After all, most of the positions in a sequence are gaps.

After expanding the data by introducing secondary structure (i.e. DATASET1 versus DATASET2), we got more information about the protein sequences, and this has resulted in increased match degrees in DATASET2 for all alignment scores.

After encoding the sequences in DATASET2, the precision of alignment is decreased more or less. It is because after re-grouping protein sequences by their physico-chemical properties, the detailed / accurate information we have for those sequences is reduced, so results are not as good as before.

The good performance of the HMMs for aligning TRAIN set shows that the model we used in the project is believable. Because of the low sequence similarity between TEST and TRAIN set, and the fact that the HMMs are built from TRAIN set, values for the TEST set are lower than for the TRAIN set in all cases. This also causes the difference between ‘DescOnly’ values to the values that include gaps (i.e., either ‘Original or ‘Replace’) in TEST set to be bigger than in TRAIN set.

The better ‘Replace’ values during aligning TEST sequences infers that the HMMs can find the rough position for substructures (i.e. overlapping can be detected) of low sequence similarity (compare to training set for the HMMs), but not exact position.

Results after ENCODE4 clearly drop compared with other performance. This may be because in ENCODE4, only secondary structure information has been used but no amino acids information. A natural interpretation is that amino acid information is important for deciding the placement of local descriptors (i.e., alignment).

Moreover, values from ENCODE4 (only secondary structure used) are much lower than values from DATASET1 (no secondary structure used). This indicates that for predicting the alignment of local descriptors to a protein sequence, the amino acid sequence is more important than the secondary structure information. The exception is to the task of only finding the rough position of the sub-structures (i.e., after replace). This is intuitive: we can find the rough alignment by only using secondary information (e.g. assigning a helical fragment to a helical part of the protein), but to find the exact alignment we need the additional information of amino acids.

Fold recognition

Table 3 compares results for DATASET1, DATASET2 and DATASET3. Table 4 shows results for DATASET3 and its ENCODE results.

In both tables, the upper three rows are results for sequences. Correct means that for a certain sequence, its predicted folds is the same as its true fold. Good refers to those cases where the prediction is either ‘correct’ or ‘partially correct’ (see detail in ‘Methods’). The bottom three lines are for folds. Correct here means that for a certain fold, at least one of the sequences that belongs to this fold has been predicted correctly (correspondingly for ‘partially correct’). “Good” contains both “correct” and “partially correct”.

Table 3

	DATASET1	DATASET2	DATASET3
Uni_Seqs:	167	167	399
Correct:	28 (0.168)	51 (0.305)	132 (0.331)
Good:	58 (0.347)	84 (0.503)	298 (0.747)
Uni_folds:	46	46	53
Correct:	16 (0.348)	26 (0.565)	30 (0.566)
Good:	22 (0.478)	31 (0.674)	41 (0.774)

Table 4

	DATASET3				
	Amino acids	ENCODE1	ENCODE2	ENCODE3	ENCODE4
Uni_Seqs:	399	399	399	399	399
Correct:	132 (0.331)	113 (0.283)	112 (0.281)	99 (0.248)	47 (0.118)
Good:	298 (0.747)	254 (0.637)	249 (0.624)	243 (0.609)	162 (0.406)
Uni_folds:	53	53	53	53	53
Correct:	30 (0.566)	25 (0.472)	26 (0.491)	25 (0.472)	12 (0.226)
Good:	41 (0.774)	44 (0.830)	44 (0.830)	44 (0.830)	35 (0.660)

For DATASET1 the HMM has 16.8% chance of predicting the fold of a sequence correctly, while 34.7% to predict the correct fold as one of the top five predictions. Values for folds show that 16 of the 46 unique folds (34.8%) have at least one sequence that can be predicted correctly, while 47.8% of the 46 unique folds can be predicted good.

In DATASET2, these predicted values improved to 30.5% (vs. 16.8%), 50.3% (vs. 34.7%), 56.5% (vs. 34.8%) and 67.4% (vs. 47.8%), respectively. As was the case for alignment, such clear improvement is caused by introducing secondary structure into DATASET2. Thus we got more information of the protein sequences, and results increased in all cases.

In DATASET3, results are similar to DATASET2 for ‘correct prediction’ of both sequences and folds, whilst the values for ‘good prediction’ in both cases show an increase (i.e., 74.7% vs. 50.3% for sequences and 77.4% vs. 67.4% for folds, respectively). This may be because DATASET3 have more groups that make the method more robust, or because the smaller PSEUDOCOUNT value used (i.e., 0.0001 instead of 0.01).

Moreover, it seems 'ENCODE' doesn't improve fold recognition so much, and only results in better prediction in 'GOOD_pred' for fold (which means that more folds can be predicted after using substitution by amino acids' physico-chemical properties). However, less can be predicted exactly after ENCODE.

As was the case for alignment, ENCODE4 shows a poorer prediction than other encode ways. From this we may infer that only using secondary structure and no amino acid information doesn’t work well. Again, much worse results from ENCODE4 than from before encoding (i.e., both using amino acids and secondary structure) in DATASET3 point out that amino acid information is necessary for fold recognition.

Furthermore, as mentioned before, the results for folds mean that at least one of the sequences that belong to this fold has been predicted correctly. Hence it is easier for folds to get higher scores than for sequences. However, the main reason for including fold-scores is to show that the sequences we predict correctly do not only come from one or a few folds, but that we in fact can handle many different folds.

Conclusion & Future

Generally, introducing secondary structure to the protein sequences brings more detailed information of the proteins, which has resulted in improved results for both alignment and fold recognition. The ENCODE ways seem not to help to predict either positions of local descriptors or folds of proteins. It infers that we were not able to show that the physical-chemical properties relate more to the protein's structure than using the amino acids themselves. Furthermore, the poorer ENCODE4 (using only secondary structure but no amino acids' information) results show that the amino acids are necessary for predicting protein structure, while the secondary structure can be used as a complement.

Allowing for sequence fragments of different length in the same descriptor group will greatly increase the size of the dataset we use and thus make the application more interesting in practice. Furthermore, BLAST can be used to build a profile from the sequence of interest (target) and similar sequences in the sequence databases. Predicting the structure of this profile rather than the single sequence may greatly increase the performance of the fold recognition procedure.

Acknowledgements

I would like to thank my supervisor Torgeir R. Hvidsten for introducing me to the research area of structural genomics and for helping me so patiently during the whole project.

References

András Fiser and Andrej Sali, Comparative protein structure modeling, Pels Family Center for Biochemistry and Structural Biology The Rockefeller University.

Bonneau R, Baker D. 2001. Ab initio protein structure prediction: progress and prospects. *Annu Rev Biophys Biomol Struct* 30:173-189.

Brenner, S.E., Koehl, P. and Levitt, M. (2000) The astral compendium for sequence and structure analysis. *Nucleic Acids Research.*, Vol. 28, No. 1 , 254–256.

Churchill, G.A. (1989) Stochastic models for heterogeneous DNA sequences. *Bull. Math. Biol.*, 51, 79-84.

Daniel J Rigden, (2006). Understanding the cell in terms of structure and function: insights from structural genomics. *Current Opinion in Biotechnology*, Volume 17, Issue 5, 457-464.

Hvidsten, T. R., Kryshtafovych, A., Komorowski, J. and Fidelis, K. (2003). A novel approach to fold recognition using sequence-derived properties from sets of structurally similar local fragments of proteins. *Bioinformatics*. 19, II81-II91.

Iddo Friedberg, Adam Godzi, (2007). Functional Differentiation of Proteins: Implications for Structural Genomics. *Structure*, Volume 15, Issue 4, 405-415

Jones DT. (1999) Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol*. 292: 195-202.

Kabsch W, Sander C. (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12): 2577-637.

Krogh, A., Brown, M., Mian, I.S., Sjolander, K. and Haussler, D. (1994) Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.*, 235, 1501-1531.

Kryshtafovych, A., Hvidsten, T. R., Komorowski, J. and Fidelis, K. (2003). Fold Recognition Using Sequence Fingerprints of Protein Local Substructures. In *IEEE Computer Society Bioinformatics Conference*, pp. 517-518. IEEE Computer Society,.

Lo Conte L, Brenner SE, Hubbard TJP, Chothia C, Murzin AG (2002). SCOP database in 2002: refinements accommodate structural genomics. *Nucleic. Acid Research.*, Vol. 30, No.1, 264-267.

Marti-Renom MA, Stuart AC, Fiser A, Sanchez R, Melo F, Sali A. (2000). Comparative protein structure modeling of genes and genomes. *Annu Rev Biophys Biomol Struct* 29:291-325.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77 (2), 257-286

Sean R. Eddy, (1998). Profile hidden Markov models. *Bioinformatics Review*, Vol. 14, 755-763.

Silvio Carlo Ermanno Tosatto, (2002). Protein Structure Prediction: Improving and Automating Knowledge-based Approaches. PhD thesis, Fakultät für Mathematik und Informatik, Universität Mannheim.

Smith, K. (2002) Hidden Markov Models in Bioinformatics with Application to Gene Finding in Human DNA, *Machine Learning Project Proposal.*, 308-761

Steven E. Brenner, (2001). A tour of structural genomics, *Nat Rev Genet 2*, pp. 801–809

Tim J. P. Hubbard1, Alexey G. Murzin1, Steven E. Brenner and Cyrus Chothia, (1997). SCOP: a Structural Classification of Proteins database. *Nucleic Acids Research*, Vol. 25, No. 1, 236–239

Zhang, C. and Kim, S. H. (2003). Overview of structural genomics: from structure to function. *Curr Opin Chem Biol*. 7(1): 28-32.

[1] http://www.wikipedia.org

[2] http://www.osc.edu/research/bioinformatics/FAQ/evalue.shtml

[3] http://www.mathworks.com

[4] http://www.mathworks.com/access/helpdesk/help/toolbox/stats/

[5] http://www.rcsb.org/pdb/home/home.do

[6] http://swift.cmbi.ru.nl/gv/dssp

[7] http://bioinf.cs.ucl.ac.uk/psipred